

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/328784548>

Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic

Thesis · October 2017

CITATIONS
58

READS
3,849

1 author:



Nour Moustafa
UNSW Canberra
262 PUBLICATIONS 17,740 CITATIONS

[SEE PROFILE](#)

**Designing an online and reliable statistical
anomaly detection framework for dealing with
large high-speed network traffic**

**Nour Moustafa Abdelhameed Moustafa
B.Sc., M.S**

A thesis submitted in fulfilment of the requirements

for the degree of

Doctor of Philosophy



UNSW
A U S T R A L I A

School of Engineering and Information Technology
The University of New South Wales
Australia

June 2017

PLEASE TYPE**THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet**Surname or Family name: **Moustafa Abdelhameed Moustafa**First name: **Nour**

Other name/s:

Abbreviation for degree as given in the University calendar: **PhD**School: **SEIT**Faculty: **UNSW Canberra**Title: **Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic****Abstract 350 words maximum: (PLEASE TYPE)**

Despite a Network Anomaly Detection System (NADS) being capable of detecting existing and zero-day attacks, it is still not universally implemented in industry and real applications, with current systems producing high False Positive Rates (FPRs) and low Detection Rates (DRs). The challenges involved in designing a NADS architecture are 1) the methodology adopted for creating as comprehensive a profile as possible from diverse normal patterns and 2) in establishing an adaptive and lightweight Decision Engine (DE) which efficiently distinguishes between legitimate and anomalous activities at high speeds in large network environments. The need for such a method to be trained and validated on a decent dataset with the characteristics of current network environments is a significant challenge.

This thesis provides substantial contributions to research on the building of a scalable, adaptive and lightweight NADS framework. It considers three aspects: a data source, relevant features and observations, and new DE approaches for achieving a reliable NADS architecture.

The first key contribution is the creation of a new dataset called UNSW-NB15 that has a hybrid of realistic modern legitimate and synthetic malicious activities, with statistical analyses and evaluations of it fully explained. Also, its complexity is assessed using existing techniques to demonstrate the extent of current sophisticated types of anomalous events.

The second core contribution is the development of a new theory for selecting important features and observations from network packets without redundancy to construct a legitimate profile from which any deviation is considered an attack, that is, establish an efficient NADS from analyses of the protocols and services of the OSI model.

The third major contribution is the development of two scalable frameworks with two new DE techniques for successfully detecting malicious activities in less processing times than current methods. These techniques, called the Geometric Area Analysis-ADS (GAA-ADS) and Dirichlet Mixture Model-ADS (DMM-ADS), are based on mixture models for modelling all possible normal patterns and detecting abnormal events that deviate from them using new outlier approaches.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

.....
Signature.....
Witness.....
Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY

Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

Copyright Statement

'I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights.

I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted. I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.'

Signed

Date:.....

Authenticity Statement

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed

Date:.....

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked at UNSW or elsewhere, during my candidature, is fully acknowledged. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed

Date:.....

Acknowledgments

First of all, my faithful gratitude goes to ALLAH, the Ubiquitous God, the most Gracious and Merciful. This work has been accomplished because of the unlimited mercy and blessings of ALLAH during my PhD journey.

I would like to express my sincere thanks and genuine gratitude to the following people who have supported and assisted me to successfully complete this work.

Of course, as Prophet Muhammad, peace and blessings be upon him, said “He who does not thank people, does not thank ALLAH”.

I wish to express my sincere gratitude to Professor Jill Slay, my principal supervisor, and Dr. Gideon Creech, my joint supervisor, for their outstanding supervision, precious advice, constant skilled guidance and insightful discussions. Despite their busy schedules, they always arranged meetings to provide me with genuine feedback that enabled me to achieve a remarkable research record in the cyber security field. Also, Dr. Shane McGrath and Dr. Paul Malcolm, researchers in the Australian DSTO, guided me in this research direction when I began my PhD journey while Mrs. Denise Russell assisted me by proofreading my thesis.

I thank my mother and offer the rewards of this work to the spirit of my father. I am also very appreciative of my brothers and sisters support for my travelling abroad to complete my PhD degree. I wish to express my gratitude and sincere appreciation to my wife, Marwa, and my little daughter, Nardeen, who are the flowers of my life as they are always patient and supportive of me doing my best work.

I would like to thank all the people at the Australian Centre for Cyber Security (ACCS), especially Professor Greg Austin, Mr Luke Garner, Dr. Elena Sitnikova, Dr. Nalin Gamagedara Arachchilage and Dr. Benjamin Turnbull. Moreover, I offer my sincere gratitude to the UNSW, Canberra, for awarded me a scholarship to

complete my PhD degree, and the school administration and IT support members who provided me with all the necessary facilities and took care of my requirements. Special thanks to Mr. Peter Newman who helped me in sniffing the network traffic of the UNSW-NB15 dataset. Last but not least, I am grateful to Professor Ruhul Sarker who guided me towards obtaining the scholarship and beginning my PhD under the supervision of Professor Jill Slay.

Abstract

Despite a Network Anomaly Detection System (NADS) being capable of detecting existing and zero-day attacks, it is still not universally implemented in industry and real applications, with current systems producing high False Positive Rates (FPRs) and low Detection Rates (DRs). The challenges involved in designing a NADS architecture are 1) the methodology adopted for creating as comprehensive a profile as possible from diverse normal patterns and 2) in establishing an adaptive and lightweight Decision Engine (DE) which efficiently distinguishes between legitimate and anomalous activities at high speeds in large network environments. The need for such a method to be trained and validated on a decent dataset with the characteristics of current network environments is a significant challenge.

This thesis provides substantial contributions to research on the building of a scalable, adaptive and lightweight NADS framework. It considers three aspects: a data source, relevant features and observations, and new DE approaches for achieving a reliable NADS architecture.

The first key contribution is the creation of a new dataset called UNSW-NB15 that has a hybrid of realistic modern legitimate and synthetic malicious activities, with statistical analyses and evaluations of it fully explained. Also, its complexity is assessed using existing techniques to demonstrate the extent of current sophisticated types of anomalous events.

The second core contribution is the development of a new theory for selecting important features and observations from network packets without redundancy to construct a legitimate profile from which any deviation is considered an attack, that is, establish an efficient NADS from analyses of the protocols and services of the OSI model.

The third major contribution is the development of two scalable frameworks with two new DE techniques for successfully detecting malicious activities in less processing times than current methods. These techniques, called the Geometric Area Analysis-ADS (GAA-ADS) and Dirichlet Mixture Model-ADS (DMM-ADS), are based on mixture models for modelling all possible normal patterns and detecting abnormal events that deviate from them using new outlier approaches.

List of publications

The following papers were produced in this Ph.D. research:

Refereed Journal Articles

- **N. Moustafa**, J. Slay, G. Creech, “Novel Geometric Area Analysis Technique for Anomaly Detection using Trapezoidal Area Estimation on Large-Scale Networks”, IEEE transactions on Big Data for Cyber security Applications, 2017.
- **N. Moustafa**, G. Creech and J. Slay. “Detecting Malicious Activity of DNS and HTTP Protocols: An Ensemble Learning Framework using Proposed Statistical Features”, Journal of Computers & Security, ELSEVIER, 2017, “in press”.
- **N. Moustafa**, and J. Slay. “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set”, Information Security Journal: A Global Perspective, 2016, p 1-14.
- **N. Moustafa**, G. Creech and J. Slay.”A Comprehensive survey: Components of Intrusion Detection systems“, 2017, “under review”.

Book Chapters

- **N. Moustafa**, G.Creech and J. Slay. "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models." Data Analytics and Decision Support for Cybersecurity. Springer, 2017. 127-156.

Conference Papers

- **N. Moustafa**, G. Creech and J. Slay. “Flow aggregator module for analysing network traffic”, the International Conference on Computing Analytics and Networking (ICCAN 2017), KIIT University, Springer, 2017.
- **N. Moustafa**, G. Creech and J. Slay. “Anomaly Detection System using Beta Mixture Models”, the International Conference on Computing Analytics and Networking (ICCAN 2017), KIIT University, Springer, 2017.
- **N. Moustafa**, G. Creech, E. Sitnikova and M. Keshk. “ Collaborative Anomaly Detection Framework for handling Big Data of Cloud Computing”, Military Communications and Information Systems Conference (MilCIS), IEEE, 2017.
- M. Keshk, **N. Moustafa**, E. Sitnikova and G. Creech. “ Privacy Preservation Intrusion Detection Technique for SCADA Systems”, Military Communications and Information Systems Conference (MilCIS), IEEE, 2017.
- **N. Moustafa**, J. Slay. “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”, Military Communications and Information Systems Conference (MilCIS), IEEE, 2015.
- **N. Moustafa**, J. Slay. “The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems”, Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 4th International Workshop on. IEEE, 2015.
- **N. Moustafa**, J. Slay. “A hybrid feature selection for network intrusion detection systems: central points and association rules”, Proceedings of the 16th Australian Information Warfare Conference (IWAR), Perth, Australia, November 2015.
- **N. Moustafa**, J. Slay. “Creating novel features to anomaly network detection using DARPA-2009 data set“, Proceedings of the 14th European Conference on Cyber Warfare and Security (ECCWS), Academic Conferences Limited, 2015.

Contents

Title page	i
Copyright Statement	iii
Originality statement	iv
Acknowledgments	v
Abstract	vii
List of publications	ix
Table of Contents	xi
List of Figures	xvii
List of Tables	xxi
List of Terms	xxv
1 Introduction	1
1.1 Overview of Cyber Security in Current Era	1
1.2 Dimensions of Cyber Crime	3
1.3 Problem Formulations and Research Questions	5
1.4 Protection against Cyber Crime	7
1.4.1 Network Data Sources	10
1.4.2 Relevant Features and Observations for NADS.	11
1.4.3 Statistics-based NADS	12
1.5 Thesis Contributions	13
1.6 Thesis Structure	14
2 Background and Related Work	17
2.1 Objectives	17

2.2	Intrusion Detection System (IDS)	17
2.2.1	Intrusion Detection Properties	19
2.2.2	Monitored environment	20
2.2.3	Detection methods	22
2.2.4	Deployment architecture	23
2.3	Characteristics of Network Anomalies	24
2.4	Evaluation Metrics for IDSs	24
2.5	Challenges of NADS	29
2.6	Components of NADS	31
2.6.1	Data source	32
2.6.2	Data pre-processing	39
2.7	Decision Engine (DE) Approaches	44
2.7.1	Classification-based approaches	45
2.7.2	Clustering-based approaches	49
2.7.3	Knowledge-based approaches	51
2.7.4	Combination-based approaches	53
2.7.5	Statistical-based approaches	55
2.8	Contemporary Network Threats	68
2.9	Chapter Conclusion	71
3	Towards Development of New Environments of Large-scale Network Datasets and Their Features for evaluating Intrusion Detection Sys- tems	73
3.1	Target of Network Dataset	73

3.2	Description of DARPA-2009 Dataset	75
3.2.1	Security Events in DARPA-2009 Dataset	76
3.3	Framework for evaluating DARPA-2009 Dataset	77
3.3.1	Pcap Transformation and Labelling	78
3.3.2	Proposed Statistical Feature Selection	79
3.3.3	Preparation of Training and Testing Sets for DARPA-2009 Dataset .	83
3.3.4	Evaluation of Four ML Algorithms.	84
3.4	Generation of UNSW-NB15 Dataset	86
3.4.1	Configuration of UNSW-NB15 Dataset Testbed	87
3.4.2	Network Traffic Analysis	91
3.5	Framework for generating UNSW-NB15 Features	91
3.5.1	Features extracted using Argus tool	95
3.5.2	Features extracted using Bro-IDS tool	95
3.5.3	Matched features	96
3.5.4	Additional Features	98
3.5.5	Labelling Process	105
3.5.6	UNSW-NB15 Security and Malware Events	107
3.5.7	File Formats of UNSW-NB15 Dataset	111
3.6	Comparisons with Other Datasets	112
3.7	Big Data Properties in UNSW-NB15 Dataset	115
3.8	Dataset Splitting for Learning Techniques	118
3.9	Complexity Analysis of UNSW-NB15 Dataset	120
3.9.1	Z-score Function	121
3.9.2	Kolmogorov-Smirnov (K-S) Test	122
3.9.3	Multivariate Skewness and Kurtosis	124

3.10 Use of Statistical Measures on training and testing sets	125
3.10.1 Feature Correlations of (TR_{I_N}) and (TS_{I_N})	126
3.10.2 Evaluation of Five ML Techniques	129
3.11 Empirical Results and Discussion	131
3.11.1 Statistical Analyses and Explanations	131
3.11.2 Feature Correlations	135
3.11.3 Complexity Evaluations using ML Techniques	137
3.12 Chapter Conclusion	139
4 Relevant Feature and observation Methods and Their Impacts on Design of Lightweight Network Anomaly Detection System	142
4.1 Introduction	142
4.2 Network Flow Analysis	145
4.2.1 NetFlow	146
4.2.2 sFlow	146
4.2.3 IPFIX	147
4.3 Aggregator Module for ADS.	147
4.3.1 Sampling Techniques	150
4.3.2 Association Rule Mining (ARM).	153
4.4 Network Feature Creation	157
4.4.1 Proposed DNS and HTTP Features	159
4.4.2 Proposed Ensemble Method for detecting DNS and HTTP Malicious Activities	165
4.5 Role of Feature Reduction (FR)	170
4.5.1 ARM Feature Selection Method	171
4.5.2 Principal Component Analysis (PCA)	173

4.5.3	Independent Component Analysis (ICA)	174
4.6	Experimental Results and Discussion	177
4.6.1	Aggregator Module	178
4.6.2	Evaluation of Proposed Features of DNS and HTTP	184
4.6.3	Evaluation of proposed ensemble method and discussion	185
4.6.4	Feature Reduction and Evaluation	190
4.7	Chapter Conclusion	199
5	Novel Statistical Decision Engines for Anomaly Detection System based on analysing Potential Characteristics of Network Features	203
5.1	Introduction	203
5.2	Network Data Analytics for Design of Effective DE	207
5.2.1	Normality measures	209
5.2.2	Linearity measures	210
5.3	Novel Geometric Area Analysis (GAA-ADS) Technique	212
5.3.1	Beta Mixture Model (BMM)	213
5.3.2	Trapezoidal Area Estimation (TAE)	219
5.3.3	Construction of Normal Profile of GAA-ADS Technique.	220
5.3.4	Testing Phase and Decision-making Method of GAA-ADS Technique	223
5.4	Novel Dirichlet Mixture Model-based ADS (DMM-ADS) Technique	225
5.4.1	Finite DMM	225
5.4.2	Training Phase of Normal Observations of DMM-ADS Technique.	228
5.4.3	Testing Phase and Decision-making Method in DMM-ADS Technique	229
5.5	Two Proposed Scalable Frameworks for ADS.	231
5.5.1	Data Sniffing and Storing Module	234

5.5.2	Data Pre-processing Module	235
5.6	Experimental Results and Analysis	237
5.6.1	Statistical Analysis and Decision Support	238
5.6.2	Performance Evaluation of GAA-ADS Technique	242
5.6.3	Performance Evaluation of DMM-ADS Technique	254
5.6.4	Comparative Study and Discussion of Both New DE Techniques . .	262
5.6.5	Clarifications of Complexity and Time Cost of Each New DE Tech- nique	264
5.7	Chapter Summary	267
6	Conclusion	270
6.1	Introduction	270
6.2	Contributions of Research	272
6.3	Limitations	275
6.4	Future directions	277
6.4.1	Issues to be resolved	277
6.4.2	Open questions	278
6.5	Final remarks	279
	References	281
A	Protocols of UNSW-NB15 dataset	313
B	Features of NSL-KDD dataset	316

List of Figures

1.1	Thesis overview	15
2.1	Architecture of classical IDS	19
2.2	ROC curves - A, B and C show levels of detection	27
2.3	Components of NADS	31
2.4	Main steps in feature selection	40
2.5	Taxonomy of network anomaly detection approaches	45
2.6	Classification types	46
2.7	NADS classifications	46
2.8	Methodologies of clusters and outliers	50
2.9	Recent top network attacks	69
3.1	Framework for analysis of samples selected from DARPA-2009 dataset	78
3.2	Architecture of UNSW-NB15 testbed network	90
3.3	Concurrent transactions of flows over simulation periods	92
3.4	Framework for creating features of UNSW-NB15 dataset	94
3.5	Statistical behaviours of normal and abnormal observations	112

3.6	Ethernet data rates transmitted and received over simulation periods	117
3.7	Probabilities of features being in training and testing sets	133
3.8	Skewness values of features in training and testing sets	134
3.9	Kurtosis values of features in training and testing sets	135
3.10	PCCs of features in training and testing sets	136
3.11	Gain ratios of features in training and testing sets	137
3.12	ROC curves of five techniques using two sets from UNSW-NB15 dataset	138
4.1	Proposed aggregator module	148
4.2	Proposed features of DNS and HTTP	161
4.3	Tools used to create DNS and HTTP features	163
4.4	Adaboost flowchart	169
4.5	Example of ARM FS method using UNSW-NB15 dataset	173
4.6	Scatterplot matrix analysis of data sample	180
4.7	Scatterplot matrix analysis of selected sample	181
4.8	Comparison of flow aggregator mechanisms	184
4.9	Correlations of proposed features	186
4.10	ROC curves of classification algorithms using DNS data source . . .	187

4.11	ROC curves of classification algorithms using HTTP data source	188
4.12	Portions of association rules using both datasets	194
4.13	ROC curves of three ML algorithms using ARM	196
4.14	ROC curves of three ML algorithms using ICA	199
4.15	ROC curves of three ML algorithms using PCA	200
5.1	Feature vectors for mining and gathering in UNSW-NB15 dataset .	208
5.2	BMM for two arbitrary variables	218
5.3	Composite trapezoidal rule	219
5.4	Parameters of finite DMM	226
5.5	Proposed framework for establishing scalable, adaptive and lightweight GAA-ADS	232
5.6	Proposed scalable framework for design of intelligent DMM-ADS .	233
5.7	Example of converting categorical features into numerical features using UNSW-NB15 dataset	236
5.8	Q-Q plots of feature vectors adopted from NSL-KDD and UNSW-NB15 datasets	240
5.9	Normal and suspicious density probabilities for some instances in both datasets	241
5.10	Correntropy plots of some instances in both datasets	243

5.11 ROC curves for original features in two datasets obtained by GAA-ADS technique for different K values	245
5.12 ROC curves obtained from GAA-ADS technique for components in both datasets with different K values	247
5.13 ROC curves obtained from GAA-ADS technique for both datasets .	248
5.14 Comparison of DRs (%) obtained by GAA-ADS technique for both datasets with increasing K values	250
5.15 Comparison of DRs (%) obtained by GAA-ADS technique with 20 K values	253
5.16 ROC curves obtained from DMM-ADS technique for both datasets with different w values	256
5.17 ROC curves obtained from DMM-ADS technique for components in both datasets with different w values	258
5.18 ROC curves using both datasets of DMM-ADS technique	259
5.19 Comparison of complexities of four existing and two new DE techniques	267

List of Tables

2.1	Confusion matrix for binary classification problems	25
2.2	Four attack classes in KDD99 dataset	35
2.3	KDD99 dataset distributions of attack and normal instances	35
2.4	Distributions of attacks in NSL-KDD dataset	36
2.5	Comparison of decision engine techniques	67
3.1	Input features	74
3.2	Sample of highest probabilities of normal and attack classes	81
3.3	Descriptions of selected features	82
3.4	Numbers of records of each attack type in first 30 pcap files	83
3.5	Numbers of records in training and testing sets	84
3.6	Confusion matrices of four techniques	85
3.7	Evaluation metrics of four algorithms	86
3.8	Comparative analysis of proposed framework and related studies . .	87
3.9	Statistics of UNSW-NB15 dataset	93
3.10	Flow features	96
3.11	Basic features	97

3.12 Content features	97
3.13 Time features	98
3.14 Additional generated features	99
3.15 Labelled features	106
3.16 Class distributions in UNSW-NB15 dataset	111
3.17 Comparisons of popular and UNSW-NB15 datasets	114
3.18 Comparisons of KDD99 and UNSW-NB15 datasets	115
3.19 Distributions in portion of UNSW-NB15 dataset	119
3.20 Features used for analyses of UNSW-NB15 dataset	132
3.21 Comparison of results obtained for KDD99 and UNSW-NB15 datasets	139
4.1 New features created from analysing application protocols of TCP/IP model	158
4.2 Proposed features of DNS and HTTP Protocols	166
4.3 Types of DNS and HTTP records	168
4.4 Example of data sample for applying SRS and ARM techniques	179
4.5 Selected sample using SRS technique	179
4.6 Example of ARM technique for selecting relevant observations	181
4.7 Comparison of performances of ARM and SRS techniques	183

4.8 Comparison of overall performances	187
4.9 Comparisons of DRs (%) and FPRs (%) using ensemble method	188
4.10 Original and UNSW-NB15 features in KDD99 dataset using ARM	191
4.11 UNSW-NB15 features using ARM	191
4.12 Performance evaluations of original and UNSW-NB15 features on KDD99 dataset	192
4.13 Performance evaluation of UNSW-NB15 dataset	193
4.14 Features selected from both datasets	195
4.15 Performance evaluation using both datasets	196
4.16 Features selected from datasets	197
4.17 Performance evaluation using both datasets	198
5.1 Examples of identifying attacks using estimated areas	225
5.2 Features selected from NSL-KDD and UNSW-NB15 datasets	238
5.3 Evaluation of overall performances of GAA-ADS technique on orig- inal features	244
5.4 Estimations of overall performances of GAA-ADS technique on prin- cipal components	246
5.5 Comparison of DRs (%) obtained by GAA-ADS technique for prin- cipal components in NSL-KDD dataset	249

5.6 Comparison of DRs (%) obtained by GAA-ADS technique for components of UNSW-NB15 dataset	251
5.7 Performance evaluation of DMM-ADS technique on original features adopted from both datasets	255
5.8 Evaluation of overall performances of DMM-ADS technique on principal components	257
5.9 Comparison of DRs (%) obtained by DMM-ADS technique for NSL-KDD dataset	260
5.10 Comparison of DRs (%) obtained by DMM-ADS technique for UNSW-NB15 dataset	261
5.11 Performance comparisons of six ADS techniques with new DEs using NSL-KDD dataset	263
A.1 UNSW-NB15 services	313
B.1 NSL-KDD features	316

List of Terms/Abbreviations

Acronyms	Description
ACCS	Australian Centre for Cyber Security
ACSC	Australian Cyber Security Centre
ADS	Anomaly Detection System
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
ARM	Association Rule Mining
BMM	Beta Mixture Model
CC	Correlation Coefficient
CIA	Confidentiality, Integrity and Availability
CP	Central Points
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DE	Decision Engine
destip	Destination Internet protocol
DMM	Dirichlet Mixture Model
DoS	Denial of Service
DR	Detection Rate
DT	Decision Tree
EM	Expectation Maximisation
FE	Feature Extraction
FNR	False Negative Rate
FPR	False Positive Rate
FR	Feature Reduction
FS	Feature Selection
GAA	Geometric Area Analysis
GMM	Gaussian Mixture Model

Acronyms	Description
GR	Gain Ratio
HIDS	Host-based Intrusion Detection System
ICA	Independent Component Analysis
IDS	Intrusion Detection System
IQR	Interquartile Range
K-S	Kolmogorov-Smirnov test
LR	Logistic Regression
MDS	Misuse-based Detection System
ML	Machine Learning
MLE	Maximum Likelihood Estimation
NADS	Network Anomaly Detection System
NB	Naive Bayes
NIDS	Newtrok-based Intrusion Detection System
PCA	Principal Component Analysis
Pcap	Packet capture
PDF	Probability Density Function
SPA	Stateful Protocol Analysis
srcip	Source Internet protocol
TAE	Trapezoidal Area Estimation
UNSW-	University of New South Wales Network Based 2015
NB15	

Chapter 1

Introduction

1.1. Overview of Cyber Security in Current Era

Currently, the use of computing and digital appliances has become widespread in all private and public sectors. Different means of communication, including computer systems, smartphones, tablet devices, wireless connectivity and other innovations, have led to a dramatic increase in humans' reliance on them to provide services and applications anywhere at any time. This clearly indicates cooperation between the virtual and physical worlds to accomplish tasks rapidly and efficiently. Cyberspace plays a very important role in contemporary societies and economies since the Internet has changed the way that people and organisations communicate and conduct business in an electronic manner. Users, companies and governments have become dependent on the Internet services as well as other devices and applications for undertaking their daily activities [1].

The use of digital devices has been exponentially increasing throughout the world. In [2], statistics reveal that approximately 1.4 billion smartphones, 150 million iPads, 38 million tablets, 255 million personal computers and 77 million Samsung electronics were sold between 2007 and 2016, all of which are connected to the Internet and emerging new communication and computing paradigms, for example, wireless networks, 3G/4G and cloud computing [1, 3]. A wireless network is any type of computer network that uses wireless data connections to link network nodes. 3G and 4G technologies, the third and fourth generations of wireless mobile telecommunications, depend on a set of standards for providing Internet services for mobile devices. Cloud computing is a form of Internet-based computing which provides shared computers, services and platforms, such as computer networks,

servers, storage, applications and services, for the handling of resources and data on demand. While these technologies provide tremendous benefits, they constantly face serious threats of cyber attacks which create various new vulnerabilities.

As each device, whether a personal computer, server, tablet or smartphone, can be hacked by a sophisticated attacking methodology, its security should begin by deterring an exploitation vector over the network to prevent malicious activity from penetrating its operating system's resources and files [4]. An attacker's stealthy attempt to find any vulnerability of a target victim performing a task on the Internet, such as downloading/uploading files, sending emails or executing financial transactions, launches malicious scripts to compromise the security perimeter of the target device. Given the increasing numbers of protocol standards and services that are often used in an insecure way, there is a significant risk of a hacker exploiting them to achieve its aim of stealing valuable information and/or financial funds, and/or corrupting device resources [5]. Therefore, the significant role of cyber security is to protect device resources against cyber adversaries.

Many cyber security criteria for countering network threats and critical systems have been designed, with common ones established for technical product assessments, for example, ISO/IEC 15408, ISO/IEC 27002, NIST SP 800-53, OCTAVE, NIST SP 800-82, ANSI/ISA 99.00.01 and NERC-CI [1, 6]. The aim of these guides and standards, including the Zachman Framework and Department of Defence Architecture Framework, is to provide theoretical frameworks that demonstrate how to achieve high system security. According to the common criteria for ICT system evaluations [6], the assets of organisations should be protected based on the efficiency and correctness of their counter-measures [1]. This means that it is vital to design intelligent systems that can identify network intrusions and prevent their harmful actions compromising a target system. To date, no framework has been able to ensure 'complete' security, i.e., efficiently identify sophisticated stealth attacks and prevent them inflicting damage on network assets.

The rest of this chapter is organised as follows. Dimensions of cyber crime are discussed in Section 1.2; Section 1.3 explains the problem formulation and research questions; Section 1.4 outlines protections against cyber crimes and the main components of intrusion detection systems; thesis contributions and structure are provided in Sections 1.5 and 1.6, respectively.

1.2. Dimensions of Cyber Crime

The threats of attackers are described as sets of malicious events that try to exploit the principles of Confidentiality, Integrity and Availability, known as the CIA triad, in computer systems [7]. As each attack has its own sophisticated vector, this creates a big challenge for detection. Any anomalous attack on networks and computer systems could lead to severe disasters which interrupt the computer security policies of the CIA triad; for instance, Denial of Service (DoS) and Distributed DoS attacks compromise computer resources by breaking the availability principle while, as malware and malicious codes hijack the execution flows of programs, violating the integrity principle [8]. According to the Australian Cyber Security Centre (ACSC) [9] and McAfee threat reports [10], the number of network threats is increasing dramatically, causing financial losses and reputational damage, the stealing of sensitive information and intellectual property, and interruptions in business environments.

Many intrusive actions result from zero-day attacks¹, a term used to define abnormal activities not previously detected by the available white-hat technology, the vectors of which are difficult to define as no prior information is known about them [11]. Attackers' exploitation vectors, targets and means of execution have changed substantially and become more complicated. It is very clear that

¹Zero-day attacks, so-called new/unknown/future/unseen attacks, which are unidentified computer-software vulnerabilities that attackers can breach to harm computer resources, data, or network systems.

black-hat hackers attempt to find new vulnerabilities for penetrating a target system/network in a dynamically low-footprint manner. They launch sophisticated exploitation vectors against a particular target to compromise it as much as possible which is called an Advanced Persistent Threat (APT). This is a recent class of threats implemented by advanced, complicated and well-resourced intrusions targeting certain information of high-profile companies and governments, often based on a long-term philosophy and involving many different steps [1, 11, 12].

The cyber crime faced by current Internet users can be classified in a broad range of malicious activities and terms. Firstly, cyber terrorism, a term describing the integration of physical terrorism with advancements in Information and Communication Technology (ICT), is the merging of terrorism and cyberspace by a cyber terrorist with the aim of launching attack events against the computer networks of a government or organisation to commit ideological, political and/or social crimes. Its concept has multiple elements, for instance, the impact, motivation and target of a hacker, with its violation of computer networks which causes critical damages to their service operations and infrastructures and lead to severe economic loss [13].

Other threats related to fraud and financial crimes, including credit card fraud, corporate espionage and fraud, and money laundering, have gained a great deal of attention. Financial fraud attacks, which aim to abuse a profit organisation's system, are becoming of increasingly serious concern [14]. Cyber extortion is a threat activity that takes place when an organisation's resources, such as website, e-mail server or network, are compromised by DoS/DDoS attacks or other types of hacking that the hacker disrupts until the organisation pays a redemption fee in return for the promise of the exploitation being stopped. According to the Federal Bureau of Investigation [15], cyber extortionists are progressively breaching company websites and networks, crippling their capability to operate, and demanding payment for the return of their services.

Finally, cyber espionage is an offensive activity intended to stealthily collect information from a computer network for intelligence purposes [16]. The hackers involved use cracking and malicious programs, including spyware and Trojan horses, to obtain private information about users, groups and governments to hack their systems. Cyber espionage has a great impact on national security and economic wealth as it damages reputations, renders companies incapable of competing in the global economy, limits business opportunities and decreases a corporate organisation's financial competitiveness. As cyber crimes are rapidly increasing using sophisticated means of exploitation, as detailed in Chapter 2, Section 2.8, they are serious incidents that impose costs for users, organisations and governments.

According to the McAfee report issued in June 2014 [17], cyber-crime incidents during 2013 affected more than 40, 20, 54, 16, 18 and 20 million people in the USA, Korea, Turkey, Germany, Australia and China, respectively, with the total cost estimated to be almost \$160 billion per year and more than 800 million individual attacks committed. Because of the cost of cyber crime, which is substantially increasing, there is a strong motivation for the development of an effective, lightweight and adaptable Anomaly Detection System (ADS) for detecting existing and new (i.e., zero-day) attacks from current large-scale networks, as described below.

1.3. Problem Formulations and Research Questions

In this PhD thesis, we try to tackle the problem of “developing an efficient framework for the identification of existing and zero-day anomalous events in a high-speed and large volume of current network traffic in which the pattern of normal data could change over time. Network flows consist of high-dimensional network data that result in reducing the performance of the Network Anomaly Detection

System (NADS). The NADS techniques developed should be able to function effectively in the presence of an intelligent adversary tailoring its intrusive activity in such a way as to actively evade/fool the detection mechanism”.

This problem is separated into the following three major sub-problems based on the components of the NADS.

- **Sub-problem 1:** generating a quality network dataset which is one of the big challenges for evaluating the credibility of NADS’ theories. It should be captured from the norm of current network environments running at high speeds with large network flows. It should also have a wide variety of authentic contemporary legitimate and malicious patterns to ensure the quality of new NADS approaches for deployment in real networks.
- **Sub-problem 2:** selecting the relevant network features and observations is also a big challenge to the norm of current network environments. The aim is to eliminate unimportant features and observations, improve the efficiency of DE approaches and establish a lightweight NADS. The automatic online selection of features and observations that can assist in discriminating between normal and abnormal observations demands a careful study.
- **Sub-problem 3:** developing an adaptive, lightweight, scalable DE approach that can distinguish between normal and malicious observations in real network environments. It relies on its capability to self-automatically adapt and effectively and efficiently identify anomalous patterns in a current high-speed large network.

Based on the above discussion and background outlined in Chapter 2, to address the above sub-problems, the following research questions are considered.

1. How can an acceptable network dataset be created for large-scale networks?
2. What are the metrics required to trust such a dataset when evaluating new NADSS?

3. Which methodology could be effective for selecting the relevant features from network packets?
4. Can a Feature Reduction (FR) methodology assist in building a lightweight NADS and to what extent is it effective?
5. How can statistical models be used to establish an adaptive, lightweight, scalable NADS?
6. To what extent can a statistics-based NADS detect zero-day attacks?
7. How can a NADS be applied in the industry rather than a Misuse-based Detection System (MDS) for efficiently detecting known and unknown attacks?

1.4. Protection against Cyber Crime

The aforementioned types of cyber crime are extremely difficult to detect and prevent based on a single security mechanism. Despite the availability of network security solutions, existing techniques, including firewall, data encryption and authentication systems that are located on the first line of defence, cannot offer complete protection, particularly against the modern philosophy of threats, for computer systems and networks. As additional lines of defence against these threats, specifically, anti-virus software and Intrusion Detection Systems (IDSs), have been considered, a complete layered defence, also known as a defence-in-depth strategy [11], is needed to combine them in order to provide a much more effective defence.

Current protection systems, such as anti-virus and firewall systems, were designed on the basis of logging attack signatures/rules and detecting attacks by firing any of these rules but, although they are effective to some extent at identifying existing malicious patterns, any new patterns or even variants of existing

activities can easily hack a victim system. An IDS has become a primary component for protecting critical computing infrastructures, with the first model of the signature-based detection, which was designed to consider particular procedures that recognise computer threats executed by attackers launching either inside or outside an organisation, proposed by Denning [18].

An IDS can be used to monitor both host- and network-based systems [19, 20]. A Host-based IDS (HIDS) monitors the traces of a host by gathering information about events which take place in a computer system [21]. It should be installed on each client to monitor it and log its operating system's behaviours to provide audit trails for recognising both anomalous and legitimate observations. In contrast, a Network-based IDS (NIDS) monitors a network's traffic to detect remote attacks occurring in its environment [22]. It has always been a powerful security solution that provides a solid line of defence against attack events before they access the resources of operating systems. The research discussed in this thesis is interested in designing a NIDS solution for large, high-speed network environments. Its architecture has to deal with the current paradigms of computing and communications that contribute to performing detection in wireless and wearable sensing nodes in relation to the current high speeds and large sizes of networks, also known as large-scale networks [23].

An IDS methodology can be classified as one of three types, namely, Misuse-based Detection System (MDS), Anomaly-based Detection System (ADS) and Stateful Protocol Analysis (SPA) [8, 24, 25]. A MDS monitors network traffic to match observed behaviours against those on a known blacklist but cannot detect new attacks. An ADS constructs a normal profile and detects any deviation from it as an attack. Although it can identify new and existing attacks, it still presents some challenges, as described below. Finally, a SPA monitors particular protocols for recognising malicious activities that attempt to exploit them but relies on vendor-developed profiles of the protocols. More details of these methodologies are provided in Chapter 2, Section 2.2.

With the goal of designing a solid line of defence against low-footprint network threats², many research studies [26–29] have focused on designing an intelligent NADS that can recognise known and future malicious network activities. A NADS has four key components, namely, a data source, data pre-processing module, Decision Engine (DE) and security responses [30]. Initially, the data source, which provides the potential network data to enable the DE to classify observations as either normal or abnormal, comprises a set of network observations, each of which has a set of features captured from network ingress traffic. Then, the data pre-processing module prepares this input data by excluding unimportant features and observations to establish a set of patterns that discriminate between normal and malicious activities, after which the DE uses a technique for classifying observations as either normal or abnormal. Ultimately, a security response is a decision taken by a program or cyber administrator(s) to stop attack actions. Details of these components are presented in Chapter 2, Section 2.6.

To apply the architecture of NADS in the industry and commercial products on large-scale networks, it still faces three major challenges [23, 26, 31, 32]. Firstly, the creation of a comprehensive profile from different possible normal patterns to prepare it for real network environments is very difficult. Secondly, the methodology for building an adaptive and lightweight DE that efficiently differentiates between legitimate and suspicious events at high speeds and in large networks is an issue. Monitoring and analysing these networks, which have large flow volumes with high transmission velocities and involve wide ranges of high dimensionality, is not easy. For modern network data sources (i.e., datasets), it is considered that, to perfectly handle big data, specific big data analysis techniques should be used [23, 33]. Big data can be defined in terms of its volume (i.e., a large amount of data), velocity (i.e., a high data-processing speed) and variety (i.e., high-dimensional, complex data) [36]. Finally, obtaining a decent-quality dataset, which should have a wide variety of modern normal and malicious observations as

²Low-footprinting network threats are strategies of stealthily abusing a particular network environment for revealing its vulnerabilities using different sophisticated methods of exploitation, such as social engineering, DoS and phishing.

well as being correctly labelled, is always a major challenge for assessing, training and validating NADSs as it is very difficult to achieve.

DE techniques have been developed based on the mechanisms of data mining and machine learning [14, 30], artificial intelligence [30, 34], knowledge-based [30, 35] and statistical [30, 36, 37] models. However, their proposed mechanisms predominantly reflect high False Positive Rates (FPRs)³ due to the difficulty of finding a solution that mitigates the aforementioned challenges. Recent research studies [26–28, 30, 31, 36, 37] have concentrated on using statistical models as the ease of simultaneously executing and determining potential characteristics of normal and suspicious network behaviours for both features and observations.

The fundamental components of NADS that are the target of this PhD research are explained in the following three subsections.

1.4.1. Network Data Sources

A network data source/dataset can be represented in a relational table as a set of observations, each of which contains a set of features captured from network packets and tagged as normal or attack based on network security events occurring at the time of connection. The aim of a network dataset is to evaluate the performances of DE approaches for efficiently detecting attack observations, with a DE approach considered acceptable if its NADS solutions present high Detection Rates (DRs)⁴ and low FPRs. Such a dataset plays a key role in training and validating DE approaches to identify their capabilities to recognise abnormal behaviours and define their potential efficacy when a NADS is deployed in a real network.

³False Positive Rate (FPR)- an evaluation criterion of learning algorithms, which is the percentage of incorrectly detected attack instances.

⁴Detection Rate (DR)- an evaluation criterion of learning techniques, which is the percentage of correctly detected attack instances.

Public network datasets have serious problems that affect their reliability for evaluating NADSs. Although the KDD99 dataset [38] and its enhanced version (NSL-KDD) [39] are the most widely used, they are outdated and their behaviours are very different from those of recent networks. Other datasets, such as CAIDA [40] and DARPA-2009 [41], are seldom used, as discussed in Chapter 2, subsection 2.6.1. Some of these datasets do not publish their ground truth tables for ensuring the correctness of the labelling process and do not have sufficient security events to evaluate modern NADS solutions. Therefore, this was the motivation for designing the new dataset UNSW-NB15⁵ which is one of the core contributions of this PhD research. The UNSW-NB15 dataset [42, 43] was generated to create a hybrid of realistic modern normal, and security and malware events. A detailed description and evaluation of this dataset are provided in Chapter 3.

1.4.2. Relevant Features and Observations for NADS

Determining relevant features, called Feature Reduction (FR) in the machine-learning field, which is defined as the process of removing the unimportant or noisy features in a data collection, comprises two procedures, Feature Selection (FS) and Feature Extraction (FE) [44]. Firstly, FS removes redundant or irrelevant features from a given dataset and then FE transforms high-dimensional space data into a lower-dimensional space, with the size of the feature space often able to be significantly decreased without the loss of a great deal of information of the original feature space.

FS methods have been used in the network intrusion detection field to eliminate unnecessary features in order to decrease the computational complexity and improve the accuracy of DE approaches, remove redundant information and assist the understanding of data. In a NADS, FR plays a significant role in effectively reducing network features in order to improve the performance of DE techniques

⁵UNSW-NB15 refers to University of New South Wales Network Based 2015, is one of the contributions in this thesis for testing performance of NIDSs

[26]. While choosing relevant features from network traffic, selecting relevant observations from network traffic, with no duplication of flows and no missing flows that might have suspicious events, is an important task for improving NADS performances.

The methods of selecting important features and observations are applied in the data pre-processing module to establish a lightweight NADS, as discussed in Chapter 2, Subsection 2.6.2. We develop a new principle, so-called ‘relevant feature and observation method’, for eliciting relevant attributes and instances from network traffic that could precisely differentiate between legitimate and suspicious activities in DE approaches for a NADS, as explained in Chapter 4.

1.4.3. Statistics-based NADS

Statistically speaking, an anomaly is a rare event which takes place between normal data points. In NADSs, statistical models are fitted using normal network data and then a statistical inference test using either a pre-defined threshold or probability condition applied to consider observations which deviate from this condition as attacks [30, 45].

Statistics-based techniques are classified as parametric and non-parametric, both of which have been widely used to develop statistical models for NADS. Parametric approaches assume that network data fit a certain distribution while non-parametric mechanisms do not make any assumptions about the statistical characteristics of the given data [26, 45]. Since statistical approaches can accurately determine and analyse features and observations of network data in order define a clear difference between normal and suspicious network activities [30, 45–47], we can use them for developing new DE techniques, as detailed in Chapter 5.

1.5. Thesis Contributions

This thesis contributes to the field of NADS by offering promising solutions to the above sub-problems through the following key aspects based on the main components of a NADS.

- **Creating a contemporary network dataset called the UNSW-NB15 dataset with deep statistical analysis and evaluation (Chapter 3)**
- the IXIA PerfectStorm tool is used to extract a hybrid of modern normal and anomalous activities from network traffic. Anomalous activities were simulated from the CVE site, which contains up-to-date malware behaviours, and the tcpdump tool used to capture approximately 100 GB of raw network traffic in the pcap format, with each pcap file containing about 1 GB for easy determination and analysis.
- **Designing a new aggregator module (chapter 4)** - based on the theory of flow-level analysis for enhancing the performance of NADS via extracting only significant observations without duplication or missing values in those observations with the capability of handling large-scale network data.
- **Developing and applying feature reduction and selection approaches for selecting relevant features (Chapter 4)** - three techniques of a new Association Rule Mining-Central Points (ARM-CP), Principal Component Analysis (PCA) and Independent Component Analysis (ICA), are used for reducing and selecting important network features.
- **Suggesting new statistical features of the DNS and HTTP protocols and ensemble-learning framework for detecting malicious activities that face these protocols (Chapter 4)** - these protocols contain some information about the user and network activities and are two fundamental protocols for Internet and network applications. We develop

a NIDS for recognising malicious activities, due to their importance in network systems. An ensemble learning method, including Decision Tree, Naïve Bayes and Artificial Neural Network techniques with the model of Adaboost to fairly distribute network data between them, is developed.

- **Developing two novel DE techniques using the statistical mixture models for recognising known and zero-day attacks (Chapter 5) -** These techniques are based on the methodology of anomaly for building a normal profile from parameters of statistical mixture models in the training phase, as these models can fit the majority of all possible normal data points accurately. The same parameters used in the training phase are applied to the testing data, and then new outlier decision-making methods are used for recognising anomalous observations, which are out of the normal profile.
- **Designing scalable NADS frameworks for the above DE techniques (Chapter 5) -** these frameworks are designed for developing a lightweight, adaptive and scalable ADS that can effectively process large-scale networks. It comprises the three modules of data sniffing and storing, data pre-processing and DE techniques. The first involves a feature set created from network traffic, the second analyses and filters network data while the third includes the proposed DE techniques for successfully detecting known and unknown malicious activities.

1.6. Thesis Structure

The rest of this thesis is organised below and its overview is presented in Figure 1.1.

Chapter 2 provides the technical background to IDS methodologies and their components, and the literature related to our research detailed in this thesis.

Figure 1.1: Thesis overview

The following NADS components and their challenges are investigated. Firstly, publicly available benchmark datasets are reviewed as data sources for assessing the performances of NIDSs and then data pre-processing methods and their roles in establishing a lightweight, adaptive NADS are explained. Finally, existing ADSs and the problems they pose for developing a lightweight, adaptive and scalable NADS, and the statistical methods related to our new DE schemes are discussed.

Chapter 3 discusses modern network dataset environments. Firstly, and most importantly, the capability of the DARPA-2009 dataset is analysed in depth to evaluate new DE approaches but some challenges regarding its correct labelling are encountered. In addition, as the patterns in this dataset are very different from the sophisticated ones of normal and malicious behaviours in real network environments, they are not sufficiently complex to be identified using existing ML techniques. This was the motivation for generating the new UNSW-NB15 dataset that resembles realistic modern network data. Its capability to generate a feature set from raw packets is examined and its statistical and complexity evaluation presented to demonstrate to what extent it can be used to efficiently assess new DE approaches.

Chapter 4 details the new theory of selecting relevant features and observations from raw network packets. Firstly, a new aggregator module is designed for collecting relevant observations and feature selection methods based on the association rule mining with central points, principal component analysis, as well as independent component analysis for adopting the important features that assist in building a lightweight NADS. Then, the new methodology involving the significant features from different network segments and application layers with depth

analysis for the HTTP and DNS protocols is discussed. The proposed features of the protocols are obtained by computing their statistical characterises and then passing them to the DE approach for recognising malicious instances.

Chapter 5 explains two new frameworks, for the first time in this field, designed for building the methodology of adaptive, lightweight and scalable NADS with two novel DE techniques using statistical models of Beta- and Dirichlet- mixture models that precisely distinguish between legitimate and malicious network vectors. The two frameworks have three main modules, namely data sniffing and storing, data pre-processing and DE. The first two modules are the same in both frameworks for eliciting and storing network data, and analysing and filtering those data, respectively, whereas the DE techniques are different in the two frameworks for successfully discovering existing and zero-day attacks.

Concluding remarks are provided in Chapter 6; protocols of the UNSW-NB15 datasets are presented in Appendix A; and the feature description of the NSL-KDD dataset used for the evaluation of the mechanisms developed in the thesis is provided in Appendix B.

Chapter 2

Background and Related Work

2.1. Objectives

This chapter discusses the background to and related work on, establishing an effective Intrusion Detection System (IDS), with the purpose of our research to build an adaptive and online Network Anomaly Detection System (NADS) that discriminates between legitimate and suspicious network traffic. A typical NADS architecture consists of four components: a data source, data pre-processing method, Decision Engine (DE) technique and security response. In this chapter, we investigate mainly data sources, data pre-processing methods and DE techniques; and their challenges. Firstly, public benchmark datasets are reviewed as data sources for evaluating the performances of IDSs and the issues they raise. Secondly, methods for data pre-processing and their roles in establishing an online NADS are discussed. Finally, existing anomaly detection (i.e., DE) approaches and the problems they pose for developing an adaptive and online NADS, and the statistical methods related to our new DE schemes are described¹.

2.2. Intrusion Detection System (IDS)

An IDS is important in the cyber security field for achieving a solid line of defence against cyber adversaries. The digital world has become the main complement of

¹

- Parts of this review has been submitted in:
Moustafa, N., Creech, G. and J. Slay."A Comprehensive survey: Components of Intrusion Detection systems", 2017, "under review"

the physical world because of the ubiquitous use of computer networks and the prevalence of applications that easily execute users' tasks in a short time and at low cost. Awareness of the need for information technology means of securing network resources against threats of attackers which has been steadily increasing. A system is considered secure if the three principles of computer security, Confidentiality, Integrity and Availability (CIA), are successfully achieved [48]. Each attack type has its own sophisticated philosophy that poses serious threats to computer networking and violates these principles. When an attacker gathers significant information about a system, it breaches the system's confidentiality and, when it interrupts legitimate operations, compromises its availability and integrity. For example, Denial of Service (DoS) attack corrupts computer resources, which breaks the availability principle, while malware code hijacks the execution flow of an application which violates the integrity principle [8].

An IDS is a mechanism for monitoring and analysing the activities which take place in a computer system or network to detect possible threats by measuring their violations of computer security principles of CIA [48–50]. The classical architecture of an IDS comprises four components [22] (Figure 2.1), namely, a packet decoder, pre-processor, DE sensor and defence response/alert module, as described below.

- The packet decoder acquires portions of raw network traffic using audit data collection tools, such as Tcpdump and Libpcap, which transfer each portion into the pre-processor for handling.
- The pre-processor captures a set of features from the raw audit data which is used later in the DE sensor. A typical pre-processor widely used in network-based IDSs is the TCP handler which analyses TCP protocols in session flows; for example, Netflow, Bro-IDS and Argus tools which examine different protocols, such as HTTP, DNS, SMTP and UDP.

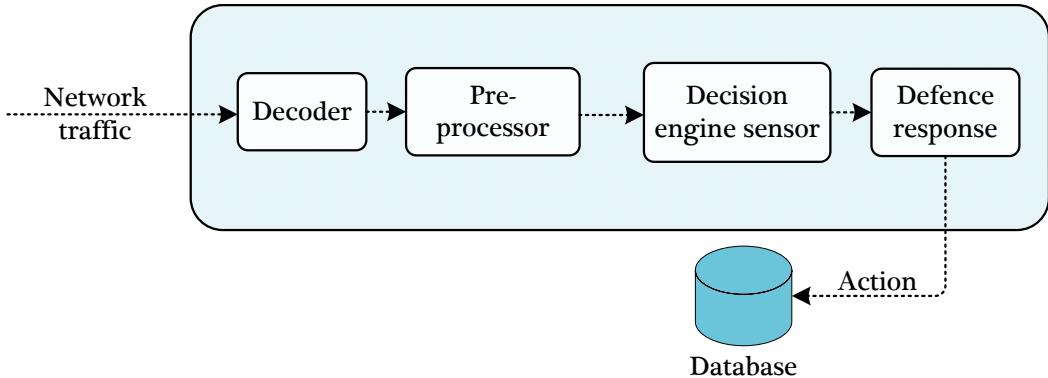


Figure 2.1: Architecture of classical IDS

- The DE sensor receives the proposed features from the pre-processor and builds a model that distinguishes attack observations from normal ones. If an attack is detected, it requests the defence response to raise an alert.
- The defence response is the process of raising alerts requested by the DE which are logged in a database and then sent to the security administrator to take action.

An IDS can be active or passive depending on the actions taken; it is passive when an anomalous behaviour is identified and an alert raised but no further action taken while an active IDS raises an alert and attempts to neutralise the anomalous behaviour by executing a predefined script action and is also called an Intrusion Prevention System (IPS).

2.2.1. Intrusion Detection Properties

An IDS can be categorised in three ways based on its properties: monitored environment; detection approaches; and deployment architecture (see [22, 51–53]), as discussed in the following.

2.2.2. Monitored environment

An IDS can be used to monitor host- and network-based environments. A host-based IDS (HIDS) monitors the events of a host by collecting information about activities which happen in a computer system [54]. A sensor should be installed in such a system to monitor hosts and log the operating system's activities to provide an audit trail [20, 54]. A HIDS can identify the improper use of an organisation's internal equipment, for instance, an employee's computer launching attacks. It is also used when a network is encrypted by distributing its load to monitor its hosts [54]. However, each operating system has different audit trials; for example, those of Linux are API calls while those of Windows are DLLs. Therefore, it is difficult to build a HIDS that can monitor different operating systems to be compatible with their platforms. As they use their audit trails which are extremely expensive which require significant amounts of storage space and increase the host server's load [20, 55]. Another drawback is that a HIDS can be exposed as soon as its host server is compromised by an attacker. This means that any existing vulnerabilities of a host server threaten the integrity of a host-based sensor [54, 55]. If an attacker breaches one of these weaknesses, this could lead to a vulnerability which is hard to detect. Also, a DoS attack can expose a host and impede the functions of a HIDS by filling the audit file system [22, 55].

Another environment for monitoring is a Network-based IDS (NIDS) which monitors network traffic to identify remote attacks that happen over a network connection [22]. It has always been an essential security solution as it provides a solid line of defence against a malicious activity before it accesses the resources of a host and records itself in the audit trials of an operating system. Although a HIDS can detect intrusions into hosts, this naturally occurs after a host's computer resources, such as its files and services, are accessed. It is clear that the best security solution is to deter known and zero-day attacks before they exploit hosts, i.e., over networks, to achieve the wisdom of 'prevention is better than cure'. As, even if a HIDS detects an attack, there is no guarantee that computer resources are

not exposed, the design of an intelligent NIDS considers a means of deterring such attacks although it is still challenging to apply this in a reliable way. Therefore, a combination of a HIDS and NIDS has been implemented to establish a hybrid IDS which can monitor decrypted network traffic and host activities [51].

A NIDS has several advantages [56]. Firstly, its network monitoring does not degrade the performance(s) of other software running on the network as it only needs to read some information from each packet coming across its network segment whereas a HIDS inspects the resources of the operating system that is running [54]. Secondly, it is quite portable as it monitors network traffic over only a certain network segment regardless of the destination's type of operating system. Thirdly, its network-based sensors can be installed on a network and its data are easily collected which is beneficial in some situations; for instance, following network topology or system resource changes that it can be moved and used as needed.

Nevertheless, one of the major drawbacks of a NIDS is scalability. Since high-speed connected networks (e.g., 100 Mbps, 10 Gbps and more) have become the norm, attackers can define them and exploit their weaknesses using low-footprint attacks, such as stealth and spy attacks [11, 57]. The encryption of a network represents a great challenge to the design of a NIDS for collecting cipher data while the tunnelling of IPsec protocols establishes a new risk by allowing the encapsulation of IPv6 traffic in IPv4 data streaming to route over non-compliant appliances which involves serious threats of IPv4 DoS and DDoS attacks [58]. A modern NIDS can deal with end-to-end encryption by extracting general and statistical information about packets, for instance, their sizes, lengths and inter-arrival times, as flow-based features [54, 59] but packet payloads, namely, packet-based features, always obfuscate. These packets have been analysed using Deep Packet Inspection (DPI) paradigms, with their classifications based on their behaviours or a hybrid of learning theories and statistical approaches [59].

As this PhD thesis concentrates on the development of a NIDS, a HIDS is not discussed in any further detail and the focus in this chapter is on NIDS technologies and approaches.

2.2.3. Detection methods

Intrusion detection methodologies are classified in three major categories: Misuse-based (MDS); Anomaly-based (ADS); and Stateful Protocol Analysis (SPA) [8, 25, 60]. A MDS monitors network traffic to match observed behaviours with attack signatures logged in a database. Although it provides higher detection rates and lower false positive rates (FPRs) to existing attacks than other categories, it cannot detect zero-day or even variants of known ones. This is a significant issue in terms of the computer security required to defend against those attacks. Moreover, a huge effort is necessary to frequently update its database, which is a set of rules for each malicious activity, generated by network security expertise [31].

An ADS creates a normal profile and detects any deviation from it as an attack. It can identify known and zero-day attacks as well as require less effort to construct its profile than a MDS. However, it still faces two challenges for application in the computer industry or even commercial products: (1) determining a way of creating a comprehensive profile from diverse legitimate behaviours; and (2) designing an architecture for establishing an adaptive detection method which efficiently distinguishes between normal and suspicious network traffic. These challenges are the main motivation for our research into designing an online and adaptive ADS.

Finally, a SPA examines and traces protocol states, specifically a pair of request-response protocols, such as a HTTP protocol. Although a SPA is roughly similar to an ADS, it relies on vendor-developed profiles of certain protocols and requires information of the relevant network's protocol standard from international standard organisations [25]. As a SPA consumes many computer resources to

inspect protocol states and is incompatible with different dedicated operating systems, an ADS is a better defence solution if its DE approach is properly designed [8, 26, 28, 31].

2.2.4. Deployment architecture

An IDS's deployment architecture is either distributed or centralised. The former is a compound system comprising multiple intrusion detection subsystems installed at different sites and connected to exchange relevant information. This helps in detecting malicious patterns which can identify corresponding attacks from multiple locations in a particular time. Conversely, a centralised IDS refers to a non-compound system which is deployed at only one site, with its architecture dependent on the organisation's size and sensitivity of the data which should be considered when designing a deployment [51].

This PhD thesis focuses on statistical approaches and how to use them to design a statistical ADS for implementation in an online and adaptive way. The rest of this chapter is organised as follows: the characteristics of network anomalies are provided in Section 2.3; Section 2.4 outlines the evaluation metrics as well as efficiency and reliability measures used for IDSs; Section 2.5 describes the particular challenges facing modern NADS; the components of NADS are presented in Section 2.6; DE approaches are discussed in Section 2.7, with details of the statistically based NADS used in the proposed DE described in subsection 2.7.5; and, finally, contemporary network threats are presented in Section 2.8 and concluding remarks in Section 2.9.

2.3. Characteristics of Network Anomalies

Anomalies are known as patterns in network traffic which behave differently from legitimate ones. Their characteristics are classified as point, contextual or collective according to the output from the detection method used [61–63]. A point anomaly occurs when a certain data observation deviates from the normal profile and, in statistical methods, is referred to as an outlier. Contextual anomalies occur when data patterns are anomalous in a particular context and appear as related behaviours which are always different from the majority of normal activities. They are also called conditional anomalies that require a notion of context to identify them from network data. Collective anomalies happen when a group of similar data instances acts anomalously compared with the entire data of a normal network.

The output from anomaly detection is often based on a baseline/threshold which is a condition that discriminates between normal and attack instances [63]. Determining this threshold is one of the significant challenges faced when designing a NADS due to the overlapping patterns of normal and attack activities. The types of output from anomaly detection can be a score or binary which affects the selection of a correct threshold. A score-based output is a numeric value of either probabilities or real numbers for each data record while a binary/label-based output is a certain value which tags each record as normal or attack; for example, the labels of the KDD99 [38] and UNSW-NB15 [43] datasets are ‘0’ and ‘1’ for normal and attack records, respectively.

2.4. Evaluation Metrics for IDSs

The performance of an IDS depends on conducting a confusion matrix (Table 2.1) for a validated IDS model which is built for any classification problem, with its size dependent on the number of classes included in the particular dataset [26].

Table 2.1: Confusion matrix for binary classification problems

		Actual	
		Negative	Positive
Predicted	Negative	TN	FP
	Positive	FN	TP

Its purpose is to compare actual and predicted labels, and it is acknowledged that an intrusion detection problem, which contains two classes, normal and attack, is defined by a 2-by-2 confusion matrix for an evaluation.

The terms TP (true positive) and TN (true negative) denote correctly predicted conditions and FP (false positive) and FN (false negative) misclassified ones. TPs and TNs refer to correctly classified attack and normal records, respectively and, conversely, FPs and FNs refer to misclassified normal and attack records, respectively [26, 57]. These four terms are used to generate the following IDS evaluation measures.

- **Accuracy** is a metric that estimates the overall percentages of detection and false alarms an IDS model produces, which reflects the overall success rate of any IDS, and is computed as

$$\text{Accuracy} = (TN + TP) / (TP + FP + TN + FN) \quad (2.1)$$

- **The Detection Rate (DR)**, also called the true positive rate (TPR) or sensitivity, is the proportion of correctly classified malicious instances of the total number of malicious instances and is computed as

$$DR = TP / (FN + TP) \quad (2.2)$$

- **The True Negative Rate (TNR)**, also called the specificity, is the proportion of correctly classified normal instances of the total number of normal instances and is computed as

$$TNR = TN / (TN + FP) \quad (2.3)$$

- **The False Positive Rate (FPR)** is the proportion of normal instances of the total number of normal instances misclassified as attacks and is computed as

$$FPR = FP / (FP + TN) \quad (2.4)$$

- **The False Negative Rate (FNR)** is the proportion of misclassified attack instances of the total number of attack instances, given as

$$FNR = FN / (FN + TP) \quad (2.5)$$

IDS approaches are evaluated using the TPR-FPR or specificity–sensitivity measure to estimate to what extent they are accurate in detecting malicious activities [26]. A perfect IDS approach could have a 100% DR while a 0% FPR reflects that all attack instances are detected without any misclassification. However, this is very difficult and demonstrates the optimal performance to be achieved in a real environment.

Another measure commonly used is the Receiver Operating Characteristics (ROC) curve. It was created from the signal processing theory and then extended to other domains, such as medical diagnosis, bioinformatics, data mining

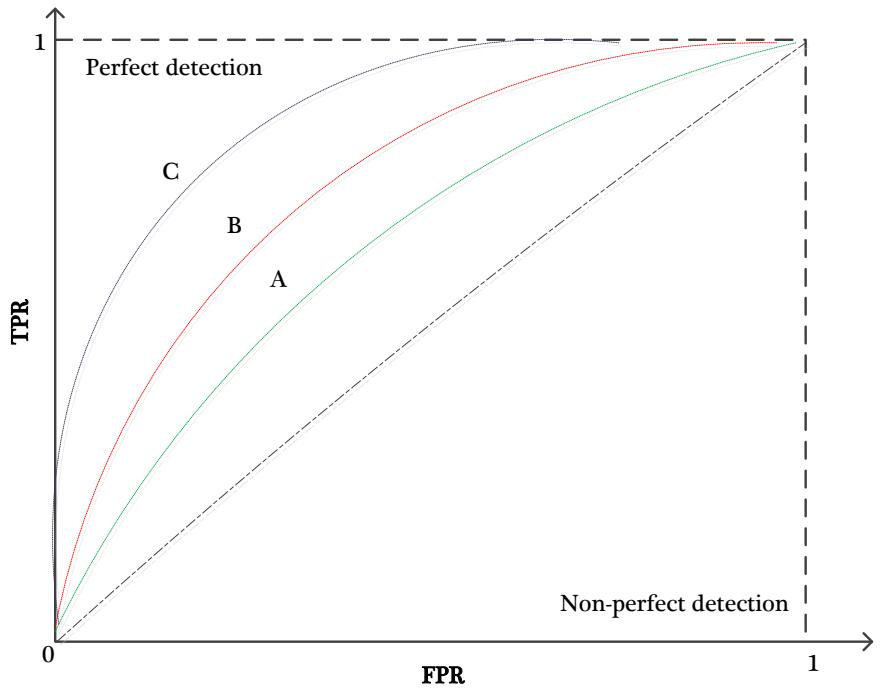


Figure 2.2: ROC curves - A, B and C show levels of detection

and machine learning as well as artificial intelligence. In an intrusion detection methodology, it represents the relationship between the TPR and FPR of a DE approach [57], as shown in Figure 2.2.

Ultimately, the F-measure criterion is a preferable measure of evaluating IDS approaches. It is a harmonic mean of precision and recall [64], that is, a statistical function for estimating the accuracy of a system by computing its precision and recall given as

$$F - \text{measure} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (2.6)$$

where precision is the fraction of the predicted positive values which are actually positive and recall the actual number of positives correctly detected, as given in equations. (2.7) and (2.8), respectively.

$$Precision = TP / (TP + FP) \quad (2.7)$$

$$Recall = TP / (TP + FN) \quad (2.8)$$

Similar to the TPR-FPR measure, when the precision and recall of an IDS approach achieve 100%, as the F-measure is the maximum, a 0% FAR and 100% DR are produced [64, 65]. This thesis uses these evaluation metrics to assess the performances and effectiveness of the proposed IDS approaches and evaluate the complexity of the UNSW-NB15 dataset with other widely applied functions.

The efficiency and reliability of IDSs can be assessed by the following measures ([26, 66]).

- **Accuracy** - indicates the overall success of correctly detecting legitimate and anomalous records [22, 26], as previously discussed.
- **Performance** – is the capability of a system to handle network traffic that deals with a high speed and low packet loss while running in real time. As, in a real network environment, the packets are different sizes, the efficacy of a NIDS relies on its capability to process a packet of any size. Moreover, CPU and memory usage could also be considered criteria for assessing a NIDS performance [22, 26]. The performance of any NIDS depends on its configuration in a network and the capacity of the network it monitors.

- **Completeness** - is the capability to detect all the vulnerabilities and attacks that attempt to breach a network [22]. This measure is more difficult to appraise than the others as it is impossible to have knowledge about malicious activities which could penetrate a user's privileges.
- **Timeliness** - indicates the capability of a NIDS to perform its inspection as quickly as possible to enable the security administrator or response engine to take action before a great deal of loss occurs [1, 66]. There is a continual delay between the detection of an attack and the response of the system which it is preferable to reduce as much as possible to prevent attack threats.
- **Profile update** – when new vulnerabilities or abuses are identified, blacklists or profiles have to be updated for new detection [1]. However, this task is a big challenge in current high-speed network traffic.
- **Stability** - a NIDS should operate consistently in different network infrastructures and steadily log identical events to allow its triggers to be easily configured [26].
- **Interoperability** - an effective NIDS is assumed to be capable of associating information from numerous sources, such as system and firewall logs, HIDSs, NIDSs and any other available source of information [67].

2.5. Challenges of NADS

Although a MDS cannot recognise zero-day attacks or even variants of existing ones, it is still a common defence solution used in commercial products. On the contrary, a NADS can detect serious threats but has often been faced with potential challenges for its effective design. These challenges, which could be explored from an anomaly-based methodology, which is the construction of a purely legitimate profile with any variation from it declared an anomaly [1, 8, 26, 36, 47, 63], are as follows.

- Constructing a comprehensive profile that involves all possible legitimate behaviours is very complex to accomplish. As the boundary between normal and abnormal behaviours is usually not accurate. There are errors of the FPRs and FNRs which occur when a normal behaviour falls in an attack region and a malicious one in a normal region, respectively. Although both errors are dangerous for network data, the FPR is often used to evaluate the performance of DE approaches because there are no alerts of attacks in real network systems.
- When designing the architecture of an adaptive and scalable NADS, it is very difficult to distinguish attacks from the normal profile as sophisticated malicious activities, such as stealth and spy attacks, can adapt to be almost the same as normal patterns. Therefore, methods for detecting such attacks have to analyse and inspect the potential characteristics of the network traffic.
- Real-time detection is also very challenging for several reasons. Firstly, the features created for network traffic may contain a set of noisy or irrelevant ones. Secondly, the lightweight of detection methods need to be carefully adopted, with respect to the above problems. These reasons increase the processing time and false alarm rate if not properly addressed.
- A decent-quality dataset is usually a major concern for evaluating, learning and validating NADS models as it should have a broad range of contemporary normal and malicious behaviours as well as being correctly labelled which is difficult.

For the above challenges, most existing ADSs address particular formulations of them. The factors involved in designing an effective NADS framework are encompassed by solving the above problems and executing the components of NADS to build an online and adaptive NADS.

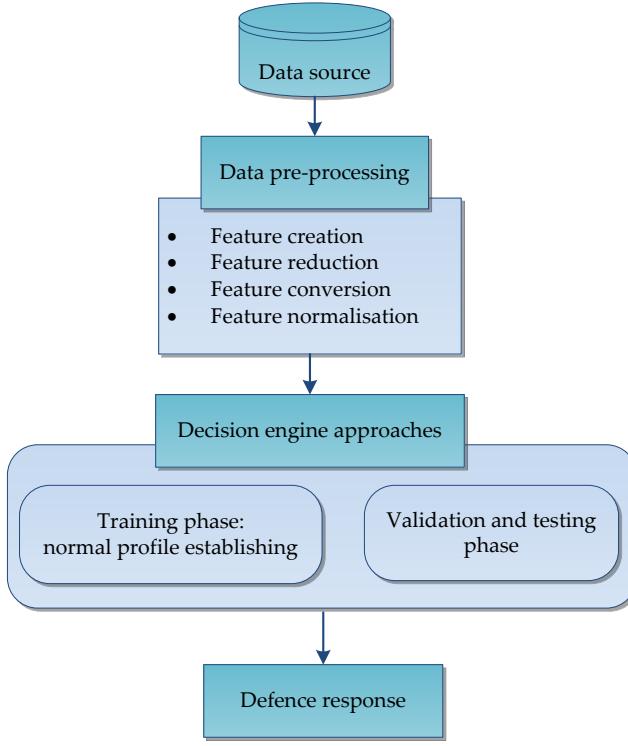


Figure 2.3: Components of NADS

2.6. Components of NADS

As depicted in Figure 2.3, a typical NADS consists of four components: a data source; data pre-processing module, DE method and security responses [30].

A fundamental component of a NADS is the data source which provides the potential information to enable the DE to classify records as either normal or attack. It includes a set of network records, each of which has a set of features generated from network ingress traffic, with existing secure servers used to capture network connections for a particular time interval. Monitoring and analysing the traffic at the destination network decrease the overhead of identifying attack records by focusing on only relevant traffic [36]. Next, the data pre-processing

prepares the input data by excluding unnecessary features to create a set of patterns that distinguish between legitimate and suspicious activities and then the DE method uses a technique to classify records as either normal or abnormal. Finally, a security response is a decision taken by the software or cyber administrators to prevent attack actions. Details of these components are provided in the following subsections.

2.6.1. Data source

In 1980, pioneering research in the area of NIDS stated that, audit data of computer systems should be carefully analysed to understand the types of threats and recognise attacks [68]. Later, the research community collectively realised that its data source is a major component of any NIDS for evaluating the performances of DE methods due to the difficulty of labelling legitimate and attack activities in live network traffic [69, 70]. Network data sources have been collected in an offline dataset which comprises a wide variety of normal and malicious records.

As previously discussed, datasets play a significant role in testing and validating NIDS methods. A decent-quality one can identify the capabilities of DE approaches to detect malicious behaviours and define their potential effectiveness when NIDSs are deployed in real network environments. In this subsection, publicly available benchmark network datasets and the reasons for designing the UNSW-NB15 dataset, which is part of this PhD thesis, are discussed (see [42, 71]).

With the high speeds and large sizes of current network environments, elicited network data has the characteristic of the big data principle, as the case of the UNSW-NB15 dataset explained in Chapter 3 - Section 3.4. Big data is typically defined in the terms of Volume, Velocity and Variety proposed by Doug Laney in 2001 [72]. Volume indicates the amount of data that is processed, with traditional computing mechanisms incapable of handling the current large sizes of network

data. Velocity refers to the speed of processing data and, although real network environments run very quickly to provide services and programs at any time and anywhere between different networks, traditional mechanisms cannot process such data very rapidly. Variety indicates the complexity of the data and to what extent they are of diverse types and dimensions, with those with high dimensions collected from different sources or having multiple different structures, treated as complex [33].

Two additional terms, Veracity and Value, determine whether a set of data is considered big data. Veracity refers to the correctness of the data, including the problems of their quality, such as noise and missing values, while Value relates to the importance and sensitivity of particular data, such as bank and government ones. If the data are not important, a big data analysis is not appropriate [73]. As traditional techniques generally cannot process the big data in real-world problems, such as those in the cyber security field, it is vital to develop intelligent, mainstream computing mechanisms capable of efficiently and effectively handling a network's big data.

Handling big data has become a significant issue for NIDS, with its challenges discussed in [32, 33, 74]. From these studies, a Hadoop-based DDoS attack detection technique for processing vast network traces to mitigate network adversaries was suggested, and a new traffic monitoring system developed to perform a Net-Flow analysis of internet traffic. A MapReduce technique was used to generate a set of features from the libpcap as input to the detection method for recognising malicious observations [75] while Kamaldeep et al. [76] proposed a quasi-real-time IDS based on open-source tools, such as Hadoop, Hive and Mahout. The big data required for evaluation are sniffed from realistic network traffic and the CAIDA dataset in order to determine intrusive traces. The benchmark network datasets are discussed as in the following subsections.

A. KDD99 dataset

The IST group at the Lincoln Laboratories in the MIT University performed a simulation involving both normal and abnormal traffic in the military network of the U.S. Air Force LAN environment to generate the DARPA 98 dataset using nine weeks of raw tcpdump files. The training set's size was approximately four GBs and comprised binary tcpdump files captured over seven weeks in almost five million connection records while the testing set consisted of two million connection records captured over the other two weeks [38].

In 1999, the DARPA 98 dataset was analysed by the BRO-IDS to capture 41 features and the class label for each connection record and called the KDD99 dataset which has been the most widely used for evaluating NIDS methods. In it, the features are divided into three groups: firstly, the intrinsic features acquired from the headers of the network packets; secondly, the content features captured from the payloads of the network packets; and, thirdly, the traffic features obtained from information about previous connections. This dataset contains five classes, one normal and the four attack ones (DoS, Probe, U2R and R2L) listed in Table 2.2 [71], with the training and testing sets comprising 22 and 15 attack types, respectively. Table 2.3 presents two samples of the KDD99 dataset distributions, namely the entirely corrected and 10% corrected training ones, used to evaluate DE approaches.

The KDD99 dataset has three main drawbacks which can affect the fidelity of evaluating the performances and effectiveness of NIDS mechanisms. Firstly, every attack data packet has a time to live (TTL) of 126 or 253 while those of most packets of current real traffic are 127 or 254 [77]. Secondly, the data distribution of the testing set is different from that of the training set because new attack records have been added to the testing set [78, 79]. This means that, as classification

Table 2.2: Four attack classes in KDD99 dataset

Denial of Service (DoS)	Attackers, such as SYN flood, Smurf and teardrop, endeavour to obstruct legitimate users from using a service provided by a system
Remote to Local (R2L)	Attackers attempt to get right of access to a machine without authorization, for example, the password-guessing attack
User to Root (U2R)	Attackers try to access the local super-user (root) privileges from the same domain user, for instance, buffer overflow attacks
Probe	Attackers, such as port scanning and ping-sweep, attempt to obtain information about the target client

Table 2.3: KDD99 dataset distributions of attack and normal instances

Dataset	DoS	U2R	R2L	Probe	Normal	Total
Corrected KDD	229853	70	16347	4166	60593	311029
10% corrected KDD	391458	52	1126	4107	97278	494021

approaches have a bias towards some observations rather than balancing legitimate and attack ones, a poor NIDS evaluation often results. Thirdly, this dataset's network traffic is very different from current network traffic in terms of network speed, network bandwidth, upgraded protocol types, attack behaviours and even normal behaviours.

B. NSL-KDD dataset

The NSL-KDD dataset [39] proposed by Tavallaei et al. [80], is an enhanced version of the KDD99 dataset and, similar to it, consists of five classes, one normal and the four attack ones DoS, Probe, U2R and R2L, with each record having 41 features and a class label. Its data has two sets: training ('KDDTrain+_ FULL' and 'KDDTrain+_20 %'); and testing ('KDDTest+_FULL' and 'KDDTest-21_new attacks') [39].

This dataset addresses some of the problems of the KDD99 dataset. First and foremost, as it does not contain superfluous and duplicated records in either the

Table 2.4: Distributions of attacks in NSL-KDD dataset

Parts	DoS	U2R	R2L	Probe	Normal	Total
KDDTrain	45927	52	995	11656	67343	125973
KDDTest	7458	67	2887	2422	9710	22544

training or testing set, NIDS methods could not be biased towards more repeated observations. Secondly, as the numbers of records in the training and testing sets are adopted from different portions of the original KDD99 dataset without any duplication, they are reliable for evaluating NIDS approaches. Moreover, they are rational and used to run experiments on the full dataset without the need to randomly select a small proportion (Table 2.4). Nevertheless, the NSL-KDD dataset cannot represent contemporary network traffic as its legitimate and attack behaviours are extremely different from those of current network traffic.

C. CAIDA datasets

The CAIDA datasets [40] are collections of different data types for analysing security-related events to validate attack detection approaches, but are limited to particular types of attacks, such as DDoS ones, with their traces the anonymised backbones of the packet headers without their payloads. The most common CAIDA dataset is the CAIDA DDoS 2007 anomaly one which includes an hour of anonymised network traffic for DDoS attacks.

A simulation was designed to consume a significant amount of network resources while linking to the internet, with the network traffic involving only attack activities to the victim, the responses of which were stored every 5 minutes in a separate pcap file for easy inspections of attack behaviours. It attempted to ensure that the pcap files were free of legitimate traffic to reliably evaluate detection methods [71]. However, these datasets did not indicate a clear ground truth about the attack activities involved and, moreover, their pcap files were not inspected

precisely to elicit features in order to discriminate attack activities from normal ones.

D. DEFCON dataset

The DEFCON dataset, which comprises a list of packets, is freely available on the internet [81]. Although most of the files are Full Packet Capture (FPC) ones, some have truncated frames. They were captured during a hacking competition called Capture-the-Flag (CFT) in which competing teams were divided into two groups, attackers and defenders. It contains only malicious activities with no legitimate traffic which is different from real network traffic. This dataset is only effective for assessing alert correlation approaches and poor for evaluating NADS ones due to its limitations of losing frames and lacking legitimate network traffic.

E. ISCX dataset

The ISCX dataset [82, 83] was designed based on the concept of profiles which contains descriptions of attacks and distribution models for a network architecture. Its records were captured from a real-time simulation conducted over seven days of normal network traffic and synthetic attack simulators. Their traces were analysed to create profiles from the HTTP, SSH, SMTP, IMAP, POP3 and FTP protocols which were used to generate the dataset’s files in a testbed environment. Several multi-stage attack scenarios were included to help in evaluating NIDS methods. However, the dataset did not provide the ground truth about attacks to reflect the credibility of labelling and, secondly, the profile concept used to build the dataset could be impossible to apply in a real complex network because of the difficulty of analysing and logging. Also, the profiles were established for protocol types to evaluate Stateful NIDSs but would increase the alarm rate for NADS.

F. Kyoto dataset

The Kyoto dataset, which was developed at the Kyoto University [84], is a set of network traffic collected from honeypot systems, with the BRO-IDS tool used to extract 24 features from those in the KDD99 dataset. These features were categorised into 14 conventional and 10 additional ones that reflected network characteristics [71]. The main drawbacks of the Kyoto dataset were that it lacked measures for labelling and describing attack behaviours or even variants of legitimate ones.

G. DARPA 2009 dataset

The DARPA 2009 dataset [41] was synthetically designed to emulate the traffic between 16 sub-networks and the internet with data collected over 10 days, from 3rd to 12th November 2009. It contains synthetic HTTP, SMTP and DNS background data traffic, and has a set of attack types such as DoS and DDoS. It consists of 7000 pcap files with almost 6.5 TB, with each file including approximately a one- or two-minute timing window.

We analysed the first 30 GBs using the TCP trace tool to evaluate the performances of some existing classification methods [85]. Although these methods provided high detection rates, the ground truth of the attack events had many faults as the same information was used to label records as normal or attacks which led to attack detection being unreliable.

From the above discussion, as the benchmark datasets have serious problems of either containing out-dated network data or ground truth faults for labelling, they are unreliable for evaluating real anomaly detection methods. This was the motivation for designing the UNSW-NB15 dataset to evaluate our proposed statistical NADS approaches, as detailed in Chapter 3.

2.6.2. Data pre-processing

Data pre-processing is a significant step in learning theories because, like data-gathering measures which are often loosely controlled and result in irrelevant or duplicated data values, network data extracted from network traffic also include these data. It filters network data by removing redundant, noisy or irrelevant information which leads to improving the performance of DE approaches for detecting attack behaviours. Data pre-processing for network data involves the creation, reduction, conversion and normalisation of feature, as described in the following.

A. Feature creation

Network features are captured from raw network packets using different tools, such as Argue, Bro-IDS, Netflow, Tcptrace and Netmate. It is impossible to operate a NIDS on raw packets without extracting a set of features, often called basic features, as in the KDD99 and UNSW-NB15 datasets. Moreover, additional features are established using both transactional flow identifiers (i.e., source and destination IP addresses, source and destination ports, and the protocol type) and transactional connection times (e.g., 10 or 100 connections per second) to define the potential characteristics of network behaviour. These features are significant for identifying attackers who scan victims in a capricious way, such as one scan per minute or per hour; for example, in the KDD99 and UNSW-NB15 datasets, the *is_sm_flw* feature could identify land or teardrop attacks [86], as further discussed in Chapter 3.

B. Feature reduction

Feature reduction is the process of removing irrelevant, redundant and/or noisy features, and can be divided into feature selection and feature extraction. The former finds a subset of the original features and the latter converts the data from

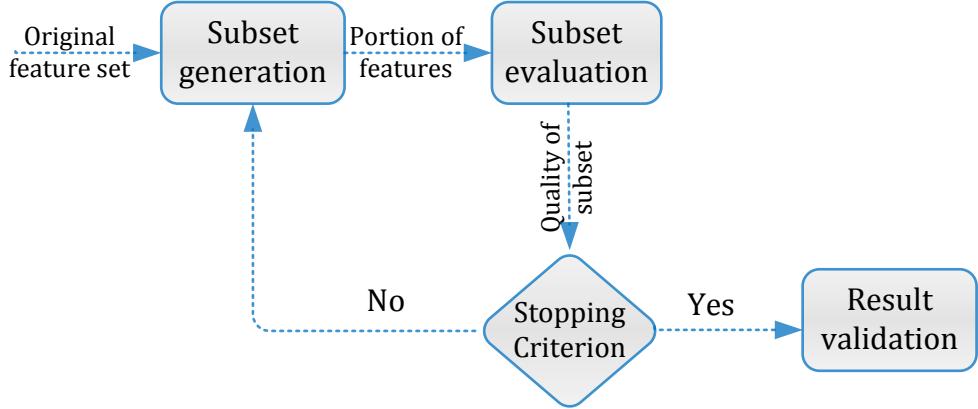


Figure 2.4: Main steps in feature selection

a high dimensional to lower-dimensional space [36]. In more detail, the size of the feature space can often be greatly decreased without losing much information.

FR is applied in the data pre-processing module for building an effective and online NADS in which it plays a significant role in efficiently and effectively detecting network attacks. As network packets have some information which might be important for identifying anomalies, they should be carefully analysed to select only the relevant information that can help a DE approach correctly detect anomalous activities. FS methods comprise four steps, subset generation, subset evaluation, a stopping criterion and result validation, as depicted in Figure 2.4 (see [26, 87, 88]).

- **Subset generation**

This is an essential heuristic search process whereby each state in the search space specifies a candidate subset for the evaluation step which is analysed using two strategies. Firstly, the starting points that affect the direction of the search have to be determined. The search, which begins with a randomly adopted subset to avoid the local optimal problem, could have an empty set

of features then added, a full set with features then eliminated or a set with features added and eliminated simultaneously. Secondly, it has to determine a search strategy, such as a sequential or random one. For a dataset with D features, there are 2^D candidate subsets, a space that enables an excessively thorough search with even only a reasonable number of features [88].

- **Subset evaluation**

Each new subset created has to be assessed using an evaluation measure which can be classified as either independent or dependent based on the learning techniques in which it is applied on the selected features [88]. Some common independent measures are information, distance and consistency which are used in filter models. A dependent measure is used in wrapper models which require a predefined learning technique for feature selection. This technique provides the criteria/criterion for determining which features should be chosen.

- **Stopping criterion**

A stopping criterion controls when a FS method should end, with common ones the minimum number of features selected, maximum number of iterations and completion of the search [88].

- **Result validation**

A simple means of validating results is estimating the output using prior information about the data. Although it is difficult to find such information in real applications, we depend on some indirect approaches via monitoring changes in performance with changes in selected features; for instance, a classification error rate can be used to measure the learning performance and check the relevance of the selected features [26, 88].

The Association Rule Mining (ARM) [89], Principal Component Analysis (PCA) [90] and Independent Component Analysis (ICA) [91] techniques are used in Chapter 4 for selecting important network features. Their methodologies are described

in Chapter 4, and the popular studies applied them in NADS are explained as follows.

Many studies [92–94] have used the ARM technique in a NADS to detect abnormal instances. Lee et al. [92] proposed an IDS that correlates program and user activities using ARM to generate the most frequent observations to identify malicious activities. Luo et al. [93] used the ARM to construct a set of rules from audit data to establish a normal profile and detect any variation from it as an attack. Yanyan and Yuan [95] developed a partition-based ARM technique for scanning the training set twice. In the first scan, the data is divided into many partitions to run easily in memory while, in the second, itemsets of the training set are created.

Nalavade et al. [94] proposed an ADS based on integrating association rules. Its algorithm establishes abnormal rules that can identify malicious observations in network data. However, this technique has some problems in terms of detecting abnormal events as it requires a huge number of instances to correctly generate normal and attack patterns. Moreover, as it takes a long time to build a model for network data, it is difficult to create online systems for detection purposes and it needs the labels of normal and attack data to efficiently generate rules. However, in Chapter 4, Section 4.5, we use it to select relevant features and observations due to its capability to proficiently correlate variables by selecting a set of network flows each time.

As several research studies have been undertaken using the ICA and PCA techniques to analyse the potential properties of network traffic and eliminate inappropriate or noisy features, we use it in the data pre-processing module to address the variety of problems inherent in big data discussed in [96, 97]. Palmieri et al. [91] suggested NADS using the ICA to elicit the basic components of network data from several network sensors, improving the performance of attack detection. In [98], a technique using the ICA mechanism was developed to detect stealthy attacks with a high detection accuracy. It was supposed that the attacker has

no information about the system, and those attacks were detected based on a measurement matrix. A similar technique for attack detection considering both full- and partial- measurements was suggested in [99].

Hornig et al. [100] proposed a category-based ADS based on the PCA mechanism to decrease the number of features in the DARPA 1998 dataset. Their experimental results demonstrated that their model produces a smaller number of features which leads to an increase in the speed of detecting an attack with the same accuracy. In [101], an adaptive IDS using a combination of SVM and PCA evaluated on the KDD99 dataset is proposed, with the results showing that PCA improves both the accuracy and processing time. Recently, Eduardo et al. [50] suggested a hybrid statistical technique using PCA, the Fisher Discrimination Ratio and Probabilistic Self-Organising Maps (SOMs) to remove unimportant features when building an adaptive IDS. In Chapters 4 and 5, the two techniques are used to estimate their effect on the overall performance of NADS.

C. Feature conversion

NIDS datasets include quantitative (i.e., numeric) and qualitative (i.e., symbolic) features. As this thesis focuses on a statistical DE that can deal with only quantitative data, a standard format for features (X) is applied to convert symbolic features into numeric ones (i.e., $X \in R$), where R indicates real numbers [47]. In other words, symbolic data are replaced with sequential numbers for ease of processing in statistical approaches.

D. Feature normalisation

This is a function for scaling the value of each feature into a specific confidence interval, such as $[0, 1]$ [47]. Its main benefit is to remove the bias from raw data without amending the statistical properties of the features. Common functions

of normalisation are the linear transformation and z-score, as given in equations (2.9) and (2.10), respectively.

$$X_{normalised} = (X - \min(X)) / (\max(X) - \min(X)) \quad (2.9)$$

$$Z = (X - \mu) / \sigma \quad (2.10)$$

where X denotes the feature values, μ the mean of the feature values and σ the standard deviation.

2.7. Decision Engine (DE) Approaches

The DE module of a NADS is clearly a critical aspect in the design of an efficient system for discovering intrusive activities in real time. Selecting the functions of DE, as well as its training and testing phases, fundamentally contributes to measuring the effectiveness of a NADS as, if it is not performed correctly, the overall protection level will be able to be easily penetrated. Many research studies have investigated DE approaches which can be classified in five categories, classification-, clustering-, knowledge-, combination- and statistical-based, as depicted in Figure 2.5. The background to these categories and related work are discussed in the next subsections.

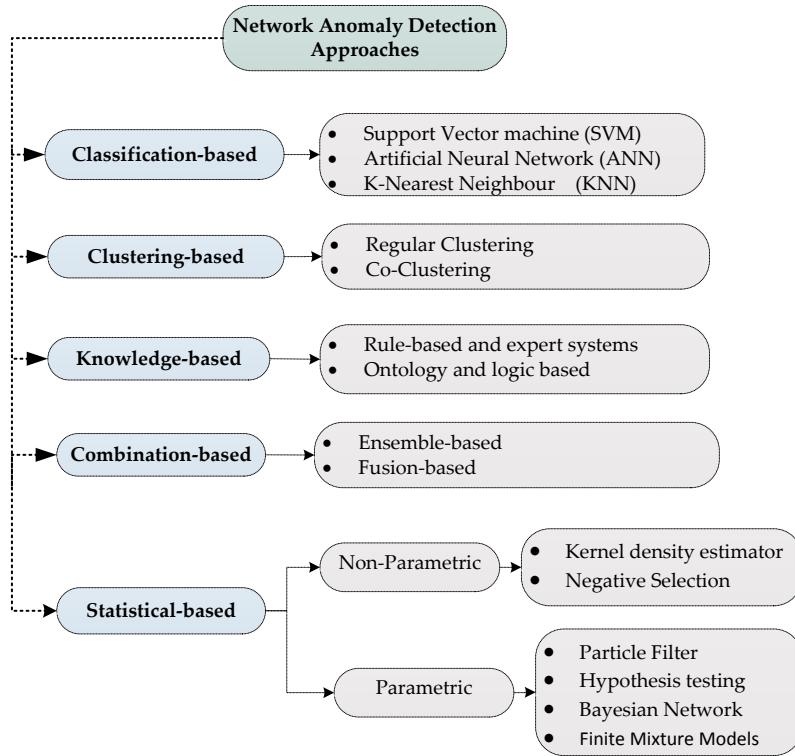


Figure 2.5: Taxonomy of network anomaly detection approaches

2.7.1. Classification-based approaches

Classification is categorising data instances in certain classes based on those in a training set while a testing set contains other instances for validating the labelling process; for example, assuming that we have two classes in which observations are labelled ‘1’ and ‘2’, these observations can be classified as linear or non-linear, as depicted in Figure 2.6.

In network anomaly detection, the data source always has high dimensions with different data types (i.e., quantitative and qualitative features), as discussed in subsection 2.6.2. Therefore, classification approaches have been used to build models that enable classifying network traffic behaviours into either two classes

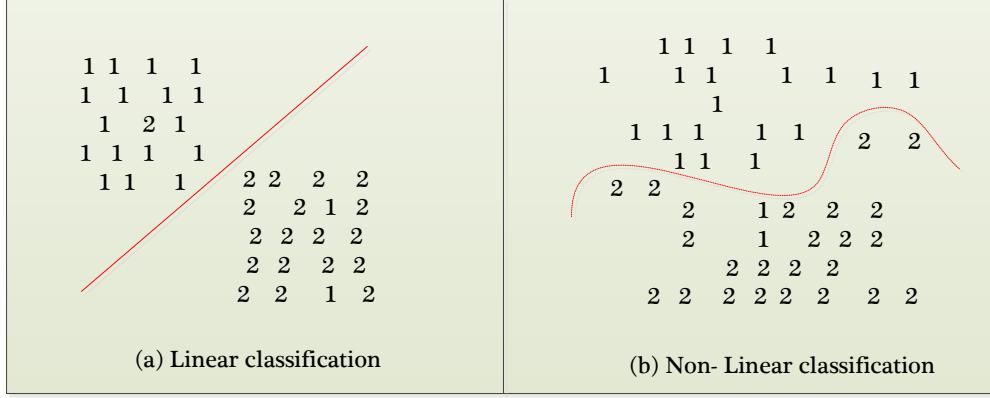


Figure 2.6: Classification types

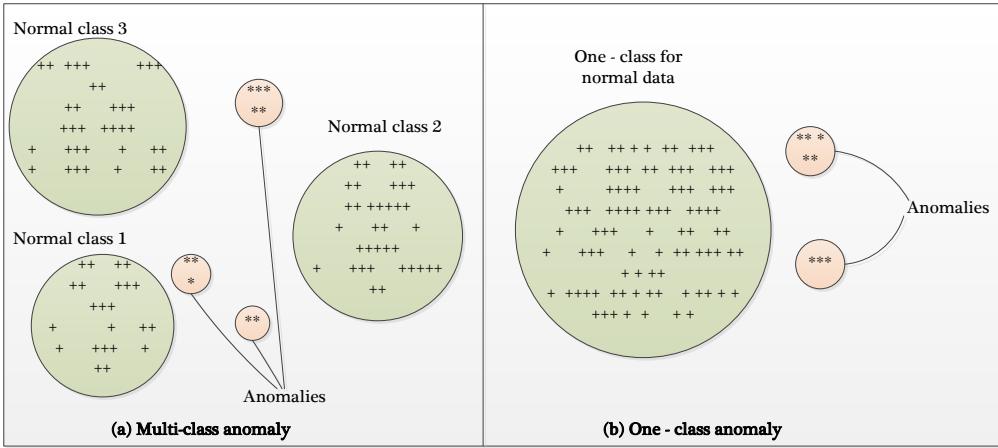


Figure 2.7: NADS classifications

(i.e., normal or attack) or a set of classes (i.e., normal with each attack as a class) [26, 102], as depicted in Figure 2.7.

The most popular classification-based techniques applied for NADSs are the Support Vector Machine (SVM), Artificial Neural Network (ANN) and K-nearest Neighbour (KNN). A typical SVM involves two steps for classifying data observations [103]; firstly, the training set is moved from the original input space into

a higher-dimensional feature space based on kernel functions to convert a linear non-separable problem into a linearly separable one; secondly, the data points are on a hyperplane with the maximal margins at the nearest data points on each side. A one-class SVM [104] uses only the training set of legitimate network data and considers any deviation from the normal patterns as an anomaly.

Wagner et al. [96] used a one-class SVM technique to build a network anomaly detection approach which detected zero-day attacks that did not belong to the normal training class. However, this technique often took a long time to train a large amount of data, such as network data. Khan et al. [97] decreased the time required for the training set using the hybridisation concept whereby they designed a NADS using a hybrid of SVM and hierarchical clustering techniques called the Dynamically Growing Self-organising Tree (DGSOT). They stated that using clustering in parallel with SVM improved both the training time and accuracy of network anomaly detection. Similarly, Horng et al. [100] proposed a NADS which included a hierarchical clustering and SVM to reduce the training time and improve detection accuracy. In [101], a least-square SVM was proposed for the design of a lightweight NADS by selecting the significant features of network data and detecting anomalies. More recently, a patent containing techniques such as SVM for achieving high rates of detection was published, with the SVM technique applied to detect anomalous behaviours of network traffic over a network node [105].

Another approach is to use ANNs that are inspired by the human brain and calculate in an entirely different way than traditional digital methods. An ANN technique is a machine-learning model that converts the input into outputs through non-linear latent processing of a set of artificial neurons [106] and has been widely applied for NADS research due to its popular use of pattern recognition methodologies. In a NADS methodology, ANNs require some information about the legitimate data class to systematically alter the interconnection neurons to learn the weights of the network and obtain a model that can discriminate attacks from

normal behaviours. There are several different types of ANN configurations, with artificial intelligence researchers aiming to optimise parameters and classify data points.

ANNs have been used with other kernel functions, specifically Multi-layer Perceptrons (MLPs), the Radial-basis Function (RBF) and Self-organizing Maps (SOMs), to improve the performance of a NADS [107], with an example of an ANN-based NADS employing a combination of an ANN and SOM to identify intrusions using resilient propagation ANNs provided by Jirapummin et al. [108]. Horeis [109] designed a NADS using a hybrid of ANN, SOM and RBF techniques which showed better results than NADSs based on only RBF networks. Liu et al. [110] proposed a NADS for detecting existing and zero-day attacks in network traffic based on an unsupervised ANN. It applied a hierarchical NADS using the Principal Components Analysis (PCA) and ANN to address the drawbacks of single-level structures. Recently, Shreya et al. [111] developed a NADS based on the collection techniques of a back-propagation neural network (KBB), K-means and Naïve-Bayes to improve the detection of malicious activities. However, ANNs usually take a long time to process a large amount of network data to determine the best neural weights for minimising classification errors as possible.

A KNN mechanism classifies each observation assigned to the class label by computing the highest confidence among the k data points nearest the query data point [30]. A KNN-based NADS creates a normal network profile and treats any deviation from it as an attack. It is a powerful DE for NADSs because it does not require learning parameters in the training phase. The KNN technique was used to design a Dependable NIDS (DIDS) based on the strangeness and isolation measures of its potential functions which could effectively identify network attacks [112]. Nevertheless, KNNs are often time-consuming and require vast amounts of storage to classify high-speed network traffic.

Other classification techniques, for instance, a decision tree, regression models

and fuzzy logic (see [38] – [41]) have also been applied to design NADSs. However, overall, classification-based IDSs rely heavily on the assumption that each classifier has to be adjusted separately and always consume more resources than statistical techniques. Ultimately, if these techniques do not successfully build normal patterns, they are not capable of detecting zero-day attacks or even variants of known ones. It is important to note that most classification techniques have been evaluated using old datasets, particularly the KDD99 dataset, and their poor performances will certainly be worse on newer datasets.

2.7.2. Clustering-based approaches

Clustering approaches are unsupervised machine-learning mechanisms which assign a set of data points to groups based on the similar characteristics of these points, such as distance or probability measures; for example, if we have unlabelled data instances in two dimensions (X and Y), we might group them into five clusters, namely C_1 to C_4 , as shown in Figure 2.8 (a). Another concept derived from clustering is outliers which denote that some data points in a dataset are more highly deviated than regularly grouped ones; for example, in Figure 2.8 (b), the data points of O_1 and O_2 are outliers while those of N_1 and N_2 are normal clusters [113].

Although there are different clustering techniques, the most popular types applied for NADSs are regular and co-clustering with the difference between their strategies of processing the observations and features of a network dataset [62, 63]. Specifically, regular clustering, such as K-means clustering, assembles data points from the observations of a dataset while co-clustering simultaneously considers both the observations and features of a dataset to provide clusters.

When using clustering to identify anomalies, three key assumptions are usually made. The first is that, as legitimate data instances often fall into a cluster

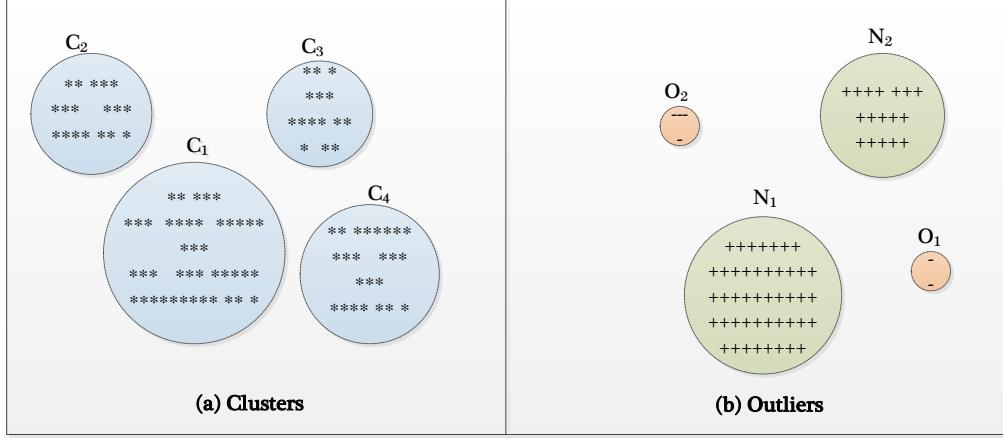


Figure 2.8: Methodologies of clusters and outliers

whereas attacks do not, in a NADS methodology, clustering identifies any data instances that do not fall into a legitimate cluster as attacks, with noise data also considered anomalous, as in [59]. A drawback of this assumption is that clustering techniques cannot be optimised to identify anomalies as the major goal of a clustering algorithm is to define clusters. Secondly, legitimate data instances are usually located near the closest cluster centroid while anomaly ones are often far away from it [63]. Techniques using this assumption consider the points farthest from the cluster centre as anomalies, with many of them suggested for designing NADSs [63] whereas, if anomalies are located in normal clusters, they cannot be correctly identified. To tackle this challenge, the third assumption is that legitimate data instances fall into vast and dense clusters and anomalies into small or spare ones. Mechanisms using this assumption identify data observations belonging to clusters with those of sizes and/or densities under a baseline considered anomalies.

Bhuyan et al. [114] designed an outlier-based NADS in networks in which legitimate data were clustered using a k-means technique and then a reference point computed for each cluster, with these points classified as attacks if they were less than a certain threshold value. Also, in [76], a NADS for large network datasets using tree-based subspace clustering and an ensemble-based labelling mechanism

for improving detection accuracy in a real network environment was proposed. Nadiammai et al. [115] analysed and evaluated k means, hierarchical and fuzzy c-means clustering techniques for building a NADS and reported that the complexity and detection accuracy of the fuzzy c-means algorithm were better than those of the others. However, this system could not work effectively on an unbalanced data problem in which the network instances of normal class are too larger than the instances of abnormal class. Jadhav et al. [116] proposed a NADS based on clustering network packets and developed a new data pre-processing function using the fuzzy logic technique for classifying the severity of attacks in network traffic data. Zainaddin et al. [117] proposed a hybrid of fuzzy clustering and ANN to construct a NADS which efficiently detected malicious events.

Clustering-based NADS techniques have several advantages. Firstly, they group data points in an unsupervised manner which shows that they do not need to provide class labels for observations, which is a very difficult process, to ensure the correct labelling of data as either normal or attack. Secondly, they are effective for clustering large datasets into similar groups to detect network anomalies, which decrease computational complexity, and perform better than classification methods. In contrast, one of clustering-based NADS drawbacks is that its clustering is highly reliant on its efficacy in profiling normal instances while another is that dynamically updating a profile for legitimate network data is time-consuming. Finally, its dependency on one of the three above assumptions is occasionally problematic for effectively recognising abnormal behaviours as it produces a high false alarm rate and, in particular, attack instances can conceal themselves in a normal cluster.

2.7.3. Knowledge-based approaches

Knowledge-based techniques establish a set of patterns from input data to classify data points with respect to class labels. In NADSs, network traffic data are examined against predefined patterns of anomalies and system vulnerabilities to

detect malicious events and raise an alarm [118]. Although these approaches can identify known attacks, they cannot determine zero-day ones unless a profile is constructed from diverse normal patterns which is extremely difficult to do as is updating it with new normal patterns [22].

Common knowledge-based NADS approaches are rule-based and expert as well as ontology- and logic-based [26]. Rule-based methods model the knowledge collected about suspicious network events which allows browsing of network traffic data to find evidence of existing vulnerabilities [119]. An expert system comprises rules which define attack events whereby network traffic data are transformed into patterns according to their relative weights in the system and an inference engine matches the predefined rules with the current state of the system to detect attack activities [120]. Rule-based and expert system approaches have been widely applied to detect suspicious network events while ontology- and logic-based ones model intrusion signatures based on a logic structure by incorporating the constraints and statistical characteristics of network traffic data [26].

Snort [121] is one of the most popular open-source and rule-based IDSs. Its rules recognise malicious network packets by matching the current packet against predefined rules and cannot detect zero-day attacks but produce a high FPR due to its methodology for identifying attack signatures [122]. Currently, Snort involves more than 20,000 rules which are usually updated by users [22]. The Petri nets tool [123], which was designed at Purdue University, is an example of a knowledge-based IDS which consists of directed bipartite graphs and Coloured Petri Nets (CPNs) representing the signatures of intrusions. Although this tool can easily represent small network data and helps in discriminating known attacks, its process for matching an attack signature with predefined rules is very difficult to execute in real network environments and takes a long processing time. Vaccaro et al. [124] proposed an intrusion-detection tool which identifies malicious statistical behaviours by establishing a set of rules that statistically depicts the behaviours

of users using logs of their activities over a certain period of time. It then matches the current activity against the stored rules to detect suspicious behaviours.

Scheirer et al. [125] proposed a syntax-based system which utilised a variable-length partition with many break points to identify polymorphic worms. It provided a semantics-aware capability to design a NIDS and could extract polymorphic shellcodes with additional stack structures and mathematical procedures. Naldurg et al. [126] suggested a framework for intrusion detection using temporal logic specifications with intrusion patterns formulated in a logic structure called EAGLE. It supported data values and parameters in recursive equations and enabled the identification of intrusions with temporal patterns. Hung et al. [127] presented an ontology-based approach for establishing a NADS according to the end-users' domain in which, as ontologies were applied as a conceptual modelling technique, a NADS could be simply built.

Knowledge-based NADS mechanisms have some advantages: firstly, they are sufficiently robust and flexible to discriminate existing attacks in small network traffic data; and, secondly, achieve a high detection rate if a significant knowledge base about legitimate and anomalous instances can be extracted correctly. On the contrary, they have FPRs due to the unavailability of biased normal and intrusion network traffic data and cannot identify rare or zero-day anomalies. Finally, their procedures for dynamically updating rules are very challenging and their processing times very expensive which are deterrents to building an online NADS [26].

2.7.4. Combination-based approaches

A combination-based methodology uses multiple mechanisms to classify data points effectively and efficiently, with most of those used for NADSs ensemble- and fusion-based mechanisms. Ensemble learning approaches integrate many techniques and consolidate them to achieve an overall accuracy which outperforms that of each

classifier [26, 128–130] and are categorised as bagging, boosting and stack generalisation [26, 131]. Firstly, bagging, so-called bootstrap aggregation, improves the detection accuracy by establishing an enhanced composite classifier which combines the findings of previously used classification techniques into one predictor. Secondly, boosting constructs an incremental ensemble by learning misclassified observations acquired from a previous model. Thirdly, stack generalisation obtains the highest generalised accuracy by utilising the output probabilities for each class label from base-level classifiers. Fusion-based approaches, which integrate the decisions coming from different classifiers, have emerged as techniques that could reinforce the final decision [132], with their taxonomy consisting of three levels, data, feature and decision. Some methods tackle the problem of high dimensionality by adopting only relevant attributes while others amalgamate classification techniques trained on diverse attributes using either hierarchical abstraction levels or the types of attributes involved [26].

Ensemble- and hybrid-based methods have been applied to design effective NADSs; for example, Chebrolu et al. [133] proposed an ensemble technique that combined two classifiers, Bayesian networks (BNs) and classification and regression trees [134]. Also, a collection of feature selection methods for designing a NADS was merged to achieve a better detection rate. Folino et al. [135] provided a distributed data mining technique for improving the accuracy of intrusion detection based on genetic programming extended with ensemble learning. Their data were distributed across several autonomous sites, with the winning module obtaining patterns from the input data and used network profiles to identify malicious observations, and achieving high accuracy. Perdisci et al. [136] established a high-speed payload NADS based on an ensemble of one-class SVM techniques for improving the accuracy of detection. Nguyen et al. [137] proposed a classification technique based on both the input features and additional ones provided by k-means clustering. These ensemble methods were computed using the classification capabilities of techniques for different local data segments provided by k-means clustering.

Giacinto et al. [138] developed a pattern recognition technique for NADS that employed a fusion of multiple classification mechanisms after five decision fusion approaches were evaluated by experiments and their performances compared. Shifflet [139] discussed a platform which allowed a hybrid of classification techniques to be executed together to build a fusion mechanism for the state of a network capable of efficiently detecting anomalous activities. Shreya et al. [111] proposed a hybrid technique for NADS using k-means clustering, a NB and back-propagation neural network, with its findings provided by the Bayesian inference to detect attack activities. Aburomman et al. [140] suggested an ensemble method which used PSO-generated weights to build a hybrid of more accurate classifiers for NADS created based on local unimodal sampling and weighted majority algorithm approaches to improve the accuracy of attack detection.

Combination-based methods are advantageous as they achieve higher accuracy and detection rates than single ones while requiring a set of controlling parameters that can be easily adjusted. However, adopting a subset of consistent and unbiased classification techniques is difficult because it depends on usinng a hybridisation measure to combine them. Also, it is evident that their computational costs for large amounts of network traffic data are high due to the number of classifiers used [26].

2.7.5. Statistical-based approaches

From the statistical aspect, an anomaly is a rare event which occurs amongst natural data events and is measured by statistical approaches which could be of the first order, such as means and standard deviations, the second order, such as correlation measures, or the third order, such as hypothesis testing, mixture models and inference approaches. In NADSSs, these approaches fit a statistical model of legitimate network data and then apply a statistical inference test, using either a threshold/baseline or probability condition, to identify deviated instances as anomalies [141]. In other words, an observation which has a low probability or does

not belong to a threshold condition can be declared an attack activity. Statistical-based approaches are classified as non-parametric and parametric [26, 141], both of which have been widely applied to develop statistical models for NADS. This research thesis also applies and develops a new DE based on such approaches which has the capabilities to discriminate among rare events and effectively analyse the potential properties of network data.

A. Non-parametric approaches

Non-parametric approaches do not make any assumptions about the statistical characteristics of given data. They create a model as they run and attempt to resolve the complexity of the data to efficiently adapt the data points. One of the simplest non-parametric statistical approaches is using histogram tools which graphically illustrate the tabulated frequencies of data [69]. In a NADS, a normal histogram is built and then new tested data points determined which, if they do not fall into the normal histogram are considered anomalous instances. For multivariate network data, feature-level histograms are established, with an overall score for a test data point attained by accumulating the scores of selected features.

The methodologies of the non-parametric methods applied in this PhD research are described below.

- **Kernel density estimator**

The kernel density estimator is a non-parametric method which bases its estimations on some kernel distributions, such as Gaussian, for all the sample space data and then integrates the local contributions of all the distributions [142]. Estimating the probability density of each sample depends on the data points that fall in a localised neighbourhood of the kernel.

The Parzen windows approach can be applied for non-parametric data density estimation to identify abnormal behaviours. A NADS-based Parzen is

built on estimating the probability density of normal data and rejecting outlier patterns as anomalies [143]. The Gaussian kernel function is chosen for such a technique for two reasons: firstly, as it is smooth, a PDF estimation also differs smoothly; and, secondly, if a radial symmetrical Gaussian is assumed, the function is identified using the data variance. A threshold is adapted based on a separate training set and the unconditional probability ($p(x)$) of a test observation (x) used to model the data distribution.

Shen et al. [142] suggested a NADS based on a non-parametric method which simulates the PDFs of some random variables. A set of kernel density estimators was established and the distribution parameters estimated to classify malicious and normal instances. These estimators were applied to enhance the SOM technique in order to produce a high detection rate and low false alarm rate. This method was extended in [144] to build a non-stationary high-dimensional PDF estimator using parallel programming to identify computer intrusions. Nicolau et al. [145] provided a one-class classification for a NADS using kernel density estimation and genetic programming, with each query point in the method classified as an attack if its density was below a certain threshold. However, this model was lazy because most of its computation was required to build the model. Yeung et al. [143] presented a NADS using a non-parametric density estimation method which utilised the Parzen window approach and Gaussian kernels to establish a profile from normal data and detect any deviations from this profile as attack instances. However, the Parzon window approach took a relatively long time during the testing phase.

- **Negative selection**

Negative Selection (NS) techniques have been widely applied for detecting anomalous network instances. The theory behind NS was inspired by the characteristics of the human immune system which can identify antigens [146], meaning that anything that is not a portion of the human body can be

detected, for example, viruses and bacteria. The aim of the immune system is to discriminate between antigens and the human body itself, with its potential process known as ‘self/non-self’ discrimination which ‘recognises’ an antigen by certain antibodies called T-cell receptors generated by an arbitrary process of genetic re-arrangements. The cells which successfully bind with normal cells are treated as normal instances while those that do not are considered anomalous ones, a process called NS. Likewise, attack detection has the essential objective of differentiating among the ‘self’ which resembles the normal operation of the monitored system and the ‘non-self’ indicating abnormal data.

Jinquan et al. [147] proposed a NS algorithm for building an adaptable NADS, with a normal training profile established using the ‘self/non-self’ space to adapt the model’s parameters. The empirical results showed that it provided a high detection rate and low FPR but its processing time was relatively high. Mostardinha et al. [146] suggested a NADS using a NS approach with a random diversity which used two kinds of agents, detectors and presenters. As the presenters passed information to the detectors, the detectors were chosen to participate in a greatly frustrated dynamic and the presenters received data from a state, the algorithm performed ‘self/non-self’ discrimination between the two agents. This detection-based methodology was that ‘if presenters prepare information that has never been available during the selection phase, they are involved in a less frustrated dynamic and their anomalous behaviours can be identified. Ramdane et al. [77] developed a NS approach called Hybrid NSA for IDS Adaptation (HNSA-IDSA) to build an effective NADS which was adapted automatically to be able to recognise low-footprint attacks. This system was evaluated on the KDD99 dataset, producing a high detection rate and low FPR.

B. Parametric approaches

Parametric approaches assume that network data follow a certain distribution, for instance, that a Gaussian distribution estimates the parameters of the given data [69]. However as, in real networking, the underlying distribution of network traffic data is not known, it is important to specify which probability distribution can fit the data with a relatively low error rate.

The Kolmogorov-Smirnov (K-S) test method [47] is one of the most popular methods used to determine the distribution of data, either Gaussian or non-Gaussian, and is used in this study to assess the best data distribution for the UNSW-NB15 dataset, as detailed in Chapter 3. The K-S test is applied using the SPSS tool and its hypotheses are H_0 and H_a which denote that the sample data are not significantly different from a normal population and are significantly dissimilar to a normal population, respectively. If the probability of finding an event is rare (i.e., less than 5%), the data will be non-Gaussian, otherwise, Gaussian [148]. As it is observed that network data do not belong to a Gaussian distribution [47], it is better to apply non-Gaussian distributions, such as a Gaussian Mixture Model (GMM), Beta Mixture Model (BMM) or Dirichlet Mixture model (DMM), to network data. The Probability Density Functions (PDFs) of these distributions have to be modelled from the ingress network data from which their parameters should be dynamically adjusted, instead of there being a static setting, to build a flexible model which distinguishes anomalies from normal observations [141].

The methodologies of the most commonly used parametric methods in this thesis are discussed in the following.

- **Particle filter**

A particle filter is an inference mechanism which measures the unknown state from a set of observations with respect to a time, with the posterior distribution established by a set of weighted particles. The dynamic state

system comprises a state transition model and observation model. The state transition function computes the changes of the object being tracked based on the previous state and system noise whereas the measurement function models the correlations among the observations and states given the observation noise [149, 150].

Lin et al. [149] suggested a NADS which applied the particle filter approach and was simulated using CPN tools to detect intrusion activities. The Coloured Stochastic Petri Nets (CSPN) technique was used for an in-depth assessment of the proposed NADS while the particle filter was applied to analyse network packets. In this technique, the packet inspection for extracting a set of features (i.e., the flow identifiers of IP addresses and port addresses and protocol types) as particles was considered. Each particle had a different weight on a time window used to build the model. During the filtering step, each packet was classified as normal or attack based on an increase or decrease in the corresponding weights of the particles, respectively. However, this scheme did not provide highly accurate attack detection for two reasons; firstly, using flow identifier features with classifiers could not be beneficial for distinguishing between normal and attack observations as their standard protocols, such as DHCP, could dynamically change IP addresses; and, secondly, adjusting and controlling network flows is very challenging because they are not continuous data for operating effectively with the particle filter methodology.

Jing et al. [151] proposed a Continuous Time BN (CTBN) model for detecting attacks that penetrated both host and network activities. In a NADS, a hierarchical CTCN was established for network traffic data and a particle filtering approach for learning the parameters. The model was aimed at recognising malicious activities if their likelihood was lower than that of normal ones. This technique was assessed using the KDD99 dataset and produced high accuracy and a low FPR. If the challenges of using the particle filter technique, as discussed in [149, 151], are properly tackled using feature

selection and discretisation methods, this technique will provide an efficient and reliable NADS for tracking.

- **Hypothesis testing**

Hypothesis testing is a simple statistical method for designing a NADS which examines whether the data samples come from the same distribution as legitimate training data and consists of both the initial hypothesis (H_0) and an alternative one (H_1), with the initial one assumed and then either accepted or rejected after estimating its probability. The estimator used separates data points into disjoint sets encompassed in both the acceptance and rejection regions [152, 153]. The two types of errors related to hypothesis testing are:

- Type *I*, which occurs when the null hypothesis is rejected even though it is correct, with the probability of this error reflecting the FPR; and
- Type *II*, which occurs when the null hypothesis is accepted even though it is incorrect, with the probability of this error reflecting the FNR.

Krishnan et al. [154] proposed an online technique of sequential hypothesis testing for detecting malicious events from clients using non-existent (NX) responses. It was designed to prevent botnets which tend to create DNS queries which extract NX responses. Its aim was to discriminate among benign and suspicious DNS queries initiating from the same client and determine their capabilities to scale to high network traffic loads. The experimental results showed that, on real-world data, this technique outperformed other well-known ones.

Abouzakhar et al. [155] used the chi-square statistical approach on the CAIDA backscatter-2008 dataset to detect suspicious activities occurring in network traffic. A normal profile was established based on a chi-square goodness-of-fit test which estimated variations between the distributions of test observations and normal profile to recognise attack behaviours. Santos

et al. [156] built a NADS for monitoring malicious network behaviours based on hypothesis testing, the BN and game theory mechanisms. They stated that combining agent-based models and adversarial hypothesis testing could improve the performances of network detection techniques. The evaluation of the model showed that it was enhanced by adversarial hypothesis testing which integrated the responses to malicious activities.

- **Bayesian network (BN)**

A BN is a joint probability distribution represented in a graphical mode for making decisions regarding uncertain data [30]. It is built on the Bayes theory which produces a hypothesis (H) of classes and observations (X) estimated as

$$p(H|X) = \frac{p(X|H)p(H)}{P(X)} \quad (2.11)$$

where $p(H)$ is the prior probability of each class without available information about observation X , $p(H|X)$ the posterior probability of observation X over the sustainable classes and $p(X|H)$ the likelihood of observation X given a class hypothesis (H).

Altwaijry et al. [157] developed a NADS based on a naive BN technique which recognised possible attacks faced by computer networks and was then extended to a multi-layer BN to improve detection accuracy. The system was assessed on the KDD99 dataset and its results for detecting rare events, such as U2R and R2L, were acceptable. Xiao et al. [158] proposed a BN Model Averaging (BNMA) classifier for detecting network attacks which was evaluated on the NSL-KDD dataset as it has fewer redundant data than the KDD99 dataset. The BNMA classifier produced significantly better accuracy detection than the Naive Bayes technique and the BN with a heuristic technique such as the genetic algorithm.

Han et al. [157] developed a naive BN NADS using the PCA which computed the highest ranked features within the PCA and used the selected features and their components as weights to improve the traditional naive Bayesian technique. The experimental results reflected that it could effectively decrease the data dimensions and improve detection accuracy. Thaseen et al. [159] designed a NADS using a combination of a naïve BN classifier, Linear Discriminant Analysis (LDA) and chi-square feature selection. The LDA was used to reduce the dimensionality of the network data and, thus, remove noisy features, the chi-square feature selection to adopt the optimal feature set to improve detection accuracy and then the naïve Bayesian classifier to classify normal and malicious data. This system was evaluated on the NSL-KDD dataset and shown to provide better accuracy and a lower false alarm rate.

- **Finite mixture models**

As a finite mixture model can be defined as a convex combination of two or more PDFs, the joint properties of which can approximate any arbitrary distribution, it is a powerful and flexible probabilistic modelling tool for univariate and multivariate data [160]. The methodology of these models is widely applied to design the BMM described in Chapter 5 which is used for NADSs. Network data are typically considered multivariate as they have d dimensions for differentiating between attack and normal instances [47]; for example, let $X = [X_1, \dots, X_d]$ be a d-dimensional random variable and $x = [x_1, \dots, x_d]$ an observation of X . The PDF of a mixture model is declared by a convex combination of K -component PDFs [101] and is given as

$$P(x|\theta) = \sum_{k=1}^K \alpha_k P(x|\theta_k) \quad (2.12)$$

where $\alpha_1, \dots, \alpha_k$ are the mixing proportions, each θ_k a set of parameters defining the k^{th} components and $\theta = (\theta_1, \dots, \theta_k, \alpha_1, \dots, \alpha_k)$ the complete set of parameters required to identify the mixture. Applying the probability conditions, α_k has to satisfy

$$\alpha_k \geq 0, \quad K = 1, \dots, k \text{ and } \sum_{k=1}^K \alpha_k = 1 \quad (2.13)$$

The mixture model is computed by the Maximum Likelihood Estimate (MLE) [141]. Assuming X data with N observations, the probability of data in which x_i are identically and independently distributed (*i.i.d*) is given by

$$p(X | \theta) = L(\theta | X) = \prod_{i=1}^N \sum_{k=1}^K \alpha_k p_k(x_i | \theta_k) \quad (2.14)$$

The MLE is derived from the set of parameters (θ) by

$$\theta^* = \operatorname{argmax}_{\theta} L(\theta | X) \quad (2.15)$$

The GMM is the mixture model most often applied for NADSs. It estimates the PDF (from equations (2.11) to (2.14)) of the target class (i.e., normal class) given by a training set and is typically based on a set of kernels rather than the number of rules in the training set. The parameters of this technique are estimated by maximising the log-likelihood of the training data (TR) computed using the EM algorithm, conjugate gradients, re-estimation techniques or Bayesian inference [45, 141]. Mixture models require a large number of normal instances to correctly estimate their parameters and it is difficult to select a suitable threshold (δ), as

in equation (2.15), which differentiates attack instances from the normal training class with a certain score.

$$\left\{ \begin{array}{l} \delta \geq score \implies \text{normal instance} \\ \text{otherwise} \implies \text{anomalous instance} \end{array} \right\} \quad (2.16)$$

This score can be defined using the unconditional probability distribution ($w(X) = p(x)$) and a typical approach for setting the threshold ($\delta = p(x)$) [36]. Another method for determining the threshold is the Cumulative Density Function (CDF) estimated by integrating $p(x)$ over an area (A). The threshold is initiated with 0 and gradually increased based on the Maximum-a-posterior (MAP) measure which, if lower for a test instance than the maximal threshold, indicates an anomaly instance. This theory is further discussed in Chapter 5 as one of the contributions of this research to designing new DE techniques which effectively detects anomalous network events. Also, an Extreme Value Theory (EVT), which is a statistical measure that considers extreme deviations from the median of a probability distribution and reflects extremely large or small values of the probability distribution used to fit a set of data, is used to adapt the threshold [45].

Fan et al. [161] developed an unsupervised statistical technique for identifying network intrusions in which legitimate and anomalous patterns were learned through finite generalised Dirichlet mixture models based on Bayesian inference, with the parameters of the mixture model and feature saliency simultaneously estimated. In [162], they extended their study to provide an efficient method for the varied learning of finite Dirichlet mixture models to design a NADS. This approach was based on the establishment and optimisation of a lower boundary for the likelihood of the model by adopting factored conditional distributions through its variables.

Greggio [163] designed a NADS based on the unsupervised fitting of network data using a GMM which selected the number of mixture components and fit

the parameter for each component in a real environment. The highest covariance matrix identified legitimate network activities, with the smaller components treated as anomalies. Christian et al. [164] proposed a NADS based on combining parametric and non-parametric density modelling mechanisms in two steps. Firstly, malicious samples were recognised using the GMM and then clustered in a non-parametric measure in the second step. While a cluster stretched to an adequate size, a procedure was identified, transformed into a parametric measure and added to the established GMM. These techniques were evaluated using the KDD99 dataset and their results reflected a high detection accuracy and low FPR. However, they would require the use of Bayesian inference to be adjusted for their efficient application in real networking.

Some research studies have used the Trapizodal Area Estimation (TAE) mechanism as a complementary function for detection purposes, in either network traffic or other domains, such as signal processing, computer vision and neuroscience. For example, Niandong et al. [165] suggested a fuzzy expert system for network forensics to determine computer crimes and produce automated digital evidence. Trapezoidal- and rectangular-shaped functions were applied as fuzzy membership functions to show a degree of truth about the number of ping attacks. Similarly, these functions were used for designing a fuzzy anomaly detection technique for reducing energy consumptions [166]. Jongsuebsuk et al. [167] proposed a fuzzy genetic mechnaism for defining suspicious network instances. The fuzzy trapezoidal rule was used to define abnormal data, whereas the genetic algorithm is used for finding appropriate fuzzy rules. Nedevschi et al. [166] proposed a lane stereovision detection mechanism based on a trapezoidal rule for lane model matching. In [156], Yuan et al. suggested a detection approach using trapezoidal rules for identifying the windshield regions of vehicles which recognised each region as a trapezoidal area by the shapes, colours and locations of parts of the vehicles. However, this mechnaism is used in Chapter 5 for developing a new DE technique for detecting existing and zero-day attacks based on area estimations of normal and abnormal instances.

Table 2.5: Comparison of decision engine techniques

DE techniques	Advantages	Disadvantages
Classification	<ul style="list-style-type: none"> - Reflect high detection rate and low false positive rate if the network data is correctly tagged 	<ul style="list-style-type: none"> - Depend on the assumption that each classifier has to be constructed separately - Take more computational resources
Clustering	<ul style="list-style-type: none"> - Group data with no dependency on the class label 	<ul style="list-style-type: none"> - Depend on the efficacy of establishing a legitimate profile
	<ul style="list-style-type: none"> - Decrease processing times 	<ul style="list-style-type: none"> - Need a higher time while updating the established profile
Knowledge	<ul style="list-style-type: none"> - Identify on known intrusive activities 	<ul style="list-style-type: none"> - Consume too much time during the training and testing phases
	<ul style="list-style-type: none"> - Provide high detection rate for existing attacks 	<ul style="list-style-type: none"> - Apply static rules for recognising suspicious events
Combination	<ul style="list-style-type: none"> - Attain high accuracy and detection rates 	<ul style="list-style-type: none"> - demand a huge effort to incorporate more than one technique
	<ul style="list-style-type: none"> - Need only a set of controlling parameters to be adapted. 	<ul style="list-style-type: none"> - Consume a long processing time than other mechanisms
Statistics	<ul style="list-style-type: none"> - Achieve higher accuracy and detection rates if a threshold of identifying attacks correctly adjusted from network data, as provided in this thesis 	<ul style="list-style-type: none"> - Need precise analysis to select the correct threshold
	<ul style="list-style-type: none"> - Do not take computational resources like other mechanisms 	<ul style="list-style-type: none"> - Demand new functions to identify attack types, such DoS and DDoS

A brief comparison between advantages and disadvantage of the existing DE techniques is demonstrated in Table 2.5.

2.8. Contemporary Network Threats

The numbers, types and complexity of network threats are increasing. Cyber adversaries can cause financial losses and reputational damage, steal sensitive information and intellectual property, and interrupt business. An attacker's philosophy almost invariably comprises two phases [11]. The first, the so-called exploitation phase, is a method for controlling the execution flow in the targeted program. At its abstract level, this can be a stack-based buffer overflow in which an intrusively long text overwrites the instruction pointers of the targeted program but also includes a full suite of methods which can be used by more sophisticated adversaries to gain control of a system while its code is running. The second phase is known as the payload phase. After successfully exploiting the execution flow to the payload, this phase performs the aim of the attacker, such as to steal information and/or disrupt computer resources. The payload process is executed through a shellcode terminal which establishes a command prompt on the hacker's computer to execute post-exploitation events.

Based on the Australian Cyber Security Centre (ACSC) [9] and McAfee threat reports [10], Figure 2.9 depicts the current top attacks which expose computer networks. Firstly, a DoS is an attempt by an attacker to prevent legitimate access to websites by overwhelming the amount of available bandwidth or resources of the computer system (e.g., zombies). When many computer systems are utilised to investigate such activities, such as applying a botnet, it is known as a DDoS attack. The number of these network attacks has been increasing, with a variety of DDoS types of attack sending more than 100 Gbps which constitute serious vulnerabilities for computer networking [168].

Secondly, another type of attack is Brute Force which endeavours to illegally obtain pairs of user names and passwords by trying all predefined pairs to gain

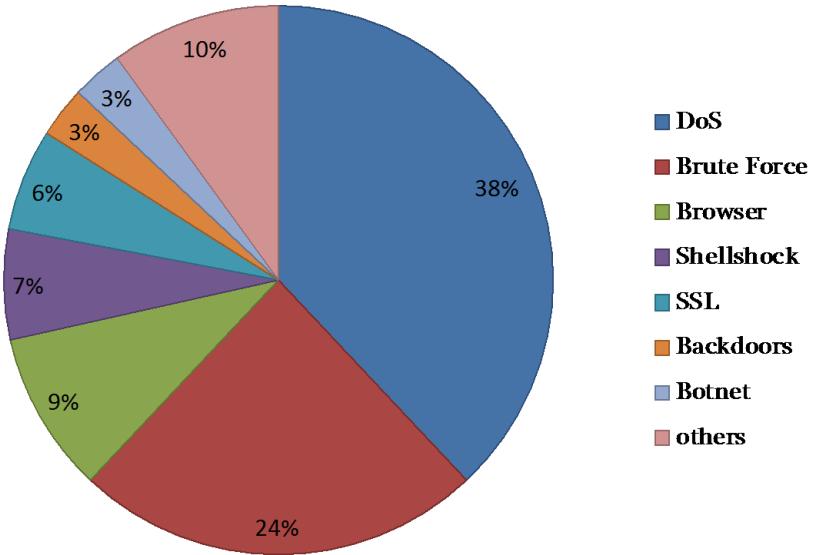


Figure 2.9: Recent top network attacks

access to network services, with automated applications often used to guess password combinations. To prevent such an attack, network administrators can place restrictions on the acceptable number of login attempts and generate a blacklist for a client whose network traffic are anomalous. This leads to the blocking of IP addresses after multiple login failures as well as limiting access to specific IP addresses [169]. Thirdly, browser-based network attacks, such as Tor, attempt to penetrate anonymous communication systems by exploiting JavaScript and HTML to create some predefined rules for correlating user activities based on the websites visited. They are often executed by an attacker penetrating a client's vulnerabilities, which are typically triggered by outdated software, and possibly tempting the user to unwittingly download malware masquerading as a fake software/application update. A common solution to browser-based attacks is to frequently update web browsers and their services, for example, Java and Flash, so that browser vulnerabilities are easily detected [10, 170].

Fourthly, Shellshock attacks relate to vulnerabilities that breach the command-line shell of Linux, UNIX and Apple OS systems called Bash. When Shellshock appeared in September 2014, many computer systems and appliances were vulnerable as they could be penetrated by a remote code execution which possibly authorised attackers to have full access and control. This permitted anomalous commands to be executed which could then download and implement anomalous scripts. Fifthly, a successful SSL attacker aims to intercept encrypted data, send them over a network and then access the unencrypted data and benefit by gaining access to applications. In April 2014, a dangerous vulnerability in the OpenSSL execution of the TLS/SSL Heartbeat extension, namely Heartbleed, was publicly released and caused the leaking of memory data. An attacker could also access private keys, confidential information and secure content which could help other cyber adversaries. Moreover, these vulnerabilities allowed attackers to continually access the private information in systems by sending a wide variety of malicious commands to susceptible servers [9, 10].

Sixthly, a Backdoors attack can be defined as a technique which exposes computers to remote access by naturally replying to particularly constructed client applications. Several of them essentially use the IRC backbone and receive commands from IRC chat clients through the IRC network [10, 171]. They are less popular attacks than others and often used as part of targeted intrusions [171] which can be custom-designed to evade security detection and provide a masked point of entry.

Seventhly, a botnet denotes the number of hijacked computer systems remotely operated by one or many malicious actors which coordinate their activities by Command and Control (C&C). Networks are regularly hit with attempts to expose their computer systems and appliances as attackers execute DDoS attacks to send spam email or implement fraudulent botnets to penetrate their targeted networks [9, 10]. Ultimately, hacktivism refers to malicious cyber-attacks designed by groups or individuals for a particular reason or with the aim of targeting a certain person

or organisation to gain access to their resources (more details are provided in [9]). These attacks could be called stealth attacks (see [11]) in which an active attacker elicits data packets from a network in order to find a way of compromising that network's security. After this security is breached, the attacker removes all traces of the network to keep it hidden and avoid discovery [10].

2.9. Chapter Conclusion

This chapter discusses the concepts and literature related to IDSs, specifically a NADS. Due to rapid advances in technologies, computer network systems need a solid layer of defence against vulnerabilities and severe threats. Although an IDS is a significant cyber security application which integrates a defence layer to achieve secure networking, it still faces challenges for being built in an online and adaptable manner. Anomaly detection methodologies which can efficiently identify known and zero-day attacks are investigated.

It has been a very challenging issue to apply a NADS instead of a MDS methodology in the computer industry which could be overcome by framing its architecture with a data source, pre-processing method and DE mechanism. A NADS is usually evaluated on a data source (i.e., a dataset) which involves a wide variety of contemporary normal and attack patterns that reflect the performances of DE approaches. Unfortunately, the publicly available benchmark datasets have serious issues which lose an evaluation's fidelity. The research presented in this thesis also presents the new UNSW-NB15 dataset which addresses the problems of the legacy datasets in Chapter 3.

The network dataset used consists of a set of features and observations that may include irrelevant ones that could negatively affect the performances and accuracy of DE approaches. Consequently, data pre-processing methods for creating, generating, reducing, converting and normalising features (see Chapter 4) are executed to pass filtered information to a DE approach which distinguishes between

anomalous and legitimate observations and has been applied based on classification, clustering, knowledge, combination and statistics discussed to demonstrate their merits and demerits in terms of building a NADS. In addition, as statistical approaches could determine the inherent characteristics of data, they are described in detail and used to develop the new DE approaches which can efficiently detect existing and zero-day attacks described in Chapter 5.

Chapter 3

Towards Development of New Environments of Large-scale Network Datasets and Their Features for evaluating Intrusion Detection Systems

3.1. Target of Network Dataset

In the 1980s, James P. Anderson released a paper that outlined for the first time a way of establishing an automated IDS [1], with its main focus on collecting and using audit data to identify potential threats which attempt to penetrate computer resources. Since then, it has been acknowledged that DE approaches need a data source to evaluate their efficiency and efficacy for detecting cyber adversaries.

NIDS datasets can be conceptualised as relational data, as reflected in Table 3.1. The input to a classification technique is a set of data records (also defined as vectors, events, samples or observations), each of which consists of columns (i.e., attributes/features) with different data types, that is, integer, floating, binary or categorical [172]. In the case of input multivariate data, whether their features are the same or different data types determines the applicability of DE techniques for processing them [173]. However, as most DE methods, particularly statistical models, can handle only numeric data, they face the challenge of using all these features.

NIDS datasets need to be labelled so that DE approaches can learn and validate their models to effectively distinguish between legitimate and suspicious

Table 3.1: Input features

	Feature 1	Feature 2	Feature 3	Class label	Type
Record 1	2	Y	0.07	1	Training
Record 2	1	X	0.02	0	Training
Record 3	3	K	0.004	1	Testing

records. For this purpose, although network’s features, which are elicited from raw network packets (i.e., pcap formats¹) using different tools, for instance, Argus, Bro-IDS and Net Flow, need to be understandable. The features elicited could include irrelevant and noisy features that have to be eliminated. It is essential that normal and suspicious activities in the acquired network data are correctly labelled to ensure the fidelity of evaluating the DE technique used. The labelling process is conducted by matching processed records according to the ground truth of anomalies. The ‘class label’ column in Table 3.1 indicates whether a record in a dataset is normal (0) or abnormal (1) [24]. It is noted that the labelling process is expensive because of the difficulty of ensuring that the ground truth of attack behaviours is precise and its records correctly matched with all the transaction records in a dataset [172].

The key contributions in this chapter are described as follows.

1. We suggest statistical features from the DARPA-2009 dataset for building an effective NIDS, with some existing ML techniques applied to evaluate its credibility for detecting abnormal activities.
2. We develop a new dataset, called the UNSW-NB15 dataset by configuring an authentic testbed environment at the UNSW cyber security lab, with the IXIA tool used to simulate current representations of normal and abnormal network traffic.

¹**Pcap** stands for **P**acket **c**apture, which includes an application-programming interface for storing network data. The UNIX operating systems implement the pcap format using the libpcap library, whilst the Windows operating systems use a port of libpcap, called WinPcap.

3. We analyse and determine these datasets in terms of statistical analysis using different statistical measures and ML techniques in order to estimate their reliability and credibility for evaluating new NIDSs.

As discussed in Chapter 2, subsection 2.6.1, public benchmark network datasets have serious problems that affect their reliability for evaluating NADSs. This was the motivation for describing and evaluating the DARPA-2009 dataset in Sections 3.2 and 3.3, respectively. As we found that it has some issues, we created the UNSW-NB15 dataset to address them, as discussed in Sections 3.4 and 3.5, respectively. The rest of this chapter is organised as follows. Section 3.6 compares the UNSW-NB15 dataset with others and Section 3.7 discusses its big data terms. Section 3.8 explains the way of splitting a dataset for training and testing machine learning (ML) techniques. The complexity analysis conducted of the UNSW-NB15 dataset is explained in Section 3.9 and its experimental results discussed in Sections 3.10 and 3.11, respectively. Finally, we conclude this chapter and suggest directions for future research.

3.2. Description of DARPA-2009 Dataset

The DARPA-2009 dataset² [85] was synthetically generated from a simulation of 16 local sub-net networks (i.e., 172.28.0.0/16) connected to the Internet. The local network traffic was not extracted but the raw packets of traffic between the local sub-nets and Internet were captured by a sniffer between the 3rd and 12th November 2009. This dataset consists of an enormous amount of network data with HTTP, SMTP and DNS protocols. It has a wide variety of security events and attack types, such as DDoS and worms, that are parameterised to demonstrate several propagation characteristics, and comprises 7000 pcap files with a total size

²The work of this study presented in this chapter has been published in:
Moustafa, Nour, and Jill Slay. "Creating novel features to anomaly network detection using DARPA-2009 data set." Proceedings of the 14th European Conference on Cyber Warfare and Security 2015: ECCWS 2015. Academic Conferences Limited, 2015.

of approximately 6.5 TBytes. Each file is less than 1 Gigabyte in size for easy analysis of its events and represents an approximately one- to two-minute time window with respect to the traffic rate of the simulation [174].

3.2.1. Security Events in DARPA-2009 Dataset

The DARPA-2009 dataset includes 46 security events occurring over 10 days, with basic information about them documented in the dataset’s ground truth spreadsheet [175]. This information includes attack types, source and destination internet protocol (IP) addresses and ports, and start and end times of offensive events [174].

The following are serious security events in this dataset.

- **DDoS attacks** - occur when many intrusion systems are used to flood a target system with many packets whereby the resources and services of this system can become corrupted and inaccessible. The DARPA-2009 dataset includes a wide variety of DDoS attacks, the scenarios of which are their connections to the destination IP address 172.28.4.7 on the HTTP port with SYN packets. Almost 100 IPs were observed penetrating the victim at the same time through the dataset and establishing multiple *SYN-flood* attacks.
- **Malware DDoS attacks** – were caused by many local clients used to launch malicious events via the IP address 205.63.202.67 and recorded as ‘client compromise’ in the ground truth spreadsheet. To log these events, the IP address 152.162.178.254 with a 499 TCP destination port was targeted from days 4 to 9.
- **Spambots** - compromised 56 local clients between the 3rd and 5th days. The two IPs 201.89.32.16 and 44.29.203.5 were used to start these malicious spam events which were labelled in the ground truth as ‘*spambot client compromise*’. A spambot program was used to download the malicious events via

three outside IP addresses, 68.91.226.37, 64.222.102.58 and 64.222.102.58, to expose the 56 local clients by sending approximately 14 KB of payload to each of them.

- **Scans** - were logged as ‘*scan/usr/bin/nmap*’ in the ground truth sheet, with some outside IP addresses scanning particular local IP addresses with specific ports to collect information about target systems in the dataset.
- **Phishing emails** - were included in the dataset by two sequences of events. The first began with the event ‘*noisy phishing email/ exploit/ malware/-trawler*’ which exploited the SMTP protocol while the second, namely, ‘*noisy client compromise + malicious download exfil*’, also exposed the SMTP protocol to penetrate mail servers embedded in the testbed setup of the dataset.

3.3. Framework for evaluating DARPA-2009

Dataset

In the first study of its kind, we propose a framework for analysing the DARPA-2009 dataset and determining its pcap files to demonstrate to what extent it is capable of evaluating the accuracy and performance of a new DE technique assessing NIDSs. Figure 3.1 presents the framework’s architecture which consists of pcap transformation and labelling processes, a proposed statistical feature selection technique, the preparation of training and testing sets and an experimental evaluation using some existing ML algorithms [85].

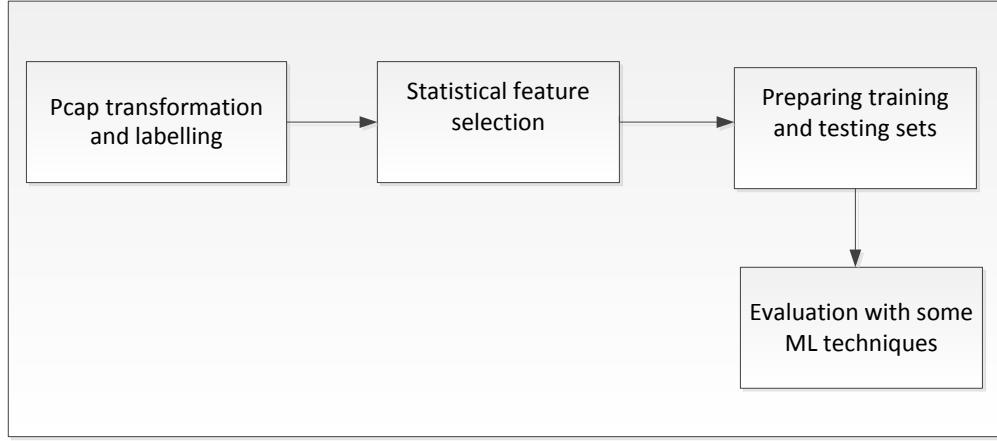


Figure 3.1: Framework for analysis of samples selected from DARPA-2009 dataset

3.3.1. Pcap Transformation and Labelling

We analysed the first 30 pcap files, which have the majority of security events in this dataset, to determine the potential of these events to measure the dataset's reliability for evaluating anomaly detection approaches. They were inspected by the *tcptrace* tool [176] developed by Shawn Ostermann at Ohio University for analysing and inspecting TCP dump files to extract the features of the TCP protocol which can discriminate between normal and abnormal observations. It can handle multiple input files produced by several packet-capture tools, including *tcpdump*, *snoop*, *etherpeek* and *WinDump*. It creates information about each network connection, for example, received re-transmissions, elapsed times, window advertisements, round-trip times and throughputs. It was installed on Linux Ubuntu 14.0.4 to generate the features of the TCP protocol using the command

- ***tcptrace -CSV -l input.pcap > output.CSV***

The features were logged in the SQL server 2008 [177] to establish a dataset containing them and their class labels, with the labelling performed by matching

the ground-truth records with those generated from tcptrace. As discussed above, the ground-truth spreadsheet includes all attack records for the 10 days covered in the DARPA-2009 dataset. To tag each record in the SQL database as either normal or attack, we used an *update* command to assign 0 for each normal record and 1 for each attack record as

- *update table1 set class=1, subclass='attack type' where srcip='0.0.0.0' and dstip='0.0.0.0' – attack record*
- *update table1 set class=0 where srcip='0.0.0.0' and dstip='0.0.0.0' – normal record*

This code updates the label columns, i.e., the class and sub-class of each record. The class label column contains 1 if a record is an attack, otherwise 0, and, if it is 1, the sub-class label column indicates the attack type, such as a phishing email, DoS or DDoS. These conditions are satisfied when srcip and dstip (i.e., the source and destination IPs, respectively) generated from the tcptrace equal their counterparts in the ground-truth spreadsheet.

3.3.2. Proposed Statistical Feature Selection

Statistical feature selection is a type of filtering approach used to eliminate unnecessary and duplicate features [178, 179]. Its goal is to reduce the number of features in a given dataset and optimise the selection of the relevant features most likely to improve the accuracy and performance of ML algorithms. In the domain of anomaly detection, these algorithms play a significant role in enhancing the efficiency and effectiveness of building a lightweight NADS by identifying the maximal number of malicious patterns with the minimal number of FPRs in a relatively low processing time [101].

The features generated are flow-based ones which can be easily extracted from packet headers without the need to rely on an inspection of the full packet which

might be encrypted. Before choosing the relevant features for building DE algorithms, the identifiers of source/destination IP addresses and ports are excluded as they could result in a high false alarm rate (FAR) if some are used for attack and normal events while simulating the dataset. The flow-based features are taken from the inter-arrival times, inter-packet lengths and directions of flows which relate to the bi-directional streams of network packets between two hosts (i.e., a client and server). In real-time network environments, generating such features from packet headers implies a low computational cost. The packet header denotes the portion of the IP data which is followed by the packet payload/body and involves IP addresses as well as other data needed for them to reach the intended destination [173]. The major reason for using statistical features is that they are a possible means of accomplishing real-time detection because many NADSs use them to construct their patterns of normal profiles [180, 181].

These features are ranked based on a proposed Ranking Probability Feature Selection (RPFS) algorithm to select the relevant features which help in distinguishing between normal and attack observations using DE techniques. Algorithm 3.1 describes the steps in the RPFS which, firstly, counts the feature values of each class and computes their probabilities. Then, the highest probabilities of each class are matched with a predefined threshold to select their features. Finally, the highest ranked features are passed to some ML algorithms for evaluating the reliability of the DARPA-2009 dataset. The RPFS algorithm is implemented using the Visual Studio Business Intelligence 2008 tool [177], with its results provided below.

Table 3.2 presents pairs of the highest probabilities of each feature value and its class. In this experiment, a 90% threshold is used to select the top-ranked features. The RPFS method is designed to select all the possible features that can affect legitimate and anomalous records, as described in Table 3.3. To pass them

Algorithm 3.1 Ranking Probability Feature Selection (RPFS)

Input: numerical features, class label {0 for normal, 1 for attack}, threshold

Output: feature selection (F)

- 1: **for** (each class) **do**
 - 2: count (C) the values (V) of each feature by:
 - 3: $A_i = V_j = \{v_1, v_2, \dots, v_j\}$
 - 4: $C(V_i) = \{c(v_1), c(v_2), \dots, c(v_j)\}$
 - 5: compute the probability (P) of each value count by:
 - 6: $P_i = \{c(v_1)/j, c(v_2)/j, \dots, c(v_j)/j\}$
 - 7: select the highest probability of each class which matches the threshold
 - 8: select the features ($F = \{f_{11}, f_{12}, \dots, f_{ij}\}$) generated from step 3
 - 9: **end for**
-

Table 3.2: Sample of highest probabilities of normal and attack classes

Feature name	Value of class=0	Probability of class =0	Value of class=1	Probability of class =1
Mss_Requested_a2b	1460	98.21%	0	99.99%
Mss_Requested_b2a	1460	94.02%	0	90.92%
Min_Win_Adv_b2a	5888	93.84%	0	90.93%
Min_Win_Adv_a2b	5888	93.77%	32120	99.99%
Max_Win_Adv_b2a	5888	92.10%	0	90.92%
Avg_Win_Adv_b2a	5888	92.10%	0	90.92%
Actual_Data_Pkts_a2b	1	91.33%	0	99.99%
Pure_Acks_Sent_b2a	2	90.31%	0	99.94%
Unique_Bytes_Sent_a2b	28	91.82%	0	99.99%
Actual_Data_Bytes_a2b	28	91.82%	0	99.99%
Max_Win_Adv_a2b	64128	90.95%	32120	99.99%

directly to ML techniques, as almost all of them are of the numeric type, they are much easier to process than categorical features.

It is observed that the first 30 pcap files have the 6 types of attack events listed in Table 3.4, including 314206 DDoS and 26387 malware DDoS records, which compromise target systems by flooding them with many packets which corrupt their resources. It is also noted from this analysis that they do not have

Table 3.3: Descriptions of selected features

Feature name	Description
Mss_Requested_a2b	Requested maximum segment size from source to destination
Mss_Requested_b2a	Requested maximum segment size from destination to source
Min_Win_Adv_b2a	Smallest window advertisement sent from destination to source
Min_Win_Adv_a2b	Smallest window advertisement sent from source to destination
Max_Win_Adv_b2a	Largest window advertisement sent from destination to source
Avg_Win_Adv_b2a	Average window advertisement sent from destination to source
Actual_Data_Pkts_a2b	Number of packets containing amount of data from source to destination
Pure_Acks_Sent_b2a	Number of packets containing valid ACK from destination to source
Unique_Bytes_Sent_a2b	Number of bytes sent (excluding re-transmissions) from source to destination
Actual_Data_Bytes_a2b	Number of bytes (including re-transmissions) from source to destination
Max_Win_Adv_a2b	Largest window advertisement sent from source to destination

packet sizes, as reflected by the ‘*Actual_Data_Bytes_a2b*’ feature in Table 3.2. It is evident that this dataset does not have packet payloads (i.e., data about attack events) for these malicious behaviours which could confirm its credibility for evaluating novel DE approaches.

The 7 records for an event called ‘*scan/usr/bin/nmap*’ refer to some outside IPs that scan certain local IPs with certain ports. Also, there are two different sets of phishing emails, each consisting of a sequence of events of ‘*client compromise*’ and ‘*no precursor client compromise exfil/sams_launch_v*’ appearing in 6 records. Finally, 1 malicious event is called ‘*c2 + tcp control channel exfil - no precursor nc*’. Both these malicious and normal instances were divided into training and testing sets in order to measure the reliability of this dataset for evaluating NIDSs.

Table 3.4: Numbers of records of each attack type in first 30 pcap files

No. of records	Attack category
314206	DDoS
26387	Malware DDoS
7	scan /usr/bin/nmap
3	client compromise
3	no precursor client compromise exfil/sams_launch_v
1	c2 + tcp control channel exfil - no precursor nc

We can observe from the above analysis that the data distributions of attack and normal observations are unbalanced. As a consequence, ML algorithms are almost always biased towards some records which produce high FARs even if their detection rates are high. Although this unbalanced problem is natural for network data containing a majority of normal records and a minority of unusual observations, ML techniques alone cannot mitigate this problem. Several solutions to this challenge were suggested in [182] but their outcomes were not sufficiently accurate to apply in real networking. A plethora of research [183–185] has used statistical methods which could improve the efficiency and efficacy of applying them for network data.

3.3.3. Preparation of Training and Testing Sets for DARPA-2009 Dataset

The process of separating a dataset into training and testing sets is an essential step in assessing ML models. For the easy analysis and building of ML algorithms, the feature values of the first 30 pcap files are recorded in one table, with 408204 records for the normal class (0) and 340607 for the attack class, which is divided into training and testing sets at a ratio of 1:10, respectively, in order to efficiently learn different attacking patterns using the ML algorithms. Table 3.5 shows the numbers of records selected for the training and testing sets for each class, 367384 and 40820 normal and 306547 and 34060 attack, respectively. This division ensures that most of the data in the training phase correctly build the model with a smaller

Table 3.5: Numbers of records in training and testing sets

Number of records	Class	Type of set
367384	0	Training
306547	1	Training
40820	0	Testing
34060	1	Testing

proportion used for the testing phase to validate correctness [186]. More details of strategies for dividing a dataset are provided in Section 3.8.

3.3.4. Evaluation of Four ML Algorithms

ML algorithms are commonly used to establish DE approaches for identifying abnormal patterns from a profile constructed for anomaly detection. In a real-time anomaly detection methodology, when establishing a normal profile, attack data are not labelled whereas normal records are labelled. Supervised ML algorithms require attack-free data in the training phase but, as they cannot be obtained in a real network environment, supervised models are generally built offline using attack and normal observations and then run in either online or offline systems. The main drawbacks of supervised techniques are their requirements to obtain correctly labelled attacks to validate their models and frequently update the data with new attacks to ensure the credibility of detection. Another type of learning algorithms is unsupervised techniques which do not require attack patterns to build its model as it groups data points based on their distances or probabilities. For detecting network attacks, an unsupervised technique produces a higher FPR than a supervised technique [30] while it is clear that using either requires carefully setting their parameters to achieve the highest DR and lowest FAR.

We apply both supervised, the Naive Bayes (NB) [187], Decision Tree (DT) [188, 189] and Artificial Neural Network (ANN) [188, 189], and unsupervised, EM clustering [190], ML techniques using their default parameters provided by the SQL

Table 3.6: Confusion matrices of four techniques

(A) Confusion matrix of NB			(B) Confusion matrix of DT		
Predicted	Actual (0)	Actual (1)	Predicted	Actual (0)	Actual (1)
0	34040	8	0	34060	10
1	20	40812	1	0	40810

(C) Confusion matrix of ANN			(D) Confusion matrix of EM		
Predicted	Actual (0)	Actual (1)	Predicted	Actual (0)	Actual (1)
0	33746	7	0	33487	8
1	314	40813	1	573	40812

Server Business Intelligence Development Studio in [177] to evaluate the credibility of the sample of the DARPA-2009 dataset for detecting attacks. While the training and testing phases are implemented on the dataset, the confusion matrix of each algorithm is calculated, from which the DR, FPR, FNR and accuracy metrics are computed using the equations in Chapter 2, Section 2.4, to estimate the reliability of this dataset for evaluating new DE approaches. The confusion matrices, which show the numbers of actual records versus those predicted by a model, for the NB, DT, ANN and EM clustering algorithms are shown in Table 3.6.

Table 3.7 shows the evaluation metrics of the DR, FPR, FNR and accuracy of the four algorithms. The DT algorithm achieves the highest DR and accuracy (100 % and 99.9 %, respectively) and the lowest FAR (0.29 %). In contrast, the EM clustering technique produces the lowest DR and accuracy (97.6 % and 98.2 %, respectively) and highest FAR (0.86 %). The results from the four algorithms are approximately similar, providing relatively high rates of detection accuracy and low FARs.

The findings from this work are compared with those from four competing studies. The six terms used to conduct this comparative analysis, feature extraction tool, feature selection method, number of features, classification technique, accuracy and FAR, are shown in Table 3.8. Butun et al. [3] used the tcptrace tool

Table 3.7: Evaluation metrics of four algorithms

	NB (%)	DT (%)	ANN (%)	EM (%)
DR	99.2	100	98.1	97.6
FPR	0.23	0.29	0.3	0.23
FNR	0.49	0	0.07	0.63
Accuracy	99.3	99.9	99.5	98.2

to extract the TCP features of the KDD99 dataset as we provided in this study of the DARPA-2009 for the first-time analysis [85]. They applied some classification techniques to evaluate the generated features without a feature selection method. Their results were not as high as the findings using the feature selection method to select the relevant features. Although, in [191], the features of the KDD99 dataset generated using the DRO-IDS tool were applied using feature selection of the correlation coefficient to evaluate a specific DE approach, the outcomes of our study are better. Heba F. et al. [192] suggested a DT algorithm for evaluating the NSL-KDD dataset using a linear correlation feature selection method to select the highest-ranked features. Their experimental results were close to the results of this study, which have a 99% accuracy and 0.23% FAR.

Ultimately, existing ML algorithms can efficiently detect attacks in the DARPA-2009 dataset, with the results significantly improved by applying feature selection methods. As a result, the patterns in this dataset are easy to detect and very different from those in current network environments that have sophisticated patterns of normal and abnormal behaviours with modern network protocol standards and higher network speeds. This was the motivation for generating the UNSW-NB15 dataset that has the norm of contemporary real network data.

3.4. Generation of UNSW-NB15 Dataset

Because of their drawbacks discussed above and in Chapter 2, subsection 2.6.1, publicly available benchmark datasets are incapable of evaluating NIDSs against

Table 3.8: Comparative analysis of proposed framework and related studies

Authors	Dataset	Feature extraction tool	Feature selection method	No. of features	Classification technique	Accuracy (%)	FAR (%)
Lazarevic et al. [3]	KDD99	Tcptrace	Not applicable	25	LOF	78.7	4
					ANN	78.9	9
					Mahalanobis	52.6	8
					Unsupervised SVM	84.2	12
Joffroy et al. [191]	KDD99	Bro-IDS	Correlation coefficient	41	PCC-R	78.7	3.2
Heba F. et al. [192]	NSL-KDD	Bro-IDS	Linear correlation	17	DT	99	0.03
Tan et al. [36]	KDD99	Bro-IDS	Multivariate correlation analysis	41	Unsupervised of MCA	95.2	1.26
Our study [85]	DARPA-2009	Tcptrace	Ranking probability	11	NB	0.993	0.07
					DT	0.999	0.02
					ANN	0.995	0.03
					EM	0.982	0.08

current network behaviours. However, designing a new and more dynamic dataset is a big challenge. It should have new normal and anomalous network traffic that can be described as a ‘big data’ problem which has to behave as realistically as possible to ensure its credibility for evaluating NIDSs.

3.4.1. Configuration of UNSW-NB15 Dataset Testbed

In this section, the configuration of the testbed network and all the processes involved in generating the UNSW-NB15³ dataset are presented. The current large

³A portion of this study has been published in:

- Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.

volume and high speed of network traffic demand a sophisticated tool capable of monitoring and testing them to ensure a satisfactory ongoing network performance and must have a significant amount of space and power to process IT infrastructures in real time. The IXIA's PerfectStorm ONE product [193] handles enterprise-scale applications and security simulations for large organisations. It can simulate millions of real-world normal and malicious traffic to test and validate the infrastructure of an entire network environment and handle 1 to 10 Gbps (Gigabytes per second) of network traffic. Its functionality incorporates portability, performance and cost-efficacy criteria which permit an organisation to accomplish enterprise-scale processing anywhere at any time to optimise its IT infrastructure [193]. The key features of this product are that it:

- is an enterprise-scale and security-testing tool that can be used anywhere and at any time;
- has a unified architecture which supports the BreakingPoint and IxLoad programs;
- reflects high performances with hardware-based acceleration for SSL and IPsec protocols;
- offers flexible interfaces with dual-speed support; and
- is a compact form factor with relatively low power requirements and high performance.

PerfectStorm ONE is a commercial product which provides traffic generation and a strike pack of network security and malware intrusions in a single rack-mountable device. It has the capability to analyse a broad range of network segments and protocols that supports generating traffic for more than 150 web applications, such as Skype, Facebook and Google services, and can concurrently simulate many

- <https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/>

user activities to offer a realistic network traffic scenario. It includes more than 4,500 live security and 28,000 live malware attacks taken from the Common Vulnerabilities and Exposures (CVE) [194] directory installed using multiple evasion techniques. It can achieve high scalability, as it is capable of configuring large network topologies involving hundreds of thousands of clients that can be simulated in a single device [195]. Therefore, we configured a network architecture applied to a large-sized business environment which includes many servers, a wide variety of hosts with different applications and services, routers, a packet sniffer (tcpdump) and a firewall system for generating the raw network packets of the UNSW-NB15 dataset, and installed it in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [196].

As shown in Figure 3.2, the IXIA traffic generator was configured by three virtual servers. Servers 1 (59.166.0.0) and 3 (149.171.126.0) were used to generate normal and server 2 (175.45.176.0) malicious network traffic. These servers and the traffic generator were interconnected by two virtual interfaces with IP addresses of (10.40.85.30) and (10.40.184.30) to link the connected private and public network devices. The servers were joined by hosts using two routers, each of which had two interfaces for linking with clients and capturing packets while running the simulation. The IP addresses of Router 1 were (10.40.85.1) and (10.40.182.1) and those of Router 2 (10.40.184.1) and (10.40.183.1). A firewall was connected to the two routers to filter the emitting traffic which showed that security events in the UNSW-NB15 dataset could not be detected by traditional signature-based tool [42].

As these security events need a NIDS as a possible means of identifying them, the tcpdump tool [197] was installed on Router 1 to extract the pcap files while the simulation was running. It is an excellent command-line packet analyser for capturing network traffic which includes the packet contents of a network interface associated with their time stamps. A pcap file is a common network capture and

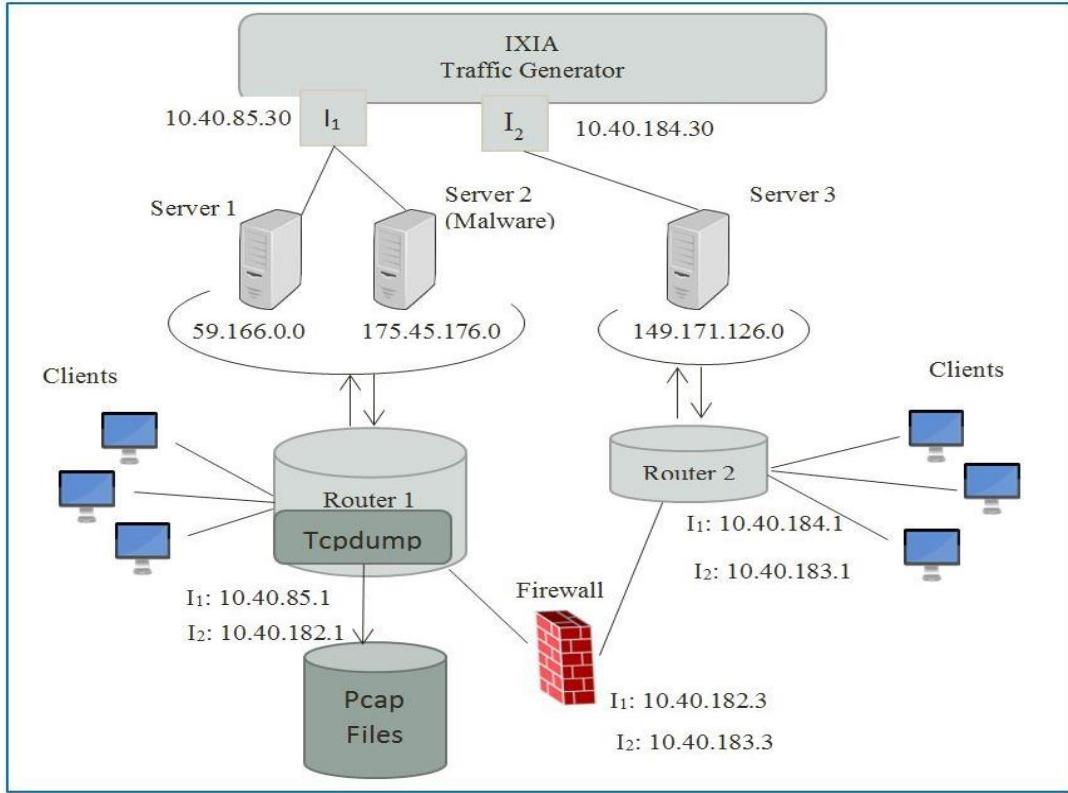


Figure 3.2: Architecture of UNSW-NB15 testbed network

exchange format which can be identified by WinPcap for Windows and LibPcap for UNIX platforms. It is a binary file comprising global and packet headers, with the former containing the time-zone indicator of packet sniffing and the latter information about the layers of the OSI Reference Model [197].

IXIA products simulate all anomalous events on the CVE website [194] which is a dictionary of publicly existing security events with its index connected directly to IXIA products to reflect real network threats. This simulation was implemented on 22nd January and 17th February 2015 in order to compare the effect of the number of attacks that might penetrate computer networks at particular times. On 22nd January, the first simulation was executed by capturing 50 GB with the configuration for generating one attack per second while the other, on 17th February, elicited 50 GB for ten attacks per second. More details of the settings of the configuration environments are provided in the UNSW-NB15 reports [32].

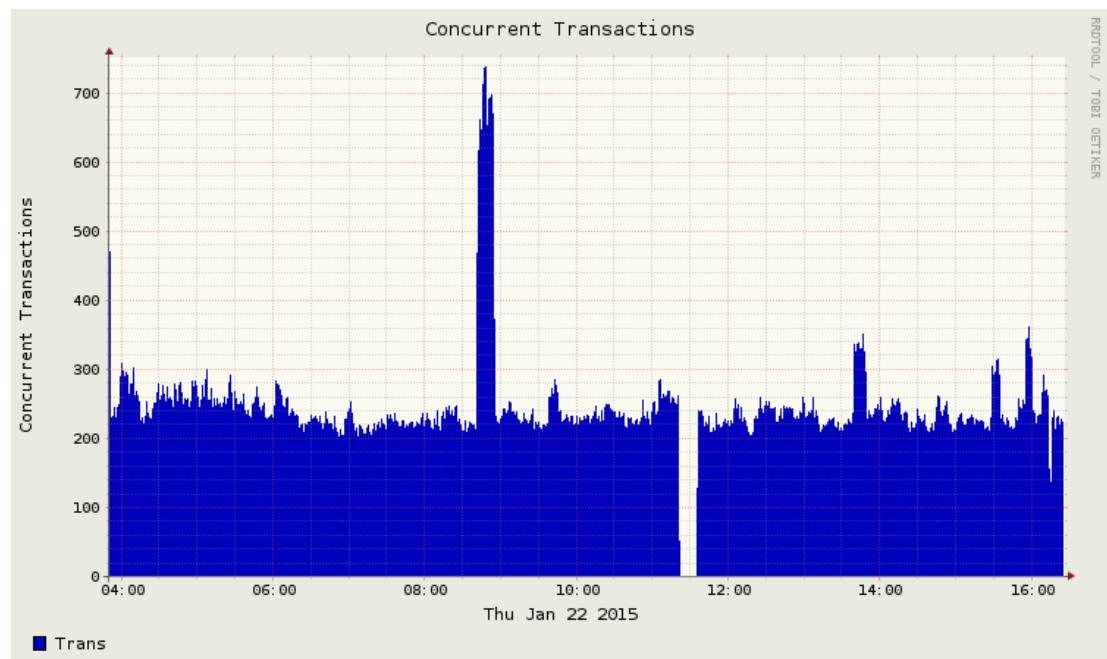
3.4.2. Network Traffic Analysis

An examination of the network traffic in this dataset demonstrates the occurrences of network flows over time, with their concurrent transactions indicating the amount of network data emulated over the simulation periods. Figure 3.3 shows the number of Kbytes on the y-axis and time on the x-axis sniffed over the simulation periods, with Figure 3.3 (A) depicting the synchronised transactions that took place over 16 hours on 22nd January 2015 and Figure 3.3 (B) those simulated over 15 hours on 17th February 2015. It is clear in both figures that approximately 280 Kbytes per second were transmitted between network nodes over the simulation periods.

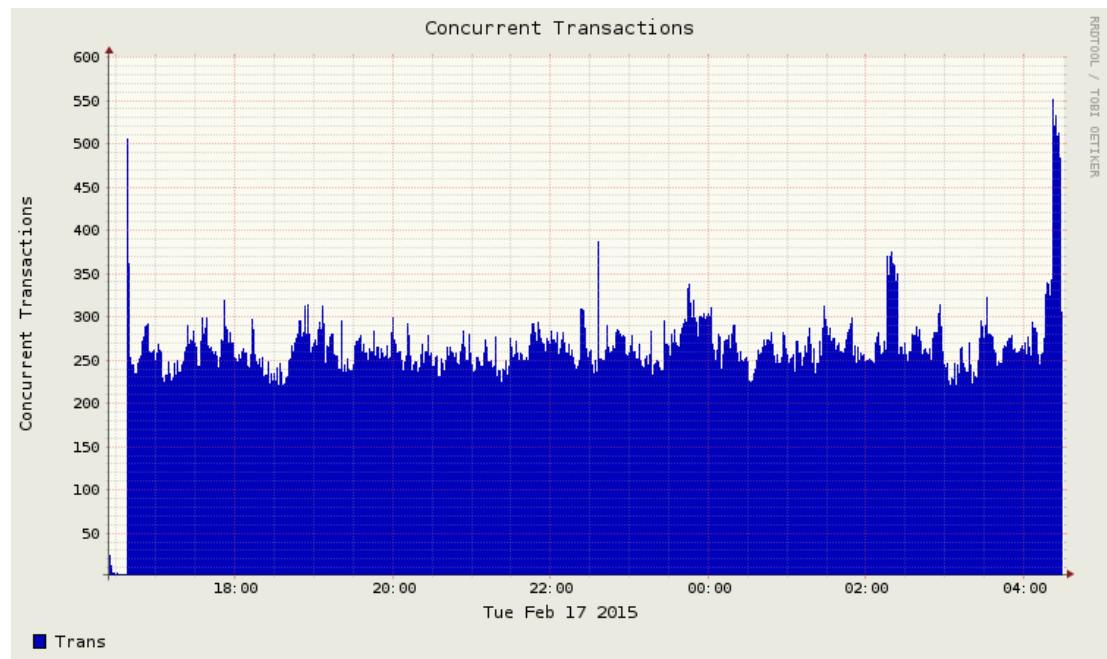
The statistics of this dataset are computed in terms of the numbers of flows, source and destination bytes, source and destination packets, protocol types, normal and attack observations, and unique source/destination IP addresses (Table 3.9). It is vital to record these statistics in order to demonstrate the extent of the complexity of this configuration environment. The functionalities of these protocols and services are explained in Appendix A.

3.5. Framework for generating UNSW-NB15 Features

This section describes a proposed framework for extracting the features of the UNSW-NB15 dataset from the raw network packets, i.e., the format of the pcap files. As shown in Figure 3.2, the tcpdump tool is used to save these files which have well-known network capture formats. However, NIDSs' classification algorithms cannot read these files as input to build their models for distinguishing between



(A)



(B)

Figure 3.3: Concurrent transactions of flows over simulation periods

Table 3.9: Statistics of UNSW-NB15 dataset

Statistical term		16 hours on 22nd Jan.	15 hours on 17th Feb.
No. of flows		987,627	976,882
Src_bytes		4,860,168,866	5,940,523,728
Des_bytes		44,743,560,943	44,303,195,509
Src_pkts		41,168,425	41,129,810
Dst_pkts		53,402,915	52,585,462
Protocol types	TCP	771,488	720,665
	UDP	301,528	688,616
	ICMP	150	374
	Other	150	374
Labels	Normal	1,064,987	1,153,774
	Attack	22,215	299,068
Unique	Src_ip	40	41
	Dst_ip	44	45

legitimate and anomalous observations. Therefore, to generate a set of useful information (features) as the input to classification algorithms, these files have to be properly inspected and analysed.

The framework for creating these features is presented in Figure 3.4. Firstly, some features are generated by the Argus [198] and Bro-IDS [199] tools and it is vital that a ‘big data’ analysis technology is used to make the processing of these files relatively easy. As the terms of big data, Volume, Velocity and Variety, can be satisfied in this dataset, as described in Section 3.7, these features are logged in a database using the MYSQL cluster CGE technology [200] which can handle a highly scalable database established for a distributed architecture to read and write dense workloads that can be accessed by SQL and NoSQL APIs. It can also support disk-based tables and enhanced memory, automatic data splitting with load balancing, and the addition of multiple nodes to a running cluster for processing big data online. Its architecture is almost the same as those of the Hadoop technologies [201] which are the most common for handling big data offline.

Secondly, the features of both tools are matched using the same source-destination IP addresses and ports as well as protocols to ensure that they correspond correctly. Also, new features created from the matched features show network flow

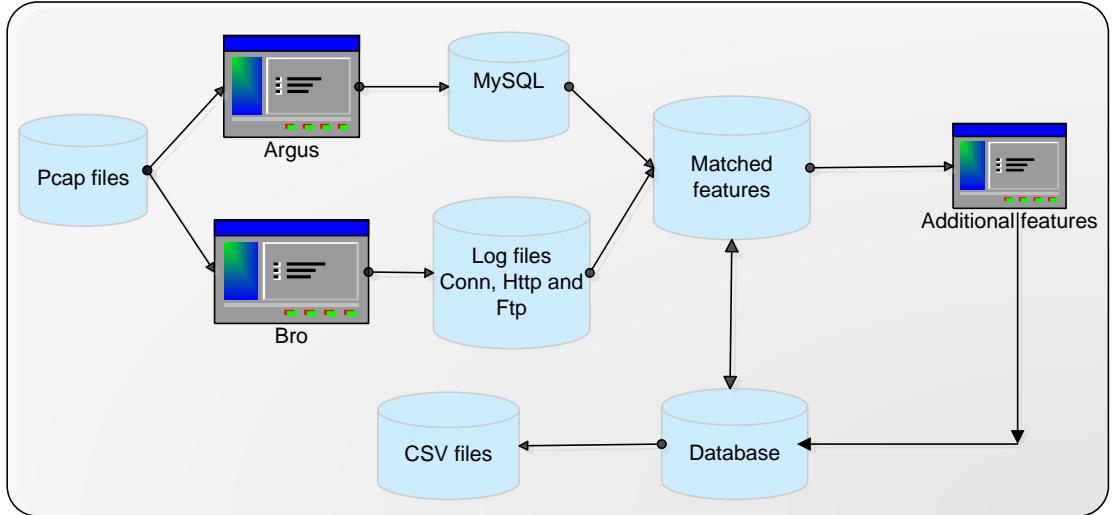


Figure 3.4: Framework for creating features of UNSW-NB15 dataset

transactions, such as source-destination IPs and source-destination ports, for each connection of 10/100 records. These features can help the examination of network observations and easily identify low-footprint attacks, as explained in subsection 3.5.3.

All features are matched with the security events provided in [43] to be correctly labelled with their data observations. Ultimately, the 47 features with labels (i.e., attack categories and sub-categories) established from the pcap files are presented in Tables 3.10 to 3.15. Their values are then logged in CSV files to make learning and validating NIDSs easier. This framework is designed and implemented on Linux Ubuntu 14.0.4, with the additional features developed using the C# programming language.

A full packet inspection is applied to create packet- and flow-based features. Packet-based features relate to analysing the headers and contents of packets while

flow-based features are bidirectional streams between two hosts, i.e., client-to-server and server-to-client [173, 202], which can evade the packet encryption problem and be handled faster than packet-based features [203]. Therefore, applying a real-time NIDS relies on the flow features, most of which are explained in the description cells in Tables 3.12, 3.13 and 3.14. Details of this framework are described in the following subsections.

3.5.1. Features extracted using Argus tool

The Argus is an open-source auditing tool developed by Carter Bullard to analyse raw network packets and create network features from them. It consists of an Argus server, which retrieves packets received by a network interface(s) available on a device and then converts them into a binary format that denotes network flows, and Argus clients which include multiple scripts that can read the binary flows executed by the Argus server and save them directly in a database [198]. The features of this dataset are generated from the following commands and are explained in subsection 3.5.3.

- `~$./rasqlinsert -M cache -m none -r /home/admin1/out/1.argus -w mysql://argusUser:root@localhost /argusData/table1 |-s -Z ltime +spkts +dpkts +sbytes +dbytes +rate +sttl +dttl +sload +dload +sloss +dloss +sintpkt +dintpkt +sjit +djit +swin +stcpb +dtcpb +dwin +tcprrt +synack +ackdat +smeansz +dmeansz`

This command indicates that the pcap files are transformed to binary ones using the Argus server, with features such as *ltime* and *spkts* extracted and saved in a MYSQL database.

3.5.2. Features extracted using Bro-IDS tool

Bro-IDS is an open-source tool used to efficiently analyse network traffic. It is also a security monitor which examines network traffic against anomalous patterns and

Table 3.10: Flow features

No.	Name	Type	Description
1	Srcip	Nominal	Source IP address
2	Sport	Integer	Source port number
3	Dstip	Nominal	Destination IP address
4	Dsport	Integer	Destination port number
5	Proto	Nominal	Transaction protocol

creates log files to record multiple network connections, including all records of connections available on LANs, application-layer transcripts, such as DNS, and HTTP services [199]. It is configured to create CONN, HTTP and FTP log files from the pcap files of the UNSW-NB15 dataset based on the command

- `~$ bro -r /home/1.pcap`

The CONN file includes all connections seen in the pcap files, the HTTP file all HTTP requests and replies, and the FTP file all the events of an FTP service. The files obtained from the Argus and Bro-IDS tools are stored in the MYSQL database for matching with these generated features via the flow features in Table 3.10.

3.5.3. Matched features

The matched features of both tools are classified in the three groups, basic, content and time, given in Tables 3.11, 3.12 and 3.13, respectively. The basic features encapsulate all the attributes elicited from TCP, UDP and IP connections which examine the essential information of packet flows. The content features include attributes which can be extracted from the information of a TCP protocol and services, such as HTTP. The time features contain attributes computed according to a time period which can help to identify malicious patterns as they properly examine network behaviours, as explained below in the statistical evaluation of this dataset.

Table 3.11: Basic features

No.	Name	Type	Description
6	State	Nominal	State of dependent protocol, such as ACC, CLO, CON, ECO, ECR and FIN
7	Dur	Float	Total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	Sttl	Integer	Source to destination time-to-live value
11	Dttl	Integer	Destination to source time-to-live value
12	Sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	Nominal	E.g., HTTP, FTP, SMTP, SSH, DNS and IRC
15	sload	Float	Source bits per second
16	dload	Float	Destination bits per second
17	spkts	Integer	Source to destination packet count
18	dpkts	Integer	Destination to source packet count

Table 3.12: Content features

No.	Name	Type	Description
19	Swin	Integer	Value of source TCP window advertisement
20	Dwin	Integer	Value of destination TCP window advertisement
21	Stcpb	Integer	Source TCP base sequence number
22	Dtcpb	Integer	Destination TCP base sequence number
23	Smeansz	Integer	Mean of flow packet sizes transmitted from source
24	Dmeansz	Integer	Mean of flow packet sizes transmitted by destination
25	trans_depth	Integer	Pipelined depth into connection of HTTP request/response transaction
26	res_bdy_len	Integer	Actual uncompressed content size of data transferred from server's HTTP service

Table 3.13: Time features

No.	Name	Type	Description
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	Start time
30	Ltime	Timestamp	Last time
31	Sintpkt	Float	Source inter-packet arrival time (ms)
32	Dintpkt	Float	Destination inter-packet arrival time (ms)
33	Tcprtt	Float	Round-trip time of TCP connection setup computed by the sum of 'synack' and 'ackdat'
34	Synack	Float	TCP connection setup time between SYN and SYN_ACK packets
35	Ackdat	Float	TCP connection setup time between SYN_ACK and ACK packets

3.5.4. Additional Features

Using the matched features, twelve additional features were created to estimate the relationships among them that could identify stealth attacks, as suggested in [36, 204, 205]. In Table 3.14, they are categorised as either general-purpose features, which consist of a set of attributes generated from the relationships of flow identifiers, or connection ones which include network connection events that scan hosts over a specific time, for instance, one scan a minute, hour or even day (s), with these kinds of features generated for each 100 network connections in order to help the detection of network adversaries [69]. The procedures for creating the additional features are elaborated below.

Algorithm 3.2 describes the steps for creating the `is_sm_ips_ports` feature using the flow identifiers (`flw`), data records (`dr`) and flag (`is_sm_flg`) as inputs. If the source and destination IP addresses are the same and the source and destination ports equal to any record, the `is_sm_flg` attribute is assigned 1, otherwise 0, a feature that can help to identify some sorts of DDoS attacks, such as land strikes [206].

Table 3.14: Additional generated features

No.	Name	Type	Description
General-purpose features			
36	is_sm_ips_ports	Binary	If the source (1) equals the destination (3) IP addresses and port numbers (2) and (4) are equal, this variable takes the value 1 else 0
37	ct_state_ttl	Integer	A number of each state (6) according to a specific range of values for the source's/destination's times-to-live (10) (11)
38	ct_flw_http_mthd	Integer	A number of flows obtained using methods such as Get and Post of an HTTP service
39	is_ftp_login	Binary	If the FP session is accessed by a user's credential, then 1 else 0
40	ct_ftp_cmd	Integer	A number of flows using a command in a FTP session
Connection features			
41	ct_srv_src	Integer	A number of connections containing the same service (14) and source address (1) in 100 connections according to the last time (30)
42	ct_srv_dst	Integer	A number of connections containing the same service (14) and destination address (3) in 100 connections according to the last time (30)
43	ct_dst_ltm	Integer	A number of connections containing the same destination address (3) in 100 connections according to the last time (30)
44	ct_src_ltm	Integer	A number of connections containing the same source address (1) in 100 connections according to the last time (30)
45	ct_src_dport_ltm	Integer	A number of connections containing the same source address (1) and destination port (4) in 100 connections according to the last time (30)
46	ct_dst_sport_ltm	Integer	A number of connections containing the same destination address (3) and source port (2) in 100 connections according to the last time (30)
47	ct_dst_src_ltm	Integer	A number of connections containing the same source (1) and destination (3) address in 100 connections according to the last time (30)

Algorithm 3.2 Creating is_sm_ips_ports feature

Input: [flw \leftarrow (srcip, sport, dstip, dsport), dr \leftarrow data records, is_sm_flg \leftarrow 0]

Output: [flw, is_sm_flg]

```
1: for (sm_flg in dr) do
2:   if (flw.srcip= flw.dstip && flw.sport==flw.dstip) then
3:     is_sm_flg  $\leftarrow$  1
4:   else
5:     is_sm_flg  $\leftarrow$  0
6:   end if
7: end for
```

Algorithm 3.3 presents the procedure for generating the ct_state_ttl feature, with the data records (dr) and counter (ct_state_ttl) provided as inputs. The time-to-live values of a source (sttl) and destination (dttl), and the state occurring in the dataset are matched with particular rules. If any of these rules are fired, any of 6 distinct values will be assigned to the ct_state_ttl attribute, a feature which shows that the ttl values have more real attack behaviours than those in the KDD99 dataset, as stated in [78].

Algorithm 3.4 presents the steps for creating the ct_flg_http_mthd feature using the data records (dr), flow HTTP methods (flw_http_mthd) and counter (ct_flg_http_mthd) as inputs, with the method attribute extracted from the HTTP log file. If a flow HTTP method equals the next included record, the counter increments by 1, otherwise remain as 0. This feature can help to identify some HTTP attacks, such as HTTP flooding ones [207].

Algorithm 3.5 describes the steps for generating the is_ftp_login feature using a FPT session (ftp_login), dr and flag (is_ftp_login) as inputs, with the user and password attributes elicited from the FTP log file. If the service attribute is

Algorithm 3.3 Creating ct_state_ttl feature

Input: [dr \leftarrow data records, ct_state_ttl \leftarrow 0]
Output: [ds.id, ct_state_ttl]

```
1: for (each record in dr) do
2:   if (ds.sttl == (62 || 63 || 254 || 255) && ds.dttl == (252 || 253) && ds.
      State == FIN) then
3:     ct_state_ttl=1
4:   else if (ds.sttl == (0 || 62 || 254) && ds.dttl == (0) && ds. State ==
      INT) then
5:     ct_state_ttl=2
6:   else if (ds.sttl == (62 || 254) && ds.dttl == (60 || 252 || 253) && ds.
      State == CON) then
7:     ct_state_ttl=3
8:   else if (ds.sttl == 254 && ds.dttl == 252 && ds. State == ACC) then
9:     ct_state_ttl=4
10:  else if (ds.sttl == 254 && ds.dttl == 252 && ds. State == CLO) then
11:    ct_state_ttl=5
12:  else if (ds.sttl == 254 && ds.dttl == 0 && ds. State == REQ) then
13:    ct_state_ttl=6
14:  else
15:    ct_state_ttl=0
16:  end if
17: end for
```

Algorithm 3.4 Creating ct_flw_http_mthd feature

Input: [flw_http_mthd \leftarrow (scrip, dstip, sport, dsport, method), dr \leftarrow data
records, ct_flw_http_mthd \leftarrow 0]
Output: [flw_http_mthd, ct_flw_http_mthd]

```
1: for (flw_http_mthd in dr) do
2:   if (flw_http_mthd == next (flw_http_mthd) && ds. method != ' ') then
3:     ct_flw_http_mthd ++
4:   else
5:     ct_flw_http_mthd  $\leftarrow$  0
6:   end if
7: end for
```

FTP, and the user and password ones have values, is_ftp_login is denoted by 1, otherwise 0. This feature can assist in identifying failures of attempts to obtain a FTP login [208].

Algorithm 3.5 Creating is_ftp_login feature

Input: [ftp_login \leftarrow (scrip, dstip, sport, dsport, service), dr \leftarrow data records, is_ftp_login \leftarrow 0]
Output: [ftp_login, is_ftp_login]

```

1: for (ftp_login in d) do
2:   if (ftp_login.service == ftp && ds.user != ' ' && ds.password != '') then
3:     is_ftp_login  $\leftarrow$  1
4:   else
5:     is_ftp_login  $\leftarrow$  0
6:   end if
7: end for
```

Algorithm 3.6 presents the procedure for creating the ct_ftp_cmd feature using a FTP command (ftp_cmd), dr and counter (ct_ftp_cmd) as inputs, with the FTP command attributes extracted from the FTP log file using the Bro-IDS tool. If any FTP command corresponds to the next one and the command attribute has a value, ct_ftp_cmd increases by 1, otherwise remain as 0. This feature can be used to determine sensitive commands launched by an attacker [122].

Algorithm 3.7 shows the steps for establishing the ct_srv_src feature using source IPs (scrip), a service with its corresponding scrip (srv_src), dr and a counter (ct_srv_src) as inputs. Step 1 implies that a loop scans all the srv_src in the dr while step 2 contains another loop that reads each subsequent 100 records from the dr. In steps 3 and 4, if the srv_src equals the next one, the ct_srv_src increases by 1, else remains as 1 or, otherwise, 0 (step 5). The steps in this algorithm are applied to create the ct_srv_dst by replacing source IPs with destination ones.

Algorithm 3.6 Creating ct_ftp_cmd feature

Input: [ftp_cmd \leftarrow (scrip, dstip, sport, dsport, command), dr \leftarrow data records, ct_ftp_cmd \leftarrow 0]
Output: [ftp_cmd, ct_ftp_cmd]

```
1: for (ftp_cmd in dr) do
2:   if (ftp_cmd == next (ftp_cmd) && ftp_cmd.command != '') then
3:     ct_ftp_cmd++
4:   else if (ftp_cmd != next (ftp_cmd) && ftp_cmd.command != '') then
5:     ct_ftp_cmd←1
6:   else
7:     ct_ftp_cmd←0
8:   end if
9: end for
```

Algorithm 3.7 Creating ct_srv_src feature

Input: [srv_src \leftarrow (scrip, service), dr \leftarrow data records, ct_srv_src \leftarrow 0]
Output: [ct_srv_src, ct_srv_src]

```
1: for (srv_src in dr ) do
2:   for (each 100 record in dr) do
3:     if (srv_src == next (srv_src)) then
4:       ct_srv_src++
5:     else if (srv_src != next (srv_src)) then
6:       ct_srv_src←1
7:     else
8:       ct_srv_src←0
9:     end if
10:   end for
11: end for
```

Algorithm 3.8 presents the steps for constructing the ct_dst_ltm feature using the destination IPs (dstip) and last time (ltime) (feature number 30) (dst_ltm), dr and a counter (ct_dst_ltm) as inputs. The two loops read the dstip and ltime (dst_ltm) of each 100 records in the dr and, if they equal the next ones, the counter increases by 1, otherwise remains as 1. The steps in this algorithm are used to generate the ct_src_ltm by replacing destination IPs with source ones.

Algorithm 3.9 describes the creation of the ct_src_dport_ltm feature using

Algorithm 3.8 Creating ct_dst_ltm feature

Input: [dst_ltm \leftarrow (dstip, ltime), dr \leftarrow data records, ct_dst_ltm \leftarrow 0]
Output: [dst_ltm, ct_dst_ltm]

```
1: for (dst_ltm in dr) do
2:   for (each 100 record in dr) do
3:     if (dst_ltm == next (dst_ltm)) then
4:       ct_dst_ltm ++
5:     else
6:       ct_dst_ltm  $\leftarrow$  1
7:     end if
8:   end for
9: end for
```

source IPs (srcip), a destination port number (dsport), the last time (ltime) features stored in (src_dport_ltm), dr and a counter (ct_src_dport_ltm) as inputs. The two loops scan all the src_dport_ltm in the dr for each 100 records with respect to the ltime attribute. If a src_dport_ltm equals the next one, it is incremented by 1, else remains as 1 or, otherwise, the counter value is 0. The same procedure is used to construct the ct_dst_sport_ltm and ct_dst_src_ltm features using their particular attributes.

Algorithm 3.9 Creating ct_src_dport_ltm feature

Input: [src_dport_ltm \leftarrow (srcip,dsport,ltime), dr \leftarrow data records, ct_src_dport_ltm \leftarrow 0]
Output: [src_dport_ltm, ct_src_dport_ltm]

```
1: for (src_dport_ltm in dr) do
2:   for (each 100 record in dr) do
3:     if (src_dport_ltm == next (src_dport_ltm)) then
4:       ct_src_dport_ltm ++
5:     else if (src_dport_ltm != next (src_dport_ltm)) then
6:       ct_src_dport_ltm  $\leftarrow$  1
7:     else
8:        $\leftarrow$  0
9:     end if
10:   end for
11: end for
```

These additional features could be applied to recognise multi-flow attacks. The theory behind using a flow watermark technology to trace current network traffic between flows is that a network flow is uniquely recognised by content-independent manipulations and, if sets of flows have the same pattern, they can be considered connected. Watermarking is a new approach that takes less computational time than traditional active monitoring techniques. As a powerful watermark is scalable, robust to packet losses and unseen, it is a remarkable alternative means of identifying correlations of the flows and flow patterns in anonymous communication systems [209]. This technology is used to activate anonymous communication systems by encrypting network traffic in order to evade attempts to exploit attacks [209] by identifying correlations of the flows and then making the anonymous devices accountable. It is observed that this technology achieves a high DR and low FPR using such additional features [210].

3.5.5. Labelling Process

The labelling process relates to associating data instances with their class labels which denote determining whether they are legitimate or anomalous. It is vital that the labelled data are accurate and represent all types of network behaviours to ensure the fidelity of a dataset for effectively and efficiently evaluating NIDSs. Obtaining a labelled set of malicious data instances containing all possible kinds of malicious patterns is more complex than determining labels for normal patterns and, also, malicious behaviours are often dynamic in real networks. As this means that new attacks not involved in a DE training set could occur, an anomaly detection methodology has been proposed [69].

All the security events which occur while running the simulation are logged by the IXIA traffic generator and used to label the observations in the UNSW-NB15 dataset. They are saved in two reports for the two days of the simulation, as published in [43], and then transformed into a ground-truth table to enable easier matching of all data instances created in the MySQL dataset, as described

Table 3.15: Labelled features

No.	Name	Type	Description
48	attack_cat	Nominal	The unsw-nb15 dataset has 9 malicious categories, i.e., Fuzzers for malicious activities, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode and Worm.
49	Label	Binary	0 for normal and 1 for attack records

in subsection 3.5.3. This table comprises 11 attributes, that is, the start time, last time, protocol type, source IP address, source port, destination IP address, destination port, attack category, attack sub-category, attack name and attack reference [43]. The SQL commands used to implement the labelling process are

- ***update table1 set class=1,subclass='attack type' where table1.srcip = gt_table. srcip and table1.sport = gt_table. sport and table1.dstip = gt_table. dstip and table1.dsport= gt_table.dsport and table1.proto = gt_table.proto and table1.stime = gt_table.stime and table1.ltime = gt_table.ltime - attack record***
- ***update table1 set class=0 where class is NULL – normal record***

The first command demonstrates the labelling of the attack instances whereby, if the attributes in *table1* (data instances) are equal to those in the *gt_table* (ground-truth table), they are labelled malicious and the rest of the records normal. It is worth mentioning that the labelling process is wisely performed to ensure the credibility of using this dataset to assess NIDSs. Table 3.15 lists the labelled attributes contained in this dataset with *attack_cat* (the attack categories) and label of each instance described.

3.5.6. UNSW-NB15 Security and Malware Events

The UNSW-NB15 dataset has a wide variety of observations of normal as well as network security and malware attacks [29] [28] [32], as shown in Table 3.16, with each security event having its own scenario which interacts with a certain protocol and service. The security and malware events included in this dataset are categorised into the following 9 types.

1. **Fuzzers for malicious activities** – the IXIA traffic generator uses fuzzing-strike lists to determine security vulnerabilities that cause computer systems to crash. Fuzzing is defined as a technique for discovering weak points in an application, operating system or network by feeding them with a massive input of random data designed to crash them. It is often used as a tool for penetration testing to identify potential bugs in software and web applications/services. Fuzzers work effectively for attacks that can crash a computer system, such as a buffer overflow, cross-site scripting, DoS, SQL injection and format bugs, by mimicking their behaviours. They show that fuzzing behaviours are very different from normal vectors and adding them to the security events in this dataset can assist in determining how current normal patterns vary from abnormal vectors and validating new NIDSs, as shown by the differences in Figure 3.5.
2. **Analysis** – this is an attack method which breaches internet applications via ports (e.g., port scans), emails (e.g., spam) and web scripts (e.g., HTML files). In a port scan, an attacker launches a port-scan listener to inspect the ports open on a victim system and then sends several packets to the victim, varying the destination port each time. Its goal is to discover the services and operating systems running on this victim. Regarding spamming, electronic messaging systems are used to repeatedly send unsolicited messages (spam), in particular advertising, on the same website or email. Finally, a hacking

web script is a type of computer security vulnerability, for example, cross-site scripting (XSS) which attackers use to inject client-side scripts into web pages browsed by other users to bypass access controls.

3. **Backdoor** – this is a technique whereby an attacker bypasses a normal stealth authentication method to secure unauthorised remote access to a device. As it is executed in the background and concealed from users, it is difficult to detect. Once an attacker gains access to a device through a back door, it can potentially alter files, steal personal information, set up malicious software and eventually take control of a computer system; for example, install a keylogging program on a victim system which allows everything written, including passwords and user identity, to be seen. As this information is stolen, the accounts of victims are compromised and open to identity theft.
4. **DoS** – this is an attack which disrupts computer resources via memory to prevent authorised requests gaining access to a device. It frequently sends unnecessary messages asking the network or computers to authenticate requests that have invalid return addresses. The network or computers will not be capable of finding the return address of the attacker while sending the authentication approval which causes the computer system to wait before closing the connection. When the server closes the connection, the attacker sends more authentication messages with invalid return addresses and, as a result, stops users from accessing their services and applications, such as email, websites and online accounts. There are several basic types of DoS attack, for example, flooding a network to obstruct legitimate network traffic in order to prevent a particular person accessing a service. The difference between DoS and DDoS attacks is that the former uses a single connection to exploit a software vulnerability or flood a target with fake requests to exhaust computer resources (i.e., RAM and CPU) whereas the latter is

launched from many connected devices distributed across the Internet and causes the victim system to stop working by sending huge volumes of traffic.

5. **Exploit** – this is a sequence of instructions that takes advantage of a glitch, bug or vulnerability and causes an unintentional or unsuspected behaviour to occur in a host or network. This behaviour normally involves anomalous events such as gaining control of a computer system or allowing a DoS attack or privilege escalation. Common types of exploits include a remote attack which penetrates the security vulnerability of a computer system to which it previously did not have access, a local access one which requires prior access to the vulnerable system and often increases the privileges of the user account implementing this exploit and a zero-day attack that is a malicious activity not yet discovered.
6. **Generic** – this is an attack perpetrated against all block-ciphers to cause a collision without respect to the configuration of the block-cipher, with common ones ‘collision’ and ‘pre-image’. The resistance of a hash function to these attacks relies firstly on the length (n) of the hash value. Regardless of how a hash function is established, an attack will usually be capable of finding ‘pre-images’ after trying approximately 2^n messages. A birthday attack is a type of cryptographic and collision one which exploits the mathematics of the birthday problem in the probability theory. It can be used to misrepresent communications between two or more systems and relies on a higher likelihood of collisions being found between random attack attempts and a fixed degree of permutations.
7. **Reconnaissance** - this can also be defined as a probe and is an attack which gathers information about a computer network to evade its security controls. An attacker typically runs a ‘ping sweep’, a network reconnaissance mechanism that uses a ping (an ICMP echo and echo-reply), in order to map a targeted network to determine its active IP addresses and responses. The attacker’s queries collect information about the victim, including its IP

address range, software version, running operating system, server location, types of devices and applications.

8. **Shellcode** – this is a small piece of code used to exploit the payload of a software vulnerability which typically launches a command shell from an attacker that could control the victim system. It is used to access a shell and give the highest privilege to users and developers, and possibly as an exploit payload to provide an attacker with command-line access to a computer system. This malicious code is injected into computer memory via exploiting the vulnerabilities of a stack or heap-based buffer overflow or formatting string attacks. Usually, a shellcode execution is triggered by overwriting a stack return address with the address of the injected shellcode. As a consequence, instead of a subroutine returning to the caller, it returns to the shellcode and spawns a shell.
9. **Worm** – this is a malware program which requires a user action to trigger it and spread it on a computer network by replicating itself based on security failures of the target computer. It causes harm to host networks by consuming bandwidth and overloading web servers. It contains ‘payloads’ which are pieces of code developed to trigger it to spread to damage a system’s resources, such as by stealing data and/or removing files.

The training and testing sets of the dataset are statistically analysed to demonstrate the dissimilarity between normal and abnormal observations, with the results depicted in Figure 3.5 normalised in the range of [0, 1] to be easier to plot. The regression analysis is applied to draw a smooth line between normal and abnormal records in order to show their relationship. When the data points of normal and attack records locate at the same line, this denotes that there is a strong correlation/similarity between these records, otherwise the records are not similar in their statistical behaviour. The graphs show that normal records clearly

Table 3.16: Class distributions in UNSW-NB15 dataset

Class type	Number of records
Normal	2,218,761
Fuzzers	24,246
Analysis	2,677
Backdoor	2,329
DoS	16,353
Exploit	44,525
Generic	215,481
Reconnaissance	13,987
Shellcode	1,511
Worm	174

deviate from abnormal records, with security events sometimes trending the same as normal records, as shown by the blue lines, although their common points are too low, averaging 0.1 - 0.3 % over all the data points.

3.5.7. File Formats of UNSW-NB15 Dataset

As previously mentioned, the raw network packets of the UNSW-NB 15 dataset were created by the IXIA traffic generator to simulate a hybrid of contemporary realistic legitimate and synthetic malicious activities. The source files in this dataset were provided in different formats, pcap, BRO, Argus and CSV, and ordered based on the dates of the simulations (22-1-2015 and 17-2-2015, respectively) [43]. The details of these simulations provided in reports [43] demonstrate that security events occurred while a simulation was running.

The CSV files in this dataset, which consist of 47 features with class labels and attack categories as stated in Section 3.5, were prepared for use in evaluating NIDSs. Four files, named *UNSW-NB15_1.CSV*, *UNSW-NB15_2.CSV*, *UNSW-NB15_3.CSV* and *UNSW-NB15_4.CSV*, contain both data records of normal and attack events [43]. Their data records are ordered according to the last-time feature

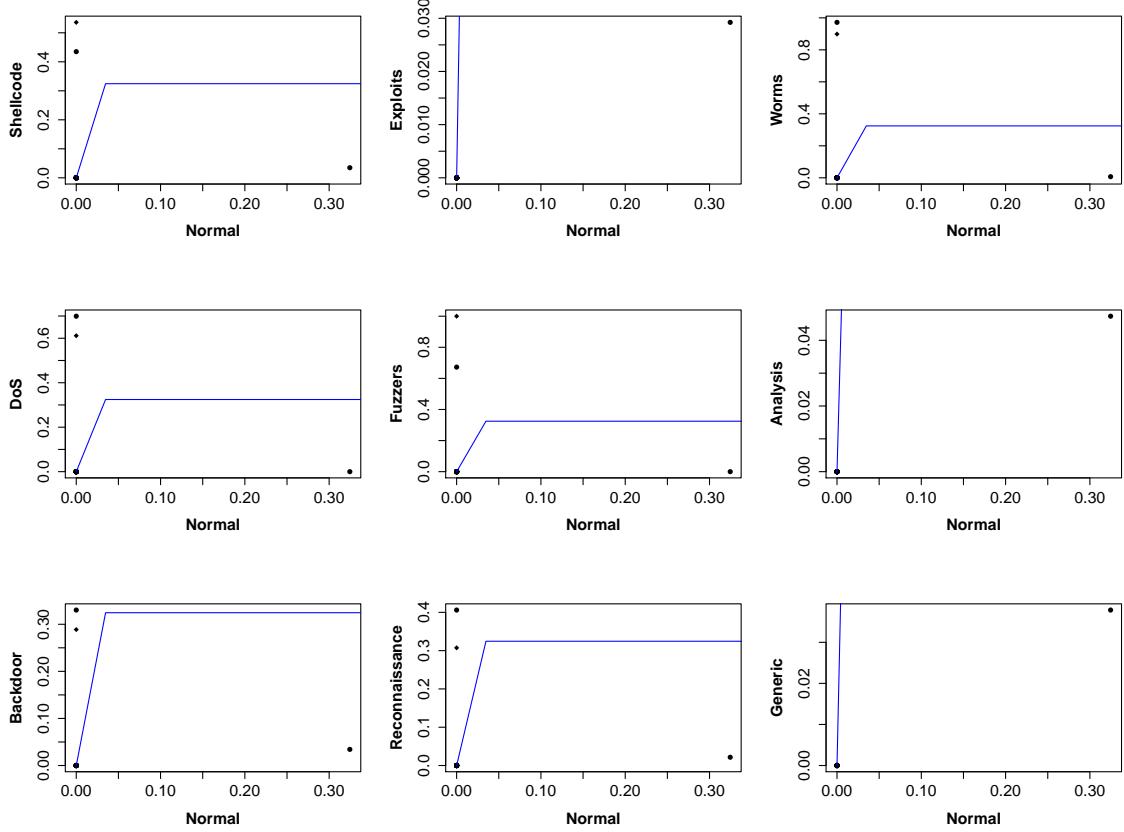


Figure 3.5: Statistical behaviours of normal and abnormal observations

(ltime) and the total number of records is 2,540,044, consisting of 700,000 in each of the first three and 440,044 in the fourth.

In [43], the ground-truth table, which is entitled *UNSW-NB15_GT.CSV*, has information about the security events, with their categories and sub-categories given in the *UNSW-NB15_LIST_EVENTS* file. Also, the portions of it proposed for training and validating DE approaches, the *UNSW_NB15_training-set.CSV* and *UNSW_NB15_testing-set.CSV* files, respectively, are described in Section 3.5.

3.6. Comparisons with Other Datasets

Many network datasets are available to the research community, as discussed in Chapter 2. A decent dataset should have a set of terms that includes a realistic

network configuration, realistic network traffic, labelled observations, total interactions and full packet captures as well as many malicious scenarios [82]. Table 3.17 presents a comparison of the most popular datasets, KDD99, NSL-KDD, CAIDA, DEFCON, ISCX, DARPA-2009 and UNSW-NB15.

All these datasets except DEFCON were simulated or established by configuring a realistic network environment while KDD99 and its enhanced version, NSL-KDD, do not involve a regular trace of the insertions of new dimensions of synthetic attack traffic in the testing phase. Although most datasets that depend on simulators often capture information about lower layers of the TCP/IP, the IXIA traffic simulator used to generate UNSW-NB15 can capture a wide range of segments from different protocols, such as TCP/IP, UDP, ICMP, DNS and HTTP. The labelling process requires a trusted ground-truth table to ensure that malicious observations are correctly tagged. The CAIDA, DEFCON and DARPA-2009 datasets are not tagged while the others are labelled. However, most do not publish their ground truth to ensure their reliability for assessing NIDSs. For this purpose, tagging of the UNSW-NB15 dataset is reported in [43].

The term ‘complete capture’ relates to the extracted traces containing all the information that could penetrate the privacy of users which can often be found in the application layer of the OSI model by analysing its services, such as DNS and HTTP. Almost all publicly available benchmark datasets eliminate any user’s information by removing some or all of the information in the packet payload, as shown in Table 3.17. The UNSW-NB15 dataset was simulated from a realistic environment and its files published in different formats without any information being removed, as discussed in Section 3.4. Although a network dataset should have a broad range of current legitimate and anomalous activities for efficiently evaluating NIDSs, most of the available datasets do not contain the current sophisticated malicious activities that require the design of an intelligent DE to define them because existing ML algorithms cannot efficiently detect them, as discussed in Chapter 2. In fact, as these datasets do not reflect current threads which have

Table 3.17: Comparisons of popular and UNSW-NB15 datasets

Datasets	Realistic network configuration	Realistic network traffic	Labelled observations	Total interaction capture	Full packet capture	Many malicious scenarios
KDD99	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i> ⁸
NSL-KDD	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i> ⁸
CAIDA	<i>True</i> ¹	<i>True</i>	<i>False</i>	<i>False</i> ⁵	<i>False</i> ⁴	<i>False</i> ²
DEFCON	<i>False</i>	<i>False</i> ⁵	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i> ⁸
ISCX	<i>True</i> ²	<i>True</i> ⁶	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i> ¹⁰
DARPA-2009	<i>True</i> ¹	<i>True</i> ⁵	<i>False</i>	<i>False</i> ⁵	<i>True</i>	<i>True</i>
Our dataset - UNSW-NB15	<i>True</i>	<i>True</i> ⁹	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i> ¹⁰
	1. Network configuration information not available 2. Basic captured network traces 3. No payload available; most simply reduced/summarised trace information 4. No payload available; in some packets, protocol, destination and flags deleted 5. Comprises no packet contents and no host or protocol information 6. Designed to include profiles of network information 7. Only malicious traffic 8. Does not reflect current trends 9. Contains a large number of protocols and services 10. Has modern security events and malware scenarios					

complex scenarios, the UNSW-NB15 dataset was designed to contain many normal and hostile observations for a new evaluation of a NIDS's efficiency.

As the KDD99 dataset is most commonly used to evaluate NIDSs, another comparison of it and the UNSW-NB15 dataset, with their numbers of sub-networks and unique IPs, simulations, data formats, attack types, feature extraction tools and numbers of extracted features, is provided in Table 3.18. The UNSW-NB15 dataset has the norm of current realistic network environments, including a larger number of sub-networks and unique IP addresses than the KDD99 dataset. The amount of data (around 100GB) generated during its 31-hour simulation is very much greater than that from the 5-week one of the KDD99 dataset (less than 1

Table 3.18: Comparisons of KDD99 and UNSW-NB15 datasets

Parameter	KDD99 dataset	UNSW-NB15 dataset
Number of sub-networks	2	3
Number of distinct IPs	11	45
Simulation	Yes	Yes
Duration of simulation	5 weeks	31 hours
Data formats	Tcpdump, BSM and data dump files	Pcap, BRO, Argus and CSV files
Attack types	Dos, Probe, U2R and R2L	Fuzzers for malicious activities, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode and Worm
Feature extraction tools	Bro-IDS	Argus, Bro-IDS and new scripts
Number of features extracted	41	47

GB) which indicates that the UNSW-NB15 dataset can be considered a big data problem, as described below. Also, its different file formats are provided to help researchers use it to produce new features and mechanisms in the network security field.

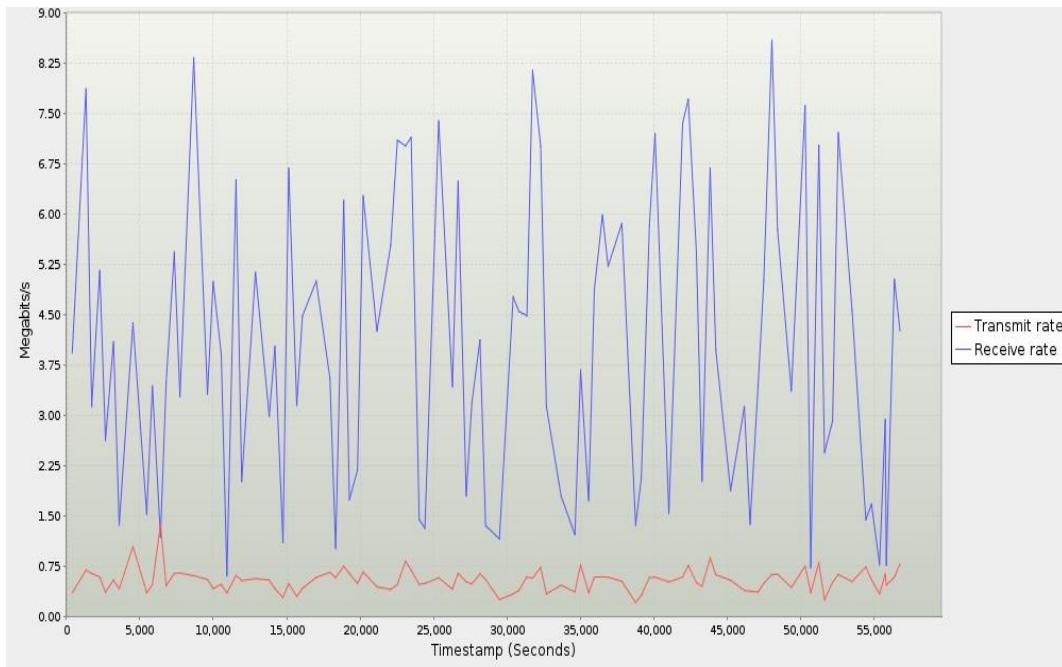
3.7. Big Data Properties in UNSW-NB15 Dataset

In this section, we discuss the UNSW-NB15 characteristics and how some challenges of security learning can be successfully evaluated using this dataset that has the properties of the ‘big data’ principle (see Chapter 2- subsection 2.6.1). Current Network datasets should have the big data terms of Volume, Velocity

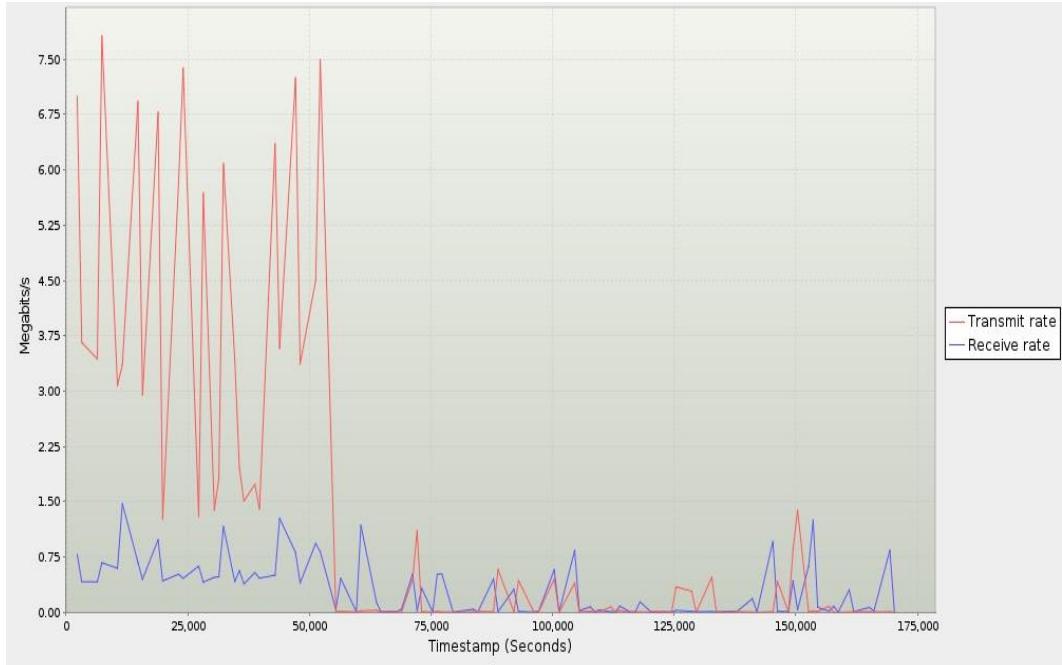
and Variety (3Vs) to ensure their credibility for training and validating new DE approaches installed over a real network with a large volume, high velocity and variety of network flows.

In the UNSW-NB15 dataset, the volume of network traffic is approximately 100 Gigabytes which consists of more than 2 million observations that require a large amount of storage and computing mechanisms with a high memory and CPU for their effective processing. In this regard, different tools and platforms are used to capture traces and extract the important features discussed above. Secondly, the velocity of network flows between a source and destination is, on average, 5-8.5 Megabytes per second, as stated in [43] and shown in Figure 3.6, which demonstrates that this dataset has high data rates across the Ethernets while sending and receiving traces which exactly mimic real network environments. It is worth mentioning that collecting traces of current network traffic is a big data problem for the design of a reliable NIDS because of the volume needed to store more information about network traffic, the velocity of collective flows at higher rates and the variety of information collected from different sources and high dimensions.

This dataset also covers a wide range of segments collected from various protocols and services during the simulation periods and logged in pcap files which were precisely determined to extract 47 features of several data types, i.e., numeric, timestamp and categorical, which could distinguish between normal and attack activities. Therefore, it has a wide variety of types with traces collected from different systems and has high dimensionality. The CSV files of the dataset were processed to ensure they contained clean data with no missing values or duplicated records in order to guarantee their veracity. Ultimately, as this dataset has the characteristics of the big data available in real network environments, it is valid for evaluating new NIDSs.



(A) 22nd January 2015



(B) 17th February 2015

Figure 3.6: Ethernet data rates transmitted and received over simulation periods

3.8. Dataset Splitting for Learning Techniques

In a learning methodology, splitting a dataset into training, validation and testing sets is essential for the training and validation of its algorithms. A training set is a set of instances for determining the learning classifier's parameters, a validation set is a set of instances used to adjust those parameters as much as possible and a testing set is a set of observations for evaluating the classifier's performance. Typically, when a dataset is separated into training and testing sets, most of the data are used for training [186]. In this case, instead of manually designing a validation set to correctly fit the parameters, cross-validation techniques are applied. Cross-validation is a model validation mechanism for measuring the extent to which a dataset is independent of observations, including in both its training and testing sets. Its goal is to fit the parameters of a predictive model as well as estimate its accuracy for validating new instances in the testing set [211].

In intrusion detection, a dataset is represented as a relational table (T) [182] containing a set of observations with their class labels (normal or attack). DE techniques train and validate these instances (I), each of which comprises features (F) with multiple data types (i.e., $\forall f \subset \{R \cup S\}$, where $\forall f$ indicates each feature in T , R real numbers and S strings). For this purpose, a portion of the UNSW-NB15 dataset is divided into an almost 60:40 ratio of training and testing sets, respectively. To avoid biasing the DE approaches and reflecting high FARs, these sets do not contain any redundant observations which ensure the credibility of the evaluations. The numbers of observations included in each category are listed in Table 3.19 which shows that there are 175,341 and 82,332 records in the training and testing sets, respectively, as reported in [43].

It is acknowledged that statistical approaches face the challenge of using different data types, i.e., numeric or nominal [212], and T can be represented as

Table 3.19: Distributions in portion of UNSW-NB15 dataset

Category	Training set	Testing set
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	583
DoS	12,264	4089
Exploit	33,393	11,132
Fuzzers for malicious activities	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worm	130	44
Total Records	175,341	82,332

multivariate data, as stated in Definition 3.1.

Definition 3.1: Let $I_{1:N} \in T, I_{1:N} = \{f_{ij} \in F\}, Y_{1:N} = \{c_i \in C\}$, where $i, j = 1, 2, \dots, N$. Assume that F is independently and identically distributed (i.i.d.), with $I_{1:N}$ and $Y_{1:N}$ defined as a column-vector matrix which indicates that a network dataset has a feature set statistically treated as multivariate data as

$$I_{1:N} = \begin{bmatrix} f_{11} & f_{12} & \dots \\ f_{21} & f_{22} & ij \end{bmatrix}, \quad Y_{1:N} = \begin{bmatrix} c_1 \\ c_i \end{bmatrix} \quad (3.1)$$

where I is the observations of T , Y the class label (C) of each I , N the number of observations and F the features of each I .

Proposition 3.1 is applied for multivariate data to transform all their feature types into numeric values as required by classification techniques, particularly statistical models.

Proposition 3.1: the same data type of numbers ($\forall F \subset \{R\}$) for features (F) is applied for ease of analysing and classifying network data. Each nominal feature (S) is assigned to a sequence of numbers ($\forall S \rightarrow R_{0:R}$), where $\{0 : R\}$ is a sequence of numbers) [213]; for example, the UNSW-NB15 dataset includes three

nominal features: protocols such as TCP and UPD; states such as CON and ACC; and services such as HTTP and FTP. Each feature is replicated by a sequence of numbers, for example, the protocol values of TCP=1 and UDP=2, and so on.

3.9. Complexity Analysis of UNSW-NB15 Dataset

The aim of analysing this dataset⁴ is to determine the extent to which its current normal and malicious patterns are complex. To achieve this target, a statistical analysis, feature correlations with and without class labels and evaluation of some existing ML algorithms, are considered. Firstly, the statistical analysis inspects the relationship between the records/observations and attributes/features which can determine the statistical characteristics of normal and suspicious observations and then the features that can precisely discriminate between them are obtained. Finally, five existing techniques are used to evaluate their capabilities for classifying these data in terms of accuracy and FARs. We use the proposed portion consisting of the training and testing sets because it comprises the majority of security events and malware as well as normal records in the dataset.

These sets are designed to be in the form of multivariate data, as in equation (3.1), namely TR_{IN} for the training set and TS_{IN} for the testing set. The first aspect is applied using three well-known statistical functions, the Kolmogorov-Smirnov (K-S) test [214, 215], multivariate skewness and kurtosis functions [216, 217] to examine the relationships and distributions of the TR_{IN} and TS_{IN} observations and features. As the values of these observations are not in a particular range that can help to interpret and exactly demonstrate the differences between normal and malicious records, we use the z-score function [218] to normalise them

⁴The work of this study presented in this chapter has been published in:
Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set." Information Security Journal: A Global Perspective (2016): 1-14.

within a specific interval, with the core theories and way of applying them on the two sets discussed below.

3.9.1. Z-score Function

The features of the (TR_{I_N}) and (TS_{I_N}) have large differences between their maximum and minimum values. However, as it is difficult to efficiently estimate them from the central trend of a normal distribution, this is expressed as a multivariate problem for the various data distributions, as in Dilemma 3.1.

Dilemma 3.1: $\forall f$ involved in I_N consists of large-scale values such that

$$-\infty < \max(f_{ij}) - \min(f_{ij}) < \infty, \quad \max(f_{ij}) \gg \min(f_{ij}) \quad (3.2)$$

which shows that the attribute values are not specified in a particular range, for example $[-1, 1]$, while a feature's (f) maximum value ($\text{Max}(f_{ij})$) is significantly greater than its minimum value ($\text{Min}(f_{ij})$). This could cause noise while fitting the data to any probability distribution as the smallest and largest values dramatically deviate from their means (M) [219].

To address Dilemma 3.1, the following z-score function, which is a linear transformation function for normalising the f_{ij} values in a specific format which facilitates comparing them in different probability distributions without modifying their original distribution, is used.

$$z_{ij} = \frac{f_{ij} - M}{\delta} \quad (3.3)$$

where the normalised values (z_{ij}) of each feature ($\forall f$) in $I_{1:N}$ are computed by subtracting the values from the mean (M) and then dividing each by its standard

deviation (δ) of that feature to estimate how far away it is from its M . The output from this function could be positive or negative depending on whether the values are above or below their means, respectively. It is acknowledged that this output is designed to standardise these data with $M = 0$ and $\delta = 1$.

3.9.2. Kolmogorov-Smirnov (K-S) Test

The K-S test is used to decide if a sample comes from a population with a particular distribution [214, 215]. While goodness-of-fit tests are often designed for univariate distributions, network data have multivariate distributions (i.e., a set of features) for determining the one that can precisely fit all the selected features. This assists the use of the best probability distribution for establishing the DE with the least noise, that is, if the normal profile is constructed by a Gaussian distribution which has its two parameters (the mean and standard deviation) estimated from normal and malicious observations detected by considering any deviation from the established normal model (i.e., profile) as an attack record. It is possible that a Gaussian distribution cannot correctly fit the majority of normal records because sophisticated attackers attempt to mimic normal behaviours which leads to attack records overlapping normal observations. Therefore, it is necessary to check the normality of network features before selecting a distribution that can fit the network data with less noise.

The K-S model tests the values of the features in the (TR_{I_N}) and (TS_{I_N}) to determine their normalities. Let $\forall f$ have x_1, x_2, \dots, x_n values in an ascending order and then the K-S test is as follows.

- H_0 (**null hypothesis**): the data follow a particular distribution.
- H_a (**alternative hypothesis**): the data do not follow that particular distribution.

- **Test function:** the K-S test is defined by computing the empirical distribution function ($F_n(x)$) of the data points and Kolmogorov distribution function ($F(x)$) which considers the proportion of feature values less than or equal to a certain value (x) as

$$F_n(x) = \begin{cases} 0 & , x < x_1 \\ k/n & , x_k \leq x < x_{k+1}, k = 1, 2, \dots, n-1 \\ 1 & , x \geq x_n \end{cases} \quad (3.4)$$

where n is the number of observations and k a counter from 1 to $n - 1$.

The Kolmogorov distribution function is defined as

$$f(x) = \sqrt{\frac{2\pi}{x}} \sum_{n=1}^{\infty} e^{-(2n-1)^2 \pi^2 / (8x^2)} \quad (3.5)$$

Based on equations (3.4) and (3.5), the formula for the K-S goodness-of-fit statistics is calculated by maximising the absolute difference between the $f(x)$ and $F_n(x)$ as

$$D_n = \max_x |f(x) - F_n(x)| \quad (3.6)$$

The well-known critical values provided in [220] are used to reject the hypothesis if the test statistics ($D_{n,\alpha}$) are greater than those values. The significance level (α) is estimated via ($P(D_n \leq D_{n,\alpha}) = 1 - \alpha$) to determine whether a feature follows a specified distribution. The best fitting of a feature has to satisfy the condition of ($\max_x |f(x) - F_n(x)| \leq D_{n,\alpha}$) as well as its confidence interval

$(F_n(x) \pm D_{n,\infty})$). The SPSS tool [221] is used to execute the K-S test with the criterion that, if the significance value of the K-S test is greater or less than 0.05, the data follow a Gaussian distribution or substantially deviate from it, respectively.

3.9.3. Multivariate Skewness and Kurtosis

Multivariate skewness and kurtosis measures are used to check the normality of the data points and their shapes [216, 217]. Apart from applying the K-S test, it is significant to analyse the skewness and kurtosis of features to estimate to what extent they follow a particular distribution. The skewness function is defined as an asymmetric measure of the probability distribution of a feature from its mean (μ), where each feature consists of a set of values (x_1, x_2, \dots, x_n) and its formula is

$$\text{skewness} = \frac{\sum_{i=1}^n (x_i - \mu)^3}{n\delta^3} \quad (3.7)$$

If the outcome is a positive or negative value, the distribution with an asymmetric tail spreads to major positive values or more negative ones, respectively.

The kurtosis function is considered a peakiness measure of the probability distribution for a feature which includes a set of values (x_1, x_2, \dots, x_n) and is computed as

$$\text{kurtosis} = \frac{\sum_{i=1}^n (x_i - \mu)^4}{n\delta^4} \quad (3.8)$$

If the result is a positive value, this distribution is much higher than a Gaussian distribution but, if negative, is in a flat shape. As stated in [70], if the skewness

and kurtosis of a feature tend to be 0, the distribution of this feature could follow a Gaussian distribution.

3.10. Use of Statistical Measures on training and testing sets

The aforementioned statistical measures are applied to the training (TR_{I_N}) and testing (TS_{I_N}) sets to ascertain the normality of their features which, statistically speaking, can determine the norms of the current legitimate and malicious patterns. If the findings from the K-S test, and multivariate skewness and kurtosis functions show that the network features follow a Gaussian distribution, a DE should be designed according to the functionality of this distribution to distinguish anomalous from normal behaviours. In contrast, if they do not follow a Gaussian distribution, other non-Gaussian distributions, such as a Gaussian Mixture Model (GMM), Beta Mixture Model (BMM) or Dirichlet distribution, should be used to build an intelligent DE to precisely fit the normal data and identify any outliers or rare events in them as anomalies.

The K-S test, and multivariate skewness and kurtosis functions are used on the data in the (TR_{I_N}) and (TS_{I_N}) to check their normality and determine the similarity of their results, as formulated in equations (3.9), (3.10) and (3.11). If these sets are extremely different in terms of their probability distributions, DE approaches will result in higher FARs because a DE model learns from particular data in the training phase and is tested on different data in the testing phase which is called an over-fitting problem in the ML field [222]. This problem means that a classifier is biased towards learning some data from the training phase which are not sufficient for efficiently building the model or do not represent the data in the testing phase.

$$sig_{TR_{I_N}} \approx sig_{TS_{I_N}} \quad (3.9)$$

$$skewness_{TR_{I_N}} \approx skewness_{TS_{I_N}} \quad (3.10)$$

$$kurtosis_{TR_{I_N}} \approx kurtosis_{TS_{I_N}} \quad (3.11)$$

Equation (3.9) checks whether the significance values (i.e., *sig*) of the features estimated from the K-S test are approximately similar in both the training and testing sets. If they are extremely dissimilar, this denotes that their probability distributions are different which negatively affect the reliability of an ADS as do different skewness and kurtosis values of these features obtained from equations (3.10) and (3.11). Based on the above discussion, the features of the (TR_{I_N}) and (TS_{I_N}) are statistically analysed in algorithm 3.10 to demonstrate the relationships between them, with their results presented in Section 3.11.

3.10.1. Feature Correlations of (TR_{I_N}) and (TS_{I_N})

A correlation analysis is another means of identifying the relationships between features in the (TR_{I_N}) and (TS_{I_N}), and, as choosing the significant features is an important step in building a lightweight ADS, two correlation techniques are used for the UNSW-NB15 dataset. Firstly, Pearson's correlation coefficient (PCC) [223, 224] estimates the correlations between features without the class label. Secondly, a Gain Ratio (GR) [224] method computes the correlations between features with

Algorithm 3.10 Determining statistical relationship of training and testing sets

Input: the features of the (TR_{I_N}) and (TS_{I_N})

Output: the relationships of the features of the (TR_{I_N}) and (TS_{I_N})

- 1: **for** ($\forall f$ in the two sets) //*for each feature involved in the training and testing sets* **do**
 - 2: convert the values of the nominal features into numerical features, as explained in proposition 3.1
 - 3: apply the z-score function, as discussed in subsection 3.9.1
 - 4: apply the Kolmogorov-Smirnov test and skewness and kurtosis functions, as discussed in subsections 3.9.2 and 3.9.3, respectively
 - 5: compare the (TR_{I_N}) and (TS_{I_N}) results using equations (3.9), (3.10) and (3.11), respectively
 - 6: **end for**
-

the class label. These two approaches are the simplest feature selection models and have lower processing times than others, as discussed below.

The aim of using them is to determine the correlation scores between these features, either with or without the class label, to select the important features which are then passed to a DE model for distinguishing between normal and attack activities. To apply these techniques on a dataset, let a dataset (T) have a set of features ($f_1, f_2, \dots, f_d \in I_{1:N}$) with its class label (C), where each feature and its class label contain many values; for example, $f_1 = \{x_1, x_2, \dots, x_N\}$, $f_2 = \{y_1, y_2, \dots, y_N\}$ and $C = \{c_1, c_2, \dots, c_w\}$, where d is the number of features, N is the number of observations and w is the number of classes involved in T . These two techniques are described below and their results provided in subsection 3.11.2.

A. Feature correlations without class label

PCC is one of the simplest linear correlation techniques for estimating the correlation scores of features [223, 224], with that of two features (f_1 and f_2)

$$PCC(f_1, f_2) = \frac{cov(f_1, f_2)}{\sigma_{f_1} \cdot \sigma_{f_2}} \quad (3.12)$$

$$= \frac{\sum_{i=1}^N (x_i - \mu_{f_1})(y_i - \mu_{f_2})}{\sqrt{\sum_{i=1}^N (x_i - \mu_{f_1})^2} \cdot \sqrt{\sum_{i=1}^N (y_i - \mu_{f_2})^2}}$$

where cov is the covariance and σ the standard deviation of these features, and $\mu_{f_1} = 1/N \sum_i^N x_i$ and $\mu_{f_2} = 1/N \sum_i^N y_i$ the means of f_1 and f_2 , respectively.

The result obtained from equation (3.12) has to be in a certain range of $[-1, 1]$ which, if close to -1 , indicates a strong correlation in an opposite direction, if close to 1 , a strong correlation in the same direction and, if close to 0 , no correlation between these features, that is, a positive or negative sign shows that these features have the same or different trends, respectively. The important features are ranked by computing the mean of each PCC feature (i.e., $\mu_{f_i} = 1/N \sum_i^N pcc_{f_i}$) which are arranged in a descending order.

B. Feature correlations with class label

The GR technique computes the ratio of an Information Gain (IG) method to the values of the features [224] which is then used to address the problem of the IG when a feature contains a set of values by comparing those values. The IG is a feature selection approach that relies on an entropy function which measures the uncertainty of the features; for example, let I be a set of observations with w distinct classes for which the entropy or expected IG are computed by

$$G(I) = - \sum_{i=1}^m p_i \log_2(p_i) \quad (3.13)$$

where P_i is the probability of observation I belonging to class C_i and is estimated by I_i/I_N .

To divide I_i into subsets for each feature (f), the expected IG is defined as

$$E(f) = - \sum_{i=1}^m G(I) \frac{I_{1i} + I_{2i} + \dots + I_{mi}}{I_N} \quad (3.14)$$

From equations (3.13) and (3.14), the encoding information gained to a feature is given by

$$Gain(f) = G(I) - E(f) \quad (3.15)$$

The splitting value of these subsets (I_{ir}) is

$$Split(I) = - \sum_{i=1}^r (|I_i/I_N|) \log_2(|I_i/I_N|) \quad (3.16)$$

In equation (3.16), the splitting value is used to divide the observations into particular portions (r) for easy selection of the important features. From equations (3.15) and (3.16), the GA is defined as $GR(f) = Gain(f)/Split(I)$, where the feature with the highest GR is used as the baseline to select the most highly ranked features.

3.10.2. Evaluation of Five ML Techniques

ML techniques are those most commonly applied to analyse a network dataset to detect malicious events, as previously explained in Section 3.4. Five existing techniques, the NB [187], DT [188, 189], ANN [188, 189], EM clustering [190] and Logistic Regression (LR) [225], are applied to the UNSW-NB15 dataset to estimate its accuracy, FARs and ROC curve, as explained in Chapter 2, Section 2.4. Each technique has its own procedures for learning and validating the data

points in the (TR_{I_N}) and (TS_{I_N}). Firstly, the NB is a conditional probability technique which classifies network observations as either normal (0) or abnormal (1) and uses the maximum a posterior (MAP) function to compute the score of each observation as

$$P(C|I) = \operatorname{argmax}_{w \in \{1, 2, \dots, N\}} P(C_w) \prod_{j=1}^N P(I_j|C_w) \quad (3.17)$$

where C is the class label, I is the observation of each class, w is the class number, $P(C|I)$ is the probability of the class given a particular observation and $\prod_{j=1}^N P(I_j|C_w)$ used to multiply the probabilities of observations to obtain the maximum output to determine whether an observation is normal or abnormal.

Secondly, the DT is a structure that looks like a flowchart which consists of roots, nodes and branches that show the rules of classification. Each node represents the rules or procedures of a feature, each branch the results of the rules and each leaf node the class label. Thirdly, the ANN uses an activation function which relies on a large number of input observations (I) to learn network patterns and is

$$f(I) = \tau(\sum_j W_j \cdot I_j) \quad (3.18)$$

where $f(I)$ is a predicted output of the class label, τ an activation function (i.e., sigmoid) and W_j the weight of each feature (F). Fourthly, the LR constructs the correlation between a dependent variable (i.e., the class label) and independent variables (i.e., the important features) using the maximum likelihood function to compute the regression parameters. Finally, the EM clustering technique depends on maximising the probability density function (PDF) of a Gaussian distribution to compute its parameters (the mean and covariance) of each feature in a dataset

and involves two steps (i.e., expectation (E) and maximisation (M)). The E step estimates the likelihood of each observation occurring in the dataset while the M step re-estimates the parameters from the E step to obtain the optimal set of parameters for building the model.

The evaluation terms (accuracy, FAR and ROC curve) demonstrate the performances of these techniques on the UNSW-NB15 dataset and the extent to which this dataset contains sophisticated patterns of current security events and malware. The proposed training and testing sets are used to learn and validate the five techniques which are then executed using the Visual Studio Business Intelligence 2008 tool [75], with their results provided in subsection 3.11.2.

3.11. Empirical Results and Discussion

As discussed above, to estimate the complexity of the UNSW-NB15 dataset, models of statistical analyses and explanations, feature correlations and complexity evaluations are applied on the proposed training and testing sets using the ML techniques, as explained in Section 3.10, with the features used listed in Table 3.20.

3.11.1. Statistical Analyses and Explanations

The SPSS tool is used for statistical explanations and determining the distribution norms of the training (TR) and testing (TS) sets, with the probabilities occurring presented in Figure 3.7. The results demonstrate that the feature distributions are non-linear and non-normal representations because the significance values of these features are less than 0.05. When the two lines of the TR and TS sets appear to be the same, the fitting percentages of these features are approximately 78% and,

Table 3.20: Features used for analyses of UNSW-NB15 dataset

Id	Names	Id	Names
1	Dur	22	Synack
2	Spkts	23	Ackdat
3	Dpkts	24	Smean
4	Sbytes	25	Dmean
5	Dbytes	26	trans_depth
6	Rate	27	response_body_len
7	Sttl	28	ct_srv_src
8	Dttl	29	ct_state_ttl
9	Sload	30	ct_dst_ltm
10	Dload	31	ct_src_dport_ltm
11	Sloss	32	ct_dst_sport_ltm
12	Dloss	33	ct_dst_src_ltm
13	Sinpkt	34	is_ftp_login
14	Dinpkt	35	ct_ftp_cmd
15	Sjit	36	ct_flw_http_mthd
16	Djit	37	ct_src_ltm
17	Swin	38	ct_srv_dst
18	Stcpb	39	is_sm_ips_ports
19	Dtcpb	40	Proto
20	Dwin	41	Service
21	Tcprrt	42	State

conversely, when the two lines are different, the non-fitting percentages are almost 22%.

As it is observed that the two are treated as if they are from the same distribution (i.e., fitting percentages of 78%), when applying any DE approach, there will be no over-fitting problem, as discussed in subsection 3.10.2. As these features cannot correctly follow a Gaussian distribution, a non-Gaussian distribution, such as the GMM, BMM or DMM, should be used to establish a new DE technique for efficiently recognising current malicious activities. These observations are also confirmed by computing the skewness and kurtosis values of these features.

The features' asymmetries in the TR and TS sets are estimated using the skewness (skw) function. In the TR set, the results for all the features, except

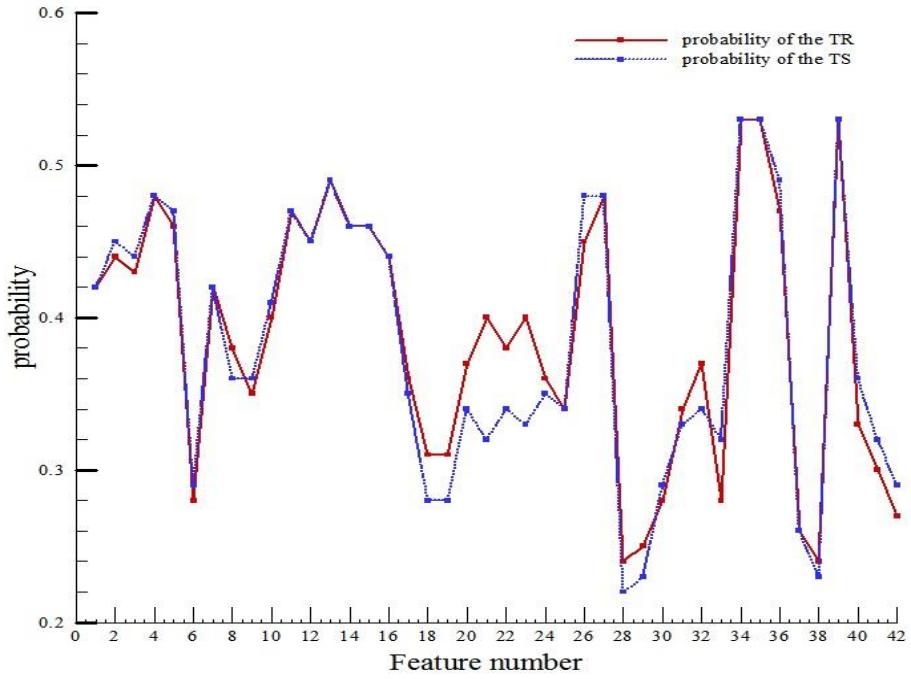


Figure 3.7: Probabilities of features being in training and testing sets

7, are positive. Therefore, most of these features are on the right-hand side and their PDFs higher or shorter than those on the left-hand side. The TS is roughly similar to the TR set, with most of these features, except 7, 17 and 20, positive. The feature skewness values of these sets are presented in Figure 3.8 in which it can be seen that the relationship percentage is approximately 82% when the two lines are similar, with the highest skewed features 25, 26, 27 and 28, and the lowest 6, 7, 8, 17, 18, 19, 29, 30, 31, 32, 37, 38 and 42.

The feature peaks of the TR and TS sets are computed via the kurtosis function. In the former, the experimental findings demonstrate that seven features, 7, 8, 17, 18, 19, 20 and 40, have negative values which represent a flatter distribution while the others have positive values. This means that the distribution of these features is higher than a Gaussian distribution. As depicted in Figure 3.9, the feature kurtosis values of the TR and TS sets clarify that their fitting percentage

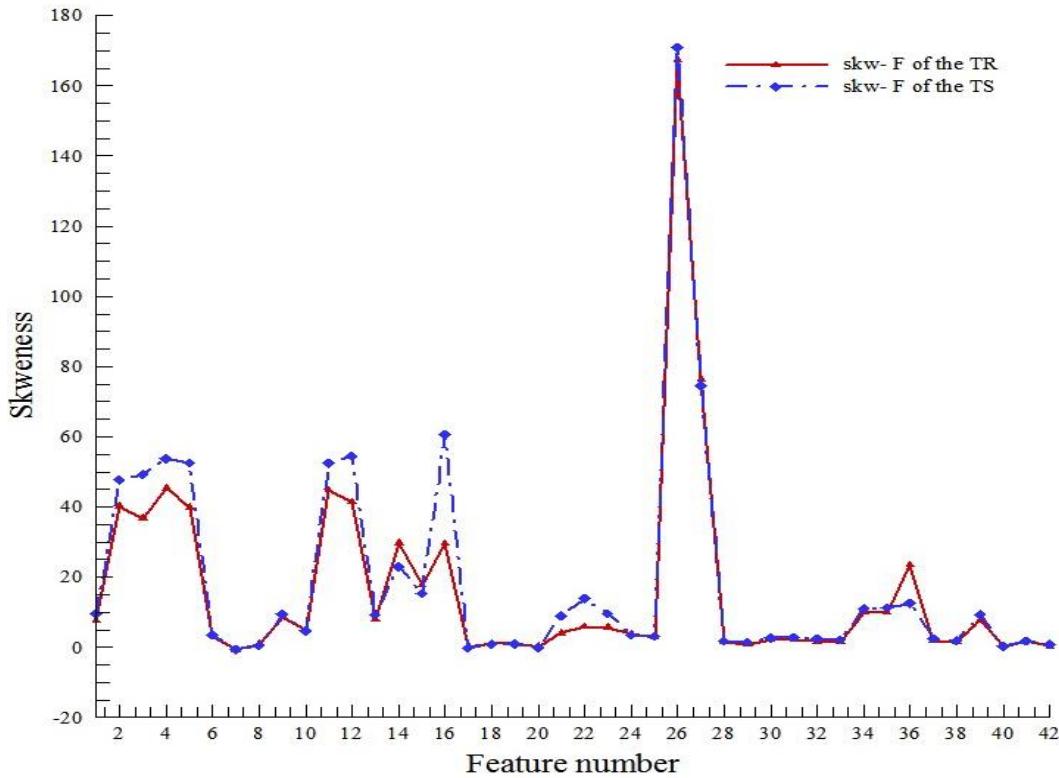


Figure 3.8: Skewness values of features in training and testing sets

is approximately 76% while the two lines maintain the same trend. The kurtosis of the TS set is higher than that of the TR in features 2-6, 10-12, 15-16, 25-28 and 35-37 while the others are relatively close to each other.

As, based on the above explanation, the features of the TR and TS sets are statistically correlated and have the same characteristics of non-linearity and non-normality, this part of the UNSW-NB15 dataset is credible for assessing NIDSs. Finally, this statistical analysis and explanation indicate that this dataset could be used to evaluate the performances of new DE approaches for effectively detecting attacks.

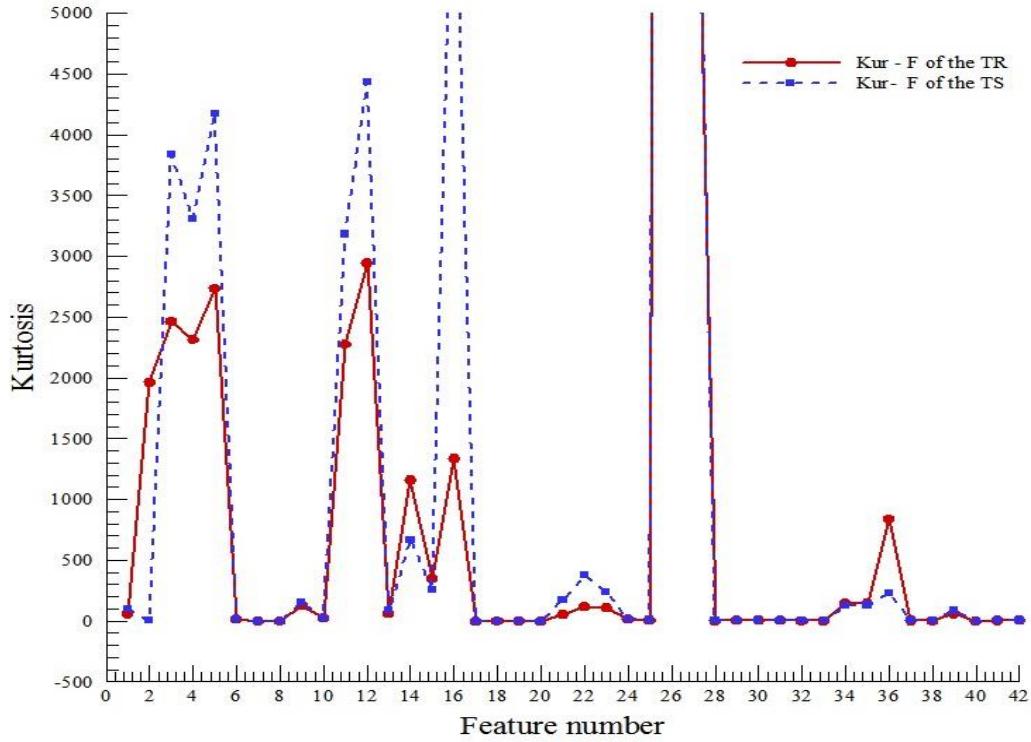


Figure 3.9: Kurtosis values of features in training and testing sets

3.11.2. Feature Correlations

The feature correlations are calculated based on two aspects, without the label by the PCC and with the label by the GR, for measuring the proportions of dependence and independence between these features. The important features can be considered when they are less independent as their statistical characteristics are different [87]. Initially, the PCC computes the score of each feature and Figure 3.10 shows that those in the TR and TS sets have almost the same correlated score. Then, these features are ranked in a certain range [-0.01, 0.11], with the highest related features, 5, 3, 6, 12, 13, 29, 31, 32, 33, 34, 38 and 39, showing lower dependencies with a less than 0.5 difference in their scores.

In contrast, the lowest correlated features are 7, 8, 10, 11, 14, 40 and 41 while the other features fall into the middle of the range with lower dependencies. In

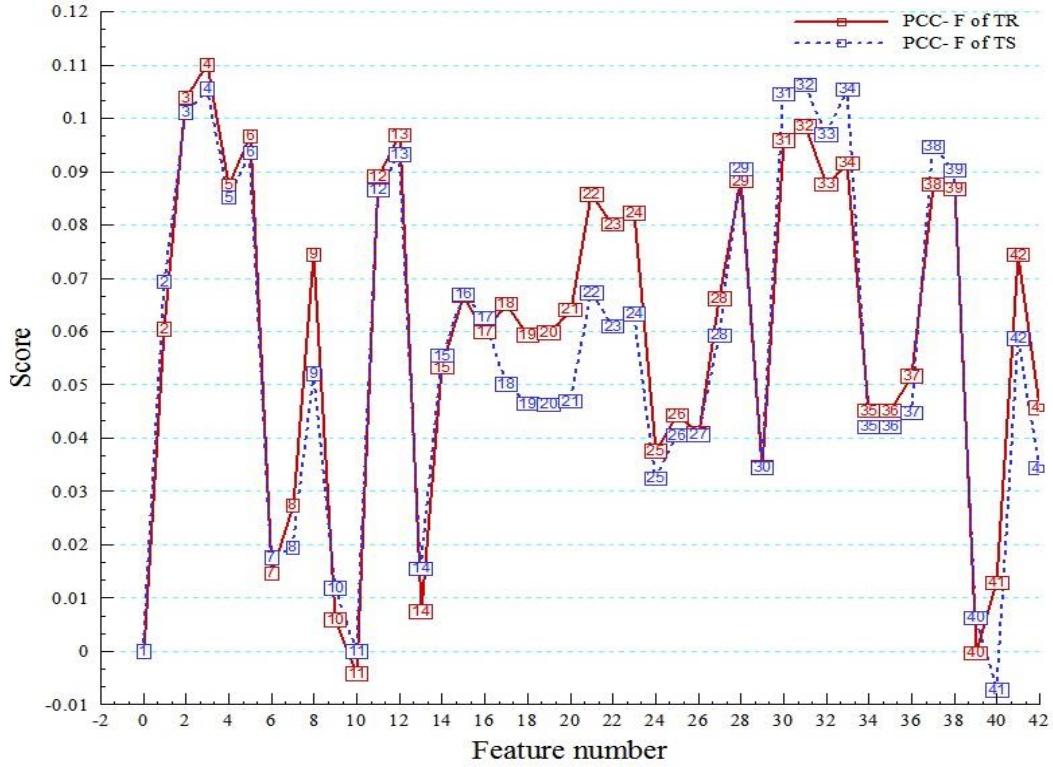


Figure 3.10: PCCs of features in training and testing sets

the two sets, the correlated features are categorised into high, middle and low, and obtain probabilities of 12/42, 23/42 and 7/42, respectively. This indicates that, as 87.5% of the high and middle features are adequately correlated, these sets are appropriate for learning and validating new NIDSs.

Secondly, in the TR and TS sets, the GR computes the correlations of the features with a class label, with Figure 3.11 showing that those of both sets are similar. The scores for these features of between 0.01 and 0.56 are classified as low, middle and high according to the ranges [0.01, 0.2], [0.21,0.3], [0.31,0.56], respectively. The lowest correlated features are 1, 2, 3, 7, 12, 13, 14, 16, 17, 27, 28, 29 , 34, 35, 36, 37, 38 and 39, the middle 4, 5, 6, 10, 11, 15, 18, 19, 20, 21, 22, 23, 24, 25, 26, 31, 32 and 40, and the highest 8, 9, 30, 33, 41 and 42, with their probabilities 18/42, 18/42 and, 6/42, respectively. Therefore, the appropriate

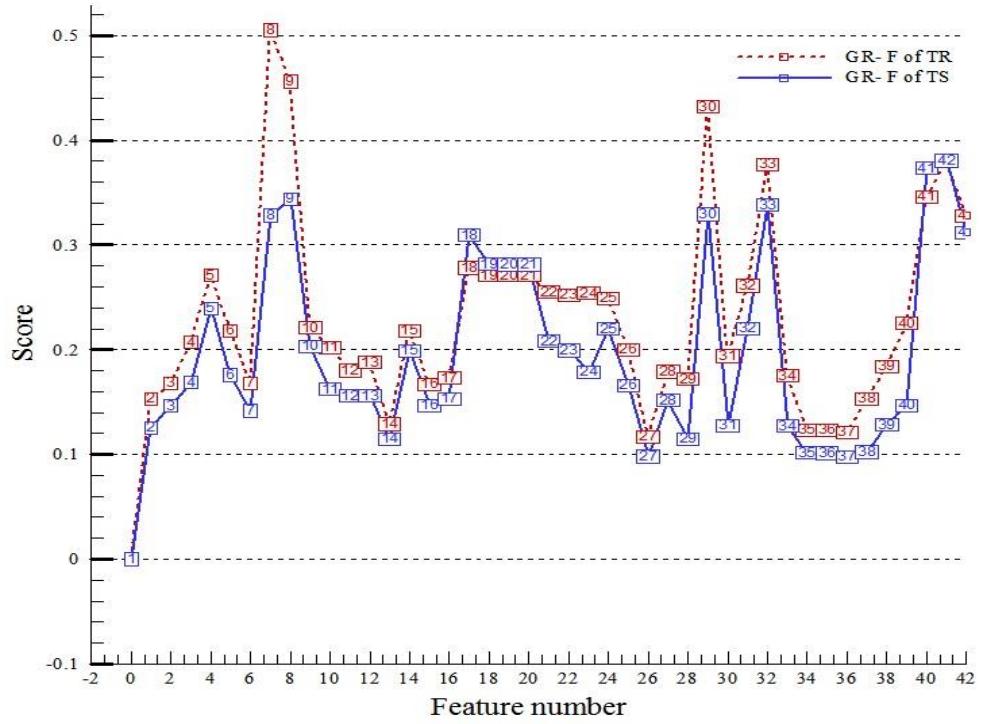


Figure 3.11: Gain ratios of features in training and testing sets

correlation rates for the high and middle correlated features are computed as 57.2% which shows that these sets are adequate for learning and validating new NIDSs.

3.11.3. Complexity Evaluations using ML Techniques

The complexity of the UNSW-NB15 dataset in terms of its accuracy, FAR and ROC curve estimated by applying the NB, DT, ANN, LR and EM clustering techniques using the default parameters of the Visual Studio Business Intelligence 2008 tool on the TR and TS sets for the features listed in Table 3.20. As can be seen in Table 3.21, the DT model obtains the highest accuracy of 85.56% and lowest FAR of 15.78%. On the contrary, the EM clustering technique achieves the lowest accuracy of 78.47% and highest FAR of 23.79%. Also, the ROC curves

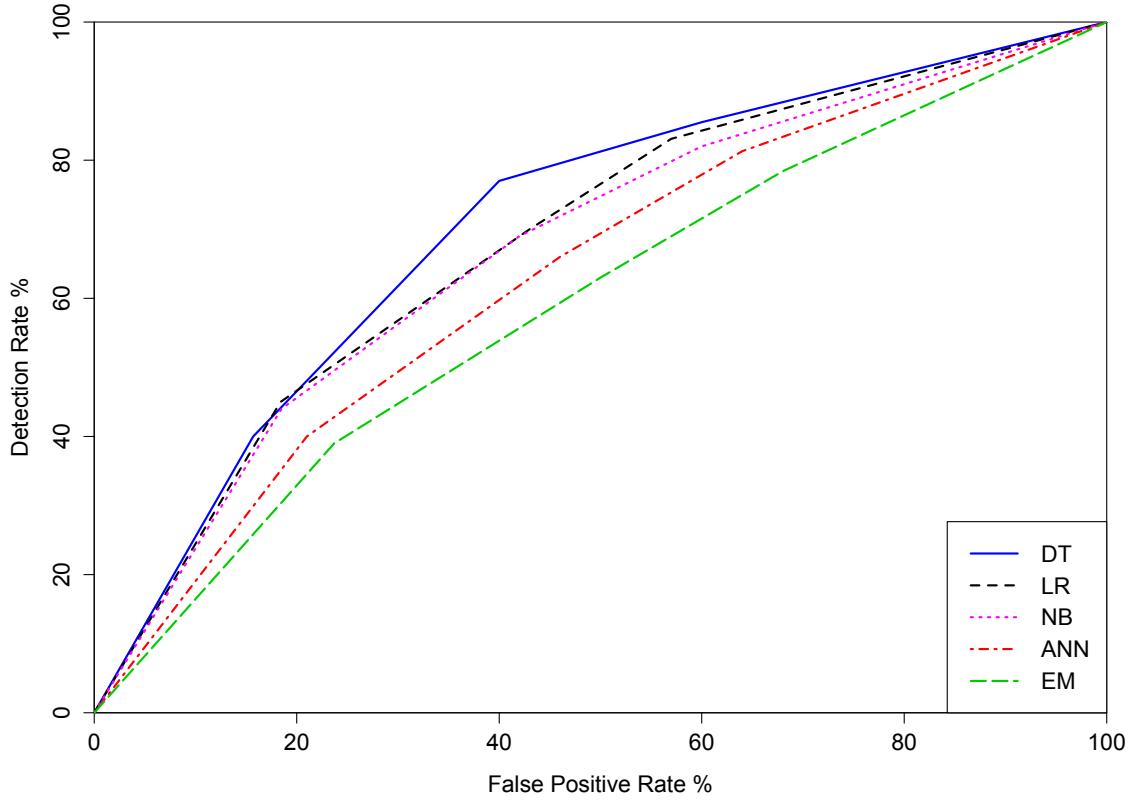


Figure 3.12: ROC curves of five techniques using two sets from UNSW-NB15 dataset

depicted in Figure 3.12, which represent the relationship between the TPR and FPR, rank the five models based on a higher TPR (i.e., DR) and lower FPR.

Table 3.21 lists the comparative results obtained from the five techniques for the KDD99 and UNSW-NB15 datasets. Generally, the accuracy and FAR of each are better using the former than the latter dataset.

There are two aspects that demonstrate the complexity of the UNSW-NB15 dataset compared with that of the KDD99 dataset. Firstly, in terms of network behaviour, the UNSW-NB15 dataset contains a variety of contemporary legitimate

Table 3.21: Comparison of results obtained for KDD99 and UNSW-NB15 datasets

Techniques	KDD99 dataset		UNSW-NB15 dataset	
	Accuracy (%)	FAR (%)	Accuracy (%)	FAR (%)
				(%)
DT [79]	94.74	6.85	85.56	15.78
LR [79]	92.75	8.95	83.15	18.48
NB [79]	95.3	5.03	82.07	18.56
ANN [79]	97.04	1.48	81.34	21.13
EM Clustering [79]	78.06	10.37	78.47	23.79

and abnormal behaviours while those in the KDD99 dataset are outdated. Also, the similarity of its normal and attack observations is another factor that indicates its complexity, that is, its variances between normal and abnormal observations are very low.

Secondly, from a statistical perspective, as the features of the TR and TS sets in the UNSW-NB15 dataset are closely correlated as they have quite similar skewness and kurtosis scores, as shown in Figures 3.9, 3.10 and 3.11, these techniques do not provide the highest DR and lowest FPR as they do for the KDD99 dataset. This means that a single ML technique cannot distinguish between normal and abnormal observations as it has a kernel function(s) for solving a particular problem. Therefore, it is clear that the UNSW-NB15 dataset is more complex than the KDD99 dataset.

3.12. Chapter Conclusion

This chapter covers two new large-scale network datasets, DARPA-2009 and our UNW-NB15. The former is one of the latest and we analysed its first 30 pcap files, which contain most of its security events, using some existing techniques and tools, such as a TCP trace tool, to generate features from its pcap files and label this portion according to its ground truth using a SQL Server 2008 tool.

To select the important features, we proposed a statistical method which can distinguish between normal and malicious observations, with some existing ML techniques applied to evaluate the DARPA-2009 dataset’s credibility for detecting abnormal activities. However, the results and analysis showed that its packets do not have a payload for attack vectors and there are some problems in its ground-truth table for correctly tagging its observations. Therefore, we provided a new dataset, namely, the UNSW-NB15, to address these challenges.

The UNSW-NB15 dataset was created by establishing an authentic testbed environment at the UNSW cyber security lab, with the IXIA tool used to simulate current representations of normal and malicious network traffic. The IXIA tool has the capability of logging the exact processing time in its ground truth report while launching malware events. This is one of the key advantages of the tool that cannot find in existing simulation network tools because it can exactly control the transnational time of sending and receiving network flows . The dataset contains 9 types of security and malware events and 47 features with their class labels generated using Argus and Bro-IDS tools and new scripts incorporating the characteristics of network packets. Some comparisons of this and other network datasets were conducted to demonstrate its credibility for evaluating new NIDSs.

The analyses and evaluations of this dataset were explained in detail. A portion of it was used to train and test the ML techniques and examining this dataset. The training and testing sets were determined based on statistical analyses, feature correlations and complexity evaluations. Firstly, the features of the two sets were transformed into numerical values for statistical processing and then normalised using the z-score function to prevent any change in their original distributions. The statistical results from the Kolmogorov-Smirnov test indicated that the two sets had the same non-normal and non-linear distribution while their feature skewness and kurtosis values were statistically similar.

The feature correlations of the training and testing sets were estimated using either the PCC method with the class label or the GR technique without the label, with the findings indicating that these features were almost independent of each other. Finally, five techniques, DT, LR, NB, ANN and EM clustering, were used to evaluate the complexity of the UNSW-NB15 dataset with a comparison of its overall results and those for the KDD99 dataset showing that the former was more complex than the latter which indicated that it could be used to reliably evaluate new methods for NIDSs.

Chapter 4 discusses the relevant features which help to differentiate between legitimate and malicious activities, and Chapter 5 explains the methodology for building new statistical NADS with lightweight, adaptable and scalable characteristics.

Chapter 4

Relevant Feature and observation Methods and Their Impacts on Design of Lightweight Network Anomaly Detection System

4.1. Introduction

A ‘relevant feature and observation method’, which plays an important role in an IDS methodology, is a new way of handling the attributes and observations of network data by eliciting and generating relevant vectors from network traffic with no duplication of flows, and with characteristics that can precisely differentiate between legitimate and suspicious activities in DE methods. We classify these techniques as feature creation, flow aggregation and feature reduction (FR) which enhance the performances and reduce the processing times of DE methods.

In this chapter, we contribute to the data pre-processing module discussed in Chapter 2, subsection 2.6.2, in terms of its feature creation and reduction processes. The former constructs a proper feature set which has the properties of normal and malicious observations while the latter eliminates irrelevant features by selecting a suitable set in a small-dimensional space. If the features are not properly generated with these properties, DE methods will not efficiently distinguish among their observations. Therefore, constructing these features requires in-depth analyses of network protocols and services for use in the DE module to detect abnormal events. In the previous chapter, we suggest new features for both the DARPA-2009 and UNSW-NB15 datasets. In this study, firstly, we extend that work by generating a new set of features via analysing protocols and services of

the OSI model [226], as explained in Section 4.4, and then choose the important features using FR methods.

Secondly, we clarify the significant role of flow aggregation in analysing network data and designing an effective and lightweight ADS. Flow aggregation has become necessary for various applications, including network planning and monitoring, as well as security management [227, 228]. With a flow aggregation step, applications, in particular an ADS, can use several statistical measures, such as packet counts and sizes of flows through network traffic for each specific time window as features that are passed to the DE method. The importance of this step is to mine only relevant observations with no duplicate or missing values to improve the performance of an ADS. In it, we investigate the simple random sampling [229] and Association Rule Mining (ARM) [89] techniques to demonstrate how they are used to implement flow aggregation.

Finally, we describe the impact of FR on the design of an effective and efficient ADS for achieving the main targets of this PhD research. For network data, this is the process of removing unimportant or irrelevant features from a collection of data to improve the detection accuracy and processing time of a DE method. While an ADS should be lightweight, to successfully achieve a high detection rate (DR) requires a huge effort. Whereas many studies have tried to develop a lightweight ADS by choosing important features using only feature selection (FS) methods, we propose the concept of ‘relevant feature and observation’ method. This concept includes feature creation to extract and generate the relevant attributes of normal and malicious activities, flow aggregation to accumulate appropriate network observations and FR to reduce the original feature set into a small set to build an efficient online ADS.

The key contributions of this study presented in this chapter are as follows.

1. A novel aggregator module is designed based on the theory of flow-level analysis using a random sampling technique or ARM algorithm to decrease

the computational resources required and tackle some limitations of existing flow tools.

2. A new ARM technique for feature selection is proposed and its performance compared with those of the PCA and ICA techniques using some ML algorithms to improve the accuracy of detecting attacks with a low processing time.
3. A set of features from the DNS and HTTP protocols and their data sources are created from the UNSW-NB15 dataset to build an effective NIDS for detecting attacks that breach network applications using a new Adaboost ensemble method including the three classification techniques DT, ANN and NB.
4. An in-depth statistical analysis and discussions of the techniques used in this chapter on the NSL-KDD and UNSW-NB15 datasets are provided to demonstrate the importance of using relevant feature methods to develop an intelligent NIDS.

The rest of this chapter is organised as follows. Section 4.2 discusses a network flow analysis and its tools. An aggregator module for an ADS, including sampling and ARM techniques, is described in Section 4.3. In Section 4.4, network features are created based on analyses of the protocols and services of the TCP/IP model. Section 4.5 includes the role of FR techniques, in particular ARM, PCA and ICA, to demonstrate how they improve the design of an online ADS. The experimental results obtained from the techniques used to select the relevant observations and features of network data are provided in Section 4.6. Finally, we conclude this chapter and provide future research directions in Section 4.7.

4.2. Network Flow Analysis

As a NIDS demands a feature creation step in its pre-processing module in order to extract relevant attributes from raw network packets. Since current network systems are large and have high-speed flows, the data of each have a large volume, velocity and variety related to the phenomenon of ‘big data’ described in Chapter 3, Section 3.7. Consequently, network data should be handled using big data analysis techniques for sniffing, storing in a database and processing purposes, such as network flow analyses and detection mechanisms.

A network flow analysis is a method for finding important information from raw packets using statistical and machine learning (ML) techniques. It involves capturing, collecting and logging network data, aggregating them for query and analysis, and analysing them to generate important information to assist a DE method to effectively discriminate between normal and abnormal activities. This information is related mainly to network management, measurement and security.

There are two ways of analysing network data, a deep packet inspection or flow-level analysis [227]. The former analyses the payload (i.e., data) of packets, takes a long processing time and faces the encryption problem while the latter collects statistical information about packets without analysing their actual data and consumes far fewer computational resources. Therefore, it is vital to use a flow-level analysis while establishing an online ADS and, given the current encryption of network systems, it is helpful to analyse only packet headers, as discussed in Section 4.4. Different tools are used to elicit flows from packets, including Argus and Bro-IDS which are described in Chapter 3, subsection 3.4.3, and NetFlow, sFlow and IPFIX which are explained in the following subsections.

4.2.1. NetFlow

This is a traffic monitoring tool developed by Darren and Barry in 1996 [230]. It describes how an ingress router collects information and statistical observations of network packets through routed sockets. Since it has become an industry standard, it is a built-in feature of several routers and switches of multiple vendors, such as Cisco and Juniper. Its functionality involves network appliances inspecting any packet arriving at the interfaces, collecting traffic statistics per flow using a particular configuration for filtering or sampling and then constructing a flow cache to extract the network data within a UDP or Stream Control Transport Protocol.

A NetFlow cache entry is established by the first packet of a flow, maintained for similar flow properties and periodically exported to collectors using flow-cache management or flow timers. The Sampled NetFlow is a variant provided by Cisco to decrease the computational resources required by reducing the number of flows. It can be configured to a specific number of packets or randomly selected intervals [227, 231].

4.2.2. sFlow

This is packet sampling of a network flow designed by InMon Inc [232] which has become an industry standard declared in RFC 3176. Its internal functionality depends on the theory of simple random sampling (discussed in subsection 4.3.1) and it is supported by several firms, such as HP, Alcatel and Extreme, which embed it in routers and switches. It can be implemented in layer 2 of a network and extract non-IP traffic information.

Its agent is a software process that combines flow samples, interface counters into datagrams and immediately transmits them to collectors using a UDP with respect to the instant forwarding of data which minimises CPU and memory usage.

Packets are usually sampled by application-specific integrated circuits to provide better performance, with the data including packet headers and switching/routing information for each minute of network traffic [227, 233].

4.2.3. IPFIX

The IP Flow Information Export (IPFIX) protocol is an IETF standard for extracting network flows using NetFlow version 9 and is specified in RFC 5101 for information-transferring protocols, RFC 5103 for exporting bidirectional flows and RFC 5102 for information modelling. It was developed to meet the fast-growing need to effectively capture network traffic, and includes an extensible and elastic data model which can be easily customised. It was also designed to support reliable and secure data transfer through UDP and TCP and is less restrictive than traditional flow tools [227].

4.3. Aggregator Module for ADS

The massive flows of current networks require an aggregator module¹ that starts by sniffing packets which finally collects in a data source to build an online ADS. Although the above tools for network flow analysis are designed for this purpose, they only aggregate and characterise network traffic using one attribute at a time, such as source/destination IP addresses or protocol types [234]. However, aggregating flows based on many attributes could determine patterns of several types of attacks and malware, such as DoS, DDoS, aggressive port scans, flash crowds and

¹

- A part of the work in this chapter has been released in the following.
Moustafa, N., Creech, G. and J. Slay. “Flow aggregator module for analysing network traffic”, the International Conference on Computing Analytics and Networking (ICCAN 2017), KIIT University, Springer, 2017.

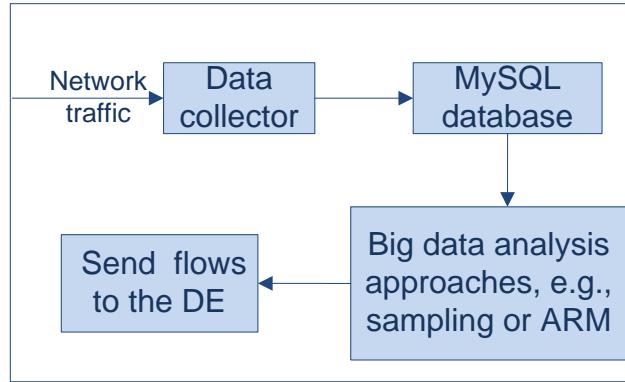


Figure 4.1: Proposed aggregator module

worm spread. Because these types generate large flows, they require careful analysis to combine all similar flows based on the attributes of the flow identifiers [228]. It is acknowledged that no IDS can efficiently detect abnormal instances when depending on only single attributes rather than generating additional features by incorporating several attributes at a time [235].

We design a new aggregator module based on the theory of flow-level analysis to reduce the computational times required by resources and address the aforementioned limitations of existing network flow analysis tools. Figure 4.1 presents the proposed aggregator module which consists of four main components. The first is a data collector which gathers network packets at the choke points of a network, for example, through switches or routers. These packets are then stored in a MySQL database for handling structured big data which can easily be installed on any platform.

We aggregate network flows using MySQL's standard functions [236] which group them based on more than one attribute of the flow identifiers rather than on only one as do existing network flow aggregation tools. Extracting such flow features offers more information about a network's normal and attack activities,

and assists in improving the performance of DE. Grouping the data of source and destination IP addresses provides a better indication of the number of flows occurring between two hosts rather than focusing on only each side separately. To accumulate all possible combinations of the flow identifiers to obtain the relevant features, we apply the ‘count’ aggregate functions which have the following properties of normal and abnormal flows:

- ***select COUNT(*) as flows, srcip, dstip from network_data group by srcip, dstip;***
- ***select COUNT(*) as flows, srcip, srcport from network_data group by srcip, srcport;***
- ***select COUNT(*) as flows, dstip, dsport from network_data group by dstip, dsport;***
- ***select COUNT(*) as flows, srcip, dstip, dsport from network_data group by srcip, dstip, dsport; and***
- ***select COUNT(*) as flows, srcip, proto from network_data group by srcip, proto.***

In the above queries, *flows* indicates the number of flows occurring between any two attributes, *srcip* the source IP address, *dstip* the destination IP address, *srcport* the source port address, *dsport* the destination port address and *proto* the protocol type. Any of these queries returns the number of flows taking place between the attributes of aggregation. However, as building a set of distinct flows is not easy, many aspects should be considered, including the fields in the datagram header, the timers and counters with parameters that rely on the header fields, resource depletion and the time of day. A flexible flow discriminator is extremely necessary because the properties of IP and internet traffic change over time. Developing a

dynamic aggregator module for determining distinct and updated flows is important for tracking the non-stationary characteristics of IP traffic and deploying an online ADS.

We use sampling and ARM techniques to select significant network flows because they have their own advantages, as detailed in subsections 4.3.1 and 4.3.2, respectively. It is very important for the design of an online ADS to obtain aggregated flows which do not have duplicated or missing values in order to improve the performance of the ADS when processing in real networks. Finally, these flows are sent to the DE method that can distinguish between abnormal and normal instances.

4.3.1. Sampling Techniques

Techniques for sampling network flows, which decrease the burden of handling large volumes of data flows during their collection, storage and analysis, are commonly chosen to reduce the data input to data mining and ML algorithms. Packet sampling is considered a foundation for a broad range of network management, monitoring and engineering applications. It provides a dynamic perception of a network's activities through an in-depth analysis of its packet headers. This packet information can be clustered to establish different network traffic statistics, aggregates, estimations and detection of attacks, for example, counts, sizes and inter-arrival times of packets as well as protocol distributions. Sampling techniques group packet information based on different metrics in order to form some clusters with data significant for identifying certain network problems [237]. The accuracy of packet sampling relies on its sampling rate, application and periodicity of the measured criteria.

The key benefits of these techniques over complete enumeration are their lower processing times and implementation costs. They have been used for aspects of network management, for example, IDSs, traffic measurement and reporting, and

traffic characterisations [227, 234]. However, packet sampling techniques have the limitation of data distortion as they sometimes omit relevant instances that could help to generate patterns contained in the data collection [232].

To address this issue, Carela et al. [238] analysed their effects on the accuracy of network traffic classification using ML algorithms and proposed a solution for reducing their impact. The authors claimed that using packet sampling rather than conventional packet analyses, such as port-based and DPI methods, to extract network flows in the training phase of a ML algorithm would improve the accuracy of detection. Zhang et al. [239] developed two methods for identifying a high-rate flow which should have short processing times as well as low memory and CPU costs. The first is a fixed sample size test which depends on a user-specified accuracy to evaluate its efficiency and the second a truncated sequential probability test which uses sequential sampling to eliminate low-rate flows.

Lee et al. [240] suggested a sampling method for network data in which flows come from the same application session, with the experimental results showing that its processing time and accuracy are better than those of the Sampled NetFlow. Shirali-Shahreza et al. [241] proposed a simple random sample (SRS) method that dynamically collects similar flows and depends on counting any received packets. Ha et al. [242] suggested a traffic sampling mechanism for software-defined networking which inspects malicious traffic whereas maintaining aggregated flows of sampled network traffic is outside the inspection capability of the IDS. In this chapter, we investigate different sampling techniques to determine which could improve the accuracy of DE with a lower processing time.

We discuss the main categories of sampling used in practice [229] and select the most suitable for the proposed aggregator module based on the following points.

- **Systematic Sampling** – which is also called interval sampling, depends on arranging the data points of a dataset according to a particular ordering of

intervals of equal size. It involves a random starting point and then proceeds to select other elements from these intervals.

- **Stratified Sampling** - handles a collection of data by dividing it into independent subsets called ‘strata’. It chooses a random element from a stratum to create a stratified sample using a simple random sampling method.
- **Cluster Sampling** - groups a dataset based on the distances, periods or probabilities between its data points with a set of clusters, the elements of which are randomly selected according to a sampling rate of each cluster.
- **Multi-stage Sampling** - organises a dataset into groups. It arbitrarily selects some groups and then chooses the same number of elements. This technique is considered an extension of cluster sampling which arbitrarily groups a data collection and then selects its observations.
- **Simple Random Sampling (SRS)** - randomly chooses a sample in which no observations of a given size are included more than once and all subsets of the observations have equal probabilities of selection. Moreover, any given pair of values has the same probability of selection as any other pair which minimises data bias and simplifies analysis. Because n samples are chosen from N packets, it is sometimes called n -out-of- N sampling [237] and, when applied to network data, each network packet has an equal probability of being selected. This technique arbitrarily generates n dissimilar numbers in the range of 1 to N and then selects all packets with a packet location equal to one of these numbers, with this step repeated for every N packet. Examples of this technique for network tools are sFlow and Netflow which use 1-out-of- N sampling procedures.

Given the high speeds and large numbers of flows in current networks, choosing only relevant flows is vital for developing a lightweight and scalable ADS. For network data, we apply the SRS technique due to its advantages [229]. Firstly, it can minimise data bias which simplifies its analysis, that is, when selecting a

subset of a data collection, the variances between the observations in this subset indicate those in the whole collection. Secondly, it can reduce computational costs because it selects only a portion of the data collection which could contain different patterns. To use this technique, it is difficult to specify the number of observations to be analysed each time, that is, the sample size.

For this technique, we adopt the sample size according to the mode (i.e., online or offline) to monitor and analyse network data. In the online mode, we use the concept of the sliding window [243] which selects the number of flows for each particular time of network traffic; for example, the aggregator module collects and analyses network flows every 1 or 2 minutes. In the offline mode, we select a specific number of observations from a dataset to be analysed at a time; for example, the aggregator module sequentially analyses every 100 records. A discussion of these techniques and the results obtained from them are provided in subsection 4.6.1.

4.3.2. Association Rule Mining (ARM)

We also propose handling large flows of network data using the ARM technique [89] to ensure that all possible observations are collected and address the limitation of data distortion that occurs in sampling techniques. This is a data mining method used to estimate the correlation between two or more variables in a dataset by determining the strongest rules that occur between their values.

To describe the ARM methodology, let $r = \{f_1, f_2, f_3, \dots, f_N\}$ be a set of variables/features and D be a dataset consisting of T transactions $\{t_1, t_2, t_3, \dots, t_N\}$. Each transaction ($t_j, \forall 1 \leq j \leq N$) has a relationship between features, where $t_j \subseteq r$. The association rule ($f_1(\text{antecedent}) \Rightarrow f_2(\text{precedent})$) subjects to the constraints of (1) $\exists t_j, f_1, f_2 \in t_j$ (2) $f_1 \subseteq r, f_2 \subseteq r$, and (3) $f_1 \cap f_2 = \emptyset$.

There are two measures, namely, support and confidence, for applying ARM on a data collection which are used to estimate the strongest association rules that have different patterns in the data. The former computes the frequency of feature

values, that is, the proportion of association of each rule (equation (4.1)) while the latter computes the frequency of a precedent if the antecedent has already taken place (equation (4.2)).

$$sup(f_1 \Rightarrow f_2) = \frac{|\#t_j | f_1, f_2 \in t_j|}{N} \quad (4.1)$$

$$conf(f_1 \Rightarrow f_2) = \frac{|\#t_j | f_1, f_2 \in t_j|}{|\#t_j | f_1 \in t_j|} \quad (4.2)$$

The ARM technique discovers all the frequent itemsets and generates the strongest rules in them, with the strongest rules in a data collection (D) defined as: 1) the estimated support of a rule is greater than a user-specified minimum support ($sup \geq minsup$); and 2) the estimated confidence of a rule is greater than a minimum confidence threshold ($conf \geq minconf$). Similarly, in the sampling technique, we select the sample size analysed using ARM according to the mode of network data processing, either online or offline. In the online mode, we use the sliding window to choose the number of flows for each specific time of network traffic and, conversely, in the offline mode, we choose a particular number of observations from a dataset to be handled each time. We suggest a Central Point (CP) function with ARM to correlate the most repeated values of each attribute to reduce processing times. This function calculates the mode of each value, numeric or categorical, in which the most frequent values of each attribute occur in a dataset, as in Example 1 [244]. Then, ARM is used to create the highest associated values of network flows.

Example 1: Computations of modes of attribute values

1. Numeric values	X= {1, 2, 1, 1, 3.2, 1}	> mode = {1}
<hr/>		
2. Categorical values	Y={‘tcp’, ‘udp’, ‘tcp’, ‘udp’,‘udp’}	> mode = {‘udp’}

Algorithm 4.1 presents the CP of the attribute values (i.e., mode). In lines 1 and 2, loops are assigned to all the attribute values, lines 3 to 12 check whether these values are categorical or numerical and then the mode for each data part is computed. Lines 13 to 17 repeat these steps until all the sample sizes in the data collection are dealt with. Line 18 retrieves the mode of the sample sizes for input to calculate the ARM values using the Apriori algorithm 4.2 [95]. There are two steps in this algorithm: the first iteratively discovers all the frequent itemsets which satisfy $sup \geq minsup$; and the second generates all the association rules that satisfy $conf \geq minconf$.

By applying these two algorithms, we can select only the relevant network flows aggregated from the MySQL database which can reduce the computational cost of resources as we employ the CP function to collect the flows with the most repeated values from the flow identifiers. As it is easy to apply this technique to collect these flows in the shape of association rules, we extend it to build a new FS method that extracts relevant features, as discussed in subsection 4.5.1. Then, these features are passed to the DE approach which distinguishes between legitimate and suspicious observations. While these observations contain all the information of the raw packets, the OSI model mines only the relevant features by analysing its protocols and services.

Algorithm 4.1 Central points of attribute values

Input: d dataset, p
Output: centres

```
1: for ( $r = 1$  to  $\text{length}(\text{row})$ ) do
2:   for ( $c = 1$  to  $\text{length}(\text{column})$ ) do
3:     if ( $d[r][c] \neq \text{categorical}$ ) then
4:        $\text{pre}[r][c] = \text{mode}(d_{1:p})$ 
5:     else if ( $d[r][c] \neq 0$ ) then
6:        $\text{centres}[r][c] = +\text{pre}[r][c]$ 
7:     else
8:        $\text{pre}[r][c] = \text{count}(d_{1:p})$ 
9:       if ( $\text{pre}[r][c] > \text{pre}[r][c+1]$ ) then
10:         $\text{centres}[r][c] = +\text{pre}[r][c]$ 
11:       end if
12:        $p = p - 1$ 
13:        $\text{row} = \text{row} - (\text{row}/p)$ 
14:     end if
15:   end for
16: end for
17: return ( $\text{centres}$ )
```

Algorithm 4.2 Steps of Apriori algorithm

Input: T transactions, α threshold of minsup, minconf
Output: L_K frequent itemset of size K

```
1:  $L_1 = \{\text{large 1-itemsets}\}$ 
2:  $k = 2$ 
3: while ( $L_{K-1} \neq \emptyset$ ) do
4:    $C_K = \text{Generate } (L_{K-1})$ 
5:   for (each condidate  $t \in T$ ) do
6:      $C_t = \text{subset } (C_K, t)$ 
7:     for (each condidate  $c \in C_t$ ) do
8:        $\text{count}[C] = \text{count}[C] + 1$ 
9:     end for
10:   end for
11:    $L_K = \{c \in C_K \mid \text{count}[C] \geq \alpha\}$ 
12:    $K = K + 1$ 
13: end while
14: return ( $L_K$ )
```

4.4. Network Feature Creation

To build an effective ADS, it is important to create relevant features from raw network packets while sniffing them. These features should have the properties of legitimate and suspicious activities occurring in a network system. To extract them from network traffic, it is vital to collect network flows at the destination nodes using ingress routers, gateways or switches to ensure that only relevant packets, which include information about network protocols and services that can be transmitted between network endpoints, are collected.

The performance of any ADS could be improved by analysing the protocols and services used to mine only the important features with different patterns through transmitting and receiving network flows. The relevant network features are collected from network flows which have unidirectional or bidirectional sequences of packets between any two endpoints (i.e., client-to-service or vice versa). The most important fields in a flow are its source/destination IPs and ports, the first and last times it is received, and its protocols, type of services and bytes transferred.

These fields are used to mine information about protocols and services based on the potential analysis of the OSI model; for example, when we extract information about any application protocol, such as FTP, and all services, our path for obtaining them is (IP -> TCP -> FTP). Table 4.1 shows the new features added to the 49 in the UNSW-NB15 dataset in Chapter 3 to provide a wide variety of features regarding all the information included in the raw packets. They are generated from an in-depth analysis of the protocols and services of the application layer of the OSI model, in particular HTTP, DNS, FTP, SMTP, SSH and SNMP, which are commonly used to send and receive data between any two endpoints, due to their significant functions in network systems, as discussed in Appendix A. The aim of this analysis is to ensure that the important features in the FR stage that can help DE approaches effectively and efficiently differentiate between normal and abnormal patterns are selected.

Table 4.1: New features created from analysing application protocols of TCP/IP model

No.	Type	Name	Description
50	Integer	len_httpurl	Length of URL from HTTP protocol
51	Integer	len_httphost	Length of values of HTTP host header
52	Integer	len_dnsquery	Length of subject of DNS query
53	Integer	len_dnsanswer	Length of resource descriptions in answer to query
54	Integer	dns_qclass	Length of qclass value specifying class of query
55	Integer	dns_qtype	Length of qtype value specifying type of query
56	Integer	len_smtpsubject	Length of SMTP contents of subject header
57	Integer	sshstatus	Status of SSH - either running or not
58	Integer	snmp_dur	Record duration of SNMP protocol
59	Integer	snmp_getrequest	Length of Getting Request message of SNMP protocol

Given the existence of ubiquitous computing, internet applications and distributed software systems, which are increasingly being installed to provide online services, face massive risks from intrusive activities. Internet protocols are currently convenient because they provide access to services and information anywhere, anytime [245] and have led to a substantial increase in the prevalence of computing resources due to users' dependence on these services and applications. Internet application resources involve web client systems, web server devices and database systems, and software interactions among them [246] occurring through using the functionalities of specific protocols, such as FTP, DNS and HTTP.

In the OSI model [226], seven layers are included in a networking system, each of which has a set of protocols that communicate with its counterparts in other networks; for instance, network *A* connects to network *B* via transport and internet layers. Internet protocols, particularly DNS and HTTP, interact directly with back-end database systems and client-server applications to store user and network activities. The DNS is a crucial protocol for most internet services and

the HTTP a staple data communication protocol for the World Wide Web [247]. Attackers breach users' privacy by exploiting the vulnerabilities of these protocols and, in the following subsection, because of their significant roles, we analyse these protocols in depth and suggest new features that can help to identify the malicious events they face.

4.4.1. Proposed DNS and HTTP Features

The DNS and HTTP are two fundamental protocols for internet applications which interact directly with user data [247]. The former is a hierarchical distributed system connected to the internet or a network which maps domain names that can be simply memorised by users to numeric IP addresses using authoritative name servers for each domain. The purpose of the DNS protocol is to prevent any conflict among reserved domain names and facilitate computer services and devices [248]. The HTTP is an application protocol that interchanges hyperlinks of users between nodes, including structured text, with its functions dependent on a request-response service which occurs in a client-server system. In other words, a client submits a HTTP request message to its server which returns a response to the client and provides resources such as HTML files [249].

Several research studies have been conducted to design features for the DNS and HTTP to identify malicious events facing these protocols; for example, Marchal et al. [250] developed a scalable distributed IDS based on collecting data from honeypot and DNS data, HTTP traffic and IP-flow records. However, they did not provide the features used for its implementation and the analysis of those data was executed in an offline manner with other systems. Valdes et al. [251] linked the alert actions of similar features in the DNS and HTTP by grouping their triggering events to recognise similar attacks and raise different alerts. Then, in [252], alarms from several NIDS are correlated in one system to accomplish high-level

explanations of attacks. Nonetheless, this technique does not provide a realistic evaluation of the proposed NIDS as how those systems are correlated is not described.

In contrast to a deep packet inspection, Nassar et al. [253] proposed a new NIDS for monitoring a flow-based network in order to improve its attack detection. The authors claimed that this system is more effective and secure for analysing data sources although a flow-based inspection without inspecting the packet payload does not provide complete information about the network traffic. Sitaram et al. [254] stated that network-based IDS problems can be solved by large cloud providers. They built a NIDS for handling big data network streams using existing big data analysis techniques, such as Hadoop, and network monitoring tools called Packet-Pig [255] which can be used to inspect packets, analyse deep networks and even capture a full packet using the Hadoop tool. Although this study, which primarily shows the efficacy of using clustering algorithms in DE, is related to our work, specifically regarding extracting features from network traffic, our research focuses mainly on extracting features from the DNS and HTTP due to their effective roles in internet applications.

Attackers exploit the vulnerabilities of victims using various deceptive approaches [256–258], for example, they create untrustworthy accounts to target users who have limited experience in registering their information about internet applications. Internet assailants breach the weak points in websites and services using a range of exploitation techniques, for instance, zero-day, polymorphic code, DNS spoofing, DoS, exploits, URL interpretation and stealth attacks [259, 260]. An ADS detects malicious activities in network traffic using a set of features from the seven layers of the OSI model [226], with an effective system acting as an ongoing deterrent against spoofing and stealing information based on selecting relevant features that affect the discrimination between normal and abnormal network records.

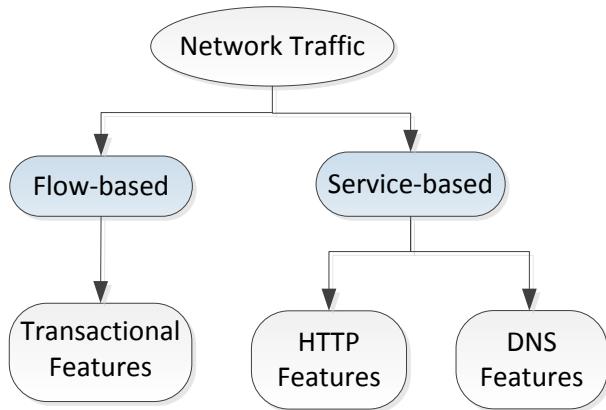


Figure 4.2: Proposed features of DNS and HTTP

We propose a methodology for extracting and generating a set of features from the DNS and HTTP to detect malicious observations using DE approaches. The tools for capturing these features from network traffic are discussed below while the features used to build an effective ADS that can detect attacks which attempt to exploit the DNS and HTTP, including flow-based and service-based features, are depicted in Figure 4.2. The flow-based features are extracted from both protocols to carefully analyse their fundamental header information to select important features. These features are widely used in online analyses of network traffic and offer high accuracy using ML algorithms if their values for legitimate observations differ from those for suspicious activities [250].

The flow-based features include the five flow identifiers of the source-destination IPs/ports, protocol types and the flow statistics called transactional features. The Tcpdump tool captures the header information of the packets to sequentially generate the flow identifiers, with the transactional features created from the interactions of these identifiers with respect to the times of the packets. These features depend on the consistency of the flow times (i.e., feature 6 ltime in Table 4.2)

for maintaining the online detection of attacks using the throughput of network traffic.

The service-based features comprise intrinsic information extracted from the application layer of the OSI model for dealing directly with the user information stored in the DNS and HTTP. Firstly, DNS features are elicited from common DNS query responses and domain names, with statistical features generated from this protocol, for example, the lengths and means of the query and answer attributes. Secondly, HTTP features are generated by analysing the HTTP requests and responses from basic information of the interacting clients and servers, such as the length and mean of the method and URL attributes. It is observed that the statistical features, which define the core information of these protocols, help to discriminate between normal and malicious instances and are easily created during online processing when implementing ML techniques to recognise malicious instances in network traffic.

A. Tools for generating DNS and HTTP Features

To create the features of the DNS and HTTP from network traffic, the three tools, tcpdump, Bro-IDS and a new extractor module depicted in Figure 4.3, are used². Firstly, the tcpdump tool captures the network traffic in the format of pcap files to analyse network flows. Secondly, the Bro-IDS tool analyses the pcap files to extract the flow-based features and general information from the DNS and HTTP stored in log files. The HTTP log file contains information of the HTTP request-response pairs and all appropriate metadata about the protocol while the DNS log file contains DNS queries and their associated responses. These files are stored

²

• A part of the study in this chapter has been submitted in.
Moustafa, N., Creech, G. and J. Slay. “Detecting Malicious Activity of DNS and HTTP Protocols: An Ensemble Learning Framework using Proposed Statistical Features”, Journal of Computers & Security, ELSEVIER, 2017, “in press”.

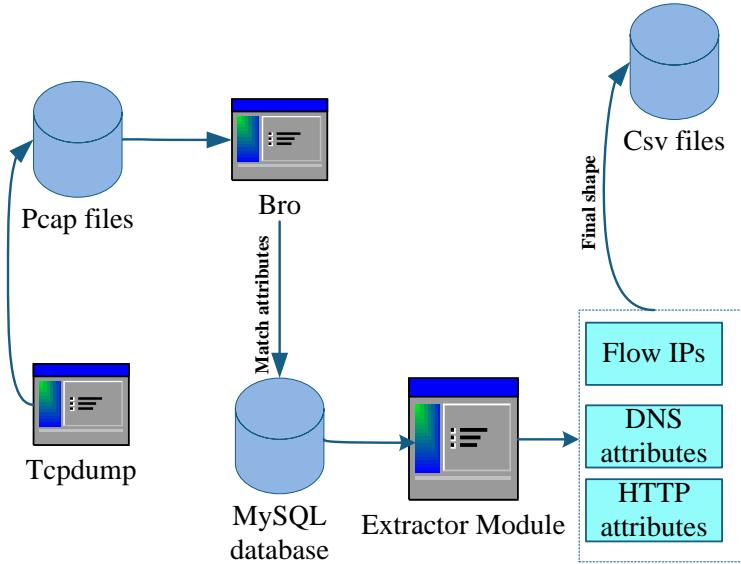


Figure 4.3: Tools used to create DNS and HTTP features

in a MySQL database for ease of generating additional statistical features and labelling instances as either normal or anomalous.

Thirdly, the newly proposed extractor module simultaneously begins to create additional statistical features from those extracted by the Bro-IDS tool and stored in the MySQL database. Using this configuration, the proposed features can be easily established through online processing without losing the packet information. To illustrate the processes in the extractor module, given a data source with a set of r records, each of which includes a set of features $\{f_1, f_2, f_3, \dots, f_N\}$, where N is the number of features. The mean (\bar{f}) and length ($\lambda(f)$) of feature values are computed using equations (4.3) and (4.4), respectively.

$$\bar{f}_k = \frac{\sum_{i=0}^k \forall f}{k} \quad (4.3)$$

$$\lambda(f) = \sum_{i=0}^r C(f) \quad (4.4)$$

where k is a certain number of rows and $C(f)$ a count of the numbers or letters of the feature values, with these two equations applied to create the proposed statistical features.

Since statistical behaviours cannot be computed for only one record, we apply the statistics for every sequentially ordered 100 records with respect to the session times of the packets. The connections between these 100 records identify the patterns of normal and attack activities and efficiently define the records' correlations [42, 261]. Then, the classification techniques use the statistical feature values to discriminate between legitimate and suspicious records. The steps for constructing the extractor module are provided in Algorithm 4.3. For every 100 records stored in the database, the lengths, rates and/or means of the existing feature values, especially categorical features, such as *query*, *answer*, *method* and *host*, are computed to create the new statistical features.

Table 4.2 lists the proposed features elicited, with 7 to 11, 17 to 25 and 32 to 36 obtained from the new extractor module and the remainder mined by the Bro-IDS tool.

To generate the DNS and HTTP features and observations, we analyse the UNSW-NB15 dataset which contains 349,000 DNS and 319,000 HTTP records, with the distributions of normal and attack vectors provided in Table 4.3. In

Algorithm 4.3 Generating statistical features of DNS and HTTP

```
Input: f [N][R], n=0, rate=0 // n is a counter of similar feature values  
Output: f_n[], f_rate[] //arrays for storing feature  
1: for (i to r) do  
2:   for (j to k) do  
3:     if (f[i][j] == f[i][j+1] then  
4:       n++  
5:     else  
6:       n=1  
7:     end if  
8:   end for  
9:   f_n[i]=n  
10:  j=m // m number of records (i.e., 100)  
11:  rate = n / m  
12:  f_rate [i]= rate  
13: end for  
14: return (f_n, f_rate)
```

order to detect abnormal observations, we apply some existing ML techniques to evaluate the capabilities of the proposed features to detect attacks that expose internet applications in the protocols.

4.4.2. Proposed Ensemble Method for detecting DNS and HTTP Malicious Activities

In order to efficiently detect DNS and HTTP malicious observations, we propose an ensemble method that consists of the three supervised ML algorithms, namely Naïve Bayes (NB) [188], Decision Tree (DT) [187, 189] and Artificial Neural Network (ANN) [187]. The algorithms are integrated as an ensemble method using the Adaboost model [262, 263] for the design of an adaptable NIDS that can identify malicious instances that breach DNS and HTTP network systems. Each algorithm is considered a weak classifier with its findings not considered sufficiently high compared with those of the ensemble method while the Adaboost model distributes network instances that can be successfully classified by these algorithms.

Table 4.2: Proposed features of DNS and HTTP Protocols

No.	Name	Type	Description
Flow Features			
1	srcip	N	Source IP address
2	sport	I	Source port number
3	dstip	N	Destination IP address
4	dsport	I	Destination port number
5	proto	N	Protocol type
6	ltime	T	Last time of connection
Transactional Features			
7	ct_dst_ltm	I	Number of connections to the same destination (3) in 100 records according to the last time (6)
8	ct_src_ltm	I	Number of connections of the same source (1) in 100 records according to the last time (6)
9	ct_src_dport_ltm	I	Number of connections of the same source address (1) and the destination port (4) in 100 records according to the last time (6)
10	ct_dst_sport_ltm	I	Number of connections to the same destination (3) and the source port (2) in 100 records according to the last time (6)
11	ct_dst_src_ltm	I	Number of connections of the same source (1) and the destination (3) in 100 records according to the last time (6)

To clarify why these classification algorithms are chosen for the proposed ensemble method, the correlation measure and scatter plots are major factors. According to the results discussed in subsection 4.6.2, the correlation measure demonstrates that the proposed features are positively linearly interrelated and the scatter plot represents a straight line with a positive trend between the features of DNS and HTTP protocols. This led to deciding which of the classification algorithms would be selected based on the potential procedures required to implement them, such as distance- and probability-based learning; for example, let there be two vectors (v_1 (normal) and v_2 (abnormal)) with the difference between them very small, i.e., $v_1 - v_2 \simeq 0$. As each algorithm is designed using a specific kernel function, such as a probability, weight or feature value, we integrate these

DNS Features			
12	query	N	Domain name subject of the query
13	q_class	I	Value specifying the query class
14	q_type	I	Value specifying the query type
15	answers	N	List of resource descriptions in answer to the query
16	ttls	I	Caching intervals of the answers.
17	len_qry	I	Length of the query (12)
18	len_ans	I	Length of the answers (15)
19	mean_class	F	Mean of the class in 100 records according to the last time (6)
20	mean_type	F	Mean of the type in 100 records according to the last time (6)
21	ct_srcqry	I	Number of the srcip (1) and the query (12) in 100 records according to the last time (6)
22	avg_ttl_ltm	F	Average value of the ttls (16) values in 100 records according to the last time (6)
23	ct_dm_src	I	Number of domains (12) to same srcip (1) in each 100 records according to the last time (6)
24	ct_dm_dstip	I	Number of domains to same dstip (3) in each 100 records according to the last time (6)
25	dm_ratio	F	The percentage of the domain in each 100 records according to the last time (6)

three types of functions to classify network data with slight variances between its normal and suspicious instances.

Therefore, we select the ANN, NB and DT because they can classify such observations effectively. The ANN relies on weighting each feature to produce accurate linear class bounds. The NB depends on computing the posterior probability that estimates all the possible data distributions to discover slight variances between each class label in the training phase. Finally, as the DT uses the feature values to divide the feature space into regions with mainly the same label, it can simply find slight variances between feature observations.

HTTP Features			
26	method	N	HTTP Request Method e.g., GET, POST, HEAD
27	host	N	Value of the HOST header
28	trans_depth	I	Pipelined depth into the connection
29	url	N	URI used in the request
30	req_body_len	I	Actual uncompressed content size of the data transferred from the client
31	res_body_len	I	Actual uncompressed content size of the data transferred from the server
32	len_host	I	Length of the HOST header value
33	len_url	I	Length of the URL
34	ratio_mth_host	F	Percentage of the method (26) with same the host (27) in each 100 records according to the last time (6)
35	ratio_mth_url	F	The percentage of the method (26) with same the url (29) in each 100 records according to the last time (6)
36	ratio_host_url	F	Percentage of the host (27) with same the url (29) in each 100 records according to the last time (6)
Type- I: Integer, F: Float, N: Nominal, T: Timestamp			

Table 4.3: Types of DNS and HTTP records

	DNS	HTTP
Normal	345561	295689
DoS	130	1814
Exploits	689	15804
Fuzzers	1008	1105
Generic	1563	1885
Reconnaissance	49	1983
Analysis	NF	369
Backdoor	NF	212
Worms	NF	139
• NF denotes not found		

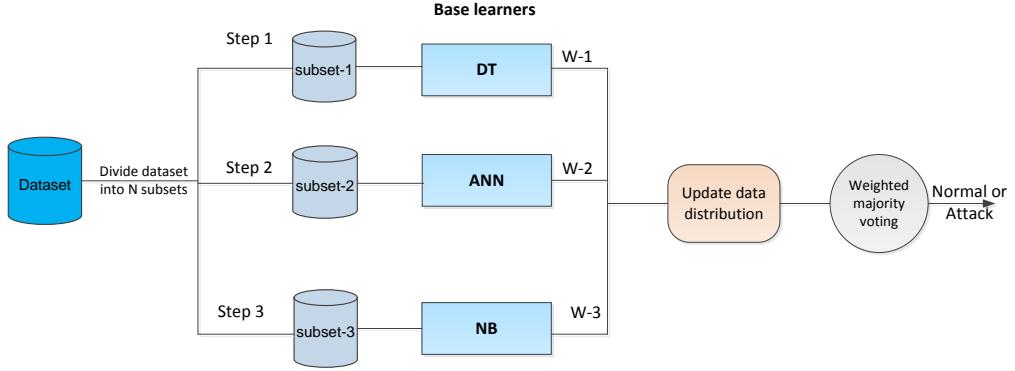


Figure 4.4: Adaboost flowchart

To distribute network data among these algorithms, we apply the boosting concept [262, 263] that establishes multiple base learners by consecutively reweighting the instances in the training phase. Each instance incorrectly classified by the first base learner obtains a higher weight in the next round in the training phase. The main idea of boosting is to regularly use a base learner to change forms in the training phase, thereby creating a sequence of base learners for a selected number of iterations. In more detail, all the instances are initiated with equal weights and then each iteration is used as a base learner for them. To distribute the data, the weights of the correctly classified observations decrease while those of the incorrectly classified observations increase. The final model attained by the boosting concept is a linear combination of many base learners weighted by their own performances.

In this chapter, we use the Adaboost technique [262, 263], which is the most widely applied to ensemble learning methods, to distribute input data on the classification algorithms used. The flowchart and steps of the Adaboost technique presented in Figure 4.4 and Algorithm 4.4, respectively, reveals how we can apply it to the proposed DNS and HTTP data sources. The results from the proposed method show better performances in terms of the DR and FPR than each algorithm used, as detailed in subsection 4.6.3.

Algorithm 4.4 Steps of Adaboost model

Input: Dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, base learning techniques L , Number of iterations I

- 1: $D_i(j) = 1/m$ // initialise the weight distribution
- 2: **for** (i to T) **do**
- 3: $h_i = L(D, D_i)$ // train a base learner h_i from D using distribution D_i
- 4: $\epsilon_i = P_{i, D_i}[h_i(x_i \neq y_i)]$ // estimate the h_i error
- 5: $w_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ // determine the h_i weight
- 6: $D_{i+1}(j) = \frac{D_i(j)}{Q_i} * \left\{ \begin{array}{l} \exp(-w_i) \text{ if } h_i(x_i) = y_i \\ \exp(w_i) \text{ if } h_i(x_i) \neq y_i \end{array} \right\}$ // update the the distribution, where Q_i is computed by (6)
- 7: $Q_i = \frac{D_i(j) \exp(-w_i, y_i, h_i(x_i))}{Q_I}$ // normalisation factor which enables D_{i+1} to be a distribution end
- 8: **end for**
- 9: **return** ($H(x) = \text{sign} \sum_{i=1}^I w_i h_i(x)$)

4.5. Role of Feature Reduction (FR)

FR, which is defined as a means of removing unimportant or noisy features from a data collection, is of two types, FS and feature extraction (FE) [44]. Firstly, FS eliminates irrelevant or redundant features from a given dataset and then FE converts the high-dimensional space data into a lower-dimensional space. FS methods have been used in the field of NIDS to eliminate unnecessary features. Their goal is to decrease the computational cost of DE methods, remove redundant information, improve the accuracy of DE and help to analyse the norm of network data [26, 264], and can be categorised as filter, wrapper and hybrid methods. The filter method does not apply any classifier to filter out irrelevant and duplicated features but uses the underlying characteristics of the training data to evaluate the best features using some independent measures, such as of distance and correlation, and does not require a long execution phase. A wrapper method employs a classifier to assess the goodness of features while a hybrid technique combines filter

and wrapper approaches. However, as both require a long execution time [88], we focus on filter methods for building an online ADS.

4.5.1. ARM Feature Selection Method

Because network systems contain a great deal of information provided by their protocols and services, network flows have many features, some of which are redundant or irrelevant. It can be seen that redundant features are a major reason for increases in the false alarm rate (FAR) and decreases in the DR. We propose an ARM FS method that generates the strongest itemsets of features by computing the support of confidence of the rules, as explained in subsection 4.3.2.

We suggest using Algorithm 4.5 to select the important features using the ARM methodology³. Line 1 executes Algorithm 4.2 which generates all the possible rules in a data collection. From lines 2 to 13, if the rules do not satisfy the constraints, they are removed, otherwise the support and confidence measures are computed using equations 4.1 and 4.2, respectively. In lines 12 and 13, the rules are chosen using the estimated support and confidence. In line 16, all the rules are ranked in descending order based on the estimated support and confidence measures. From lines 17 to 22, the significant features are selected when the constraints of $sup \geq minsup$ and $conf \geq minconf$ are achieved with respect to the number of features [265].

³

- A portion of the study provided in this chapter has been released in the following.
Moustafa, N., & Slay, J. (2015). A hybrid feature selection for network intrusion detection systems: central points and association rules, Proceedings of the 16th Australian Information Warfare Conference (IWAR 2015), Perth, Australia, November 2015.
- Moustafa, N., & Slay, J. (2015). The significant features of the UNSW-NB15 and the KDD99 Data sets for Network Intrusion Detection Systems. Proceedings of the 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS 2015), collocated with RAID2015, Kyoto, Japan, November 2015.

Algorithm 4.5 Proposed ARM feature selection method

Input: D (training set), minsup (minimum support threshold), minconf (minimum confidence threshold) , C (class label), X (number of required features)

Output: F (feature subset)

```
1:  $R = \text{Apriori}(D, \text{minsup}, \text{minconf}, C)$ 
2: for ( $i=1$  to  $\text{length}(R)$ ) do
3:   if ( $R[i]==R[i+1]$ ) then
4:      $\text{count}[i] = \text{count}[i] + 1$ 
5:   else
6:      $\text{count}[i]=1$ 
7:   end if
8:    $\text{filter\_R}[i] = R - R[i]$ 
9: end for
10: for (( $j=1$  to  $\text{length}(\text{filter\_R})$ ) do
11:   if ( $\text{count}[j]<=1 \text{ || } R[j] \notin C$ ) then
12:      $\text{sup}[j] = \text{count}[j] / \text{length}(\text{filter\_R})$ 
13:      $\text{conf}[j] = \text{count}[j] / \text{length}(D[j])$ 
14:   end if
15: end for
16:  $\text{Sort}(\text{filter\_R}, \text{sup}, \text{conf})$ 
17: for ( $m=1$  to  $X$ ) do
18:   if (( $\text{sup} >= \text{minsup} \text{ && } \text{conf} >= \text{minconf}$ ) then
19:      $F = F + (\text{extracted\_features}(r), C)$ 
20:   end if
21: end for
22: return ( $F$ )
```

Figure 4.5 presents an example that illustrates the execution of Algorithm 4.3 using the UNSW-NB15 dataset. In the rule-set, the values of Minconf and Minconf are set to 0.4, with only the rules that satisfy these values selected. Then, in the feature-set, only two features ($X = 2$) are selected for each class type in the dataset based on the setting parameters. If a set of features is selected for all types in the dataset, we sort all the ranked features and base the cut-off on only the number of features (X).

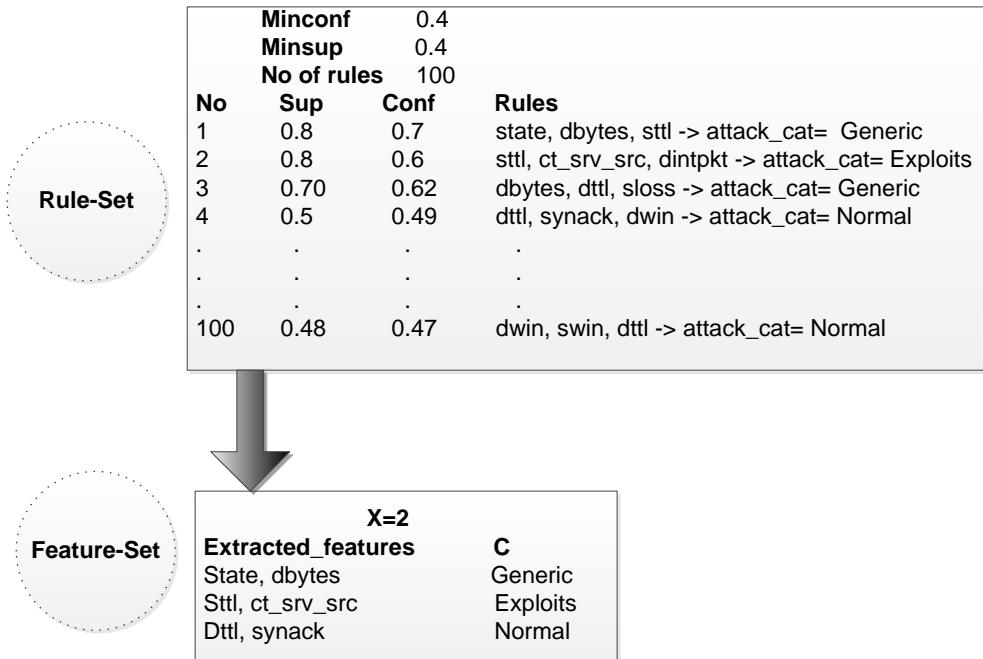


Figure 4.5: Example of ARM FS method using UNSW-NB15 dataset

4.5.2. Principal Component Analysis (PCA)

The PCA ranks a set of variables/features based on the highest variance for each variable and creates a new dimensional space of uncorrelated variables by removing those with low variances [90] which works well for data that are Gaussian distributed [91]. The first principal component of a transformation is the linear combination of the original variables with the highest variances while the other components with the lowest are sequentially omitted [90, 266]. To describe the PCA methodology, let a dataset (X) be a set of records (x_1, x_2, \dots, x_d) with $nx1$ vectors such that each record has a vector of length n. The mean for each variable is defined by $\mu = 1/d \sum_{i=1}^d x_i$ and a deviation from it denoted by $\varphi_i = x_i - \mu$.

$$cov_{dxd} = \frac{1}{n} \sum_{i=1}^n \varphi_i \varphi_i^T = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^T (x_i - \mu) \quad (4.5)$$

In equation (4.3), the covariance matrix ($cov()$) is computed to measure the score of the linear correlation between two variables, where φ_i^T is the transposed matrix of φ_i . A detailed description of covariance can be found in [90].

To reduce high dimensions using PCA, the eigenvalues and corresponding eigenvectors of the cov are always computed using SVD [90, 91]; for example, assume that $(k_1, q_1), \dots, (k_g, q_g), \dots, (k_l, q_l)$ are l eigenvalue-eigenvector pairs of the cov . Then, the g eigenvectors corresponding to the largest eigenvalue, where g refers to the internal dimensions of the feature subspace, can be represented as

$$\frac{k_1 + k_2 + \dots + k_g}{\sum_{i=1}^l k_i} \geq w \quad (4.6)$$

where W is the ratio of the variance in the subspace to the total variance in the original space and the lxg matrix (N) in which the variables contain g . The projection of the principal components (F_i) from the original feature is

$$F_i = N^T \varphi_i = N^T(x_i - \mu) \quad (4.7)$$

The input to learning techniques in DE is the original features (x_i) or principal components (F_i) selected. The steps for applying the PCA technique to select the most important features of network data and their principal components are provided in Algorithm 4.6.

4.5.3. Independent Component Analysis (ICA)

The ICA is a generative model which generalises the PCA technique [91]. It mines unidentified hidden components from multivariate data, that is, linear mixtures of

Algorithm 4.6 Steps for applying PCA to reduce features

Input: feature set (F)

Output: subset of features (F'), principal components (FC)

- 1: compute covariance matrix of F
 - 2: compute eigenvectors and eigenvalues from covariance matrix
 - 3: sort eigenvalues in descending order and choose k eigenvectors that correspond to k largest eigenvalues, where k number of dimensions of new feature subspace ($k \leq d$)
 - 4: construct projection matrix (W) from selected k eigenvectors
 - 5: transform original dataset (X) via W to obtain k -dimensional feature subspace
 - 6: select highest-ranked N of F' and their principal components (FC)
-

some latent variables, using only the assumption that the unknown components are mutually independent and non-Gaussian from a statistical perspective. Independence, which indicates that the information carried by one component cannot be derived from any others [267], is broadly used in several applications, such as data compression and data analysis. It converts original features into a set of independent attributes by maximising the non-Gaussian data of new components.

While it is acknowledged that PCA does not work well for data that is not Gaussian distributed, ICA fits non-Gaussian data, which is the norm for network data, very well [91, 267], as explained in Chapter 2, Section 3.9. Therefore, we use ICA to select relevant features and compare its results with those obtained from PCA to decide which will be effective for the design of an online and scalable ADS.

The general ICA technique is given as

$$x = As + n \quad (4.8)$$

where x is a m -dimensional feature observation, s the vector of assumed n -dimensional independent components, A a constant $m \times n$ mixing matrix with

$m \geq n$ and n a noise term. As, in an ADS, we assume that network traffic is logged and labelled in a noise-free environment, the ICA model can be reformulated as

$$x = As \quad (4.9)$$

and

$$s = Wx \quad (4.10)$$

where W is the un-mixing matrix, called a mapping function, for projecting x to s .

ICA makes the best guesses of A and s given x with the constraint of maximising the non-Gaussian data so that these independent components are suitable representations of the data. To solve equation (4.8), we describe the constraint of maximising the non-Gaussian data as minimising the mutual information between n variables (s_i), where $i = \{1, \dots, n\}$, as

$$MI(s_1, s_2, \dots, s_m) = \sum_i H(s_i) - H(s_d) \quad (4.11)$$

where H is the differential entropy. Although the mutual information is usually non-negative, if it is zero, equation (4.9) is expressed as

$$\begin{aligned} \sum_i H(s_i) &= H(s_o) \Rightarrow \\ \sum_i \int p(s_i) \log p(s_i) ds_i &= \int p(s_o) \log p(s_o) ds_o \end{aligned} \quad (4.12)$$

and

$$p(s_d) = p(s_1)p(s_1)...p(s_m) \quad (4.13)$$

This means that the selected features are statistically independent. Because mutual information is the normal way of estimating the independence of variables, it could be used as the standard for determining the appropriate ICA transformation. The steps for applying the ICA technique to choose the most relevant features of network data and their components are presented in Algorithm 4.7.

Algorithm 4.7 Steps for applying fast-ICA to reduce features

Input: feature set (F)

Output: subset of features (F'), principal components (FC)

- 1: set initial weight vector (W) to generate each FC
 - 2: Compute $w^+ = Exg(wTx) - Eg'(wTx)w$
 - 3: Compute derivatives of contrast functions (G) for steps 4 and 5
 - 4: $g1(u) = \tanh(a_1.u)$
 - 5: $g2(u) = u \cdot \exp(-u^2/2)$
 - 6: Compute $w = w^+ / \|w^+\|$ (normalisation step)
 - 7: If not converged, go to step 2
 - 8: (converged if norm $(w_{new} - w_{old}) > \xi$ or norm $(w_{old} - w_{new}) > \xi$, where $\xi = 0.0001$)
 - 9: Apply above steps to generate N features of F' and their components (FC) of W
-

4.6. Experimental Results and Discussion

In this section, the empirical results for the two main aspects considered in this chapter are discussed. Firstly, those obtained from the aggregator module, i.e., the

sampling and ARM techniques that show the relevant observations in a dataset, are described in subsection 4.6.1. Secondly, the important features selected using ARM, PCA and ICA as well as their evaluations using some ML techniques are explained in subsection 4.6.2.

4.6.1. Aggregator Module

As current networks send and receive many flows, the aggregator module is important for the design of an online ADS. As the sequences in a flow often relate to different activities, such as retrieving a web page or chatting, although uncorrelated to each other, they might still belong to the same user. While monitoring network traffic, as an ADS cannot observe all flows due to the high speeds and large sizes of current networks, this leads to many packets which could contain malicious information being dropped. Therefore, we propose selecting only relevant observations, which do not have duplications or inappropriate activities, using the SRS and ARM techniques, and demonstrate their influence on the design of an online ADS.

Table 4.4 presents an example of a data sample with 10 observations, 5 normal (0) and 5 attack (1), taken from the UNSW-NB15 dataset for applying the SRS and ARM techniques. It shows the flow identifiers (i.e., *srcip*, *sport*, *dstip*, *dsport* and *proto*) and the label for each instance which describe the results obtained from these techniques for selecting relevant flows in real networks.

For the SRS technique, we select half the observations that could have suitable information about network flows, as listed in Table 4.5. Although, by selecting only half, relevant information that has some suspicious instances may be omitted, given the large number of flows, selecting only distinct observations will improve the performance of an online ADS because the statistical characteristics of the data sample and selected sample shown in Figures 4.6 and 4.7, respectively, are approximately similar.

Table 4.4: Example of data sample for applying SRS and ARM techniques

No.	Srcip	Sport	Dstip	dsport	Proto	Label
1	149.171.126.14	179	175.45.176.3	33159	Tcp	0
2	175.45.176.3	22592	149.171.126.16	143	Tcp	0
3	175.45.176.2	61809	149.171.126.19	161	Udp	0
4	175.45.176.0	45235	149.171.126.16	21	Tcp	0
5	175.45.176.0	15816	149.171.126.10	5060	Udp	0
6	175.45.176.0	3716	149.171.126.15	80	Tcp	1
7	175.45.176.2	7434	149.171.126.16	80	Tcp	1
8	175.45.176.0	16495	149.171.126.10	80	Tcp	1
9	175.45.176.2	9710	149.171.126.15	32780	Udp	1
10	175.45.176.1	15982	149.171.126.14	5060	Udp	1

Table 4.5: Selected sample using SRS technique

srcip	Sport	Dstip	Dsport	proto	label
175.45.176.0	16495	149.171.126.10	80	tcp	1
175.45.176.2	61809	149.171.126.19	161	udp	0
175.45.176.1	15982	149.171.126.14	5060	udp	0
149.171.126.14	179	175.45.176.3	33159	tcp	0
175.45.176.0	3716	149.171.126.15	80	tcp	1

The scatterplot matrices presented in these two figures demonstrate the statistical properties of the samples and their selected parts using the SRS technique which is a good way of determining if there are many linear correlations among features. This is particularly helpful for analysing specific features that could have similar correlations. In each figure, the features are written in a diagonal line from top left to bottom right and then plotted against each other, with the plots on the lower left-hand side mirror images of those on the upper right-hand side.

The SRS technique is very effective for summarising network data to determine their patterns and active protocols, services and source/destination IP addresses. However, as it can be observed that there are small differences between the features in the two figures, it could remove important flows that have information about malicious activities. Therefore, for the design of an effective ADS, it is better to

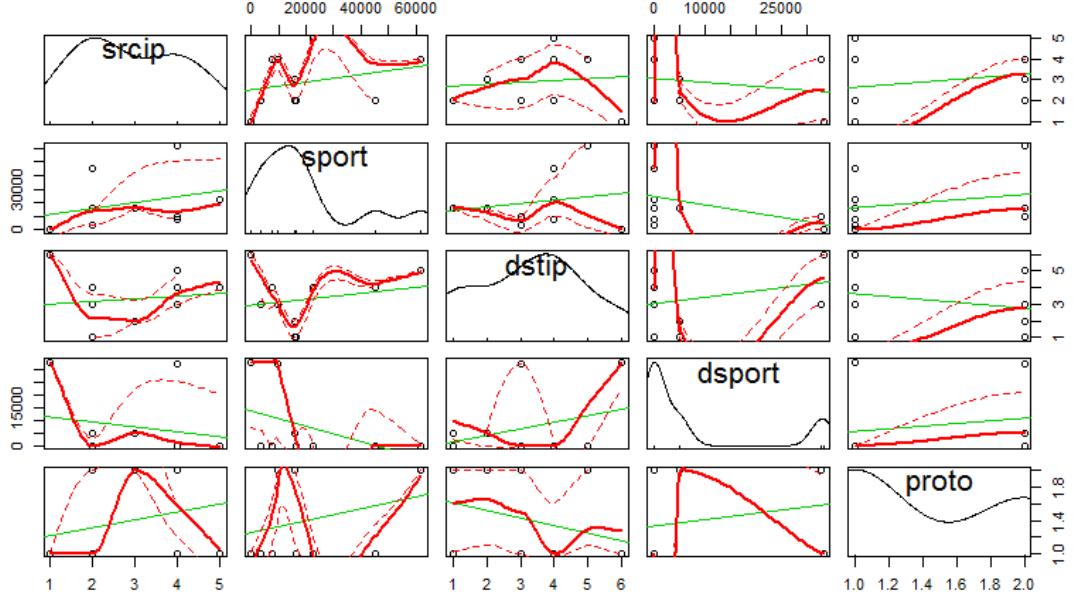


Figure 4.6: Scatterplot matrix analysis of data sample

use correlation methods to collect relevant observations occurring in networks in order to not miss any distinct observations. The aim when using such approaches is to collect all the appropriate flows in a network without losing any between two different endpoints and then aggregate them to ensure a full analysis of network traffic with a low processing time, as described in Section 4.3.

In flow sampling, a subset of all incoming packets is chosen to detect suspicious events which, if observed, leads to imprecise traffic analysis because some malicious instances could be neglected. Moreover, using sampling makes the execution of protocol analysis and DPI difficult as not all protocols and servers can be identified. To address the problem of missing important information, we apply the ARM technique using the CPs of attributes to select all observations that have pair values repeated more than once; for example, Table 4.6 lists all such itemsets

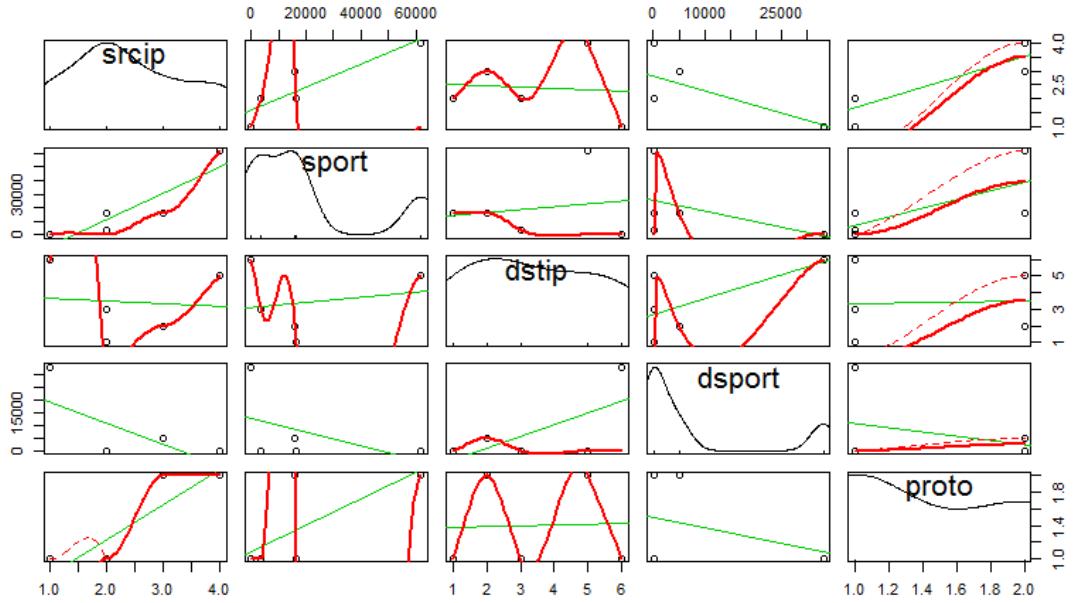


Figure 4.7: Scatterplot matrix analysis of selected sample

Table 4.6: Example of ARM technique for selecting relevant observations

CP >= 2	Itemset	No.
3	dstip=149.171.126.16, proto= tcp	2, 7
3	dsport=80, proto= tcp	6, 7, 8
2	dsport= 5060, proto= udp	5, 10
2	srcip= 175.45.176.0, proto=tcp	4, 6, 8
2	srcip= 175.45.176.0 , dstip= 149.171.126.10	5, 9
2	srcip= 175.45.176.2, proto= udp	3, 9

in Table 4.4 with CPs ≥ 2 , i.e., selected to ensure the appropriate collection of observations and reduce the processing time of an ADS for the same activities occurring between the same two endpoints. We use only the observations from these itemsets while running the DE method which achieves a better performance for collecting observations for a particular time period because it depends on their frequent values and, if these values are the same for consecutive observations, only one is selected.

A. Evaluation of aggregator module

As previously mentioned, in the proposed aggregator module, the SRS technique can be used to select relevant observations to monitor network activities, such as capacity and performance, and the ARM technique to aggregate all the distinct flows across network traffic to establish an effective IDS. These two techniques are developed using the ‘R programming language’ on Linux Ubuntu 14.04 with 16 GB RAM and an i7 CPU processor. To conduct the experiments, we select random samples from the CSV files in the UNSW-NB15 dataset with different sizes of between 50,000 and 250,000 to ensure that only relevant records are chosen. We evaluate the performances of these techniques in terms of their processing delays and distinct records identified (defined below), and then compare them with those of some existing techniques.

- **Processing delay** – a data analysis normally waits for a flow aggregator to finish handling the flow data and, the shorter the processing delay, the more timely the activities in the data analysis phase.
- **Distinct records** – are the flow records for which a technique is capable of excluding duplicates in a data collection as the result of a flow being processed at many network points. Duplications of flow records lead to increases in computational time and the difficulty of achieving online protection.

We aggregate the flows based on the five flow identifiers of the source/destination IPs/ports, protocol types and transactional features. If these identifiers are duplicated, the techniques should retrieve only the distinct records and thereby improve the processing time of any security application, such as an IDS and network monitoring system. In Table 4.7, the experimental results show that the overall processing delays of the SRS technique are approximately half those of the ARM mechanism. However, the latter can find the majority of duplicated records without losing any observation, which might be an anomalous record, while the

Table 4.7: Comparison of performances of ARM and SRS techniques

No. of records	ARM technique		SRS technique	
	Processing delay (seconds)	Distinct records	Processing delay (seconds)	Distinct records
50,000	0.43	22,543	0.24	18,654
100,000	0.82	63,639	0.47	54,273
150,000	1.21	89,876	0.71	63,876
200,000	2.83	124,701	1.13	76,554
250,000	3.23	163,854	1.56	93,376

SRS method drops several observations which have to be analysed to determine if they are abnormal.

Although the ARM technique takes a slightly longer processing time than the SRS technique, it can precisely find different patterns in network data which helps the IDS efficiently discriminate between normal and abnormal instances. The time required for online processing can be reduced using the sample size to select n samples at a time, a method which can improve the establishment of an effective IDS by analysing the incoming network flows one by one and determining if there are abnormal instances in them.

The experimental results obtained from the ARM and SRS techniques demonstrate that our aggregator module performs better than the NetFlow, sFlow and IPFIX tools in terms of the processing time and generation of distinct records as most relevant flows are generated with relatively low processing times, as shown in Figure 4.8. This is because the first step in our module aggregates the flows using five flow identifiers whereas the other tools aggregate using only one attribute. Moreover, we use the MySQL database to store these flows for easy processing by other applications, specifically an ADS, which is the main focus of our research. In conclusion, we recommend using the ARM technique with the aggregator module to detect abnormal events to ensure that all the observed records are analysed,

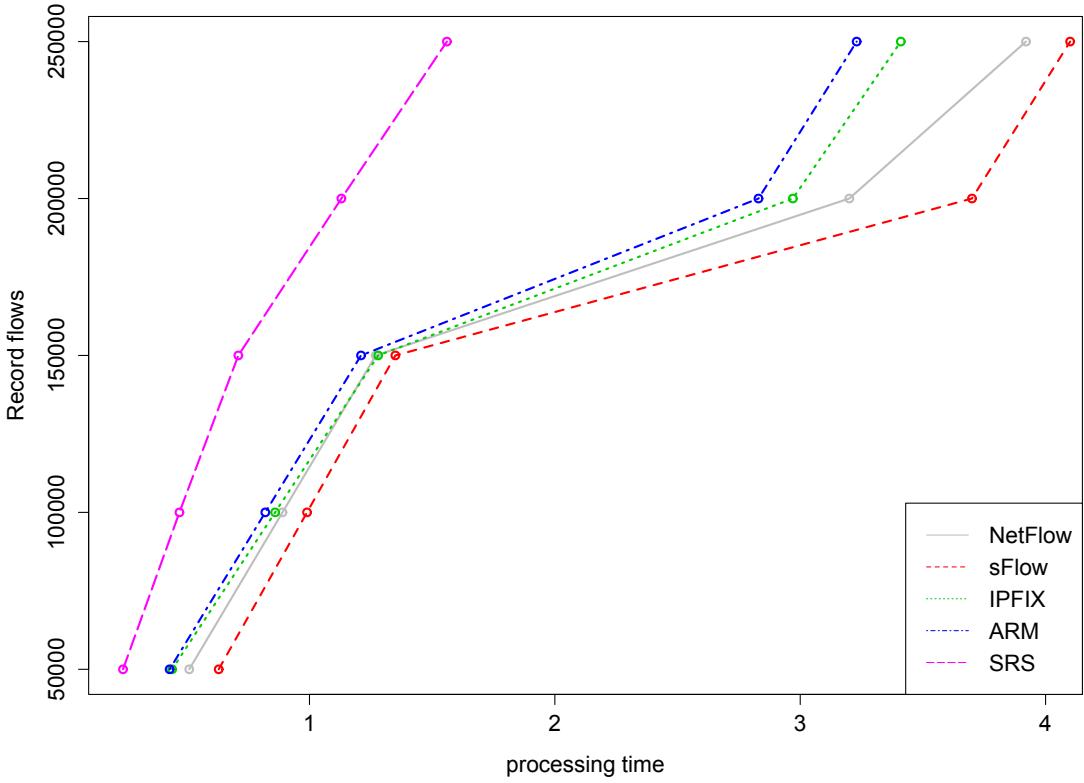


Figure 4.8: Comparison of flow aggregator mechanisms

and the SRS technique if only monitoring a network's activities, such as capacity and user events.

4.6.2. Evaluation of Proposed Features of DNS and HTTP

The proposed features of the DNS and HTTP in the UNSW-NB15 dataset listed in Table 4.2 are used to evaluate the performances of some existing ML algorithms for detecting suspicious observations facing these protocols. To determine the importance of these features, firstly, they are analysed using the correlation coefficient technique to estimate the strongest correlations between them and, secondly, the

evaluation of the proposed ensemble method including DT, NB and ANN is to estimate the performance in terms of accuracy, DR and FAR.

Firstly, the correlation coefficient estimates the strength and direction of a linear relationship between features. If it is close to 1 or -1, the features are positively or negatively linearly correlated, respectively, and the scatter plots show straight lines with positive and negative trends, respectively while, if it is close to 0, there is a weak linear correlation between the features. Features are considered useful in DE for detecting abnormal instances if they are uncorrelated to each other but correlated to the predictor (i.e., a normal or attack class label) [268].

We determine the correlation coefficients for the DNS and HTTP data extracted from the UNSW-NB15 dataset using the R programming language and plot them in a correlation matrix to demonstrate the relationship between two features in a range of [-1, 1]. As shown in Figure 4.9, those between the numeric features of the DNS and HTTP are grouped in three clusters to estimate their correlation scores. From top to bottom, the clusters range from the strongest to weakest relationships and it is clear that the features are not related to each other because their correlation scores are low. Therefore, the proposed features are relevant for classifying normal and attack records of both protocols. Also, the higher DRs of the ML techniques reflect the significance of these features for building an ADS which can efficiently detect both DNS and HTTP attacks.

4.6.3. Evaluation of proposed ensemble method and discussion

The overall performance assessment of the DT, NB, ANN and proposed ensemble method in terms of accuracy, DR and FPR is explained using the DNS and HTTP data sources described in Table 4.8, with the ROC curves representing the relationships between the DR and FPR depicted in Figures 4.10 and 4.11.

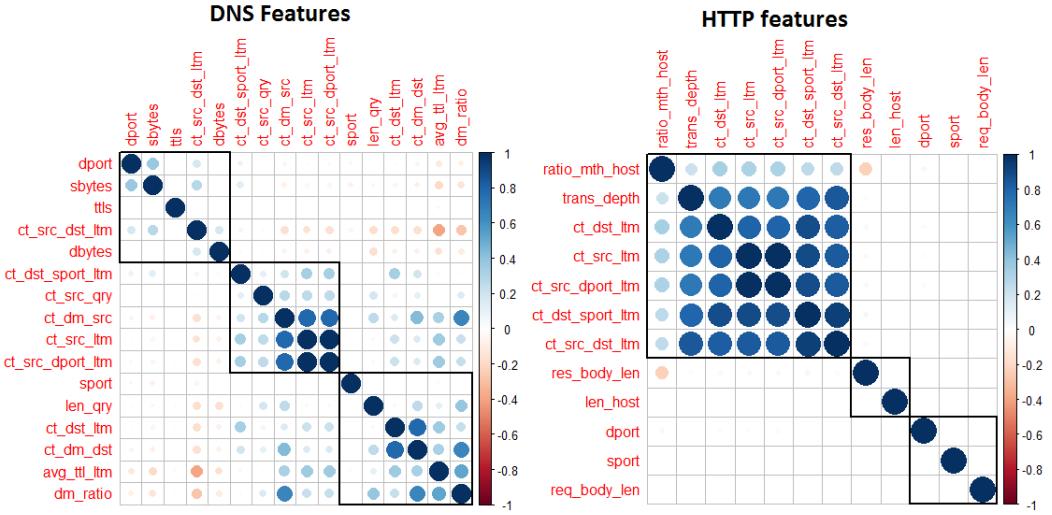


Figure 4.9: Correlations of proposed features

On the one hand as, using the DNS data source, the ensemble method's accuracy and DR are 99.54% and 98.93%, respectively, and its FPR 1.38%, it outperforms the DT, NB and ANN algorithms. The DT technique achieves a 95.32% accuracy, 91.15% DR and 5.22% FPR, the ANN algorithm a 92.61% accuracy, 91.48% DR and 7.87% FPR, and the NB technique a 91.17% accuracy, 90.78% DR and 8.27% FPR.

On the other hand, using the HTTP data source, the ensemble method outperforms each classification algorithm separately, producing a 98.97% accuracy, 97.02% DR and 2.58% FPR. The DT algorithm is the second best with a 97.13% accuracy, 96.34% DR and 3.43% FPR, the ANN technique next with a 96.27% accuracy, 95.53% DR and 4.26% FPR, and the NB technique last with a 97.13% accuracy, 96.43% DR and 3.43% FPR.

Table 4.8: Comparison of overall performances

Technique	DNS data source			HTTP data source		
	Accuracy	DR	FPR	Accuracy	DR	FPR
	(%)	(%)	(%)	(%)	(%)	(%)
DT	95.32	94.15	5.22	97.13	96.34	3.43
NB	91.17	90.78	8.25	95.91	95.25	4.18
ANN	92.61	91.48	7.87	96.27	95.53	4.26
Ensemble method	99.54	98.93	1.38	98.97	97.02	2.58

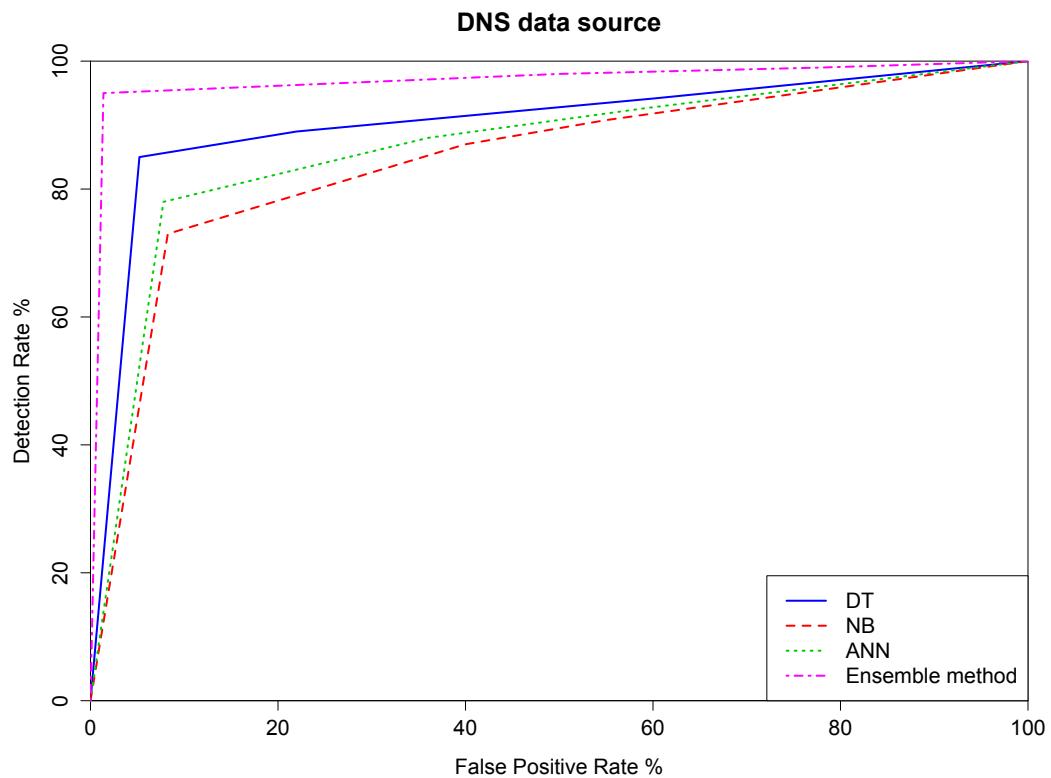


Figure 4.10: ROC curves of classification algorithms using DNS data source

The results obtained from the ensemble method for the different record types using both data sources are presented in Table 4.9 in which it is clear that the DR and FPR are generally between 95.25% and 99.86%, and 0.01% and 0.61%, respectively. These rates demonstrate that this method using the proposed features can efficiently identify the attack types that attempt to breach networks via the DNS and HTTP protocols. Also, the DRs of the normal data using the two data sources are more than 99% which reflect the lowest false negative rates when

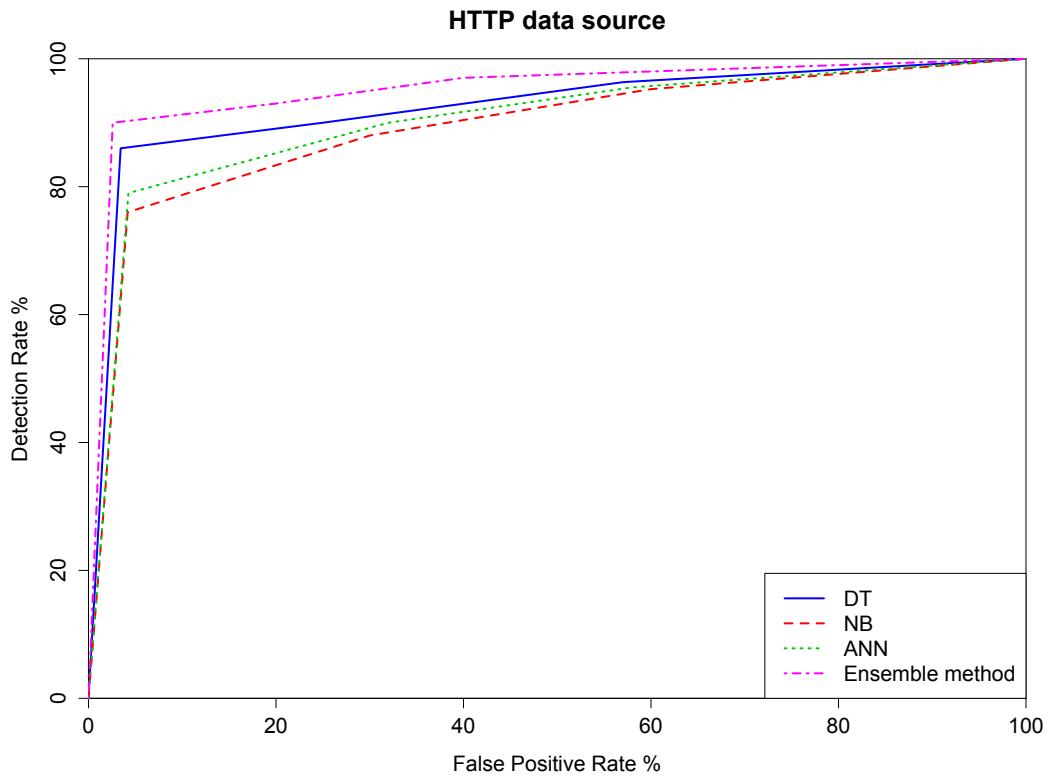


Figure 4.11: ROC curves of classification algorithms using HTTP data source

Table 4.9: Comparisons of DRs (%) and FPRs (%) using ensemble method

Record type	DNS data source		HTTP data source	
	DR	FPR	DR	FPR
Normal	99.53	0.24	99.01	0.15
DoS	98.22	0.31	98.54	0.42
Exploits	96.57	0.52	95.25	0.57
Fuzzers	99.86	0.01	99.23	0.02
Generic	99.43	0.13	96.63	0.72
Reconnaissance	99.78	0.04	98.57	0.24
Analysis	NA	NA	99.2	0.38
Backdoor	NA	NA	95.25	0.61
Worms	NA	NA	99.62	0.05

detecting normal instances.

There are two major reasons that the proposed features and ensemble method can easily recognise suspicious activities that try to expose network applications

within the DNS and HTTP protocols. Firstly, the proposed features are generated using statistical analysis measures, in particular, the mean, length and number of occurrences of event, which can accurately detect small variations between normal and malicious instances. As previously discussed, the scatter matrix plots and correlation coefficient measure, which are used to estimate the similarity and strength of the proposed features, demonstrate that these features can significantly distinguish between the normal and malicious patterns of both protocols using DE methods.

Secondly, using the Adaboost model enhances the performance of the proposed ensemble method compared with those of each algorithm involved in the method. The ensemble method was built based on the Adaboost model, which distributes the input data to process the base learners with an error function. This function computes the error value of each observation to decide which of the base learners can successfully classify this observation. This is because the adaptive boosting can deal with small variations in the feature observations by computing the error function and, if one classifier cannot correctly recognise a specific instance, another one is applied to do so. As modern malicious activities attempt to mimic legitimate behaviours, there are slight differences between these types of behaviour in the UNSW-NB 15 dataset. Therefore, the proposed ensemble method can effectively identify malicious observations from these data sources.

The classification algorithms in the ensemble method are selected based on statistical interpretations of the network data which indicate that the DT can recognise data observations with their corresponding class labels using the most information to decide whether a vector is an anomaly. Then, the ANN can weigh slight differences in features to discover malicious observations using an activation function. Finally, the NB can take the decision that a vector is either normal or attack depending on the posterior of that vector with a particular baseline. By integrating these techniques and distributing the network data sources of the DNS and HTTP protocols using the Adaboost model, the NIDS' performance can be

enhanced in terms of accuracy, DR and FPR without affecting the processing time compared with that of each classification algorithm.

4.6.4. Feature Reduction and Evaluation

A. Measuring relevant features

It is necessary to determine the extracted features are relevant, that is, have the characteristics of normal and malicious observations, for improving the performance of DE approaches for establishing an ADS. To ensure that the features of the UNSW-NB15 dataset are relevant, we previously applied some statistical measures and ML algorithms. Here, we create the same features of the UNSW-NB15 dataset from the tcpdump files of the KDD99 dataset to estimate to what extent they include different patterns of normal and malicious observations using the ARM technique. We use the second week of the KDD99 data as it includes different abnormal activities.

The ARM FS technique is developed using the Visual Studio C# 2008 - Business Intelligence package. Its parameters (i.e., minsup and minconf) are set to three different values (0.4, 0.6 and 0.8) to select the strongest rules from which the highest-ranked are chosen. Random sample sizes of between 100,000 and 250,000, which have all the normal and attack types in the datasets, are chosen with a 10-fold cross-validation used to select the relevant features and measure performances. Tables 4.10 and 4.11 present the most important features selected from the original and UNSW-NB15 features in the KDD99 and UNSW-NB15 datasets, respectively, which are ranked to select eleven significant features to each class of data as either normal or abnormal. The original features of the KDD99/NSL-KDD used in these comparisons are provided in Appendix B.

Table 4.10: Original and UNSW-NB15 features in KDD99 dataset using ARM

Type	Feature numbers in original KDD	Feature numbers in UNSW-NB15
Normal	3, 37, 36, 30, 38, 12, 22, 39, 16, 20, 15	37, 13, 46, 12, 36, 26, 5, 35, 33, 16, 22
DoS	30, 4, 39, 25, 26, 38, 5, 32, 2, 3, 37	45, 23, 20, 43, 31, 15, 46, 18, 22, 38, 12
Probes	36, 23, 24, 12, 6, 39, 8, 10, 16, 7, 11	20, 19, 5, 12, 37, 13, 46, 42, 33, 45, 17
R2L	1, 10, 7, 6, 21, 38, 9, 8, 3, 27, 31	11, 17, 10, 14, 42, 39, 19, 12, 20, 5, 46
U2R	13, 17, 14, 10, 15, 14, 30, 33, 31, 36, 3	9, 46, 42, 14, 22, 24, 34, 37, 32, 13, 6

Table 4.11: UNSW-NB15 features using ARM

Type	Feature numbers
Normal	11, 34, 19, 20, 21, 37, 6, 10, 11, 36, 47
DoS	6, 11, 15, 16, 36, 37, 39, 40, 42, 44, 45
Fuzzers	6, 11, 14, 15, 16, 36, 37, 39, 40, 41, 42
Backdoors	6, 10, 11, 14, 15, 16, 37, 41, 42, 44, 45
Exploits	10, 41, 42, 6, 37, 46, 11, 19, 36, 5, 45
Analysis	6, 10, 11, 12, 13, 14, 15, 16, 34, 35, 37
Generic	6, 9, 10, 11, 12, 13, 15, 16, 17, 18, 20
Reconnaissance	10, 14, 37, 41, 42, 43, 44, 9, 16, 17, 28
Shellcode	6, 9, 10, 12, 13, 14, 15, 16, 17, 18, 23
Worms	41, 37, 9, 11, 10, 46, 23, 17, 14, 5, 13

We use the features provided in Tables 4.10 and 4.11 to evaluate the performances of the two ML algorithms NB and EM clustering. The evaluation criteria are the accuracy and FAR of each that demonstrate the extent to which generating relevant features could improve the performances of DE algorithms. Table 4.12 shows the accuracy and FAR of each algorithm for the original and UNSW-NB15 features in the KDD99 dataset. For the original features, the performance of the NB is better than that of the EM clustering as it achieves a 92.1% accuracy and 8.5% FAR while, on average, those of the EM clustering are 88.9% and 9.5%, respectively. Similarly, for the features in the UNSW-NB15 dataset, the NB performs better than the EM clustering, achieving a 93.4% accuracy and 6.7% FAR compared with the EM clustering's 90.5% accuracy and 10.4 FAR. Comparing the

Table 4.12: Performance evaluations of original and UNSW-NB15 features on KDD99 dataset

Type	Original features (KDD99)				UNSW-NB15 features			
	NB		EM		NB		EM	
	Accuracy (%)	FAR (%)	Accuracy (%)	FAR (%)	Accuracy (%)	FAR (%)	Accuracy (%)	FAR (%)
Normal	96.7	3.2	93.2	7.5	97.4	3.1	94.7	6.2
R2L	91.1	8.7	91.3	8.3	92.1	7.8	90.6	10.5
DoS	95.7	5.3	93.7	7.2	95.9	5	93.5	8.8
Probes	92.8	8.2	90.8	9.1	93.2	6.4	93.1	7.6
U2R	83.8	17.1	75.8	15.6	88.7	11.6	80.6	18.8
Average	92.1	8.5	88.9	9.5	93.4	6.7	90.5	10.4

two feature sets, firstly, the accuracy of the normal and attack types is greater than 80% and the FAR less than 19% and, secondly, the UNSW-NB15 features perform better than the original features as the first are generated by analysing different protocols and services to exactly determine some attributes that can differentiate between normal and attack observations.

In Table 4.13, the performance evaluation of each type of attack in the UNSW-NB15 dataset is presented. The experimental results obtained from the NB algorithm, a 72.2% accuracy and 25.3% FAR, are better than those from the EM algorithm of a 67.2% accuracy and 32.9% FAR. They demonstrate that these algorithms cannot recognise some record types with a high accuracy and low FAR but detect generic records better while the others are not high, with between a 50% and 82% accuracy.

The above results illustrate that the proposed ARM FS method can be used to generate the highest-ranked features of the KDD99 and UNSW-NB15 datasets, with those of the UNSW-NB15 dataset generated from the KDD99 dataset reflecting both contemporary normal and attack records, an improvement on the original

Table 4.13: Performance evaluation of UNSW-NB15 dataset

Type	UNSW-NB15 features			
	NB		EM	
	Accuracy (%)	FAR (%)	Accuracy (%)	FAR (%)
Normal	82.1	17.5	77.6	32.9
Analysis	60.2	38.5	57.3	42.5
Backdoor	77.6	12.5	70.5	29.1
DoS	73.1	12.6	65.3	33.3
Exploits	65.6	35.2	74.5	24.3
Fuzzers	76.2	22.9	60.5	37.5
Generic	95.3	6.3	93.1	7.5
Reconnaissance	78.9	20.6	71.6	26.9
Shellcode	60.5	40.2	50.2	47.3
Worms	52.5	47.6	51.1	48.3
Average	72.2	25.3	67.2	32.9

features. Nonetheless, the ML algorithm used on the UNSW-NB15 dataset cannot efficiently discriminate between normal and suspicious instances because of the similarity of their relative values. As the variations between these instances are low, there is interference between them which leads to limited detection as the UNSW-NB15 dataset is very complex due to its wide variety of recent normal and abnormal activities, as discussed in Chapter 3, subsection 3.9.3.

B. ARM evaluation

The ARM FS technique is applied on the NSL-KDD and UNSW-NB15 datasets to select the important features appropriate for all their normal and anomalous types. We choose the NSL-KDD dataset and our UNSW-NB15 dataset because it does not contain repeated instances and handles the problem of imbalances between observations in each class, as explained in Chapter 2, Section 2.6. From the two datasets, we select random sample sizes of between 100,000 and 250,000 which have all the normal and suspicious types in the datasets, with a 10-fold cross-validation used to choose the relevant features and estimate the performances of some existing ML algorithms.

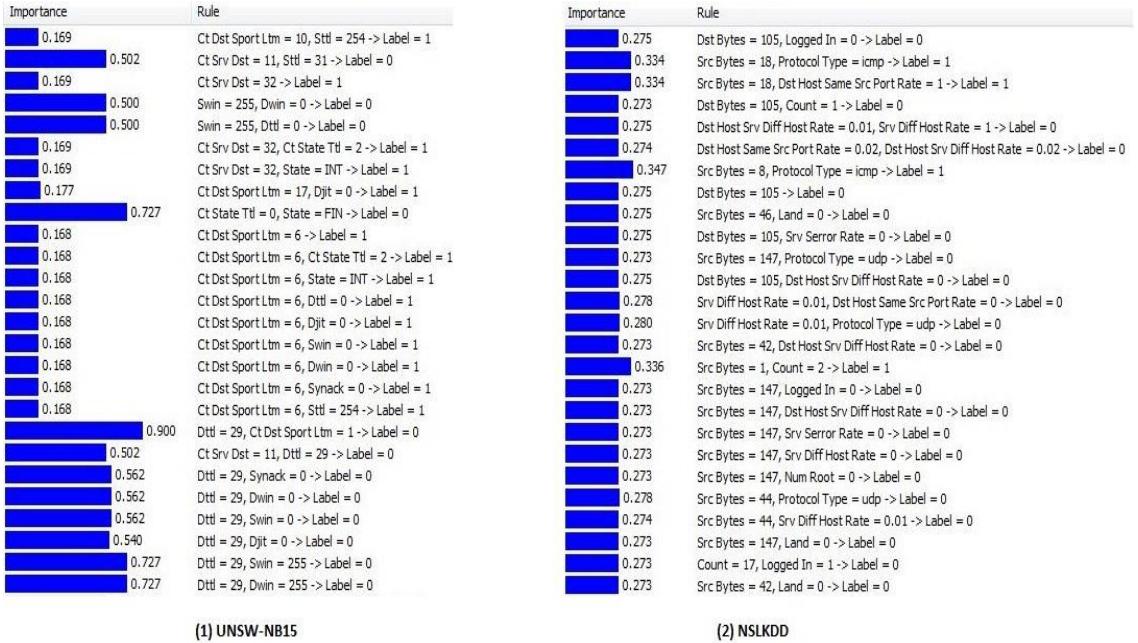


Figure 4.12: Portions of association rules using both datasets

This technique is developed using the Visual Studio C# 2008 - Business Intelligence package, the parameters (i.e., minsup and minconf) of which are adjusted with three different values (0.4, 0.6 and 0.8) to select the strongest rules from which the highest-ranked features are selected. The reason for choosing these values is that their probabilities are divided into three degrees: low (0 - 0.4); medium (0.4 - 0.6); and high (0.61- 1). Some rules and their importance (i.e., an average of each rule's support and confidence) for the NSL-KDD and UNSW-NB15 datasets are presented in Figure 4.12. Each rule contains a set of features for each class, either normal (0) or attack (1).

The most important features are selected from the rules with higher levels of importance. We select the eleven features for each dataset listed in Table 4.14 to reduce the processing time while applying DE as, for less than this number, DE evaluations provide lower accuracies and higher FARs. The processing time is

Table 4.14: Features selected from both datasets

Dataset	Selected features
NSL-KDD	dst_bytes, dst_host_srv_diff_host_rate, srv_diff_host_rate, Land, dst_host_same_src_port_rate, Count, src_bytes, logged_in, protocol_type, num_root, srv_rerror_rate
UNSW-NB15	State, Dttl, Synack, Swin, Dwin, ct_state_ttl, ct_src_ltm, ct_srv_dst, Sttl, ct_dst_sport_ltm, Djit

short as we apply the CP function to generate the rules from the most frequent values of the features which include all possible patterns in each dataset as there are strong associations between the datasets' feature vectors.

To assess performances using the features selected from the two datasets, three ML algorithms, namely, EM clustering, Logistic Regression (LR) and NB, are used. The evaluation criteria are estimated in terms of the accuracy and FAR to assess the effects of these features and how they could improve performance at a lower computational cost of resources. The evaluation results obtained from these algorithms are provided in Table 4.15, with the relationships between their DRs and FPRs represented by the ROC curves in Figure 4.13 to demonstrate their potential performances using the ARM technique.

In the NSL-KDD dataset, the LR algorithm achieves the best output of a 90.3% accuracy and 9.2% FAR, followed by the NB with a 93.8% accuracy and 6.5% FAR and, finally, the EM clustering with a 90.3% and 9.2% FAR. Similarly, in the UNSW-NB15 dataset, the LR algorithm achieves the best results, a 88.2% accuracy and 12.6% FAR, NB the second best, a 90.7% accuracy and 9.7% FAR, and the EM clustering the worst, a 88.2% accuracy and 12.6% FAR.

Table 4.15: Performance evaluation using both datasets

Techniques	NSL-KDD		UNSW-NB15	
	Accuracy	FAR	Accuracy	FAR
	(%)	(%)	(%)	(%)
EM	90.3	9.2	88.2	12.6
LR	95.1	5.6	90.7	9.7
NB	93.8	6.5	85.5	16.3

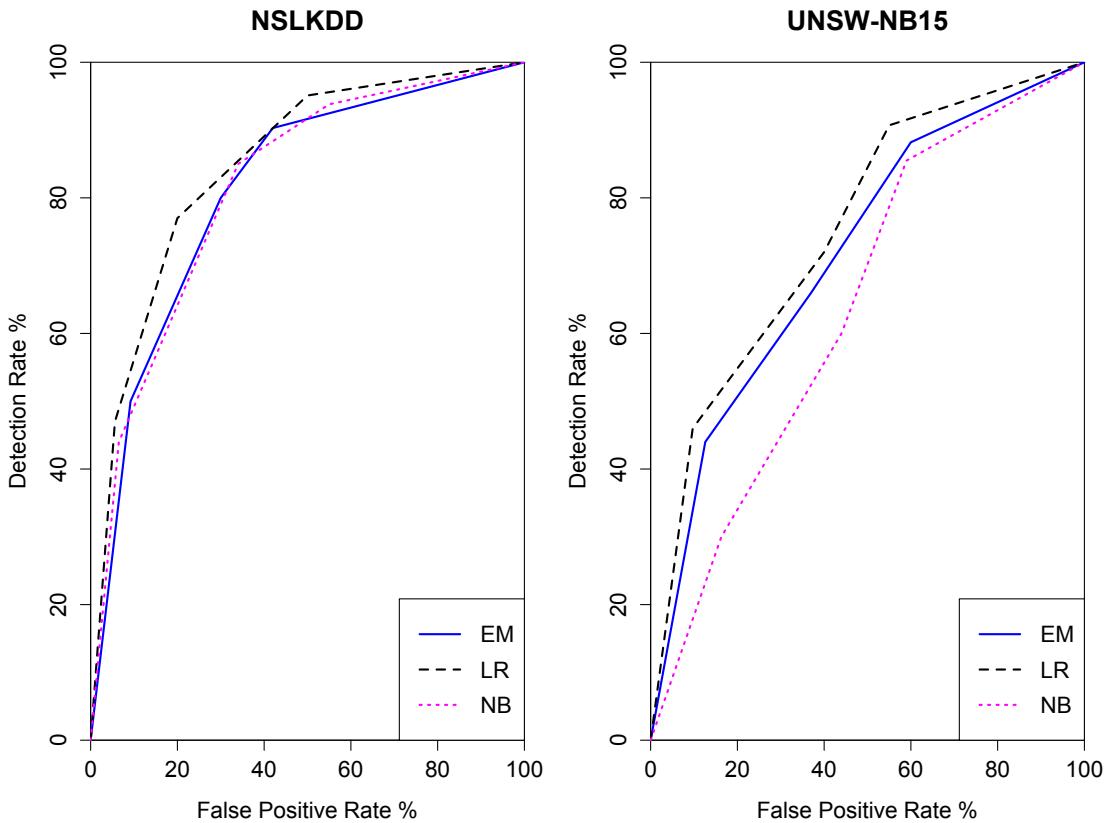


Figure 4.13: ROC curves of three ML algorithms using ARM

C. PCA and ICA evaluations

The significant features adopted from the two datasets to generate their components using the PCA and ICA are based on their higher variances, as explained in subsections 4.5.2 and 4.5.3, respectively. We select eleven features from the NSL-KDD and UNSW-NB15 datasets for applying the ML techniques to reduce the processing time and provide a fair comparison of the three FS methods, as

Table 4.16: Features selected from datasets

Dataset	Selected features
NSL-KDD	srv_count, dst_host_srv_count, count, dst_host_same_srv_rate, dst_host_count, hot, srv_diff_host_rate, rerror_rate, srv_error_rate, dst_host_srv_error_rate, dst_host_rerror_rat
UNSW-NB15	ct_dst_sport_ltm, tcprtt, dwin, ct_src_dport_ltm, ct_dst_src_ltm, ct_dst_ltm, smean, dmean, service, proto, dtcpb

shown in Table 4.16.

To evaluate performances using the features adopted from the two datasets, the same ML algorithms used for the ARM technique, EM clustering, LR and NB, are applied. Table 4.17 presents their results using the component features of both the PCA and ICA techniques. Firstly, using ICA technique, for the NSL-KDD dataset, the LR algorithm produces the highest accuracy of 95.1% and lowest FAR of 5.2%, the NB the second ranking, a 94.9% accuracy and 5.8% FAR, and the EM clustering the lowest, a 93.4% accuracy and 7.8% FAR. Similarly, for the UNSW-NB15 dataset, the LR algorithm achieves the best performance, a 92.4% accuracy and 8.7% FAR, the NB a 90.2% accuracy and 11.4% FAR, and the EM clustering a 89.3% accuracy and 12.4% FAR.

Secondly, using the PCA technique , for the NSL-KDD dataset, the LR algorithm achieves the best performance of a 95.7% accuracy and 4.9% FAR, the NB the second best, a 93.5% accuracy and 6.9% FAR, and the EM clustering the worst, a 93.4% accuracy and 7.8% FAR. Similarly, for the UNSW-NB15 dataset, the LR algorithm attains the best results of a 95.6% accuracy and 5.8% FAR, the NB a 93.7% accuracy and 7.5% FAR, and the EM clustering a 90.7% accuracy and 11.8% FAR.

Table 4.17: Performance evaluation using both datasets

Techniques	ICA				PCA			
	NSL-KDD		UNSW-NB15		NSL-KDD		UNSW-NB15	
	Accuracy	FAR	Accuracy	FAR	Accuracy	FAR	Accuracy	FAR
	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)
EM	93.4	7.8	89.3	12.4	92.6	8.8	90.7	11.8
LR	95.1	5.2	92.4	8.7	95.7	4.9	95.6	5.8
NB	94.9	5.8	90.2	11.4	93.5	6.9	93.7	7.5

There are two reasons for the ML algorithms performing better on the NSL-KDD than UNSW-NB15 dataset. Firstly, the latter has many values of normal and suspicious instances that are almost the same while the former does not. Secondly, the data distributions of the NSL-KDD dataset's training and testing sets are different due to the insertion of new attacks into the testing set which clearly distinguish between its normal and abnormal instances while executing ML algorithms. However, these data distributions in the UNSW-NB15 dataset are approximately the same because its normal and abnormal instances were created from the same network.

To compare the results obtained from the three FS methods, ARM, PCA and ICA, we observe that the last two often provide better evaluation results than the ARM using ML algorithms, as evidenced by the ROC curves of these algorithms in Figures 4.14 and 4.15, respectively. This is because the ARM technique deals directly with the values of features while the others transform the feature space into another space based on the highest variances between features which can greatly help DE techniques to find differences between normal and suspicious instances. However, the ARM method can provide promising results when selecting relevant observations, as explained above.

Regarding the PCA and ICA techniques, there are only small differences in the evaluation performances of the ML algorithms as their internal methodologies

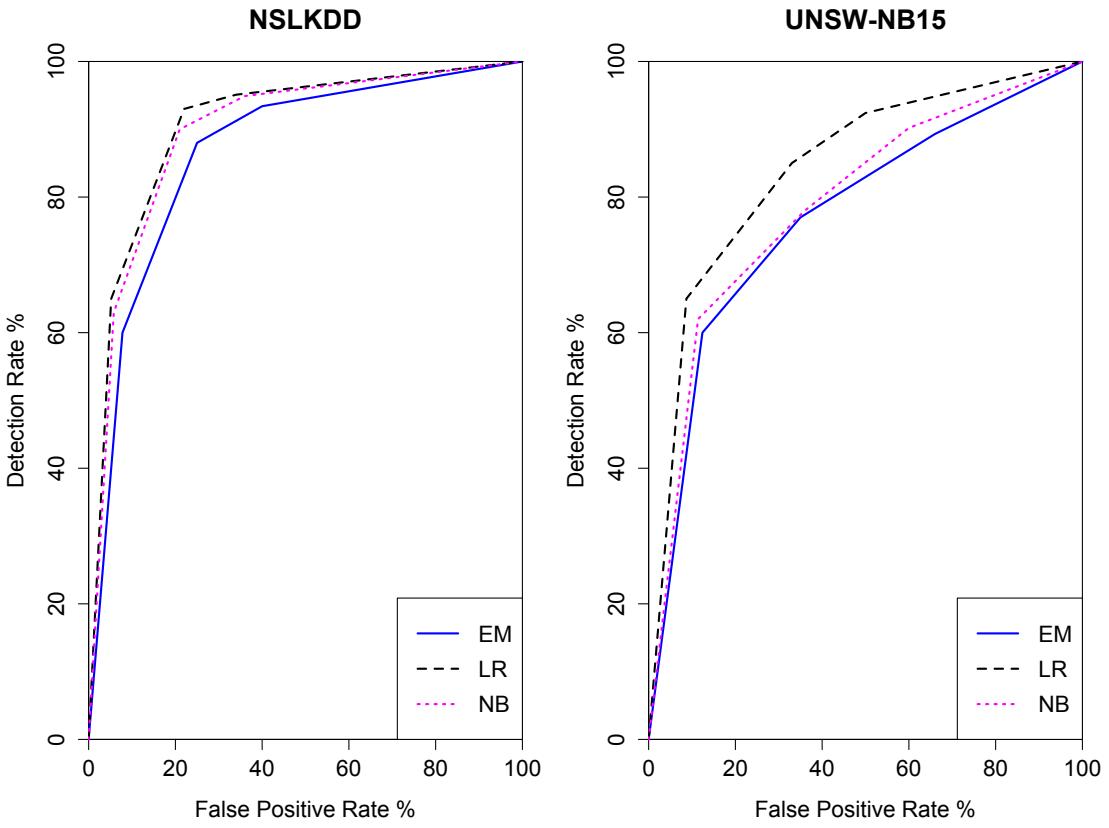


Figure 4.14: ROC curves of three ML algorithms using ICA

appear to be similar based on variances. It can be observed that more evaluation algorithms use the ICA than PCA technique. In the next chapter, we will use the PCA in the feature reduction model due to its simplicity of execution and the improved performances on the existing ML algorithms. We will use it with the new ADS techniques and will describe their effects on each type of attacks in both datasets, showing their efficiency for designing a scalable, lightweight and adaptive ADS.

4.7. Chapter Conclusion

This chapter covers the importance of the two steps of feature creation and reduction in a data pre-processing module for the design of an effective online ADS.

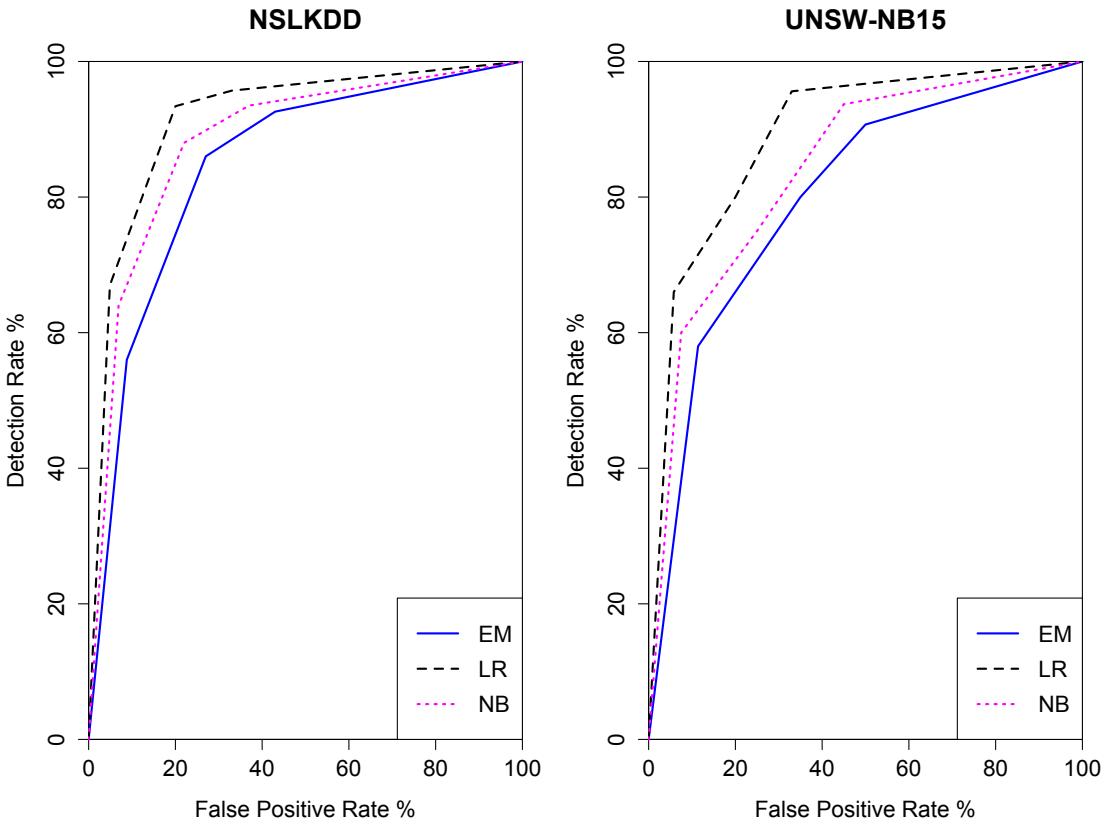


Figure 4.15: ROC curves of three ML algorithms using PCA

Firstly, feature creation is a significant step in constructing suitable attributes that include patterns of legitimate and suspicious activities for improving the performance of the DE technique. Consequently, common protocols and services need to be analysed carefully to generate all the feature sets that could contain information about normal and attack activities. Then, the values of these features should be aggregated to select only relevant and distinct observations, thereby reducing the computational costs of the DE technique used.

With the high speeds and large sizes of current network systems, flow aggregation methods are important for several applications, such as network monitoring and security management. However, as existing tools cannot ensure the aggregation of flows based on more than one attribute, many observations could be repeated which increases an ADS' overhead processing. Therefore, we propose two significant methods that involve only relevant observations. The first is the

SRS technique that selects a subset from all observations, including the majority of statistical characteristics of all the observations, and is effective for summarising network traffic. The second is the ARM technique with CP values which selects all observations that have a pair of duplicated values of more than one feature. This is a data mining method used to compute the association between two or more features in a dataset that finds the strongest rules occurring between their values while the CP function determines the most frequent values using a/the mode measure. This approach is very effective for implementing an online ADS.

Finally, the importance of a FR method for the design of a lightweight ADS is discussed. FR is the process of eliminating unsuitable features from network data to enhance the detection accuracy and processing time of a DE method, with three FR techniques, ARM, PCA and ICA, used and evaluated by some ML algorithms. The PCA technique extracts a set of features based on the highest variance for each feature and constructs a new dimensional space of uncorrelated features by eliminating low variance ones while the ICA technique elicits unknown components from multivariate data, the components of which are mutually independent and non-Gaussian data.

The experimental results obtained from these techniques show that the PCA and ICA often offer better performances than ARM using some ML algorithms although ARM can provide good results when choosing significant observations. There are small differences between the evaluation performances of the ML algorithms in the PCA and ICA techniques, with a slight improvement on the performances of ML algorithms using the PCA technique. If the network features are not carefully generated for finding clear differences between normal and suspicious observations, the feature selection techniques cannot efficiently select the important network features.

In the next chapter, we discuss the methodology for constructing a new statistical NADS using the PCA technique which has the characteristics of being

lightweight, adaptable and scalable, and demonstrates the influence of each type of attack.

Chapter 5

Novel Statistical Decision Engines for Anomaly Detection System based on analysing Potential Characteristics of Network Features

5.1. Introduction

Providing a defence against the current sophisticated methodologies of attackers has become one of the most significant challenges in the cyber security field, with its principle defined as a strategy for identifying malicious activities and preventing them breaching computer and network systems. To successfully achieve this aim, different layers of defence should be deployed throughout these systems and, since hacker methodologies attempt to dynamically change by mimicking normal activities, a so-called defence-in-depth approach for protecting against any of them using multiple security techniques and tools has appeared [269].

These layers of defence can be grouped into two types, signature and anomaly, depending on their procedures for designing security mechanisms. Most existing solutions, including firewalls, access control, authentication and misuse intrusion detection systems, were designed based on the signature concept in which a system matches patterns occurring in hosts or networks with a well-known blacklist of suspicious rules. However, as they cannot identify any zero-day attacks, that is, new malicious behaviours, the anomaly methodology recently appeared as a second type of defence. It constructs a profile of normal observations, any deviation from which is considered an anomaly, in order to detect new attacks.

A complete layered strategy can be designed by integrating the two types to ensure a far more comprehensive defence against existing and new anomalous activities, the purpose of which is to try to breach the principles of Confidentiality, Integrity and Availability (CIA) in computer systems [42, 270]. Each attack type has its own sophisticated methodology that poses a serious challenge for its successful detection, for instance, a DoS attack corrupts computer resources which breaches the availability principle whereas malware codes hijack the execution flow of programs, thereby breaking the integrity principle [8]. To achieve a complete layered defence strategy against low-footprint attacks, many research studies have proposed new or improved security mechanisms based on the concepts of signature/misuse and anomaly.

When designing the architecture of an ADS methodology for large-scale applications in industry, there are two challenges. The first relates to constructing a full profile from the majority of normal activities which is a very difficult task because of the large sizes and high speeds of current networks that dynamically change their behaviours. The second concerns adopting a suitable methodology for establishing an adaptable, scalable and lightweight DE approach which can efficiently distinguish between normal and malicious observations, an arduous task when dealing with large, high-speed network environments. Current networks include several interconnected devices, programs and platforms for offering services to users and organisations any time, anywhere. A promising ADS should monitor and analyse these services, and effectively and efficiently detect any malicious flows in a network with large volumes, high-velocity transmissions and high dimensionalities (i.e., variety). The properties of volume, velocity and variety of network traffic are considered the phenomena of big data that a proficient ADS should be capable of handling to address which we propose statistical solutions in this chapter.

The key reason for using statistical models in this PhD thesis is that they can effectively determine and inspect the intrinsic potential characteristics of normal

and abnormal network activities for both attributes/features and observations/instances, as fully explained in later section. However, developing an efficient DE requires an accurate analysis to identify normal and abnormal network characteristics and specify a particular baseline/threshold. To achieve this, we focus mainly on the DE module discussed in Chapter 3, Section 2.7 which is obviously the main task in the design of an intelligent IDS for detecting known and unknown attacks in real time. Choosing a DE approach and its training and testing phases essentially contributes to evaluating the efficacy of an ADS by determining whether it is correctly trained and validated for identifying normal patterns (i.e., building an accurate profile from diverse normal instances) and treating any instance outside this profile as an anomaly.

In this chapter, for the first time in this field, we suggest two novel scalable ADS frameworks for effectively detecting known and unknown anomalous activities based on the statistical approaches described in Sections 5.3 and 5.4, respectively. These frameworks have three major modules, namely, data sniffing and storing, data pre-processing and a DE. The first two, which are similar in the two frameworks, capture and store network data, and analyse and filter them, respectively, while the novel DE techniques that efficiently define different malicious behaviours are different. The performances of these frameworks are appraised based on two well-known datasets, the NSL-KDD, which is an upgraded version of the KDD99 and widely used to evaluate new ADSs, and our UNSW-NB15 which includes a broad range of contemporary legitimate, security and malware observations.

The key contributions of this chapter are as follows.

1. We use statistical approaches that can define the potential properties of network data, i.e., data normality and linearity, in order to decide on the best scalable, adaptive and lightweight DE approach for successfully handling large-scale networks.

2. We suggest two new scalable and adaptable ADS frameworks, GAA-ADS and DMM-ADS, each of which has three modules that capture, process and recognise attacks, for effectively identifying suspicious observations, designed based on the anomaly methodology, each DE module creates a normal profile in the training phase and estimates the patterns in the testing phase using the same parameters. To identify abnormal observations, we suggest new decision methods that check each tested observation which, if it falls within the legitimate profile, is considered legitimate, otherwise anomalous.
3. Discussions of the in-depth mathematical analyses of these frameworks assist in ascertaining their use in other domains. Moreover, their computational complexities and processing times reveal that they can be deployed simply for online processing.
4. Performance evaluations of these frameworks are carried out using two NIDS datasets: the NSL-KDD which is an improved version of the KDD99; and UNSW-NB15. Although the KDD99 dataset is outdated and does not contain modern patterns of malicious activities, it is still widely used to assess NIDSs while the UNSW-NB15 dataset is capable of detecting modern malicious activities.

The remainder of this chapter is organised as follows. Section 5.2 discusses network data analytics used for the design of an effective DE. The newly proposed GAA-ADS and DMM-ADS methods are described in Sections 5.3 and 5.4, respectively, and their frameworks for identifying known and zero-day attacks explained in Section 5.5. Then, the empirical results obtained from the new DE techniques are discussed in Section 5.6 and the chapter concluded in Section 5.7.

5.2. Network Data Analytics for Design of Effective DE

Given the large sizes and high speeds of current networks, monitoring and analysing their data have become significant for several reasons. Firstly, a network data analysis increases visibility to users, systems and programs by accumulating network flows which assist in tracking the network bandwidths of users and systems, and ensuring robust service delivery. Secondly, it can measure performance bottlenecks and reduce bandwidth consumption. Thirdly, an IDS technology which monitors and analyses network traffic using a protocol analysis can identify potential attacks, for example, UDP spikes. Finally, a network data analysis also helps to monitor the abusive use of unusual protocols for a particular device to uncover potential data stealing. More specifically, it can identify abnormal events and unauthorised access to sensitive data via analysing peer-to-peer protocols and information in the different layers of the OSI model [271].

Several mechanisms and technologies for gathering, handling, analysing and visualising large-scale networks, which are multi-disciplinary in nature and derive from the domains of mathematics, statistics and information technology, have been designed. Some have origins in academia while others were developed by corporations using online business techniques to analyse data, including statistically based methods and data mining approaches, MySQL CGE and Hadoop technologies, and visualisation tools, for example, NetFlow and spatial information flows [272].

As discussed in Chapter 3, the UNSW-NB15 dataset consists of a huge number of feature vectors and includes the potential characteristics of normal and anomalous observations. These features are both packet- and flow-based, with the former helping to examine the packet payload and headers, and the latter collecting significant information from the packet headers, for example, the number

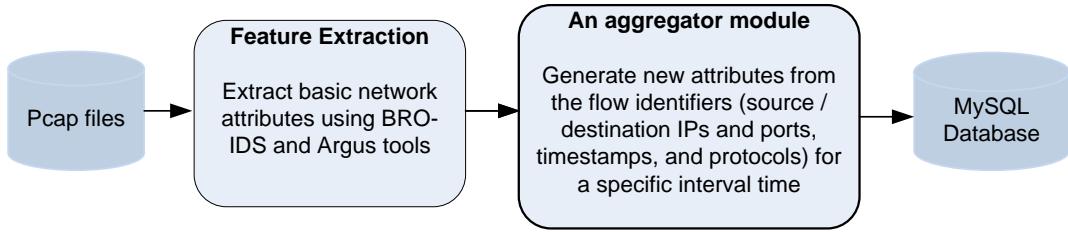


Figure 5.1: Feature vectors for mining and gathering in UNSW-NB15 dataset

of source/destination IPs for a specific time window, and an inter-packet length, packet direction and inter-arrival times of packets.

As shown in Figure 5.1, the pcap files of the UNSW-NB15 dataset were processed by the BRO-IDS and Argus tools to extract the fundamental features from raw packets. Then, we proposed a new aggregator module for correlating network flows, as explained in Chapter 4, Section 4.3, to remove redundant and irrelevant observations to improve the performance of an ADS and network analysis tool. These flows are combined for each 100 network observations, with packets with the same source/destination IP addresses and ports, timestamps and protocols gathered in one observation. It is observed that this module can effectively analyse and determine a network's properties, for example, its capacity, bandwidth, and normal and suspicious patterns.

A statistical analysis of network observations helps to determine the choice and design of the type of modelling that accurately fits the network data in order to identify data outliers as anomalies. Some statistical measures are the Kolmogorov-Smirnov (K-S) test, the results obtained from which are discussed in Chapter 3, Q-Q plots for defining data normality and density, and correllentropy plots for recognising data linearity. Ascertaining the normality and linearity of network data can help to determine which learning models can precisely fit these data and

improve the performance of an ADS for detecting suspicious instances in a low processing time. These measures are discussed in the following two subsections.

5.2.1. Normality measures

In order to validate the normality or non-normality of network data, the K-S test and Q-Q plots are used. A normality test is a statistical measure for assessing whether particular data follow a Gaussian/normal distribution. The major reason for using it is that it can perfectly estimate the goodness of fit of data by defining the potential statistical parameters that can specify to what extent these data vary from a normal distribution in terms of skewness and kurtosis, as described in Chapter 3, Section 3.8.

Firstly, the K-S test, one of the most common techniques for this purpose, is applied. It is acknowledged that, if the network data do not accurately fit a Gaussian distribution, mixture models, specifically a Gaussian Mixture Model (GMM), BMM and DMM, are used to precisely recognise outliers. Secondly, Q-Q plots are applied to represent the network data that do not fit a Gaussian distribution. A Q-Q plot is a graphical representation designed to draw two data samples of quantiles against each other. If these samples are from the same distribution, their points are almost in a straight line, with those not located close to or on the line considered outliers/anomalies [101].

These measures assist in identifying network patterns of normal and attack instances as well as choosing and designing the best DE model for recognising abnormal instances as outliers, as discussed in the experimental results in subsection 5.6.2. Ultimately, statistical analysis mechanisms are quite significant for determining potential procedures for making decisions regarding the recognition and prevention of suspicious activities in network data.

5.2.2. Linearity measures

In order to check the linearity or non-linearity of network data, their density and correntropy plots are used, primarily because they can estimate the similarities between normal and suspicious feature vectors. The mathematical function of a density plot computes the probability densities of normal and attack observations to determine the difference between them while that of a correntropy plot calculates the mean difference between two kernel density functions of given feature observations. These functions are fundamental means of smoothing and specifying the boundaries of feature values to clearly demonstrate both their dissimilarities and similarities [273].

Firstly, the density plot is obtained by computing the probability distributions of normal and abnormal instances to measure the extent to which they vary from each other in either a linear or non-linear representation. Then, we apply the correntropy measure [274] to compute the similarities between the proposed feature vectors and accurately define variations between legitimate and suspicious observations. If there are clear differences, this is statistical evidence of the significance of these features for recognising abnormal activities. This measure, which is one of second-order statistics and a non-linear similarity technique for describing the relationships among given feature vectors, is less sensitive to outliers than the Mean Square Error (MSE) and is widely used [57, 274].

The mathematical procedure for calculating the correntropy of two random variables (f_1 and f_2) is

$$V_\sigma(f_1, f_2) = E[\kappa_\sigma(f_1 - f_2)] \quad (5.1)$$

where $E[.]$ is the mathematical expectation of the features and $K_\sigma(.)$ the Gaussian kernel function, where σ is the kernel size represented by

$$K_\sigma(.) = \frac{1}{\sqrt{2\Pi\sigma}} \exp\left(-\frac{(.)^2}{2\sigma^2}\right) \quad (5.2)$$

Mathematically speaking, the joint probability density function ($P_{F_1, F_2}(f_1, f_2)$) is usually unknown whereas a finite number of vectors ($\{f_i, f_j\}_{i,j=1}^M$) is accessible. Therefore, the correntropy is computed by

$$\hat{V}_{M,\sigma}(A, B) = \frac{1}{M} \sum_{i,j=1}^M K_\sigma(f_i - f_j) \quad (5.3)$$

To use the correntropy measure for multivariate network data, as provided in equation (5.4), we compute it for both normal and suspicious feature observations as

$$I_{1:N} = \begin{bmatrix} f_{11} & f_{12} & .. \\ f_{21} & f_{22} & f_{ij} \end{bmatrix}, Y_{1:N} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad (5.4)$$

where I indicates the vectors of network data, Y the class label (C) of each vector, N the number of vectors and F the set of network features.

If the results reveal that there is a difference between the values of the legitimate and malicious observations, this is vital evidence that confirms the significance of the proposed network features in the UNSW-NB15 dataset for identifying suspicious instances, as discussed in subsection 5.5.2.

5.3. Novel Geometric Area Analysis (GAA-ADS) Technique

The theory behind the new GAA-ADS technique¹ is accurately modelling network data by computing the area for each network observation that includes a set of features using the TAE method estimated from the BMM parameters and distances of observations. It depends on the anomaly methodology and constructs a profile from legitimate areas in the training phase, with areas of attacks in the decision-making method considered when they deviate from this profile.

In the training phase, a profile is established from normal network observations by integrating the BMM parameter estimations and distances between the mean of normal observations and each observation. Then, in the testing phase, these computed parameters are used to estimate the area for each upcoming observation. In both phases, the TAE is calculated from the findings obtained by integrating the BMM and distances between observations for each individual record, as discussed in subsection 5.3.3.

In the decision-making method, the areas in the legitimate profile are divided into a number of ordered intervals to identify anomalous observations and decrease the processing time compared with that required to determine the area of each testing observation. If the area of a testing vector is located within well-known normal ranges, this vector is an anomaly. Since the decision-making method relies on the anomaly methodology, it is possible to recognise an attack vector without requiring any prior or relevant information about attack types. Moreover, the

¹

- A Part of this study has been released in:
Moustafa, N., J. Slay, and Creech, G. “Novel Geometric Area Analysis Technique for Anomaly Detection using Trapezoidal Area Estimation on Large-Scale Networks”, IEEE transactions on Big Data, 2017.
Moustafa, N., Creech, G. and J. Slay. “Anomaly Detection System using Beta Mixture Models”, the International Conference on Computing Analytics and Networking (ICCAN 2017), KIIT University, Springer, 2017.

proposed GAA-ADS technique does not need frequent updates of the attack signatures, as does misuse-based detection, but regular updates of the normal threshold like anomaly-based detection. However, its powerful performance depends on the number of ranges of legitimate areas that prevents overlapping between normal and abnormal areas, as explained in subsection 5.3.4.

5.3.1. Beta Mixture Model (BMM)

Although a GMM can model any random distribution with appropriate mixture components, some of these components do not correctly specify edges when the observed data are semi-bounded or bounded [275]. The features of network data cannot precisely fit a normal distribution because they do not follow its symmetric and unbounded boundary (i.e., $]-\infty, \infty[$). We observe that, in the NSL-KDD and UNSW-NB15 datasets, they are in the semi-bounded $[0, N]$ range, where N refers to an asymmetric integer or real number. A powerful distribution that can model semi-bounded data is a beta distribution as it has a more flexible shape than a normal distribution [275–277] and represents arbitrary variables that have a finite range (i.e., $[a, b], a, b \in \mathbb{R}$), particularly $[0, 1]$, as described in the following subsection.

The PDF of a beta distribution can be computed by

$$Beta(x; v, \omega) = \frac{1}{beta(v, \omega)} x^{v-\omega} (1-x)^{\omega-1}, v, \omega > 0 \quad (5.5)$$

where x is the normalised network features, v and ω the shape parameters that model the beta distribution, $beta(v, \omega)$ the beta function which equals $(\Gamma(v)\Gamma(\omega)/\Gamma(v+\omega))$ and $\Gamma(\cdot)$ the gamma function with $\Gamma(c) = \int_0^\infty \exp(-t)t^{c-1}dt$. If x is an arbitrary variable in the beta distribution computed by equation (5.5), its mean can be calculated by

$$\mu = \frac{v}{(v + \omega)} \quad (5.6)$$

Because network data have sets of features that include patterns of normal and attack observations, these features should be fitted by a mixture model designed to model data with more than one variable. A mixture model can be defined as a flexible probabilistic technique for representing and determining multivariate data [278]. Mathematically speaking, the features of network datasets are multivariate as they consist of more than two variables [47], with each denoted as a component in the mixture model. In [275, 279], the bounded property data are proficiently fitted by the BMM with less model complexity than the GMM.

In the GAA-ADS technique, the BMM is the first step in calculating the PDFs of the network features. Although it is perceived that network observations are independent and identically distributed (*i.i.d*) [36, 47], multivariate data are, in many cases, statistically dependent. However, for any random variable (x) involving L elements, the dependence between elements x_1, \dots, x_L is signified by a mixture model even if each particular component can only represent an observation with independent elements. A multivariate BMM for these observations is defined as a PDF computed by

$$\begin{aligned} f(x; \pi, v, \omega) &= \sum_{i=1}^I \Pi_i Beta(X, v_i, \omega_i) \\ &= \sum_{i=1}^I \Pi_i \prod_{l=1}^L Beta(x_l, v_{li}, \omega_{li}) \end{aligned} \quad (5.7)$$

where I indicates the number of mixture components ($X = \{x_1, \dots, x_L\}$, $\Pi = \{\Pi_1, \dots, \Pi_I\}$, $v = \{v_1, \dots, v_I\}$, $\omega = \{\omega_1, \dots, \omega_I\}$), Π_i the mixing component ($\sum_{i=1}^I \Pi_i =$

$1, 0 < \pi < 1$), $\{v_i, \omega_i\}$ the parameter vectors of the i^{th} mixture component, $Beta(X; v_i, \omega_i)$ the component-conditional parameters, and $\{v_{lI}, \dots, \omega_{lI}\}$ the parameters of the beta distribution for feature x_l .

A. Parameter estimation for BMM

The Maximum Likelihood Estimation (MLE) method used to estimate the parameters of the BMM in equation (5.7) models the observed data. Based on its process, the best parameters ($\theta = \{\nu_1, \dots, v_I, \omega_1, \dots, \omega_I, \pi_1, \dots, \pi_I\}$) maximise the log-likelihood function as

$$\begin{aligned} \mathcal{L}(\theta | X) &= \sum_n^N \log \left[\sum_i^I \Pi_i Beta(x_n; v_i, \omega_i) \right] \\ &= \sum_n^N \log \left[\sum_i^I \Pi_i \prod_{i=1}^L Beta(x_n; v_i, \omega_i) \right] \end{aligned} \quad (5.8)$$

The MLE finds the optimal value of θ by handling x_n as ‘incomplete’ data. The latent variables ($z_n = (z_{n1}, \dots, z_{nI})^T$) indicate an observation that has one element with a value of 1 and the rest ($x_n.x_n$ and z_n) values of 0 which are considered ‘complete’ data, with the likelihood function reformulated as

$$\begin{aligned} \mathcal{L}(\theta | X, Z) &= \sum_n^N \sum_{i=1}^I z_{ni} [\log \Pi_i + \log Beta(x_n; v_i, \omega_i)] \\ &= \sum_n^N \sum_{i=1}^I z_{ni} [\log \Pi_{ni} + \sum_{l=1}^L \log Beta(x_{ln}; v_{li}, \omega_{li})] \end{aligned} \quad (5.9)$$

To iteratively compute θ , the Expectation-maximisation (EM) technique is used. In the ‘E step’, the expected value of z_n is calculated as the posterior probability of x_n being computed from the i^{th} component which refers to the currently appraised parameters by

$$\bar{z}_{ni} = E[z_{ni}] = \frac{\Pi_i Beta(x_n; \hat{v}_i, \hat{\omega}_i)}{\sum_{m=1}^I \Pi_i Beta(x_n; \hat{v}_m, \hat{\omega}_m)} \quad (5.10)$$

In the ‘M step’, the probability (π_i) and parameters ($\theta = \{\nu_1, \dots, \nu_I, \omega_1, \dots, \omega_I\}$) are re-computed given the expected values of the latent variables for maximising their log-likelihoods, with the updated mixture weight computed by

$$\pi_i = \frac{1}{N} \sum_{n=1}^N E[z_{ni}] \quad (5.11)$$

To calculate the two independent parameters ($\hat{v}_{li}, \hat{\omega}_{li}$), their log-likelihoods are simultaneously maximised by

$$\frac{\partial E[\mathcal{L}_C(\theta | X, Z)]}{\partial v_{li}} = \sum_{n=1}^N \frac{\partial \log Beta(x_{ln}; v_{li}, \omega_{li})}{\partial v_{li}} \quad (5.12)$$

$$= \sum_{n=1}^N \bar{z}_{ni} \{ \log x_{ln} - [\psi(v_{li}) - \psi(v_{li} + \omega_{li})] \}$$

where $\psi(\cdot)$ denotes the digamma function calculated by

$$\psi(x) = \frac{\partial \log \Gamma(x)}{\partial x} \quad (5.13)$$

Symmetrically,

$$\frac{\partial E[\mathcal{L}_C(\theta | X, Z)]}{\partial \omega_{li}} = \sum_{n=1}^N \bar{z}_{ni} \frac{\partial \log Beta(x_{ln}; v_{li}, \omega_{li})}{\partial \omega_{li}} \quad (5.14)$$

$$= \sum_{n=1}^N \bar{z}_{ni} \{ \log(1 - x_{ln}) - [\psi(\omega_{li}) - \psi(v_{li} + \omega_{li})] \}$$

To compute the local curvature of $E[\mathcal{L}_C(\theta | X, Z)]$, the Hessian matrix (\mathcal{H}) is used [279] via

$$\begin{aligned} \mathcal{H}\{E[\mathcal{L}_C(\theta | X, Z)]\} &= \begin{bmatrix} \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial v_{li} \cdot \partial v_{li}} & \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial v_{li} \cdot \partial \omega_{li}} \\ \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial \omega_{li} \cdot \partial v_{li}} & \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial \omega_{li} \cdot \partial \omega_{li}} \end{bmatrix} \quad (5.15) \\ &= \begin{bmatrix} \psi'(v_{li} + \omega_{li}) - \psi'(v_{li}) & \psi'(v_{li} + \omega_{li}) \\ \psi'(v_{li} + \omega_{li}) & \psi'(v_{li} + \omega_{li}) - \psi'(\omega_{li}) \end{bmatrix} \end{aligned}$$

where $\psi'(x) = d\psi(x)/dx$. Because $\psi'(v_{li} + \omega_{li}) - \psi'(v_{li}) < 0$, as the indicator $|\mathcal{H}\{E[\mathcal{L}_C(\theta | X, Z)]\}| < 0$ demonstrates that the Hessian matrix is a negative definite one with a local maximum at $(\{\hat{v}_{li}, \hat{\omega}_{li}\})$ declared by

$$\begin{bmatrix} \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial v_{li}} \\ \frac{\partial^2 E[\mathcal{L}_C(\theta | X, Z)]}{\partial \omega_{li}} \end{bmatrix} = 0 \quad (5.16)$$

The two equations for the BMM parameters (v_{li}, ω_{li}) updated from equation (5.12), which are described in [279], can be computed by

$$\psi(\hat{v}_{li}) - \psi(\hat{v}_{li} + \hat{\omega}_{li}) = \frac{\sum_{n=1}^N \bar{z}_{ni} \log x_{ln}}{\sum_{n=1}^N \bar{z}_{ni}} \quad (5.17)$$

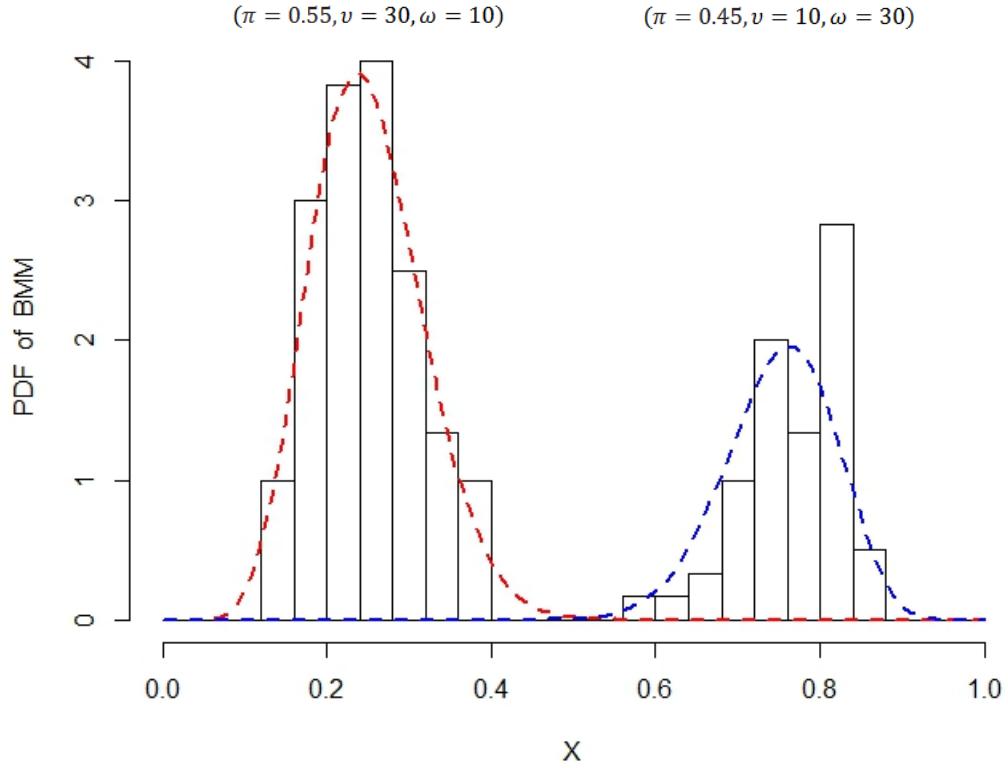


Figure 5.2: BMM for two arbitrary variables

$$\psi(\hat{\omega}_{li}) - \psi(\hat{v}_{li} + \hat{\omega}_{li}) = \frac{\sum_{n=1}^N \bar{z}_{ni} \log(1 - x_{ln})}{\sum_{n=1}^N \bar{z}_{ni}}$$

To illustrate the modelling of the BMM, its parameters are calculated using the EM technique for two given random variables (x_1 and x_2). In Figure 5.2, let the parameters of x_1 (π, v, ω) be 0.55, 30 and 10, respectively and those of x_2 (π, v, ω), 0.45, 10 and 30, respectively. Then, the parameters of the BMM are estimated for the features of the network datasets described in Table 5.2 which is the first step in the proposed GAA-ADS technique for efficiently modelling network data.

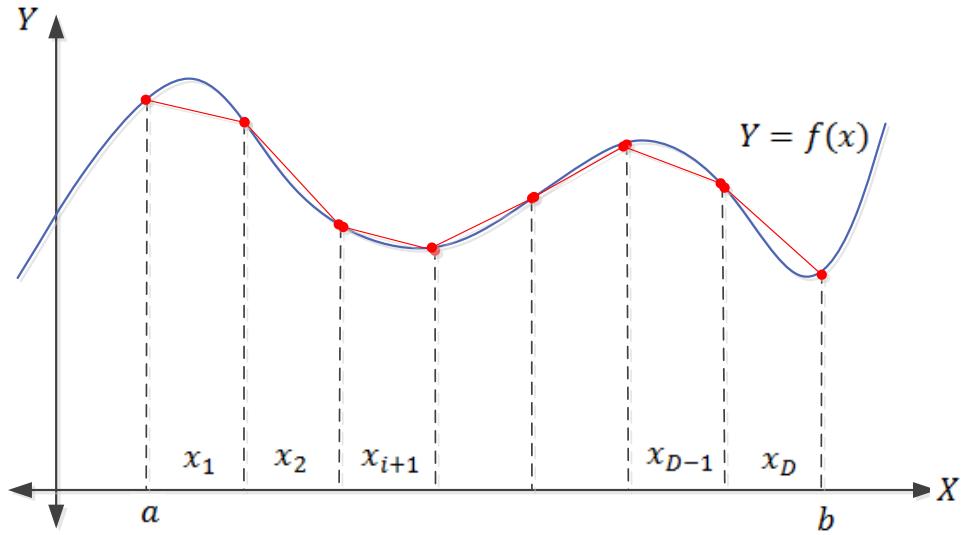


Figure 5.3: Composite trapezoidal rule

5.3.2. Trapezoidal Area Estimation (TAE)

The aim of the GAA-ADS technique is to identify the area of each vector as either normal or abnormal. To achieve record-by-record detection in the decision-making method, the TAE computes the area for each record (r) so that it has some features ($x_{1:D}, r_{1:n} = \{x_1, x_2, \dots, x_D\}$) and tends to be very precise for measuring unequally spaced points. Each PDF of the BMM (i.e., an area under the curve) denotes the area of each network feature which is considered $f(x)$ in the TAE.

The trapezoidal rule belongs to one of the numerical integration families called ‘Newton-Cotes formulas’ [280]. Its purpose is to appraise $V_n = \int_a^b f(x_{1:d})dx$, where a and b refer to the lower and upper boundaries of each feature/variable (x), respectively, $f(x)$ the PDF computed by equation (5.7) and D the number of features created from the feature reduction method.

When a trapezoidal rule is applied to multivariate data, it is called a ‘composite trapezoidal rule’, as depicted in Figure 5.3. It is obtained by integrating each sub-interval ($[x_{d-1}, x_d]$, where $d = 1, 2, \dots, D$) considering the interval points $a = x_1 < x_2 < \dots < x_D = b$ as

$$\int_a^b f(x)dx = \sum_{i=1}^d \int_{x_{d-1}}^{x_d} f(x)dx \quad (5.18)$$

$$\approx \frac{1}{2} \sum_{i=1}^d (x_d - x_{d-1}) [f(x_{d-1}) + f(x_d)]$$

In Figure 5.3, as the variables are assumed to be from a uniform grid [281], they are considered to be of equal length. Therefore, the total geometric area of each observation used in the construction of the normal profile and testing phase is calculated by

$$area(V) = \int_a^b f(x)dx \quad (5.19)$$

$$= \frac{b-a}{D} [f(x_1) + 2 \sum_{i=1}^{D-1} f(x_i) + f(x_D)]$$

5.3.3. Construction of Normal Profile of GAA-ADS Technique

Constructing a normal profile that has information from only normal observations is an essential step in applying the anomaly methodology. The validity of a purely

legitimate profile can be ascertained by providing secure normal traffic which ensures the credibility of detection. Given a set of normal vectors ($r_{1:n}^{normal}$) in which each vector involves a set of features, where $r_{1:n}^{normal} = \{x_1, x_2, \dots, x_D\}^{normal}$, the normal profile includes only statistical information from these vectors, that is, the estimated parameters of the BMM and distances between the means of both the BMM and trapezoidal area. The distance between each vector and the mean of the BMM ($distance_n$) refers to the absolute distance between the mean of all normal vectors ($\mu = 1/N \sum_{i=1}^N v_i / (v_i + \omega_i)$) and that of each legitimate vector ($\mu_n = 1/D \sum_{d=1}^D v_{nd} / (v_{nd} + \omega_{nd})$) which is computed by

$$distance_n = |\mu - \mu_n| \quad (5.20)$$

While the absolute distance is effectively used in cluster and outlier detection techniques, in the proposed GAA-ADS one, it detects dissimilarities between normal and anomalous observations because, as stated in [274], a distance measure can accurately estimate dissimilarities between different PDFs. It is practically proven that, if the areas of normal vectors are roughly similar to those of attack vectors, the sum of each PDF ($f(x_{nD})$) and absolute distance ($distance_n$) distinguishes between them.

Algorithm 5.1 presents the suggested steps for constructing a normal profile (*prof*), with the parameters of the BMM (π, v, ω) computed for all the normal vectors ($r_{1:n}^{normal}$) using equations (5.8) to (5.17) and then used to calculate the PDFs of the features ($x_{1:D}$) using equations (5.5) to (5.7). To make a clear distinction between normal and abnormal records, the sum of the PDFs and absolute distances, namely (*filterset_n*), is estimated and the TAE of each observation ($area_n^{normal}$) computed by it using equation (5.19).

Algorithm 5.1 Construction of normal profile of GAA-ADS technique

Input: normal records ($r_{1:n}^{normal}$)
Output: normal profile ($prof$)

- 1: **for** (all $r_{1:n}^{normal}$) **do**
- 2: $[\Pi_d, v_{nd}, \omega_{nd}] \leftarrow$ estimate the parameters (π, v, ω) of the BMM using equations (5.8) to (5.19).
- 3: $f(x_{nd}) \leftarrow$ compute the PDFs using equations (5.5) to (5.7) based on the parameters estimated in Step 2.
- 4: $\mu_n = 1/D \sum_{d=1}^D v_{nd}/(v_{nd} + \omega_{nd})$ using equation (5.6)
- 5: **end for**
- 6: $\mu = 1/N \sum_{i=1}^N v_i/(v_i + \omega_i)$ using (5.6)
- 7: $distance_n = |\mu - \mu_n|$ using (5.20)
- 8: $filterset_n \leftarrow (f(x_{nd}) + distance_n)$
- 9: $area_n^{normal} \leftarrow$ compute the TAE for the $filterset_n$ using (5.19)
- 10: sort $area_n^{normal}$ and divide them into K ranges $([min_{Ki}, max_{Ki}])$
- 11: $prof \leftarrow (\Pi_d, v_{nd}, \omega_{nd}, \mu_n, [min_{Ki}, max_{Ki}])$
- 12: **return** ($prof$)

Step 10 in Algorithm 5.1 takes a long processing time as it compares each area of a testing record with all the estimated legitimate areas ($area_n^{normal}$). In order to significantly reduce this time, the $area_n^{normal}$ are sorted and divided into K_i ranges, with each K_i referring to a minimum and maximum value (min_{Ki} and max_{Ki} , respectively). Mathematically speaking, all the possible values of ranges (i.e., K_{values}) of N observations in a dataset are computed by

$$K_{values} = \{\lfloor N/2 \rfloor, \lfloor (N-1)/2 \rfloor, \lfloor (N-2)/2 \rfloor, \dots, \lfloor 4/2 \rfloor\} \quad (5.21)$$

$$Subject to, K > 1, N \geq 4$$

In equation (5.21), the upper and lower values of K are ($\lfloor N/2 \rfloor$) and ($\lfloor 4/2 \rfloor$), respectively, and, as the values following after the lower one are 1 (i.e., $K = 1$), they are excluded as they denote the same original interval; for example, in Table 5.1, we have $N = 6$ normal areas with their $K_{values} = \{\lfloor 6/2 \rfloor = 3, \lfloor 5/2 \rfloor =$

$2, \lfloor 4/2 \rfloor = 2 \} = \{2, 3\}$, computed using from equation (5.21), and these K_{values} are used to generate K_i ranges. These ranges reduce the number of areas of attack falling into normal areas, particularly if normal and attack areas are closed. As a result, normal records can be detected easily at real time if their areas fall into a K_i range, i.e., between \min_{K_i} and \max_{K_i} , otherwise they are considered attack records, as detailed in Algorithm 5.2. The estimated parameters $(\Pi, v_{nd}, \omega_{nd}, \mu_n, [\min_{K_i}, \max_{K_i}])$ are stored in the normal profile (*prof*) for the testing phase and decision-making method for detecting attacks.

5.3.4. Testing Phase and Decision-making Method of GAA-ADS Technique

In the testing phase, the testing area ($area_n^{testing}$) of each observed record ($r^{testing}$) is calculated using the estimated parameters of the legitimate profile (*prof*). Algorithm 5.2 presents the steps in this phase and the decision-making method for detecting the areas of attack records. Steps 1 to 4 estimate the ($area_n^{testing}$) using the stored normal parameters $(\Pi_d, v_{nd}, \omega_{nd}, \mu_n)$.

The decision method is presented in steps 5 to 15 in which each area of a testing record ($area_n^{testing}$) is compared with the area ranges of the normal profile $[\min_{K_i}, \max_{K_i}]$. If $(area_n^{testing})$ falls within any $[\min_{K_i}, \max_{K_i}]$ range, it is considered a normal record, otherwise an attack one.

Table 5.1 presents examples of suspicious observations identified based on the areas estimated using the TAE. The training phase consists of 6 areas of normal observations and the testing phase 3 of normal and 3 of attack observations. The ranges in the training phase are divided into K=2 and K=3 and called ‘Case 1’ and ‘Case 2’, respectively. Case 1 has 2 ranges, (0.3, 0.5) and (0.55, 0.82), labelled ‘11’

Algorithm 5.2 Anomaly detection using TAE area estimation of GAA-ADS technique

Input: observed record $(r^{testing})$, normal profile $(\Pi_d, v_{nd}, \omega_{nd}, \mu_n, [min_{Ki}, max_{Ki}])$, flag=1 // a record is normal or not

Output: normal or attack record

```

1:  $distance^{testing} = |\mu_n - \mu^{testing}|$ 
2: compute  $f(x^{testing})$  using the parameters  $\Pi_d, v_{nd}, \omega_{nd}$ 
3:  $filterset^{testing} \leftarrow (f(x^{testing}) + distance^{testing})$ 
4:  $area^{testing} \leftarrow$  compute the TAE for the  $filterset^{testing}$ 
5: for (i to length ( $K$  ranges) ) do
6:   if ( $area^{testing} \geq min_{Ki}$  &&  $area^{testing} \leq max_{Ki}$ ) then
7:     flag = 0
8:     break
9:   end if
10: end for
11: if (flag==0) then
12:   return (normal)
13: else
14:   return (attack)
15: end if
```

and ‘12’, respectively, and Case 2 has 3 ranges, (0.3, 0.35), (0.5, 0.55) and (0.73, 0.82), labelled ‘21’, ‘22’ and ‘23’, respectively. In steps 5 to 15 in Algorithm 5.2, the detection of attack observations indicates that all the records in Case 1 except ‘0.72’ and all those in Case 2 are correctly classified. Therefore, the K ranges have a great impact on the DR of the GAA-ADS technique. With a gradual increase in K from 2 to 3, the DR increases and successfully detects all the suspicious areas.

Although the proposed GAA-ADS technique produces promising results using the NSL-KDD and UNSW-NB15 datasets, there are some abnormal areas which might fall into normal ones due to their close area estimates and reflect overlapping between these areas, as presented in the proposed framework and experimental results in Section 5.5. We propose a new ADS technique based on the DMM and interquartile range (IQR) approaches for detecting existing and zero-day attacks with slight differences between normal and suspicious observations in the following section.

Table 5.1: Examples of identifying attacks using estimated areas

Training phase	0.3	0.35	0.5	0.55	0.73	0.82
Testing phase	normal areas			attack areas		
	0.55	0.79	0.34	0.1	0.72	0.9
Case 1	K=2					
	Ranges label	Min	Max			
	11	0.3	0.5			
	12	0.55	0.82			
decision making						
Record areas	0.55	0.79	0.34	0.1	0.72	0.9
Ranges label	12	12	11	0	12	0
detection	normal	normal	normal	attack	normal(false)	attack
Case 2	k=3					
	Ranges label	min	max			
	21	0.3	0.35			
	22	0.5	0.55			
	23	0.73	0.82			
decision making						
Record areas	0.55	0.79	0.34	0.1	0.72	0.9
Ranges label	22	23	21	0	0	0
detection	normal	normal	normal	attack	attack	attack

5.4. Novel Dirichlet Mixture Model-based ADS (DMM-ADS) Technique

This section discusses the mathematical aspects of estimating and modelling network data using the DMM and explains the proposed methodology for developing an intelligent ADS for effectively handling large-scale networks.

5.4.1. Finite DMM

Since a finite mixture model can be considered a convex combination of two or more PDFs, the joint characteristics of which can estimate any random distribution, it is a powerful probabilistic modelling tool for multivariate data, such as network data [161]. Its graphical model parameters are depicted in Figure 5.4 in which

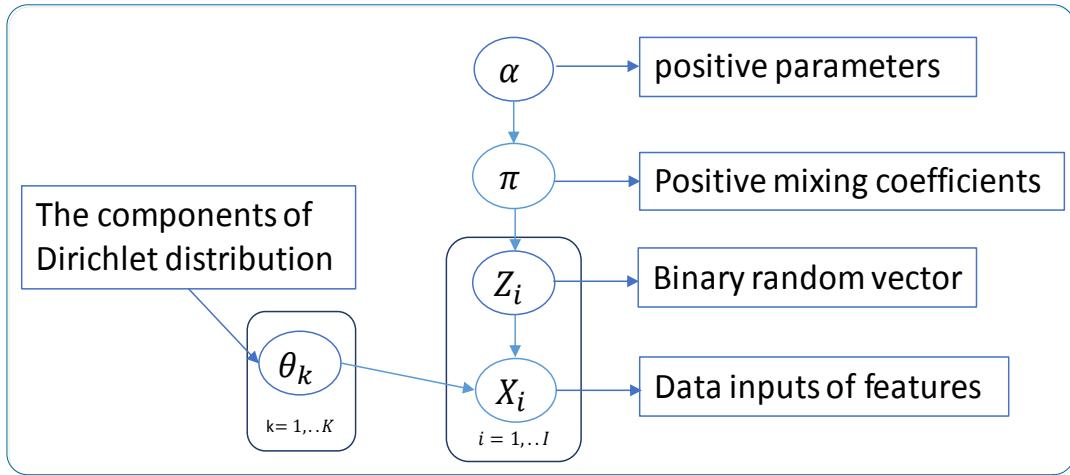


Figure 5.4: Parameters of finite DMM

the symbols in circles refer to the arbitrary variables and model parameters that demonstrate the conditional dependencies between variables².

A finite mixture of Dirichlet distributions with K components is computed by [162, 278]

$$p(X|\pi, \alpha) = \sum_{i=1}^K \pi_i Dir(X|\alpha_i) \quad (5.22)$$

where $\pi = (\pi_1, \dots, \pi_K)$ indicates the mixing coefficients, which are usually positive, with their summation 1, $\sum_{i=1}^K \pi_i, \alpha = (\alpha_1, \dots, \alpha_K)$, and $Dir(X|\alpha_i)$ refers to the Dirichlet distribution of component i with its own positive parameters ($\alpha = (\alpha_{i1}, \dots, \alpha_{iS})$) computed by

²

- A Part of this study has been released in:
Moustafa, N., Creech, G and J. Slay. "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models." Data Analytics and Decision Support for Cybersecurity. Springer, 2017. 127-156.

$$Dir(X|\alpha_i) = \frac{\Gamma(\sum_{s=1}^S \alpha_{is})}{\prod_{s=1}^S \Gamma(\alpha_{is})} \prod_{s=1}^S X_s^{\alpha_{is}-1} \quad (5.23)$$

where $X = (X_1, \dots, X_S)$, S is the dimensionality of X and $\sum_{s=1}^S x_s = 1$, $0 \leq X_s \geq 1$ for $s = 1, \dots, S$. It is worth noting that a Dirichlet distribution is used as a posterior one to directly model the data rather than as a prior one for multinomial data.

Considering a set of N independent identically distributed (*i.i.d*) observations ($X = \{X_1, \dots, X_N\}$) assumed to be shaped from the mixture distribution in equation (5.23), the probability function of the DMM is

$$p(X|\pi, \alpha) = \prod_{l=1}^N \left\{ \sum_{i=1}^K \Pi_i Dir(X_l|\alpha_i) \right\} \quad (5.24)$$

It is important to understand that, as the finite mixture model in equation (5.24) is a latent flexible one, for each observation (X_i), a D -dimensional binary random vector ($Z_i = \{Z_{i1}, \dots, Z_{iD}\}$) is defined, where $Z_{is} \in \{0, 1\}$, $\sum_{i=1}^D$ and $Z_{is} = 1$ if X_i follows component i , else 0. The latent variables ($Z = \{Z_1, \dots, Z_N\}$), which are actually concealed and do not appear clearly in the model, and its conditional distribution can be given by the mixing coefficients (π) and calculated by

$$p(Z|\pi) = \prod_{l=1}^N \prod_{i=1}^K \pi_i^{Z_{li}} \quad (5.25)$$

The likelihood function of the latent variables, which represents the conditional distribution of a dataset (X) given the class labels (Z), can be formulated as

$$p(X|\pi, \alpha) = \prod_{l=1}^N \prod_{i=1}^K Dir(X_l|\alpha_i) \quad (5.26)$$

Given a dataset (X), a significant problem is determining the learning theory for the mixture parameters, that is, both computing the parameters and specifying the number of components (K). Firstly, to estimate the parameters of the finite DMM, we use the variational inference presented in [64]. Secondly, we use the number of components based on the principal components of the PCA technique that generate the highest variations of features to find the differences between normal and malicious features, as described in subsection 5.6.3.

In summary, the three main DMM parameters (π, α, Z) are estimated using the variational learning theory to build an intelligent ADS that can efficiently detect known and zero-day attacks. They are used to model network data in a novel technique called DMM-ADS which is proposed for the first time in this PhD thesis and includes training and testing phases for learning and testing network data. In the training phase, these parameters and the IQR are estimated to establish a legitimate profile, with anomalous observations detected in the testing phase, as elaborated in the following two subsections.

5.4.2. Training Phase of Normal Observations of DMM-ADS Technique

The proposed methodology for correctly applying anomaly detection depends mainly on training normal data in the training phase while constructing a normal profile and treating any variation from it as an attack in the testing phase. This can be achieved by considering a set of normal observations ($r_{1:n}^{normal}$) in which each observation includes a set of features, such that $r_{1:n}^{normal} = \{x_1, x_2, \dots, x_D\}^{normal}$, with the normal profile involving only statistical characteristics about legitimate observations of ($r_{1:n}^{normal}$). In this methodology, the normal profile contains the estimated parameters (π, α, Z) of the DMM for estimating the PDF of the Dirichlet distribution ($Dir(X|\pi, \alpha, Z)$) for each feature vector in the training set.

The suggested steps for constructing a legitimate profile ($prof$), with the parameters of the DMM (π, α, Z) estimated for all the legitimate observations $(r_{1:n}^{normal})$ using the equations presented in [162], are described in Algorithm 5.3. Then, the PDFs of the network features $(X_{1:D})$ are computed using equations (5.22) to (5.26). Following that, the IQR is estimated by subtracting the first from the third quartile of the PDFs [161] in order to specify a dynamic threshold for distinguishing suspicious observations in the testing phase. It is acknowledged that quartiles can divide data into adjacent intervals with equal probabilities for easily specifying data boundaries [282].

Algorithm 5.3 Establishment of normal profile of DMM-ADS technique

Input: normal instances $(r_{1:n}^{normal})$
Output: normal profile $(prof)$

- 1: **for** (each record i in $(r_{1:n}^{normal})$) **do**
- 2: estimate the parameters (π_i, α_i, Z_i) of the DMM as in [162]
- 3: calculate the PDFs using equations (5.22) to (5.26) based on the estimated parameters of Step 2
- 4: **end for**
- 5: compute $lower = quartile(PDFs, 1)$
- 6: compute $upper = quartile(PDFs, 3)$
- 7: compute $IQR = upper - lower$
- 8: $pro \leftarrow ((\pi, \alpha_i, Z_i), (lower, upper, IQR))$
- 9: **return** (pro)

5.4.3. Testing Phase and Decision-making Method in DMM-ADS Technique

In the testing phase, the Dirichlet PDF $(PDF^{testing})$ of each testing observation $(r^{testing})$ is calculated based on the same parameters computed for the legitimate profile $(prof)$. Algorithm 5.4 presents the steps in this phase and the

decision-making method for recognising the Dirichlet PDFs of suspicious observations, with step 1 estimating the PDF of each $r^{testing}$ using the same legitimate parameters(π_i, α_i, Z_i) stored in the normal profile.

Algorithm 5.4 Testing phase and decision-making method in DMM-ADS technique

Input: observed instance ($r^{testing}$), pro

Output: normal or attack

- 1: calculate the $PDF^{testing}$ using equations using the parameters (π_i, α_i, Z_i)
 - 2: **if** $((PDF^{testing} < (lower - w * (IQR)) \parallel (PDF^{testing} > (upper + w * (IQR))))$
then
 - 3: **return** (attack)
 - 4: **else**
 - 5: **return** (normal)
 - 6: **end if**
-

Steps 2 to 6 are the fundamental steps in the decision-making method. More specifically, the IQR of the legitimate observations is calculated to discover the outliers/anomalies of any ($r^{testing}$) in the testing phase. This is achieved by treating the observations below the lower baseline (i.e., $lower - w * (IQR)$) or above the upper baseline (i.e., $upper + w * (IQR)$), where w refers to the interval values between 1.5 and 3 [282]. This interval mathematically proves that any data from the same distribution can be varied in this particular range, with any data points located outside this range considered outliers [282]. Similarly, in network anomaly detection, we consider that the detection decision relies on handling any $PDF^{testing}$ located outside this range as an anomalous instance which produces the promising results discussed in subsection 5.6.3.

5.5. Two Proposed Scalable Frameworks for ADS

In this section, we propose two scalable frameworks for designing an adaptive and lightweight ADS that can reliably recognise malicious observations in large-scale networks. Each framework involves three modules: data sniffing and storing; data pre-processing; and a novel technique. The main difference between these two frameworks is their DE modules, with those in Figures 5.5 and 5.6 the new GAA-ADS and DMM-ADS techniques discussed in Sections 5.3 and 5.4, respectively.

In the first module, a set of features is extracted and generated from network ingress traffic with existing secure servers used to elicit network connections for a well-defined time window. The second module determines and filters network data in three steps. Firstly, feature conversion replaces the symbolic features with numeric ones because the proposed ADS techniques can process only numeric features. Secondly, feature reduction uses the PCA technique to select a small number of uncorrelated original features and their principal components to improve the ADS's performance and accuracy. Finally, feature normalisation converts the original subset of features or their principal components into a fixed range of [0, 1], an essential step in computing the beta distribution used in the two new ADS techniques.

The third module consists of the two new DE techniques (i.e., GAA-ADS and DMM-ADS) discussed above which detect suspicious instances based on carefully analysing a network's normal and suspicious data. In their training phases, normal observations are computed using some statistical parameters based on the potential

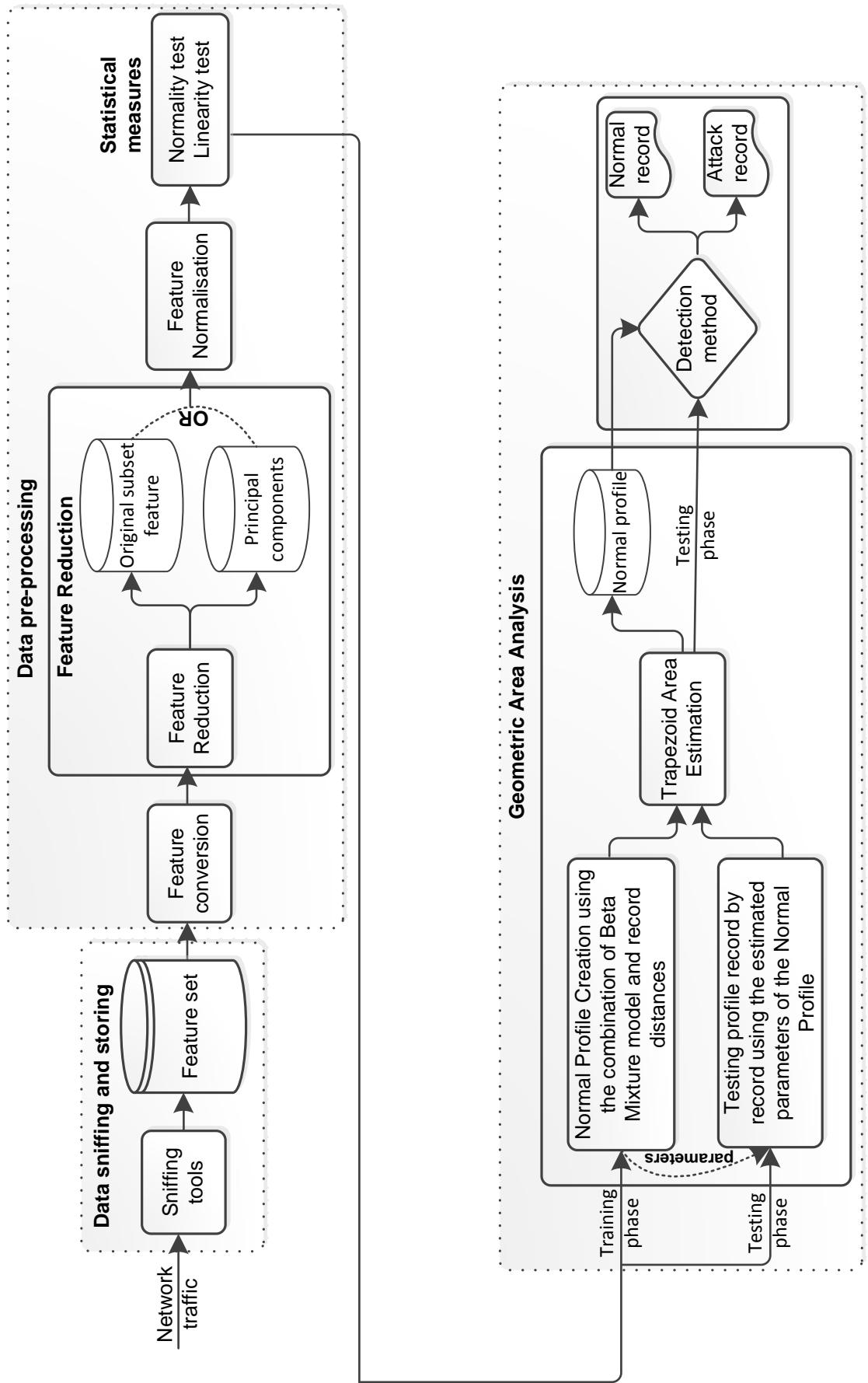


Figure 5.5: Proposed framework for establishing scalable, adaptive and lightweight GAA-ADS

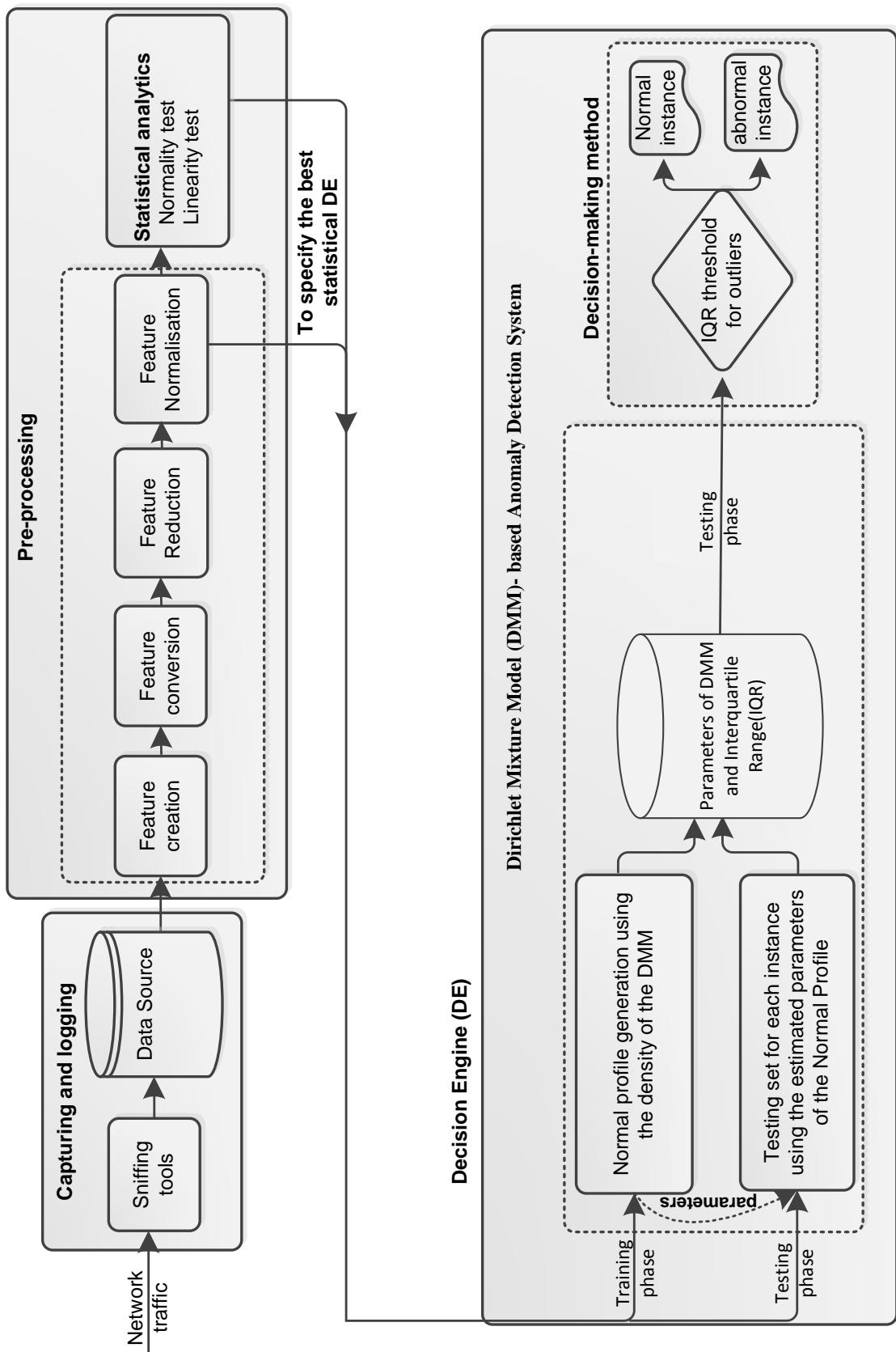


Figure 5.6: Proposed scalable framework for design of intelligent DMM-ADS

process of each technique for building a normal profile from them. Then, the same parameters are used to compute the testing observations while considering any variation from normal areas an anomaly based on the proposed decision-making methods. The first two modules in these frameworks are detailed in the following two sections.

5.5.1. Data Sniffing and Storing Module

As discussed in Chapter 3, the IXIA PerfectStorm tool [193] is used to simulate current realistic legitimate and malicious network traffic, with its testbed configured to handle a large-scale network [42, 43]. The Tcpdump tool is used to sniff raw packets from the network interface while the Bro and Argus tools as well as some new scripts are applied to extract and generate a set of features from these packets. To accumulate network flows in network systems, it is vital that ingress devices, such as routers and switches, are used to obtain these features. These flows are collected at the destination nodes of each network with respect to their source/destination IPs and protocols (i.e., flow identifiers) for a specific time window using the new aggregator module presented in Chapter 4, Section 4.3 to decrease the overhead incurred in identifying suspicious activities. We create the features for the UNSW-NB15 dataset by developing an extractor module which aggregates packets using the flow identifiers for each 100 connection records for the easy analysis, determination and classification of network data [42, 43].

This module comprises steps for sniffing network data and logging them for processing by the DE module which are similar to those for the design of the UNSW-NB15 dataset. An IXIA PerfectStorm tool [193], which has the capability to determine a broad range of network segments and extract traffic from several web applications, for example, Facebook, Skype, YouTube and Google, is used to mimic modern realistic normal and suspicious events, as depicted in Figures 5.5 and 5.6, respectively. Moreover, it simulates the majority of security events and malicious scripts which are difficult to find using other existing tools. The

configuration of the UNSW-NB15 testbed is used to simulate a large-scale network and the Tcpdump tool to sniff raw packets from the network’s interface while the Bro, Argus tools and other scripts elicit a set of features from network traffic for evaluating the new IDS methodologies.

These features are logged using the MySQL Cluster CGE technology [57] which has a highly scalable and real-time database, that enables a distributed architecture to read and write intensive workloads for access by SQL or NoSQL APIs. It can also handle memory-optimised and disk-based tables as well as automatic data partitioning with load balancing, and insert nodes into a running cluster to process online big data. Although this technology has a similar architecture to the Hadoop tools [283] that are the most common for handling big offline data, our target in this PhD research is to develop a scalable, adaptive and online ADS that identifies malicious behaviours. Therefore, the MySQL Cluster CGE technology for an ADS is used to store network data which are then passed to the pre-processing module for analysis and filtering.

5.5.2. Data Pre-processing Module

The sets of features in the NSL-KDD [39, 80] and UNSW-NB15 [42, 43] datasets are used to process compatible input to the proposed GAA-ADS and DMM-ADS techniques through the three steps of feature conversion, feature reduction and feature normalisation.

A. Feature conversion

Although these datasets include both quantitative and qualitative attributes, the proposed DE techniques can process only quantitative features because, as they are statistical models, they can handle only numeric values. Consequently, a unified format for the features (X) is used to convert a symbolic feature into a numeric one; for instance, the UNSW-NB15 dataset contains three symbolic features, protocol

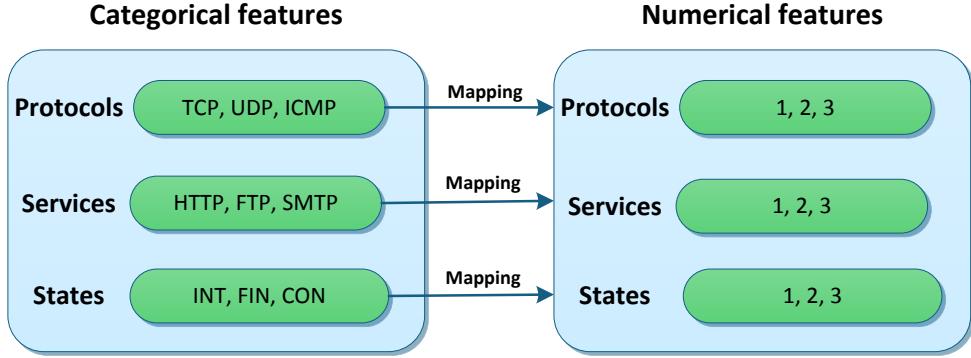


Figure 5.7: Example of converting categorical features into numerical features using UNSW-NB15 dataset

types (e.g., TCP and UPD), states (e.g., CON and ACC) and services (e.g., HTTP and FTP), as shown in Figure 5.7. In each dataset, this function replaces the values in the features with ordered numbers, such as CON=1 and ACC=2.

B. Feature reduction (FR)

Feature reduction is a method for removing inappropriate, redundant and/or noisy features. In [44], it is divided into feature selection, which discovers a subset of the original features, and feature extraction which transforms the data from a high-dimensional space into a lower-dimensional one. In [90, 284, 285], the PCA, which is one of the best-known linear FR techniques, has the advantages of requiring less memory storage, data transfer and processing time, and has better detection accuracy than other methods [23, 90, 133, 286]. Therefore, after extracting features from network data, the first phase in establishing a lightweight ADS is feature reduction for which the PCA technique is used, as explained in Chapter 4, subsection 4.5.2. This technique ranks network features based on the highest variance of each and generates a new dimensional space of uncorrelated features by eliminating low-variance features [90, 286]. The input to the GAA-ADS and

DMM-ADS techniques is the original features or principal components adopted (Table 5.2).

C. Feature normalisation

After reducing the feature set, a necessary step in the data pre-processing module is feature normalisation which is a method for scaling the value of each feature into a particular range, with its key advantage removing the bias from raw data without modifying their statistical properties. The beta distributions used in the proposed DE techniques are designed to model data in a certain range, such as $[0, 1]$, for the features (x_i) input to the DE techniques which are normalised into that range by the linear transformation computed as

$$x_i^{\text{normalised}} = (x_i - \min(x)) / (\max(x) - \min(x)) \quad (5.27)$$

5.6. Experimental Results and Analysis

This section discusses the experimental results and statistical analyses related to the proposed DE techniques and compares them with those obtained from recent peer mechanisms to demonstrate their efficiency and effectiveness for designing an intelligent ADS for a large-scale network.

A. Pre-processing Phase

The two novel DE techniques (i.e., GAA-ADS and DMM-ADS) are evaluated using 15 original features and their principal components from the NSL-KDD and UNSW-NB15 datasets selected using the PCA mechanism, as listed in Table 5.2. The largest number of features with higher variances is selected to reveal that each

Table 5.2: Features selected from NSL-KDD and UNSW-NB15 datasets

Datasets	Selected features
NSL-KDD	<i>srv_count, dst_host_srv_count, count, src_bytes, dst_host_same_srv_rate, dst_host_count, srv_diff_host_rate, srv_error_rate, dst_host_srv_error_rate, diff_srv_rate dst_host_rerror_rate, rerror_rate, is_guest_login, num_outbound_cmds, dst_host_srv_diff_host_rate</i>
UNSW-NB15	<i>ct_dst_sport_ltm, tcprtt, dwin, sjit, ct_state_ttl ct_src_dport_ltm, dbytes, ct_dst_src_ltm, ct_dst_ltm, smean, dmean, service, proto, dtcpb, ct_src_ltm</i>

feature can affect the performances of the DE techniques. To correctly apply the learning theory, these features do not depend on each other but on the predictor (i.e., class label) to improve the efficacy of developing a ML technique.

The new DE techniques are developed using the ‘R programming language’ on Linux Ubuntu 14.04 with 16 GB RAM and an i7 CPU processor. In order to carry out experiments on each dataset, we select random samples from the ‘full’ NSL-KDD dataset [39] and the CSV files of the UNSW-NB15 dataset with diverse sample sizes of between 100,000 and 300,000 (s). Each legitimate sample is approximately 60 to 75% of the total sample size, with some used to construct the legitimate profile in the training phase and others employed in the testing phase. The performances of the GAA- and DMM-ADS techniques are assessed using a 10-fold cross-validation of the sample sizes to determine their impacts with no bias towards some samples.

5.6.1. Statistical Analysis and Decision Support

A statistical analysis has a great impact as it interprets network data patterns and reveals to what extent suspicious observations are different from normal observations. Three statistical measures, Q-Q, density and correntropy plots, are applied

to the network data to find these differences.

Firstly, a Q-Q plot is a graphical representation that determines if a set of data comes from a normal theoretical distribution. Network features are considered to be from a normal distribution if their values are located on the same theoretical distribution line, as shown in Figure 5.8 in which (A) and (B) indicate that the selected features of the two datasets do not follow the theoretical distribution lines (i.e., red ones) and there are much greater differences between them than in their feature value lines.

Therefore, these network features do not properly fit a Gaussian distribution, as also proven using the K-S test in Chapter 3. Since the BMM and DMM are two of the best models for fitting non-normal distributions because they model more than one feature by accurately specifying their boundaries, they are used in the GAA-ADS and BMM-ADS techniques, respectively.

Secondly, the density probabilities of normal and malicious observations are estimated using some samples from the NSL-KDD and UNSW-NB15 datasets to reveal to what extent these observations vary, as shown in Figure 5.9. As, in the NSL-KDD dataset, those of the normal instances vary between 0 and 0.20, with their values specified as between -50 and 0, and those of the abnormal instances between 0 and 0.5, with their values ranging from -30 to 0. It is acknowledged that they are somewhat different. Similarly, in the UNSW-NB15 dataset, the density probabilities of the normal and suspicious observations also vary slightly. These findings emphasise that the proposed statistical decision-making methods in the two DE techniques can dramatically recognise malicious observations due to the differences between them and legitimate observations in the two datasets.

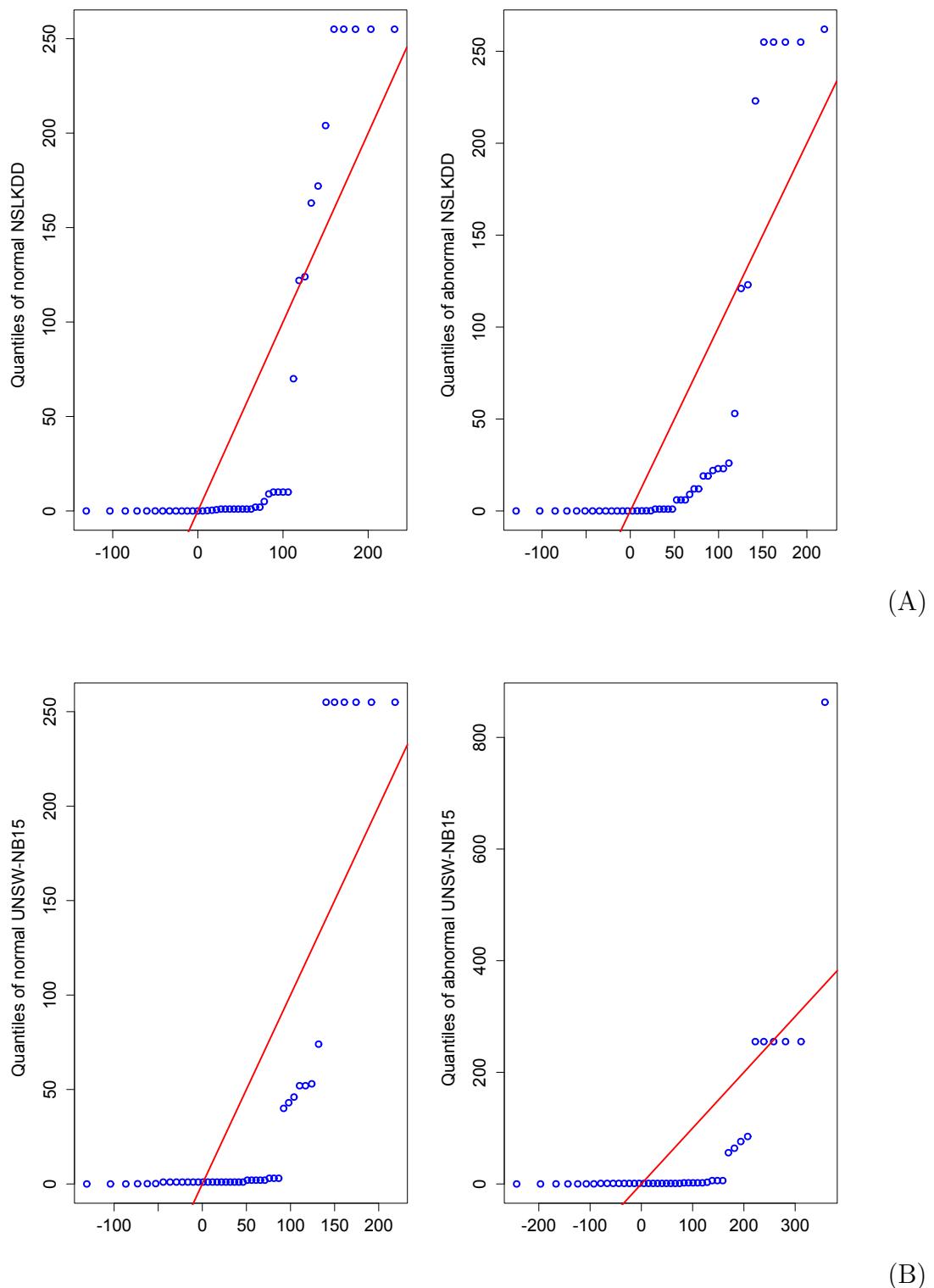


Figure 5.8: Q-Q plots of feature vectors adopted from NSL-KDD and UNSW-NB15 datasets

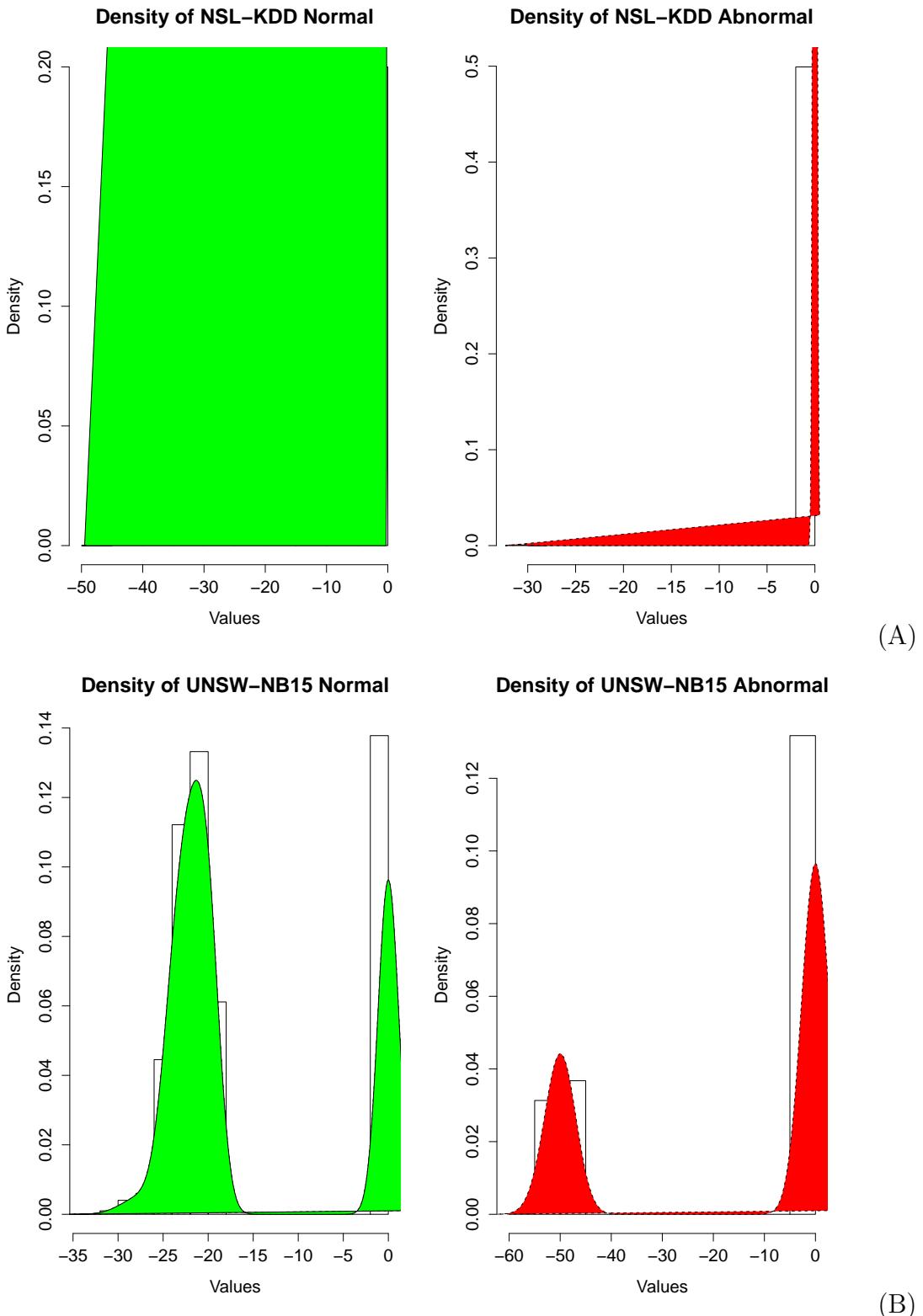


Figure 5.9: Normal and suspicious density probabilities for some instances in both datasets

Thirdly, the correntropy plots in Figure 5.10 also statistically illustrate the small differences between the values of normal and suspicious features for 100 instances in both datasets which simplifies the role of the two novel DE techniques for successfully recognising abnormal observations. The key reason for identifying variations between normal and malicious instances is that most of the features in both datasets were generated based on analysing their potential statistical properties, such as their interracial times and packet counts from header packets. These properties lead to enabling clear distinctions to be made between normal and attack observations, thereby improving the performances of the proposed DE techniques.

Overall, the above statistical measures demonstrate that network features cannot be plotted in a linear representation and modelled in a Gaussian distribution because recent anomalous activities attempt to mimic normal ones. Thereby, they highlight the need to choose decision-making methods that can satisfy these statistical constraints in order to establish an intelligent NADS that can identify small variations between legitimate and suspicious observations with respect to the high speeds and large sizes of current networks.

5.6.2. Performance Evaluation of GAA-ADS Technique

This subsection explains the performance evaluation of the proposed GAA-ADS technique on the original features and their principal components selected by the PCA technique which uses some criteria, in particular, the accuracy, DR and FPR, to assess efficiency and effectiveness.

A. Performance of GAA-ADS technique on original features

In order to provide a better overview of the performance of the GAA-ADS technique on the original features, the overall FPR, accuracy and DR are shown in

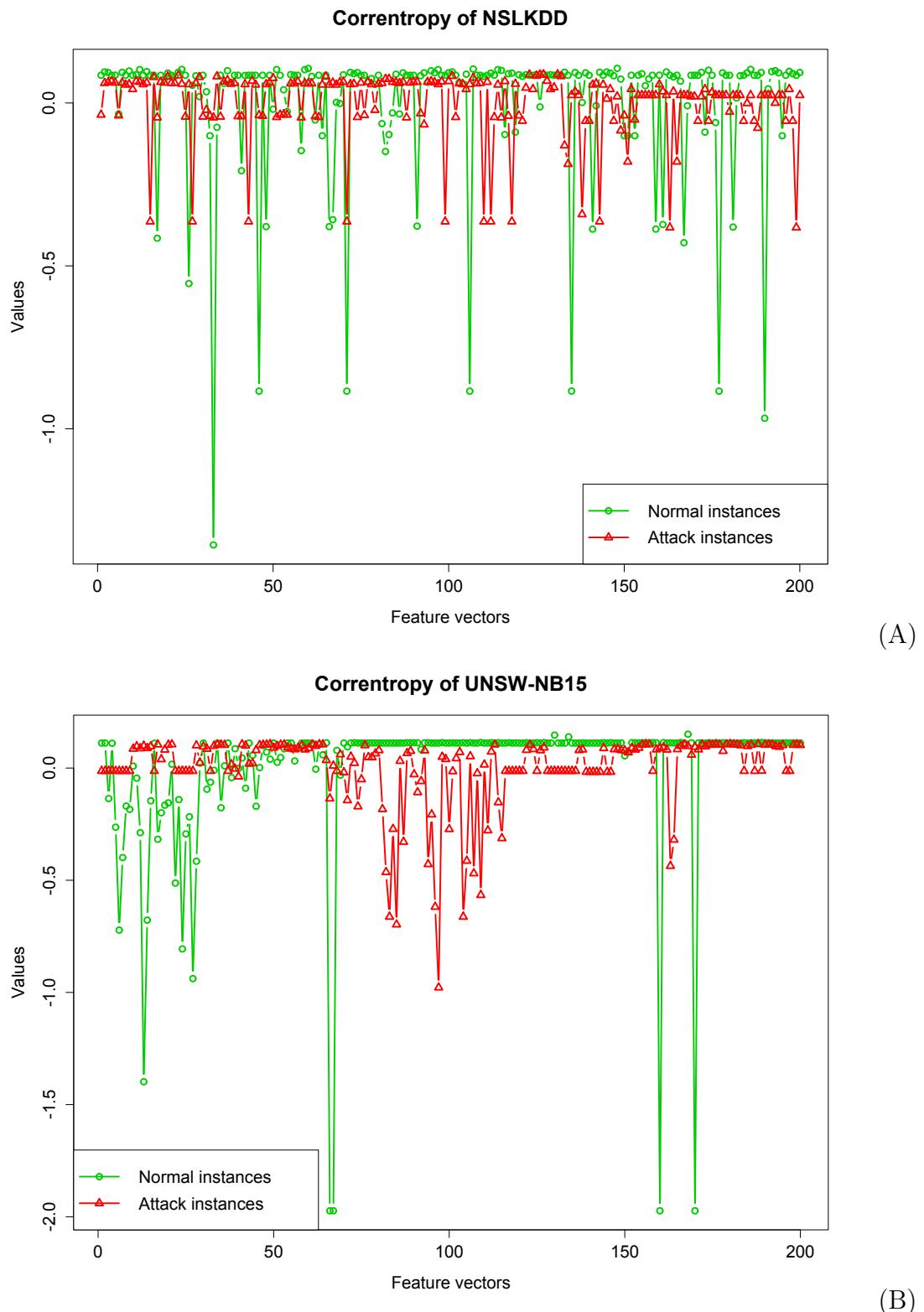


Figure 5.10: Correntropy plots of some instances in both datasets

Table 5.3: Evaluation of overall performances of GAA-ADS technique on original features

K value	NSL-KDD			UNSW-NB15		
	DR	Accuracy	FPR	DR	Accuracy	FPR
2	92.1%	92.3%	3.0%	75.1%	77.4%	8.3%
4	92.9%	93.6%	2.6%	81.3%	82.1%	7.4%
6	95.3%	95.5%	0.8%	85.2%	85.7%	7.0%
8	95.4%	95.6 %	0.7%	89.8%	90.2%	6.9%
10	98.1%	98.8%	0.4%	91.2%	91.8%	5.8%

Table 5.3 while, in Figure 5.11, the Receiver Operating Characteristics (ROC) curves represent the relationships between the DR and FPR with several K values. It is observed that the gradual rise in the K value from 2 to 10 with even numbers enhances the overall DR and accuracy and reduces the overall FPR. In the NSL-KDD dataset, the overall DR and accuracy increase from 92.1% to 98.1% and 92.3% to 98.8%, respectively, while the overall FPR decreases from 3.0% to 0.4%. Likewise, in the UNSW-NB15 dataset, the overall DR and accuracy increase from 75.1% to 91.2% and 77.4% and 91.8%, respectively, while the overall FPR decreases from 8.3% to 5.8%.

B. Performance of GAA-ADS technique on principal components

A summary of the performances of the GAA-ADS technique on the 15 principal components in terms of the overall FPR, accuracy and DR is presented in Table 5.4 while Figure 5.12 shows the ROC curves for the DR and FPR with different K values. Its overall performance on the principal components outperforms that on the original features by 1-2% as the former are generated based on the highest variations between the areas of normal and suspicious observations. In the NSL-KDD dataset, when the K value gradually increases from 2 to 10, the overall DR and accuracy increase from 94.2% to 99.6% and 95.0% to 99.7%, respectively,

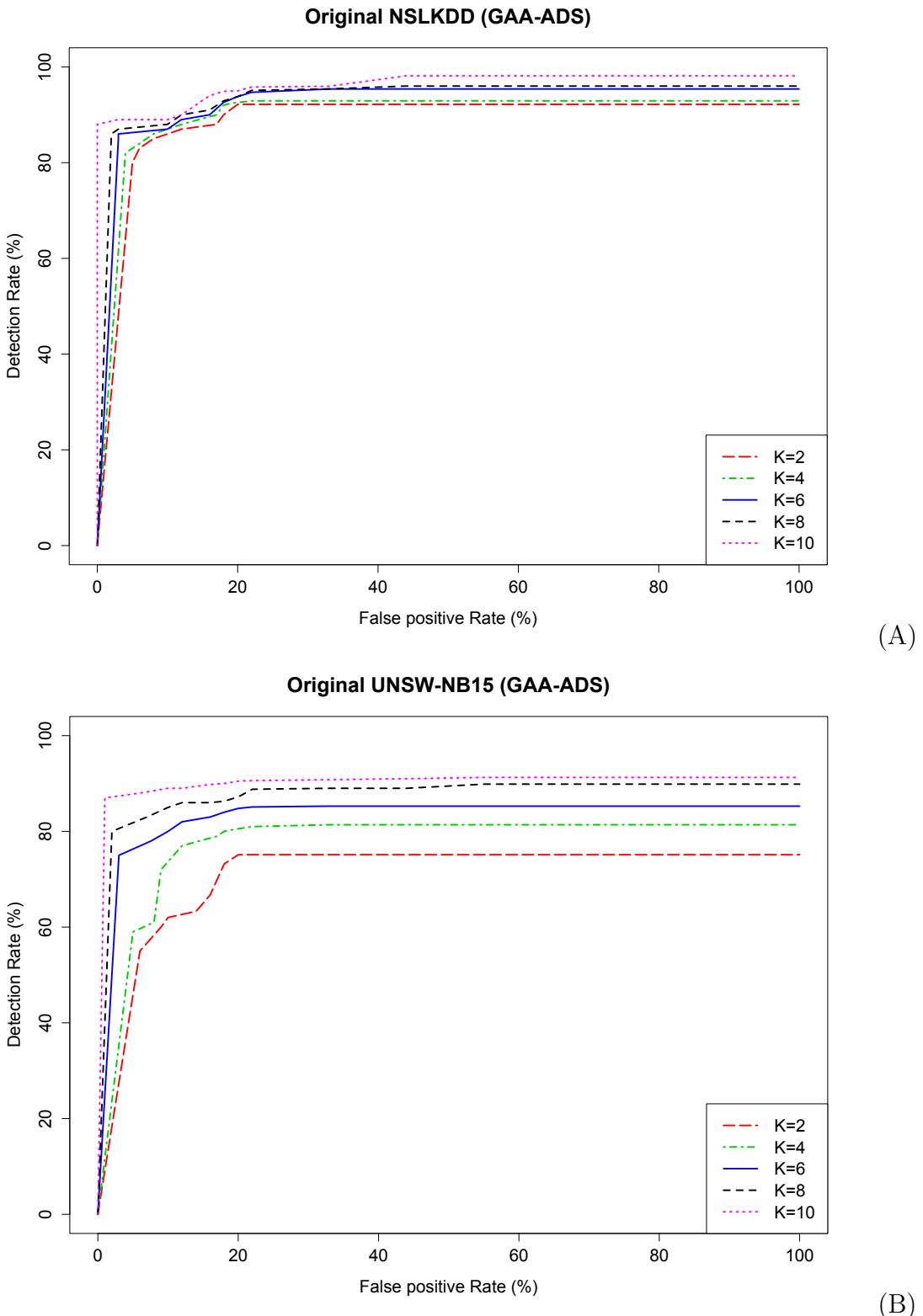


Figure 5.11: ROC curves for original features in two datasets obtained by GAA-ADS technique for different K values

Table 5.4: Estimations of overall performances of GAA-ADS technique on principal components

K value	NSL-KDD			UNSW-NB15		
	DR	Accuracy	FPR	DR	Accuracy	FPR
2	94.2%	95.0%	1.1%	75.4%	77.6%	8.2%
4	95.1%	95.3%	0.7%	85.2%	86.0%	6.3%
6	96.4%	97.7%	0.2%	87.1%	88.2%	6.1%
8	98.7%	98.8%	0.2%	91.2 %	92.7%	5.9%
10	99.6%	99.7%	0.2%	91.3%	92.8%	5.1%

while the overall FPR decreases from 1.1% to 0.2%. Similarly, in the UNSW-NB15 dataset, the overall DR and accuracy increase from 75.4% to 91.3% and 76.6 % to 92.8%, respectively, while the overall FPR decreases from 8.2% to 5.1%.

C. Performance comparisons of GAA-ADS technique

In Figure 5.13 (A), the ROC curves obtained from the above five evaluations of the two datasets indicate that the performances of the GAA-ADS technique on the principal components of the NSL-KDD dataset, which increasingly improve the DRs from 94.2% to 99.6%, are superior to those on the original features which increase the DRs from 92.1% to 98.1% whereas the areas of the FPRs under the ROC curves for the principal components (i.e., [0.4% - 3.0%]) are approximately 50% of those for the original features (i.e., [0.2% - 1.1 %]). On the other hand, Figure 5.13 (B) shows that the differences between the ROC curves for the original features and principal components in the UNSW-NB15 dataset are low, with changes in their DRs and FPRs approximately 0.2 % and 0.7%, respectively.

Tables 5.5 and 5.6 present comparisons of the DRs of the record types with the K values on the components of the NSL-KDD and UNSW-N15 datasets, respectively, which indicate that, as the K value increases, the DR gradually improves,

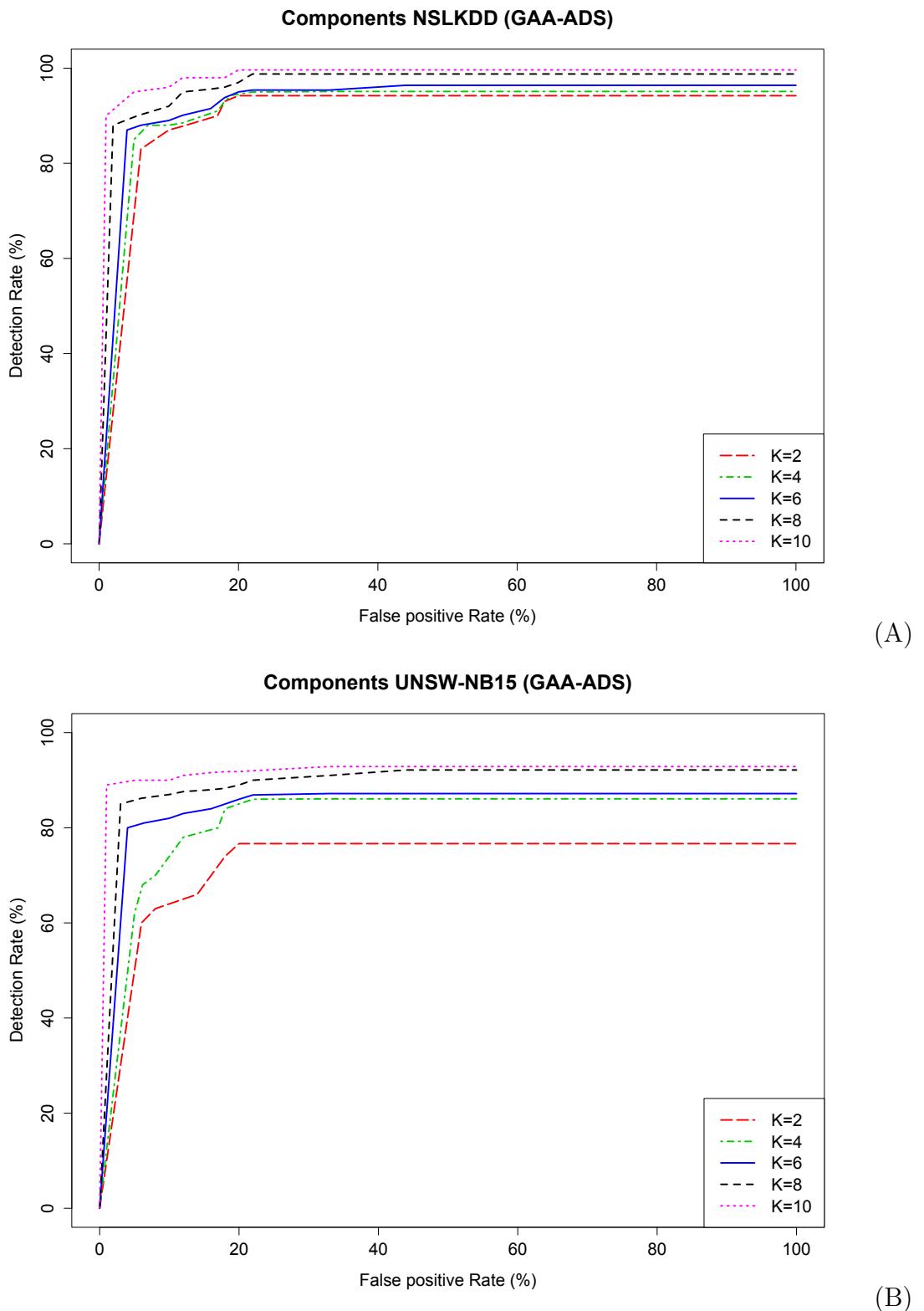


Figure 5.12: ROC curves obtained from GAA-ADS technique for components in both datasets with different K values

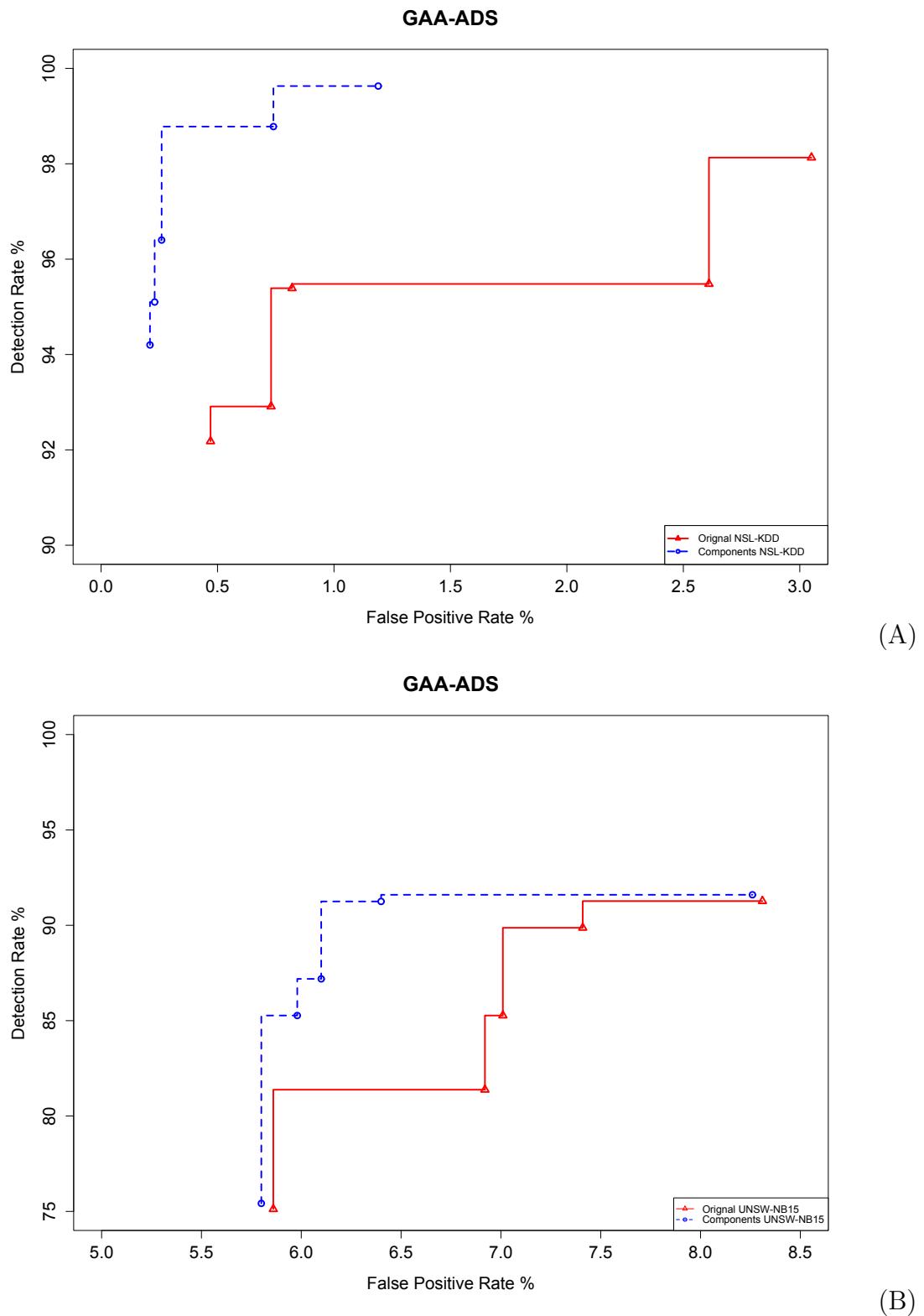


Figure 5.13: ROC curves obtained from GAA-ADS technique for both datasets

Table 5.5: Comparison of DRs (%) obtained by GAA-ADS technique for principal components in NSL-KDD dataset

Record type	K values				
	2	4	6	8	10
Normal	95.5%	96.2%	98.1%	99.3%	99.4%
Probe	93.2%	94.7%	96.2%	98.8%	99.6%
DoS	98.1%	98.3%	98.6%	100%	100%
U2R	93.7%	93.7%	95.1%	97.9%	98.4%
R2L	94.2%	94.3%	95.3%	96.7%	98.9%

as illustrated in Figure 5.14. Table 5.5 reveals that the GAA-ADS technique can identify the majority of record types in the NSL-KDD dataset with DRs varying between 93.2% and 100% whereas the DRs of normal observations increase from 95.5 % to 99.2%, with the lowest FNRs observed as the K value gradually increases from 2 to 10. Likewise, the DRs of malicious types (i.e., Probe, DoS, U2R and R2L) slowly increase from, on average, 93.2% to 98.5%.

Table 5.6 shows that the GAA-ADS technique recognises record types in the UNSW-NB15 dataset, with DRs moderately increasing from 42.0% to 93.0% while the DRs of normal records increase from 79.3% to 93.0% when the K value increases from 2 to 10. However, the DRs of anomalous types do not always progressively increase; for instance, the Analysis, Backdoor, Fuzzers, Reconnaissance and Shell-code attacks do not receive the highest DRs with the highest K values and their differences from previous values are low (approximately 1-2%) while the DRs of the other types of suspicious instances, DoS, Exploits, Generic and Worms, increase with increases in the K values. Since it can be seen that, for these types of observations, the variances of their components are close, their areas sometimes fall into each other when identifying areas of abnormal instances.

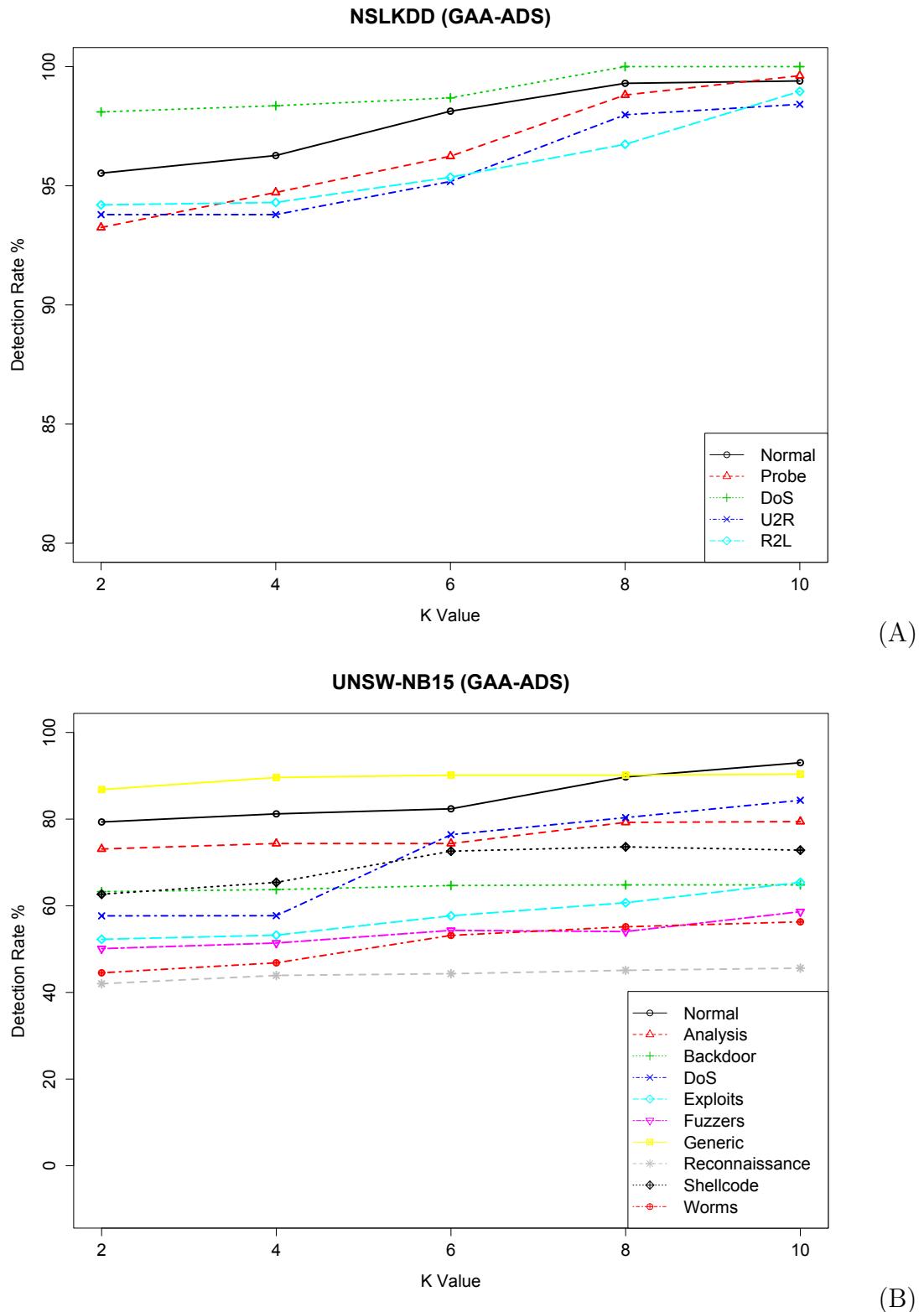


Figure 5.14: Comparison of DRs (%) obtained by GAA-ADS technique for both datasets with increasing K values

Table 5.6: Comparison of DRs (%) obtained by GAA-ADS technique for components of UNSW-NB15 dataset

Record type	K values				
	2	4	6	8	10
Normal	79.3%	81.2%	82.3%	89.7%	93.0%
Analysis	73.1%	74.3%	74.3%	79.2%	79.4%
Backdoor	63.2%	63.7%	64.6%	64.8%	64.8%
DoS	57.6%	75.7%	76.4%	80.3%	84.3%
Exploits	52.2%	53.2%	57.6%	60.7%	65.4%
Fuzzers	50.1%	51.4%	54.3%	54.0%	58.6%
Generic	86.8%	89.6%	90.1%	90.1%	90.3%
Reconnaissance	42.0%	43.9%	44.3%	45.1%	45.6%
Shellcode	62.6%	65.4%	72.6%	73.6%	73.8%
Worms	44.5%	46.8%	53.1%	55.1%	56.2%

Although this technique generally performs better on the NSL-KDD than UNSW-NB15 dataset, in some cases, some malicious observations in the UNSW-NB15 dataset are not high because of the lower variances between them and normal observations. As this dataset was simulated using a sophisticated network architecture to include a modern range of suspicious activities, the GAA-ADS technique can protect current networks because it could find all the malicious activities in real networks by synchronously monitoring their different nodes. In [205], it is claimed that deploying a framework in which each ADS collaborates with the others could efficiently protect network nodes. Therefore, the GAA-ADS technique could be a reasonable solution for detecting abnormal activities in real networks.

Estimating the performances of the GAA-ADS technique is carried out using two measures of the variances of its features/principal components and K values. Firstly, if the former are very high, the DR will be higher and the FPR lower. According to the experimental results, the principal components have the higher possible variances and always produce better detection accuracy than the original features because their variations between normal and anomalous observations make their computed areas different based on accurate estimations of their TAEs. Because the TAE estimates the precise areas of features, we use it to calculate the total area of each observation using the estimated BMM and distances between

observations because it can identify slight differences between the areas of the vector types.

Secondly, as gradual increases in the K value enhance the performance of the GAA-ADS technique, we test all the possible K values calculated using equation (5.21) on the two datasets. We observe that, for each two consecutive intervals (i.e., even and odd), when K equals even numbers from 2 to 10, the DR ranges are moderately higher than those of the odd ones, as depicted in Figure 5.15 in which the DRs and 20 K values are compared. Because these even ranges contain different dissimilarities of legitimate areas in small intervals, they significantly decrease the overlapping between normal and malicious areas. However as, by using K values greater than 10, the DRs do not improve, we should use only those from 2 to 10 to reduce the processing time and provide the best DRs and lowest FPRs.

D. Advantages and limitations of GAA-ADS technique

The GAA-ADS technique has some advantages and limitations. On the one hand, it depends on recognising the precise areas of normal instances computed using the TAE and considering any deviation from them as an anomalous instance. The statistical approach for building this technique confirms its capability to effectively recognise known and zero-day attacks in large-scale networks because its normal profile includes only estimated parameters from the network data which can be automatically modified by adapting a particular K value. Also, as it does not require any prior information about malicious instances, it can be effectively deployed in real network environments without any effort expended in the training phase.

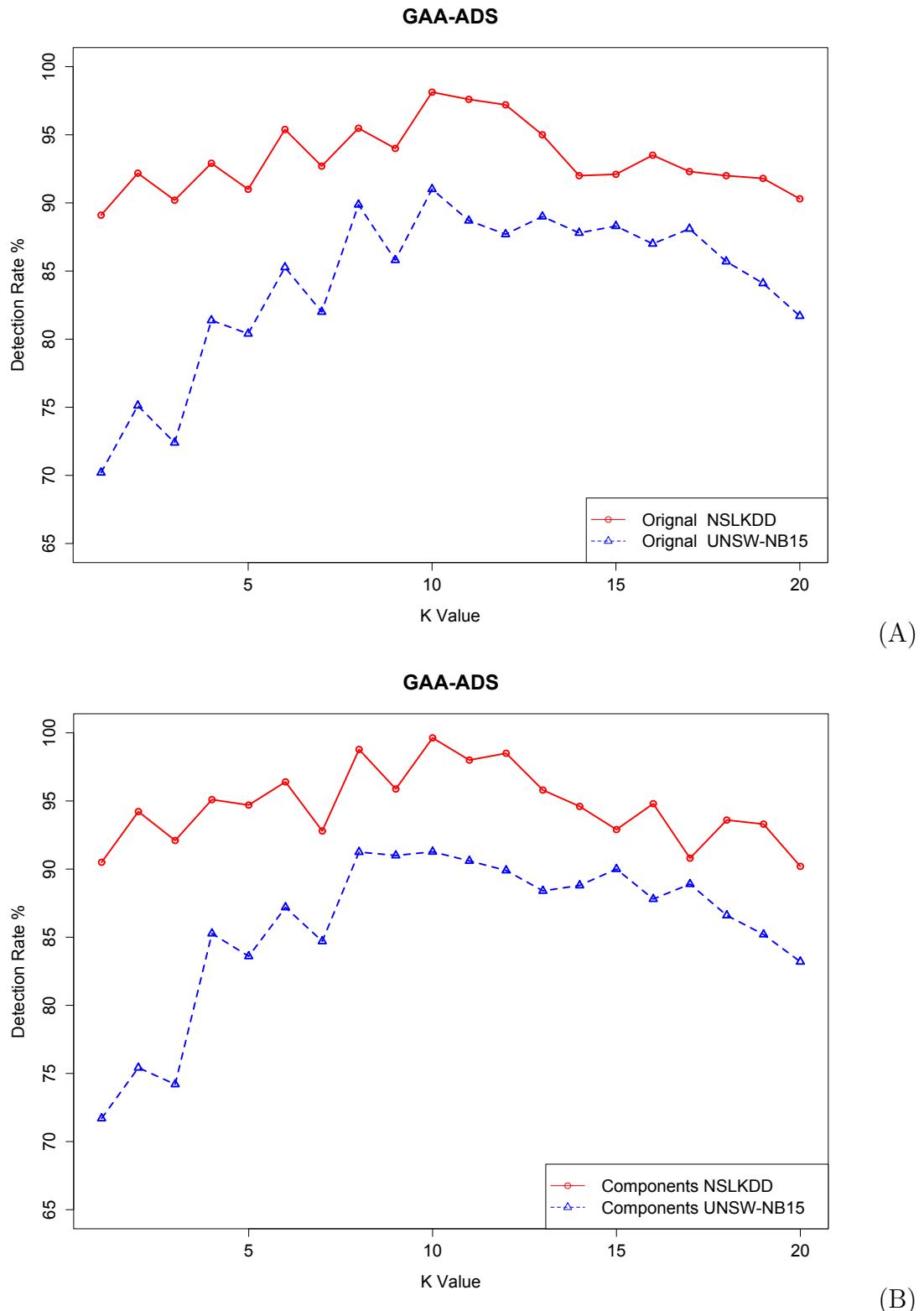


Figure 5.15: Comparison of DRs (%) obtained by GAA-ADS technique with 20 K values

Conversely, selecting the appropriate K value requires a careful analysis of the relevant network data. In current networks, some types of malicious activities, such as stealth and spy attacks, try to mimic legitimate behaviours [287]. Firstly, we design the GAA-ADS technique to identify small variations between normal and suspicious areas but as, in current sophisticated attacks, these areas sometimes overlap, it is necessary to select a K value that distinguishes between them. Then, we develop the new DMM-ADS technique for substantially addressing this challenge. It can identify attacks without defining their types as it does not require any attack information in the training phase but, if it is necessary to identify these types, they should be labelled by estimating their areas. Also, as it can handle only numeric features, a feature conversion step is implemented in the pre-processing module. Finally, because the performance of this technique will slightly decrease if the variances between the selected features are not high, we use the PCA to adopt the most highly varied features by testing the features and their principal components.

5.6.3. Performance Evaluation of DMM-ADS Technique

This subsection explains the performance evaluation of the efficiency and efficacy of the novel DMM-ADS technique on the original features and their principal components chosen by the PCA using different evaluation criteria, in particular, the accuracy, DR and FPR.

A. Performance of DMM-based ADS technique on original features

The performance evaluation of the DMM-based ADS technique is conducted on the original features adopted from the two datasets by the PCA mechanism, with the overall DR, accuracy and FPR presented in Table 5.7. Figure 5.16 shows the ROC curves which demonstrate the relationships between the DRs and FPRs

Table 5.7: Performance evaluation of DMM-ADS technique on original features adopted from both datasets

w value	NSL-KDD			UNSW-NB15		
	DR	Accuracy	FPR	DR	Accuracy	FPR
1.5	93.0%	93.1%	3.0%	86.1%	88.2%	8.2%
2	94.2%	93.8%	0.8%	89.3%	89.8%	7.3%
2.5	97.9%	97.8%	0.6%	93.6%	94.1%	5.6%
3	98.9%	99.2%	0.2%	95.5%	95.8%	4.7%

using the w value. It can be seen that the stable increase in this value between 1.5 and 3 improves the overall DR and accuracy while decreasing the overall FPR.

In more detail, when the w value gradually increases from 1.5 to 3, the overall DR and accuracy improve from 93.0% to 98.9% and 93.1% to 99.2%, respectively, and from 86.1% to 95.8% and 88.2% to 95.8%, respectively, while the overall FPR decreases from 3.0% to 0.2% and 8.2% to 4.7%, in the NSL-KDD and UNSW-NB15 datasets, respectively.

B. Performance of DMM-based ADS technique on principal components

In order to provide an overall evaluation of the performances of the DMM-ADS technique on the 15 principal components in terms of the overall FPR, accuracy and DR, the results are presented in Table 5.8 while Figure 5.17 depicts the ROC curves for the DR and FPR with different w values. Similarly, as the w value gradually increases from 1.5 to 3, the performance of this technique on the original features increases the overall DR and accuracy and reduces the overall FPR.

Nonetheless, the overall performance on the principal components is better than that on the original features by 1.5-2.5% as the principal components increase on the basis of the highest variations which reveal the highest variances in

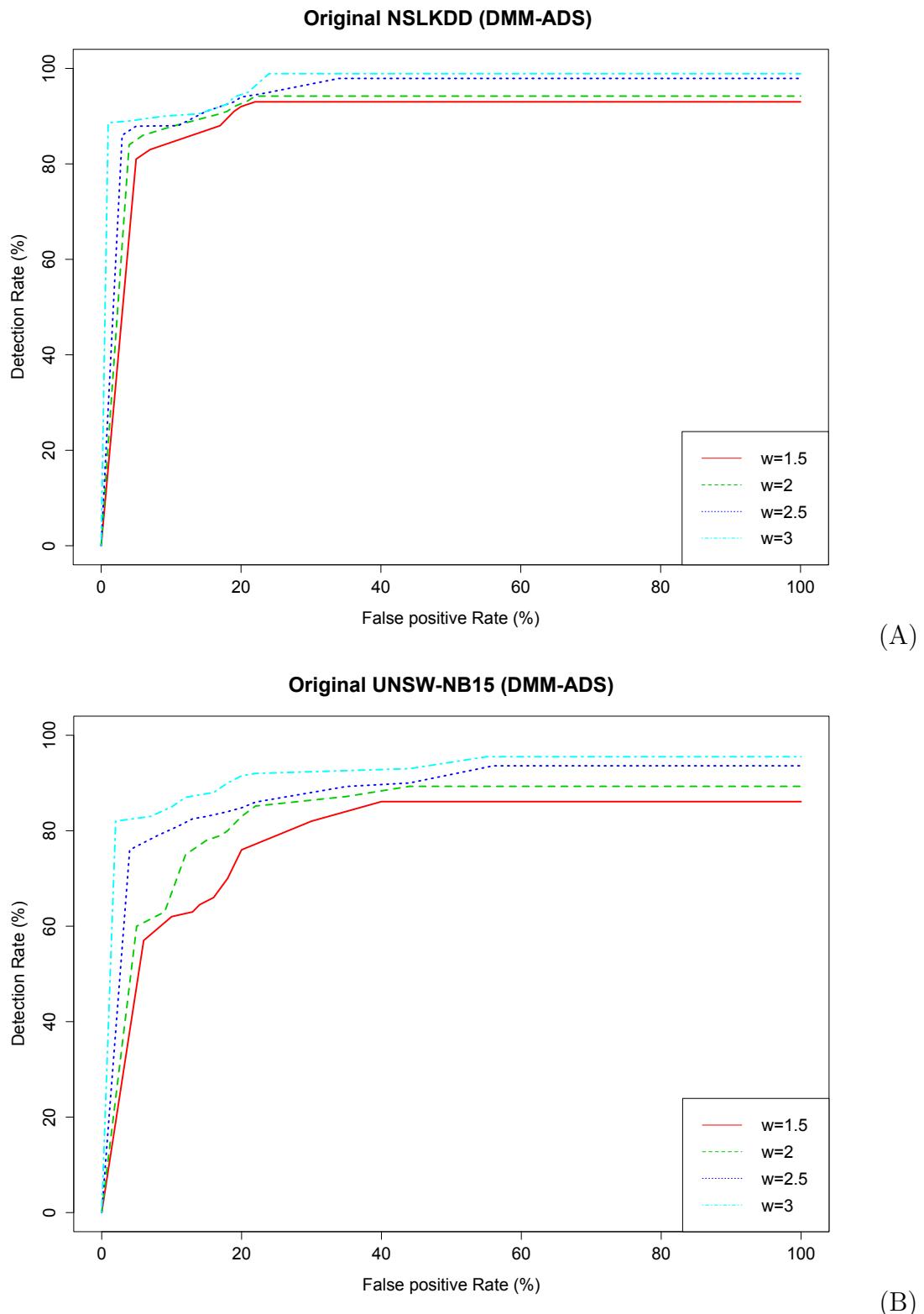


Figure 5.16: ROC curves obtained from DMM-ADS technique for both datasets with different w values

Table 5.8: Evaluation of overall performances of DMM-ADS technique on principal components

w value	NSL-KDD			UNSW-NB15		
	DR	Accuracy	FPR	DR	Accuracy	FPR
1.5	94.6%	95.4%	0.9%	90.2%	91.6%	7.4%
2	96.4%	96.7%	0.6%	93.4%	93.9%	6.2%
2.5	98.2%	98.8%	0.3%	94.8%	95.3%	5.1%
3	99.8%	99.9%	0.1%	96.2%	97.5%	3.4%

the estimated densities of legitimate and malicious instances. While the w value regularly increases from 1.5 to 3, the overall DR and accuracy increase from 94.6% to 99.8% and 95.4% to 99.9%, respectively, and from 90.2% to 96.2% and 91.6% to 97.5%, respectively, while the overall FPR decreases from 0.9% to 0.1% and from 7.4% to 3.4% for the NSL-KDD and UNSW-NB15 datasets, respectively.

C. Performance comparisons of DMM-ADS technique

The performances of the DMM-ADS technique are illustrated using the above five evaluations on the two datasets. Figure 5.18 (A) shows that those on the principal components progressively enhance the DRs from 94.6% to 99.8% which are better than those on the original features which steadily increase the DRs from 93.0% to 98.8% while the FPRs of the principal components vary between 0.2% and 0.9%, nearly 65% of those of the original features (i.e., [0.3% - 3.0 %]). Figure 5.18 (B) displays the ROC curves for the original features and principal components in the UNSW-NB15 dataset which demonstrate that there is a clear difference between them as changes in their DRs and FPRs are approximately 0.6% and 0.9%, respectively.

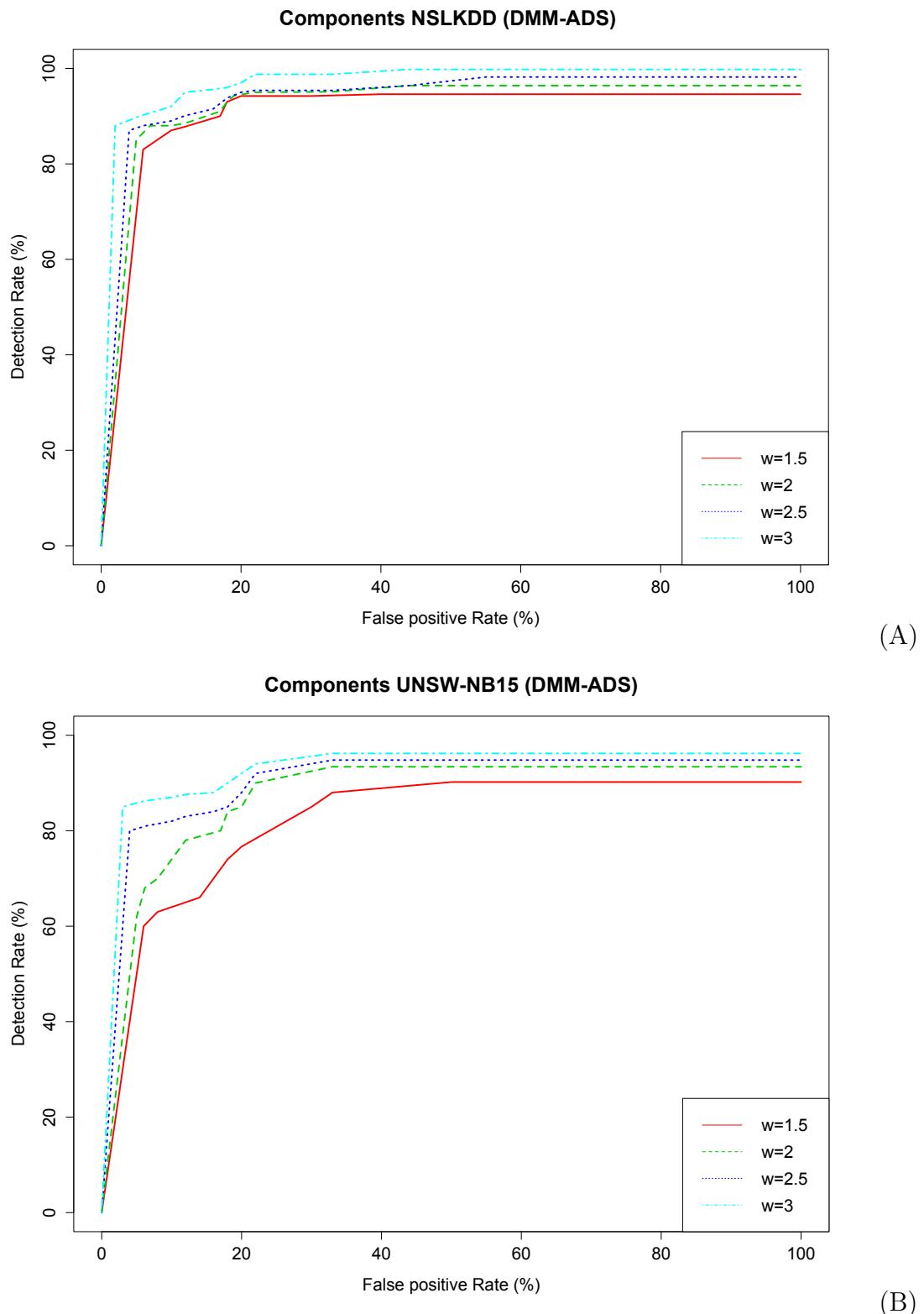


Figure 5.17: ROC curves obtained from DMM-ADS technique for components in both datasets with different w values

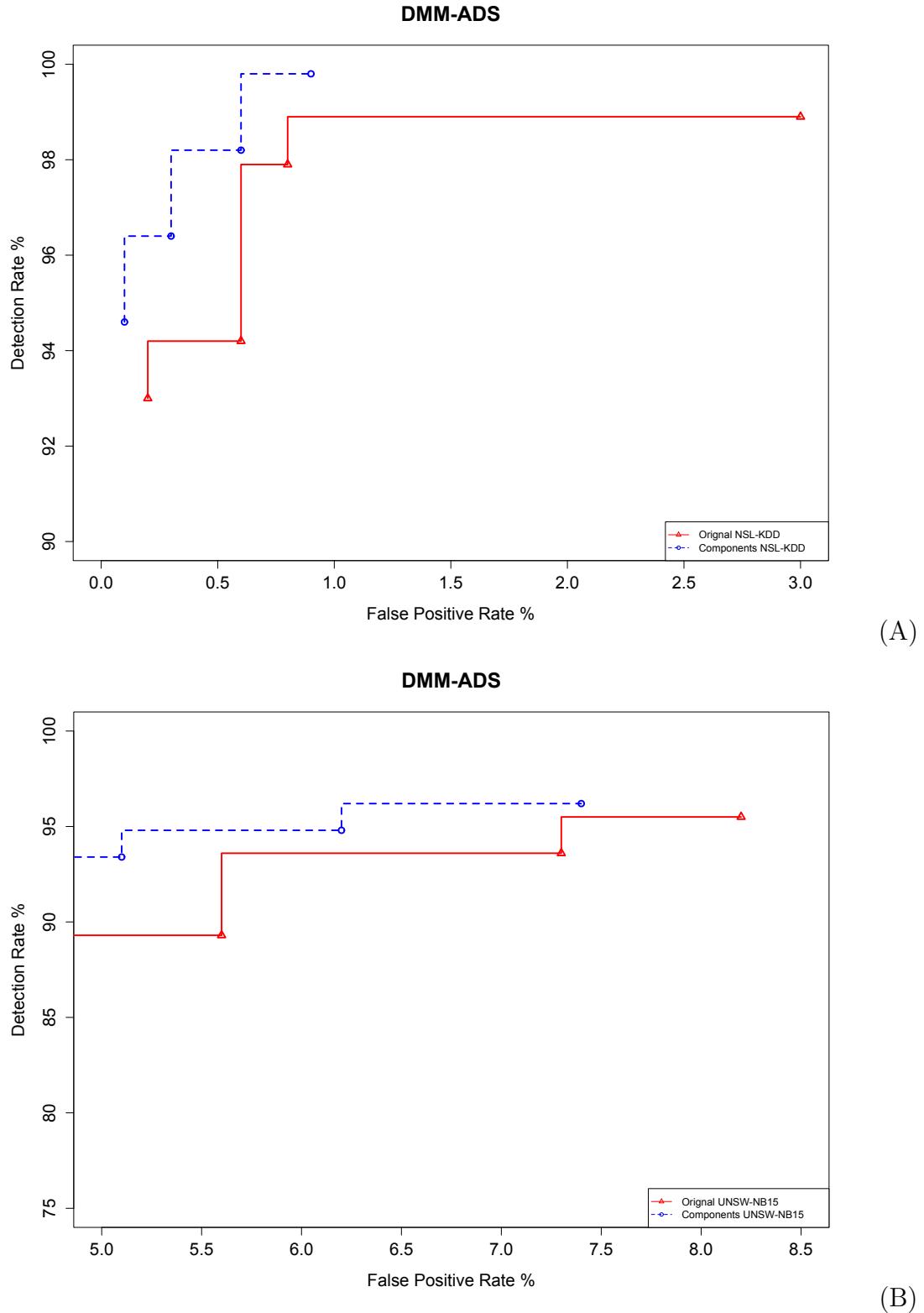


Figure 5.18: ROC curves using both datasets of DMM-ADS technique

Table 5.9: Comparison of DRs (%) obtained by DMM-ADS technique for NSL-KDD dataset

Instance type	w values			
	1.5	2	2.5	3
Normal	96.7%	97.2%	97.3%	99.80%
DoS	97.0%	98.0%	98.8%	99.7%
Probe	92.6%	93.7%	95.3%	97.8%
R2L	95.1%	93.1%	95.1%	95.8%
U2R	92.6%	90.8%	93.2%	94.0%

Tables 5.9 and 5.10 present comparisons of the DRs of the record types in the NSL-KDD and UNSW-N15 datasets, respectively, which increasingly improve with gradual increases in the w values from 1.5 to 3. It is obvious in Table 5.9 that the DMM-ADS technique can identify the majority of record types in the NSL-KDD dataset with a normal DR fluctuating between 96.7% and 99.8%, and the lowest FNR when the w value varies between 1.5 and 3. Likewise, the DRs of the malicious types steadily improve from an average of 94.3% to one of 98.4%.

Table 5.10 demonstrates that the DMM-ADS technique recognises observations types in the UNSW-NB15 dataset, with normal DRs varying between 83.4% and 94.0% while the w value gradually increases from 1.5 to 3. Similarly, the DRs of the attack types gradually increase from an average of 78.9% to one of 94.5%.

The Shellcode, Fuzzers, Reconnaissance and Backdoor attacks do not attain the highest DRs with gradual increases in the w values between 1.5 and 3 while the DRs of the other types of attacks, Generic, DoS, Exploits and Worms, are better because of the small similarities between their malicious and normal observations. It can be observed that the variances in the features selected for these instances from the UNSW-NB15 dataset are sometimes close to their probability densities,

Table 5.10: Comparison of DRs (%) obtained by DMM-ADS technique for UNSW-NB15 dataset

Instance type	w values			
	1.5	2	2.5	3
Normal	83.4%	83.0%	89.70%	94.0%
DoS	89.1%	89.0%	90.2%	99.5%
Backdoor	63.1%	72.2%	74.2%	71.1%
Exploits	42.3%	78.2%	82.1%	81.0%
Analysis	73.8%	76.3%	80.0%	84.1%
Generic	78.1%	89.4%	88.5%	87.4%
Fuzzers	43.1%	49.1%	50.8%	52.8%
Shellcode	42.2%	51.6%	52.0%	52.2%
Reconnaissance	56.1%	54.10%	57.5%	67.2%
Worms	37.0%	45.2%	47.20%	50.3%

especially with the recent sophisticated attacks that try to mimic normal observations. Therefore, the probability densities in some instances overlap each other while before reaching the threshold of the proposed decision-making method.

The major reason for the DMM-ADS technique performing better than the others, including the GAA-ADS technique, is that the DMM can accurately fit the bounds of each feature. This is because its modelling includes a set of probability distributions, including prior, likelihood and posterior, of the network data in order to precisely calculate the probability density of each feature vector. Also, the lower-upper IQR approach can effectively specify the baseline boundaries between normal and anomalous observations. However, despite the DMM-ADS technique achieving the highest DRs and lowest FPRs for the NSL-KDD dataset, its performance for the UNSW-NB15 dataset is somewhat worse due to the small variations between normal and abnormal observations which indicate the complex patterns of contemporary attacks that almost mimic normal ones. Overall, the DMM-ADS technique slightly outperforms the GAA-ADS technique because it can model network data using some probability distributions that can accurately estimate the small differences between legitimate and malicious observations while the latter models these data in a specific range that slightly increases the chances of these instances overlapping.

D. Advantages and limitations of DMM-based ADS technique

The DMM-ADS technique has many advantages. Firstly, it can be implemented on large-scale networks to recognise suspicious activities in real time as the preparation of its training and testing phases relies on only estimating some parameters for the construction of the legitimate profile. Because the DE method applies the lower-upper IQR functions as a baseline, it can define the class label of each observation without depending on other observations. Also, it is very easy to update the legitimate profile parameters with respect to selecting the best baseline that can improve the performance of this technique in terms of achieving a low processing time, high DR and accuracy and low FPR.

To ensure the best performance of this technique, a huge amount of pure legitimate instances is required to produce the highest DRs and lowest FARs. In future, it will be further developed to detect malicious types, such as DoS, Exploits and Backdoors, rather than processing only binary classifications (i.e., normal and abnormal). Also, to determine obvious differences between normal and abnormal instances, the PCA technique will be used to decrease the number of network features with the highest variations between their observations to considerably enhance the performance of the proposed DMM-ADS technique.

5.6.4. Comparative Study and Discussion of Both New DE Techniques

We compare the performances of the two proposed DE techniques (i.e., GAA-ADS and DMM-ADS) with those of six state-of-the-art NIDS approaches, namely, the Multivariate Correlation Analysis (MCA) [36], Computer Vision Technique (CVT) [205], Triangle Area Nearest Neighbours (TANN) [288], Artifical Immune System (AIS) [289], Euclidean Distance Map (EDM) [290] and Filter-based Support Vector Machine (FSVM) [101]. As demonstrated in Table 5.11, the experimental results

Table 5.11: Performance comparisons of six ADS techniques with new DEs using NSL-KDD dataset

Technique	DR	FPR
EDM [36]	94.20%	7.20%
MCA [290]	96.20%	4.90%
TANN [288]	91.10%	9.40%
CVT [205]	95.10%	5.00%
AIS [289]	90.10%	9.80%
FSVM [101]	92.20%	8.70%
GAA-ADS (original features)	98.10%	0.40%
GAA-ADS (components)	99.60%	0.20%
DMM-ADS (original features)	98.90%	0.20%
DMM-ADS (components)	99.80%	0.10%

clearly show the superiority of these new DE techniques in terms of their DRs and FPRs using the NSL-KDD dataset.

The first four state-of-the-art techniques were developed to identify only DoS attacks, for which they attain better DRs, but not for U2R, U2L and Probe malicious observations. Because they rely on computing the distances and correlations between legitimate and suspicious observations as different attacks, specifically stealth and spy intrusion activities [287], and dramatically mimic legitimate instances, they overlap the legitimate profile and reduce detection accuracy.

The FSVM and AIS techniques were developed to learn from legitimate and malicious instances in the training phase based on the principle of rule-based learning. They usually demand a massive number of instances to successfully learn different patterns which are difficult to find in real network environments. Assessments of them reveal that their DRs are better for DoS and Probe attacks, of which there are sufficient observations, but worse for rare intrusive activities, such as U2R and U2L attacks.

The GAA-ADS technique can achieve better performances than the six state-of-the-art techniques for recognising attack types with different K values, as shown in Tables 5.3, 5.4 and 5.5, as it correctly estimates the area of each observation with

the K intervals including small variances that can properly distinguish between normal and abnormal instances. This relies on computing the BMM of the features and the distances between instances to concurrently reveal the potential differences between suspicious and normal network traffic from data observations and features.

Ultimately, the DMM-ADS technique can achieve better performances than the other approaches for identifying attack types with different w values, as shown in Tables 5.7, 5.8 and 5.9, as the DMM can perfectly fit the boundaries of each network feature because its modelling involves some probability distributions for accurately estimating the probability densities of each feature vector. Furthermore, the lower-upper IQR baseline can exactly specify the boundaries between legitimate and suspicious instances.

5.6.5. Clarifications of Complexity and Time Cost of Each New DE Technique

The computational complexity and time cost of data processing the new DE techniques are analysed in order to demonstrate their effective and reliable performances. As discussed in Section 5.3, the GAA-ADS technique consists of four basic steps: 1) estimating the complexity of the PCA based on the Eigen decomposition of a covariance matrix which is $O(ND \times \min(N, D))$ [283]; 2) estimating the BMM parameters ($\text{Beta}(\pi, v, \omega)$); 3) estimating the distance between the mean of normal observations and each observation ($distance_n$); and 4) estimating the TAE for each record ($area(V)$). The four big O notations are combined in these steps to compute the total complexity of the GAA-ADS technique that processes N network observations, each with D features.

Firstly, in the training phase, the $\text{Beta}(\pi, v, \omega)$ parameters consume $O(ND^3)$, with the estimation of D features within their observations. The ($distance_n$) takes $O(ND)$ as all the observations while the features are handled only once, with

$\text{area}(V)$ producing $O(N)$ because of estimating all the network features. Secondly, the testing phase and decision-making method consume $O(ND)$ because they use the principle of ‘record by record’ detection. Therefore, for all observations, the overall complexity of the GAA-ADS technique is $O((ND \times \min(N, D)) + ND^3 + ND + 1)$. Since the ND^3 term becomes larger than the others, the final overall computational complexity of this technique is $O(ND^3)$. However, as in [36], because the D features are identically and independently distributed (*i.i.d*) in each feature vector and concurrently executed, the overall computational complexity is $O(1)$.

The DMM-ADS technique explained in Section 5.4 has three main steps: 1) computing the complexity of the PCA based on the Eigen decomposition of a covariance matrix which is $O(ND \times \min(N, D))$, as in the GAA-ADS technique; 2) computing the DMM parameters ($\text{Dir}(\pi, \alpha, Z)$); 3) by calculating the lower and upper IQR for the normal profile; and 3) integrating the three big O notations in these steps to calculate the total complexity of the DMM-ADS technique that handles N network instances, each with D features. In the training phase, the ($\text{Dir}(\pi, \alpha, Z)$) parameters take $O(ND^3)$ with the D features within their samples approximated while the lower and upper IQR generates $O(N)$. Similar to the GAA-ADS technique, the testing phase and decision-making method take $O(ND)$ due to applying the concept of ‘record by record’ detection. Therefore, for all the records, the overall complexity of the DMM-ADS technique is $O((ND \times \min(N, D)) + ND^3 + N + ND)$. Then, like the GAA-ADS technique, as the $O(ND^3)$ term becomes larger than the others, its final overall computational complexity is $O(ND^3)$, and with the D features (*i.i.d*) in each record simultaneously performed, the overall computational complexity is $O(1)$.

Comparing the four state-of-the-art techniques, the MCA technique [36] produces $O(ND^2)$ and $O(ND^4)$ in its training and testing phases, respectively, and, like the proposed techniques, its features are *i.i.d* and its overall computational complexity $O(1)$ while that of the EDM technique [290][is similar. These two

mechanisms have overall computational complexities similar to those of the proposed techniques which reduce to $O(D)$ due to computing the correlations between all possible combinations of the network features. The TANN technique [288] is very complex because the computational complexities in its training and testing phases are $O(NDL^2)$ and $O(N^2DL^2)$, respectively, with L the number of clusters used to establish the triangular areas. Finally, the overall complexities of the CVT are $O((2ND^2)$ and $O(ND^4))$ in its training and testing sets, respectively [205] , larger than those of the proposed techniques.

Figure 5.19 compares the complexities of the four ADS techniques and the new GAA- and DMM-ADS techniques using the mathematical equation of the big O notation ($f(x) = O(g(x))$), where $g(x)$ denotes the estimated overall computational complexity and x real numbers. Since the number of observations increases, the GAA- and DMM-ADS techniques run faster than the others. The complexity of the two techniques is lower than the other techniques, as they execute faster than the others with the same x value. In more detail, the lines of the two techniques are close to the y-axis (i.e., complexity), whilst the lines of the other techniques are far than our techniques when compensating with the same x value in the big O equation.

Also, an examination of the times taken indicates that the two new DE techniques are faster in terms of processing speed than the others. The GAA- and DMM-ADS techniques can process almost 23,096 and 24,190 observations per second, respectively, and the MCA, CVT and EDM approximately 23,692, 19,267 and 12,044 per second, respectively. Overall, the new techniques can run, on average, 1.04%, 2.21% and 10.34% faster than the MCA, CVT and EDM techniques, respectively.

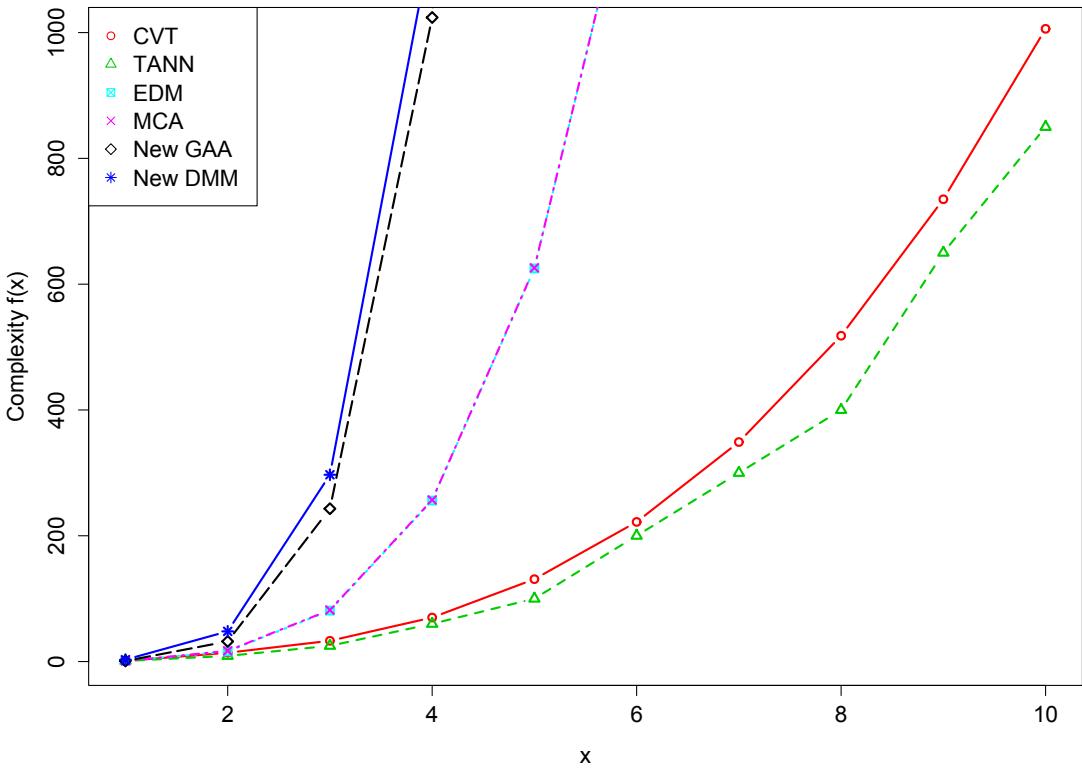


Figure 5.19: Comparison of complexities of four existing and two new DE techniques

5.7. Chapter Summary

This chapter discusses two novel ADS techniques, namely the GAA- and DMM-ADS, developed based on the potential statistical analysis of network data which are used as the DE modules in two proposed frameworks for identifying known and unknown attacks. Each framework consists of three main modules, namely, data sniffing and storing, data pre-processing and DE. The first two are similar in the two frameworks, with their roles to extract and log network data, and analyse and filter these data, respectively. However, the DE modules are considerably different in terms of identifying different suspicious behaviours.

Because of the large volumes, high velocities and wide varieties of current network data, in order to perfectly identify normal and suspicious activities, the task of monitoring and determining these data in depth has become essential. Some statistical methods, in particular, the Kolmogorov-Smirnov test and Q-Q plots, are used to determine the normality of network data, and density and correntropy plots to define their linearity. Identifying these properties of network data can assist in specifying which statistical learning techniques can accurately fit them to significantly enhance the performance of an ADS for efficiently recognising malicious observations in a low processing time. As, according to the statistical literature, these measures reveal that network data precisely follow a Gaussian distribution and are not linear, and the best models for fitting data with these properties are mixture models, in particular, the BMM and DMM, we design DE approaches based on them.

In order to build an intelligent ADS, the novel GAA-ADS technique, which is based on a BMM, the computed parameters and density probability of which are used to estimate the TAE for each network observation, and the distances between observations are used. The TAE technique can provide more correct estimations than the BMM and determine the potential properties of network feature vectors that help to distinguish between known and zero-day intrusive attacks. It is used as a DE module in one of the proposed frameworks for establishing a lightweight, scalable and adaptable ADS. Its performance evaluation is conducted using the NSL-KDD and UNSW-NB15 datasets in the data capture and storage module.

The impact of choosing data with the highest variations using the PCA technique for both the original features selected and their principal components is evident in the data pre-processing module which demonstrates the effective role of the PCA technique. The results reveal that the GAA-ADS technique performs better on the principal components than original features. Moreover, the findings from a comparative study of it and six other ADS techniques in terms of detection accuracy, computational complexity and time cost indicate that its overall DR is

better than and its computational complexity equal to or better than the others. However, as there is some overlap between the areas of normal and anomalous instances, the DMM-ADS technique is proposed.

The DMM-ADS technique, which is developed based on the methodology of anomaly detection, computes the density of Dirichlet distributions for the legitimate profile in the training set and creates the probability densities in the testing set using the parameters calculated in the training phase. The DMM can fit and define the boundaries of network data better than other mixture models as it consists of a set of combined probability distributions. Also, it is more suitable for fitting streaming data, such as those generated from networks and, for grouping any data, can achieve better accuracy than other mixture models. The PCA technique is used to reduce these dimensions, with its impact estimated using the original features selected and their principal components. Then, a decision-making method for identifying existing and zero-day anomalies by specifying a threshold of the lower-upper IQR for the legitimate profile due to its effective role in identifying outliers and considering any variations from it as an attack is designed.

The performance evaluation of the DMM-ADS shows that it performs better than some recent ADS techniques, including the GAA-ADS, in terms of the DR, FPR, complexity and processing time. However, the proposed techniques have some limitations; in particular, they require a huge amount of pure normal observations and need a new function for defining each attack type to considerably ensure reliability and efficiency, as detailed in the next chapter.

Chapter 6

Conclusion

6.1. Introduction

This thesis makes substantial contributions to the field of Network Intrusion Detection Systems (NIDSs), in particular, Network Anomaly Detection Systems (NADSs). Existing IDS methodologies were designed based on the principle of misuse/signature that monitors network data in order to compare observed patterns with those on a designated blacklist. However, this principle cannot identify zero-day attacks (i.e., new ones with no signatures logged in a blacklist) or even variants of existing malicious activities. Consequently, the principle of anomaly detection was developed to discover those activities by creating a profile from normal patterns and identifying any variations from it as attacks.

However, although this methodology has not been used in any industrial applications, its implementations in academia have encountered three major problems over the last decade. First and foremost, there is no dataset that includes a broad range of recent network normal and malicious observations for accurately evaluating NIDSs. Secondly, it is difficult to apply the ADS methodology as two questions must be considered: 1) how is a normal profile with the credibility to encompass the majority of network patterns designed and developed ?; and 2) what are the most effective online Decision Engine (DE) methods for detecting anomalous flows given the current high speeds and large sizes of existing network environments ?

The contributions of this research study address the aforementioned challenges to a considerable degree, with some of its limitations offering directions for future

research, as discussed in Sections 6.3 and 6.4, respectively. The main components of the NADS considered are a data source, data pre-processing module and DE.

Firstly, in Chapter 3, the capability of the DARPA 2009 dataset to generate the important features for building an effective NADS is analysed in depth from which it is determined that it has some limitations that make it difficult to test new NIDSs. Consequently, a new network dataset, called UNSW-NB15 , which contains contemporary legitimate and anomalous network observations, is developed. The analyses and assessments of it demonstrate that its observations can successfully test new DE approaches. By applying learning theories, a portion of the dataset is used to train and test classification techniques. The training and testing sets are created and evaluated based on statistical analyses, feature correlations and complexity evaluations to demonstrate this dataset's credibility for evaluating NIDSs. It has been published and widely used by researchers and developers in both academia and industry, such as Oracle.

Secondly, in Chapter 4, methods for selecting relevant network observations and features from network data with no redundancy are developed to help to establish an effective and lightweight NADS. More specifically, an aggregator module designed using the theory of network flow analysis for reducing the computational resources of deploying NIDSs in real network systems is proposed. Then, feature reduction and selection mechanisms are applied in the data processing module to build DE techniques. Moreover, a new set of features is designed from the DNS and HTTP protocols with data sources created from the UNSW-NB15 dataset to build an effective NIDS for detecting attacks that breach important network applications, including some user information.

Finally, for the first time in this field, in Chapter 5, two novel frameworks for constructing the methodology for developing adaptive, lightweight and scalable NADS using two new DE approaches based on statistical models that efficiently differentiate between normal and suspicious observations are designed. These frameworks have three fundamental components, namely, data sniffing and

storing, data pre-processing and DE. Although the first two, which extract and log network data, and analyse and filter them, respectively, are similar in both frameworks, the DE approaches for effectively identifying different anomalous network activities are different.

The rest of this chapter is organised as follows. Section 6.2 elaborates the key contributions of this study, with the limitations of the proposed techniques provided in Section 6.3. Section 6.4 discusses future research directions related to the main aim of this thesis and, finally, concluding remarks are provided in Section 6.5.

6.2. Contributions of Research

The key contributions of this research are as follows.

- **The recent design of an effective network dataset, the UNSW-NB15 dataset,** created using the IXIA PerfectStorm tool to capture a hybrid of authentic modern legitimate and suspicious network observations from network traffic, with suspicious events implemented based on recent CVE activities, including up-to-date malware ones, and the tcpdump tool used to extract nearly 100 GB of raw network packets logged in some pcap files, each of which consists of approximately one GB to make analysis easier.

The Argus and Bro-IDS tools and some new scripts executed to mine 47 features and their class labels reflect the dataset's diversity in terms of high dimensionality while its velocity is, on average, 5-10 Megabytes per second between sources and destinations. This shows that high data rates transmit through the Ethernets, precisely mimicking realistic network systems. This dataset contains 2,540,044 observations logged in four CSV files, with the parts of it used to train and test NIDS techniques consisting of 175,341 and

82,332 records, respectively. It comprises ten different classes, one normal and nine security events and malware (i.e., Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Fuzzers for suspicious activities, Shellcode and Worms). Evaluations of the statistical and machine-learning algorithms demonstrate the credibility of using this dataset to evaluate a NADS because it includes sophisticated attack vectors that demand deep and accurate analysis for their successful detection.

- **The design of a new aggregator module** based on the principle of flow-level analysis for improving the performance of a NADS via extracting only significant observations which have no duplications or missing values. This module consists of four main functions: 1) collecting network packets at the destination points of a network; 2) logging data which stores packets using the technology of MySQL CGE to make it easier to analyse and apply DE approaches; 3) using big data analysis techniques, specifically association rule mining (ARM) and simple random sampling (SRS), to select only relevant network flows; and 4) sending data to DE approaches for detecting abnormal observations. The aggregator module's technique demonstrates its capability to aggregate network flows based on more attributes than the NetFlow, sflow and IPFIX tools which overcomes the main drawbacks of existing tools and can outperform them in terms of processing time and the generation of distinct records. The empirical results show that the ARM and SRS techniques can generate the majority of important flows with low processing times and no redundant network flows.
- **The development and application of feature reduction and selection methods** for selecting relevant features, that is, a new ARM-Central Points (ARM-CP) approach for feature selection, and the Principal Component Analysis (PCA) and Independent Component Analysis (ICA) for reducing and selecting important network features. The experimental results using the NSL-KDD and UNSW-NB15 datasets show the effectiveness

of these techniques for considerably improving the NADS performances, with the PCA relatively superior due to its principal components being capable of determining clear difference among network features. In more detail, the ARM technique deals directly with the values of attributes while the PCA and ICA transform the feature space into another space based on the highest variations between features, thereby significantly improving the DE's performances.

- **The proposal of a set of features for the DNS and HTTP and an ensemble framework** for recognising the anomalous activities they face which contain some information about user and network activities and two fundamental protocols for internet and network applications. Then, a NIDS for identifying malicious observations that attempt to breach a network via these protocols using a suggested ensemble of learning methods, including the Decision Tree, Naïve Bayes and Artificial Neural Network techniques with the model of Adaboost to fairly distribute network data among them, is developed. The experimental results reveal the significance of these proposed features for detecting malicious observations that try to expose networks via any DNS/HTTP, with the ensemble method performing better than each technique involved.
- **The development of two new DE techniques based on statistical mixture models for identifying existing and zero-day attacks**, with the first called a Geometric Area Analysis (GAA-ADS) based on the anomaly methodology which creates a normal profile and considers any variation from it an attack. The process involved in this technique is computing the Trapezoidal Area Estimation (TAE) for each instance from the calculated parameters of the Beta Mixture Model (BMM) and the distances of instances. Then, a new decision-making method is built based on dividing the normal areas into intervals and considering any area in the testing phase outside them

an anomaly. The performance of this technique is evaluated using the NSL-KDD and UNSW-NB15 datasets, with the empirical results showing that it achieves a higher DR and lower FPR in less processing time using the original and principal components of the PCA technique for reducing network features than state-of-the-art methods. However as, sometimes, normal areas fall into attack ones, a new Dirichlet Mixture Model-based ADS technique (DMM-ADS) is developed.

This technique, which is also based on the methodology of anomaly detection, computes the densities of Dirichlet distributions to create a normal profile in the training set and then generate patterns in the testing set using parameters estimated from the training set. To detect existing and zero-day attacks, a decision-making method is designed which specifies a threshold of the lower-upper Interquartile Range (IQR) for the normal profile and considers any deviation from it an attack. This technique is compared with the GAA-ADS technique using the same configuration environments and the results show that it performs better than its peer techniques, including the GAA-ADS technique, in terms of its DR, FPR and processing time.

- **The design of two scalable NADS frameworks for the above DE techniques** for building a lightweight, adaptive and scalable ADS which can effectively handle large-scale networks. It consists of the three modules of data sniffing and storing, data pre-processing and DE techniques. The first comprises a feature set generated from network traffic, the second analyses and filters network data while the third effectively recognise existing and zero-day malicious observations.

6.3. Limitations

The major limitations of the main components of the NADS are explained in the following.

Firstly, the UNSW-NB15 dataset requires the addition of new attack vectors, in particular DDoS attacks which are similar to DoS ones but originate from multiple sources to hack and stop computer resources. More specifically, the strategy of a DDoS hacker is to flood its victim with upcoming traffic using different sources which makes it extremely difficult to either prevent its actions or distinguish between legitimate and suspicious events while they spread through several points over a network. This type of attack requires further investigation to generate the important features that help DE techniques to differentiate between legitimate and suspicious vectors.

Secondly, the methods for choosing the relevant flows and features from network traffic, as explained in Chapter 4, are developed separately, with their roles to extract and create important observations and features from network data which have no duplications or missing values. Those selected should have properties that can potentially discriminate between normal and abnormal instances using a DE technique. While these proposed mechanisms first select significant observations and then choose relevant features, determining one method for selecting both will be very useful for improving the overall performances of IDSs.

Finally, the proposed statistical DE mechanisms are designed based on the anomaly methodology for creating a legitimate profile in the training set and considering any variation from it in the testing phase an attack. However, this methodology has three main limitations. Firstly, it requires a wide variety of normal observations in the training phase to ensure that the boundaries of the mixture models contain a majority of normal activities. Although these DE methods perform well on existing datasets, they require testing on real network environments, as discussed in the following section.

Moreover, as these techniques can handle only numeric data, to be used for network forensics that require the processing of actual data, they need a new module to link the processed and actual data to provide a genuine IDS that has a forensic capability as well as its main role of identifying abnormal events. Last

but not least, these techniques also require new functions that can define attack types, such as DoS and DDoS, to determine a methodology for detecting these attacks. They will also assist the deployment of an efficient and reliable IDS that can serve in the network forensic field to effectively generate rules and policies against attackers.

6.4. Future directions

While this research provides some substantial advances in the field of IDSs, following are suggested directions, in the form of issues to be resolved and open questions, for future research on overcoming the aforementioned limitations.

6.4.1. Issues to be resolved

- A decent dataset that contains recent host and attack vectors is necessary. It requires configuring a realistic network environment, with host and network activities captured concurrently to evaluate both host- and network-based IDSs. It will be extremely effective for building a hybrid IDS that can detect intrusive behaviours from host and network environments.
- Relevant feature methods that can simultaneously select relevant features and observations from network data are needed. They should be capable of addressing this challenge by choosing important network flows from the large numbers of packets usually received in existing network systems, and then selecting significant features that has the potential characteristics of intrusive activities. The main aim would be to develop one mechanism for selecting both important features and vectors to enhance the overall performances of IDSs.
- It would be interesting to integrate the proposed DE techniques with two new components, that is: 1) linking the statistically processed and original

data to extend the role of an IDS to investigate e-crimes and combine this field with a network's forensic field; and 2) designing a new function that can determine each attack type and easily identify the methodologies of attackers that continually try to develop new hacking methods.

- Investigating and applying the proposed NADS frameworks in large-scale applications, in particular, cloud computing and industrial control systems, such as Supervisory Control and Data Acquisition (SCADA) system, will be an important focus of research because these systems have the same network architectures. With the new era of the Internet of Things (IoT), that is, networked interconnections of everyday objects often associated with their ubiquitous use, these systems need to be protected against malicious activities.

Since cloud computing and SCADA systems are currently fully dependent on the internet, they require a lightweight, adaptive and scalable NADS for detecting the anomalous activities they frequently face. The deployment of a NADS framework in these large-scale environments is often difficult as they have many nodes that are either centralised or distributed. Moreover, the high speeds and large amounts of data transferring between these nodes often affect the performance of a NADS.

6.4.2. Open questions

Based on the above issues, the following open questions are proposed for future research.

- What effective testbed configurations can be used to create a decent dataset for evaluating both host- and network-based IDSs?
- What is an efficient methodology for aggregating and selecting the relevant features of network traffic?

- How can the proposed DE techniques be used for network forensics?
- What functions can be effectively integrated with the proposed frameworks to define malicious types of attacks?
- How can statistical approaches clearly distinguish between normal and suspicious instances?
- How can the proposed frameworks be used in large-scale environments such as cloud computing and SCADA systems?

6.5. Final remarks

Given their large sizes and high speeds, current networks face serious threats from several exploitation techniques. As no single security system is capable of successfully combating all possible intrusive activities, the principle of defence-in-depth should be applied to deploy many layers of security to identify and prevent all types of malicious vectors. Anomaly-based methodologies should be implemented in different control systems, such as firewall and antivirus tools, in order to effectively define existing and zero-day attacks, and reduce the number of vulnerabilities facing computer and network systems.

This thesis makes significant contributions to the field of IDSs by applying the anomaly-based detection methodology for identifying abnormal instances in large-scale network data through: 1) creating the UNSW-NB15 dataset that has a broad range of contemporary normal and abnormal observations for effectively evaluating new DE methods; 2) proposing a new method for selecting important observations and features with the characteristics of anomalous activities that can be detected by the proposed statistical DE techniques; 3) developing two frameworks and DE methods based on mixture models for creating a normal profile that has all possible patterns of normal network data, with any deviations from it considered an attack using new outlier methods.

In summary, the proposed frameworks can effectively and efficiently recognise both known and unknown attack vectors and detect them in real network environments as they can automatically adapt by estimating their statistical parameters from network data. It is vital that the security research community and industrial applications consider the principle of anomaly-based detection for different security mitigation and control systems as it can be easily executed without requiring the generation of signatures of existing attacks, thereby ensuring that zero-day attacks which do not have signatures logged in existing blacklists can be successfully identified.

References

- [1] Sergio Pastrana Portillo. Attacks against intrusion detection networks: evasion, reverse engineering and optimal countermeasures. 2014.
- [2] Digital device statistics. April 2017.
- [3] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1):266–282, 2014.
- [4] Joseph S Nye Jr. Deterrence and dissuasion in cyberspace. *International Security*, 41(3):44–71, 2017.
- [5] Maurizio Martellini, Stanislav Abaimov, Sandro Gaycken, and Clay Wilson. Vulnerabilities and security issues. In *Information Security of Highly Critical Wireless Networks*, pages 11–15. Springer, 2017.
- [6] Public Key Infrastructure and Token Protection Profile. Common criteria for information technology security evaluation. *National Security Agency*, 2002.
- [7] Rossouw Von Solms and Johan Van Niekerk. From information security to cyber security. *computers & security*, 38:97–102, 2013.
- [8] Salvatore Pontarelli, Giuseppe Bianchi, and Simone Teofili. Traffic-aware design of a high-speed fpga network intrusion detection system. *IEEE Transactions on Computers*, 62(11):2322–2334, 2013.
- [9] The acsc threat report. May 2016.
- [10] The macafee threat report. April 2017.

- [11] Gideon Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks*. PhD thesis, University of New South Wales, 2014.
- [12] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.
- [13] Rabiah Ahmad, Zahri Yunos, and Shahrin Sahib. Understanding cyber terrorism: The grounded theory method applied. In *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*, pages 323–328. IEEE, 2012.
- [14] EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569, 2011.
- [15] Cyber-extortion-attacks. September 2016.
- [16] Kriangsak Kittichaisaree. Cyber espionage. In *Public International Law of Cyberspace*, pages 233–262. Springer, 2017.
- [17] Estimating the global cost of cybercrime. September 2016.
- [18] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.
- [19] Shelly Xiaonan Wu and Wolfgang Banzhaf. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, 10(1):1–35, 2010.
- [20] Hisham A Kholidy and Fabrizio Baiardi. Cids: A framework for intrusion detection in cloud systems. In *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, pages 379–385. IEEE, 2012.

- [21] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.
- [22] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013.
- [23] Saad Y Sait, Akshay Bhandari, Shreya Khare, Cyriac James, and Hema A Murthy. Multi-level anomaly detection: Relevance of big data analytics in networks. *Sadhana*, 40(6):1737–1767, 2015.
- [24] Wenke Lee, Salvatore J Stolfo, and Kui W Mok. A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 120–132. IEEE, 1999.
- [25] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.
- [26] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.
- [27] David J Weller-Fahy, Brett J Borghetti, and Angela A Sodemann. A survey of distance and similarity measures used within network intrusion anomaly detection. *IEEE Communications Surveys & Tutorials*, 17(1):70–91, 2015.
- [28] Derek Lin. Anomaly detection system for enterprise network security, August 18 2015. US Patent 9,112,895.
- [29] Ruth Bernstein and Andrey Dulkin. Systems and methods for detection of anomalous network behavior, May 19 2016. US Patent 20,160,142,435.

- [30] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*. CRC press, 2016.
- [31] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maci-Fernandez, and Enrique Vazquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.
- [32] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, and Chittaranjan Hota. Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences*, 278:488–497, 2014.
- [33] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):1, 2015.
- [34] Jungwon Kim, Peter J Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. Immune system approaches to intrusion detection—a review. *Natural computing*, 6(4):413–466, 2007.
- [35] Koral Ilgun, Richard A Kemmerer, and Phillip A Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering*, 21(3):181–199, 1995.
- [36] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarsi Nanda, and Ren Ping Liu. A system for denial-of-service attack detection based on multivariate correlation analysis. *IEEE transactions on parallel and distributed systems*, 25(2):447–456, 2014.
- [37] Wentao Fan, Nizar Bouguila, and Hassen Sallay. Anomaly intrusion detection using incremental learning of an infinite mixture model with feature selection. In *International Conference on Rough Sets and Knowledge Technology*, pages 364–373. Springer, 2013.
- [38] The darpa98 and kddcup99 datasets. April 2017.

- [39] The nslekdd dataset. April 2017.
- [40] The caida datasets. April 2017.
- [41] The darpa-2009 dataset. darpa scalable network monitoring (snm) program traffic. packet clearing house. 11/3/2009 to 11/12/2009. October 2015.
- [42] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *Military Communications and Information Systems Conference (MilCIS), 2015*, pages 1–6. IEEE, 2015.
- [43] The unsw-nb15 dataset. April 2017.
- [44] Pavel Pudil and Jana Novovičová. Novel methods for feature subset selection with respect to problem knowledge. In *Feature Extraction, Construction and Selection*, pages 101–116. Springer, 1998.
- [45] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.
- [46] Jingyue Pang, Datong Liu, Yu Peng, and Xiyuan Peng. Anomaly detection based on uncertainty fusion for univariate monitoring series. *Measurement*, 95:280–292, 2017.
- [47] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016.
- [48] Alireza Shameli-Sendi, Mohamed Cheriet, and Abdelwahab Hamou-Lhadj. Taxonomy of intrusion risk assessment and response system. *Computers & Security*, 45:1–16, 2014.
- [49] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Shahid Anwar. Intrusion response systems: Foundations, design,

and challenges. *Journal of Network and Computer Applications*, 62:53–74, 2016.

- [50] Shahid Anwar, Jasni Mohamad Zain, Mohamad Fadli Zolkipli, Zakira Inayat, Suleman Khan, Bokolo Anthony, and Victor Chang. From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. *Algorithms*, 10(2):39, 2017.
- [51] Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan D Payne. Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1):12, 2015.
- [52] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.
- [53] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Christos Tachtatzis, and Robert Atkinson. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv preprint arXiv:1701.02145*, 2017.
- [54] Daesung Moon, Sung Bum Pan, and Ikkyun Kim. Host-based intrusion detection system for secure human-centric computing. *The Journal of Supercomputing*, 72(7):2520–2536, 2016.
- [55] Katherine E Price. *Host-based misuse detection and conventional operating systems'audit data collection*. PhD thesis, Purdue University, 1997.
- [56] Jian Peng, Kim-Kwang Raymond Choo, and Helen Ashman. User profiling in intrusion detection: A review. *Journal of Network and Computer Applications*, 72:14–27, 2016.
- [57] Gideon Creech and Jiankun Hu. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE Transactions on Computers*, 63(4):807–819, 2014.

- [58] Manish Kumar, M Hanumanthappa, and TV Suresh Kumar. Encrypted traffic and ipsec challenges for intrusion detection system. In *Proceedings of International Conference on Advances in Computing*, pages 721–727. Springer, 2013.
- [59] Roni Bar-Yanai, Michael Langberg, David Peleg, and Liam Roditty. Re-altime classification for encrypted traffic. In *International Symposium on Experimental Algorithms*, pages 373–385. Springer, 2010.
- [60] Gisung Kim, Seungmin Lee, and Sehun Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700, 2014.
- [61] Manmeet Kaur Marhas, Anup Bhange, and Piyush Ajankar. Anomaly detection in network traffic: A statistical approach. *International Journal of IT, Engineering and Applied Sciences Research (IJIEASR)*, 1(3):16–20, 2012.
- [62] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.
- [63] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.
- [64] Fairuz Amalina Narudin, Ali Feizollah, Nor Badrul Anuar, and Abdullah Gani. Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, 20(1):343–357, 2016.
- [65] Yun Wang. *Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection: Modern Statistically-Based Intrusion Detection and Protection*. IGI Global, 2008.
- [66] Phillip A Porras and Alfonso Valdes. Live traffic analysis of tcp/ip gateways. In *NDSS*, 1998.

- [67] Glenn A Fink, BL Chappell, TG Turner, and KF O'Donoghue. A metrics-based approach to intrusion detection system evaluation for distributed real-time systems. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 8–pp. IEEE, 2001.
- [68] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. *Machine Learning*, 101(1-3):59–84, 2015.
- [69] ARi Vasudevan, E Harshini, and S Selvakumar. Ssenet-2011: a network intrusion detection system dataset and its comparison with kdd cup 99 dataset. In *Internet (AH-ICI), 2011 Second Asian Himalayas International Conference on*, pages 1–5. IEEE, 2011.
- [70] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.
- [71] Monowar H Bhuyan, Dhruba K Bhattacharyya, and Jugal K Kalita. Towards generating real-life datasets for network intrusion detection. *IJ Network Security*, 17(6):683–701, 2015.
- [72] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6:70, 2001.
- [73] Paul Zikopoulos, Krishnan Parasuraman, Thomas Deutsch, James Giles, David Corrigan, et al. *Harness the power of big data The IBM big data platform*. McGraw Hill Professional, 2012.
- [74] Alvaro A Cardenas, Pratyusa K Manadhata, and Sreeranga P Rajan. Big data analytics for security. *IEEE Security & Privacy*, 11(6):74–76, 2013.
- [75] Yeonhee Lee and Youngseok Lee. Toward scalable internet traffic measurement and analysis with hadoop. *ACM SIGCOMM Computer Communication Review*, 43(1):5–13, 2013.

- [76] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Nado: network anomaly detection using outlier approach. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, pages 531–536. ACM, 2011.
- [77] Chikh Ramdane and Salim Chikhi. A new negative selection algorithm for adaptive network intrusion detection system. *International Journal of Information Security and Privacy (IJISP)*, 8(4):1–25, 2014.
- [78] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lin-coln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.
- [79] Hashem Alaidaros, Massudi Mahmuddin, Ali Al-Mazari, et al. An overview of flow-based and packet-based intrusion detection performance in high speed networks. 2011.
- [80] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE, 2009.
- [81] The defcon dataset. April 2017.
- [82] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaei, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.
- [83] The iscx dataset. April 2016.
- [84] The kyoto dataset. April 2017.
- [85] Nour Moustafa and Jill Slay. Creating novel features to anomaly network detection using darpa-2009 data set. In *Proceedings of the 14th European*

Conference on Cyber Warfare and Security. Academic Conferences Limited, page 204, 2015.

- [86] Tatsuya Baba and Shigeyuki Matsuda. Tracing network attacks to their sources. *IEEE Internet Computing*, 6(2):20–26, 2002.
- [87] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.
- [88] You Chen, Yang Li, Xue-Qi Cheng, and Li Guo. Survey and taxonomy of feature selection algorithms in intrusion detection system. In *International Conference on Information Security and Cryptology*, pages 153–167. Springer, 2006.
- [89] Yanchang Zhao and Sourav S Bhowmick. Association rule mining with r. *A Survey Nanyang Technological University, Singapore*, 2015.
- [90] Petros Xanthopoulos, Panos M Pardalos, and Theodore B Trafalis. Principal component analysis. In *Robust data mining*, pages 21–26. Springer, 2013.
- [91] Francesco Palmieri, Ugo Fiore, and Aniello Castiglione. A distributed approach to network anomaly detection based on independent component analysis. *Concurrency and Computation: Practice and Experience*, 26(5):1113–1129, 2014.
- [92] Wenke Lee, Salvatore J Stolfo, et al. Data mining approaches for intrusion detection. In *Usenix security*, 1998.
- [93] Jianxiong Luo and Susan M Bridges. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, 15(8):687–703, 2000.
- [94] Kamini Nalavade and BB Meshram. Mining association rules to evade network intrusion in network audit data. *International Journal of Advanced Computer Research*, 4(2):560, 2014.

- [95] Zhang Yanyan and Yao Yuan. Study of database intrusion detection based on improved association rule algorithm. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 4, pages 673–676. IEEE, 2010.
- [96] Cynthia Wagner, Jérôme François, Thomas Engel, et al. Machine learning approach for ip-flow record anomaly detection. In *International Conference on Research in Networking*, pages 28–39. Springer, 2011.
- [97] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal - The International Journal on Very Large Data Bases*, 16(4):507–521, 2007.
- [98] Mohammad Esmalifalak, Huy Nguyen, Rong Zheng, and Zhu Han. Stealth false data injection using independent component analysis in smart grid. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 244–248. IEEE, 2011.
- [99] Jinsub Kim, Lang Tong, and Robert J Thomas. Subspace methods for data attack on state estimation: A data driven approach. *IEEE Trans. Signal Processing*, 63(5):1102–1114, 2015.
- [100] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert systems with Applications*, 38(1):306–313, 2011.
- [101] Mohammed A Ambusaidi, Xiangjian He, Priyadarsi Nanda, and Zhiyuan Tan. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE transactions on computers*, 65(10):2986–2998, 2016.

- [102] Inho Kang, Myong K Jeong, and Dongjoon Kong. A differentiated one-class classification method with applications to intrusion detection. *Expert Systems with Applications*, 39(4):3899–3905, 2012.
- [103] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.
- [104] Prabaharan Poornachandran, S Praveen, Aravind Ashok, Manu R Krishnan, and KP Soman. Drive-by-download malware detection in hosts by analyzing system resource utilization using one class support vector machines. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*, pages 129–137. Springer, 2017.
- [105] Igor Balabine and Alexander Velednitsky. Method and system for confident anomaly detection in computer network traffic, February 20 2015. US Patent App. 14/627,963.
- [106] Mohammed Saber, Ilhame El Farissi, Sara Chadli, Mohamed Emharraf, and Mohammed Ghaouth Belkasmi. Performance analysis of an intrusion detection systems based of artificial neural network. In *Europe and MENA Co-operation Advances in Information and Communication Technologies*, pages 511–521. Springer, 2017.
- [107] MR Gauthama Raman, Nivethitha Somu, Kannan Kirthivasan, and VS Shankar Sriram. A hypergraph and arithmetic residue-based probabilistic neural network for classification in intrusion detection systems. *Neural Networks*, 2017.
- [108] Chaivat Jirapummin, Naruemon Wattanapongsakorn, and Prasert Kanthamanon. Hybrid neural networks for intrusion detection system. In *Proc. of ITC-CSCC*, pages 928–931, 2002.

- [109] Timo Horeis. Intrusion detection with neural networks—combination of self-organizing maps and radial basis function networks for human expert integration. *Computational Intelligence Society Student Research Grants*, 2003.
- [110] Guisong Liu, Zhang Yi, and Shangming Yang. A hierarchical intrusion detection model based on the pca neural networks. *Neurocomputing*, 70(7):1561–1568, 2007.
- [111] Shreya Dubey and Jigyasu Dubey. Kbb: A hybrid method for intrusion detection. In *Computer, Communication and Control (IC4), 2015 International Conference on*, pages 1–6. IEEE, 2015.
- [112] Liwei Kuang. Dnids: a dependable network intrusion detection system using the csi-knn algorithm. 2007.
- [113] Mahdi Soltanolkotabi, Emmanuel J Candes, et al. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.
- [114] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. An effective unsupervised network anomaly detection method. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 533–539. ACM, 2012.
- [115] GV Nadiammai and M Hemalatha. An evaluation of clustering technique over intrusion detection system. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 1054–1060. ACM, 2012.
- [116] Ambarish Jadhav, Avinash Jadhav, Pradeep Jadhav, and Prakash Kulkarni. A novel approach for the design of network intrusion detection system (nids). In *Sensor Network Security Technology and Privacy Communication System (SNS & PCS), 2013 International Conference on*, pages 22–27. IEEE, 2013.

- [117] Dahlia Asyiqin Ahmad Zainaddin and Zurina Mohd Hanapi. Hybrid of fuzzy clustering neural network over nsł dataset for intrusion detection system. *Journal of Computer Science*, 9(3):391, 2013.
- [118] Akara Prayote. *Knowledge based anomaly detection*. PhD thesis, The University of New South Wales, 2007.
- [119] Kriti Chadha and Sushma Jain. Hybrid genetic fuzzy rule based inference engine to detect intrusion in networks. In *Intelligent Distributed Computing*, pages 185–198. Springer, 2015.
- [120] Teresa F Lunt and R Jagannathan. A prototype real-time intrusion-detection expert system. In *Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on*, pages 59–66. IEEE, 1988.
- [121] The snort tool. April 2017.
- [122] Hannes Holm. Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 4895–4904. IEEE, 2014.
- [123] Bartosz Jasiul, Marcin Szpyrka, and Joanna Śliwa. Malware behavior modeling with colored petri nets. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 667–679. Springer, 2014.
- [124] Hank S Vaccaro and Gunar E Liepins. Detection of anomalous computer session activity. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pages 280–289. IEEE, 1989.
- [125] Walter Scheirer and Mooi Choo Chuah. Syntax vs. semantics: competing approaches to dynamic network intrusion detection. *International Journal of Security and Networks*, 3(1):24–35, 2008.
- [126] Prasad Naldurg, Koushik Sen, and Prasanna Thati. A temporal logic based framework for intrusion detection. In *International Conference on Formal*

Techniques for Networked and Distributed Systems, pages 359–376. Springer, 2004.

- [127] Shao-Shin Hung and Damon Shing-Min Liu. A user-oriented ontology-based approach for network intrusion detection. *Computer Standards & Interfaces*, 30(1):78–88, 2008.
- [128] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.
- [129] Daniel B Araya, Katarina Grolinger, Hany F ElYamany, Miriam AM Capretz, and Girma Bitsuamlak. An ensemble learning framework for anomaly detection in building energy consumption. *Energy and Buildings*, 2017.
- [130] Leandros A Maglaras, Jianmin Jiang, and Tiago J Cruz. Combining ensemble methods and social network metrics for improving accuracy of ocsvm on intrusion detection in scada systems. *arXiv preprint arXiv:1507.02825*, 2015.
- [131] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers & Security*, 65:135–152, 2017.
- [132] Vrushank Shah, Akshai K Aggarwal, and Nirbhay Chaubey. Performance improvement of intrusion detection with fusion of multiple sensors. *Complex & Intelligent Systems*, pages 1–7, 2016.
- [133] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers & security*, 24(4):295–307, 2005.

- [134] Álvaro Herrero, Martí Navarro, Emilio Corchado, and Vicente Julián. Rt-movicab-ids: Addressing real-time intrusion detection. *Future Generation Computer Systems*, 29(1):250–261, 2013.
- [135] Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano. An ensemble-based evolutionary framework for coping with distributed intrusion detection. *Genetic Programming and Evolvable Machines*, 11(2):131–146, 2010.
- [136] Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Data Mining, 2006. ICDM’06. Sixth International Conference on*, pages 488–498. IEEE, 2006.
- [137] Huu Hoa Nguyen, Nouria Harbi, and Jérôme Darmont. An efficient local region and clustering-based ensemble system for intrusion detection. In *Proceedings of the 15th Symposium on International Database Engineering & Applications*, pages 185–191. ACM, 2011.
- [138] Giorgio Giacinto, Fabio Roli, and Luca Didaci. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern recognition letters*, 24(12):1795–1803, 2003.
- [139] Jason Shifflet. A technique independent fusion model for network intrusion detection. In *Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics*, volume 3, pages 13–19. Citeseer, 2005.
- [140] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A novel svm-knn-psو ensemble method for intrusion detection system. *Applied Soft Computing*, 38:360–372, 2016.
- [141] Nauman Shahid, Ijaz Haider Naqvi, and Saad Bin Qaisar. Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: a survey. *Artificial Intelligence Review*, 43(2):193–228, 2015.

- [142] Xiaoping Shen and Sonali Agrawal. Kernel density estimation for an anomaly based intrusion detection system. In *MLMTA*, pages 161–167. Citeseer, 2006.
- [143] Dit-Yan Yeung and Calvin Chow. Parzen-window network intrusion detectors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 385–388. IEEE, 2002.
- [144] Kyle Caudle, Christer Karlsson, and Larry D Pyeatt. Using density estimation to detect computer intrusions. In *Proceedings of the 2015 ACM International Workshop on International Workshop on Security and Privacy Analytics*, pages 43–48. ACM, 2015.
- [145] Miguel Nicolau, James McDermott, et al. One-class classification for anomaly detection with kernel density estimation and genetic programming. In *European Conference on Genetic Programming*, pages 3–18. Springer, 2016.
- [146] Patricia Mostardinha, Bruno Filipe Faria, André Zúquete, and Fernão Vistulo de Abreu. A negative selection approach to intrusion detection. In *International Conference on Artificial Immune Systems*, pages 178–190. Springer, 2012.
- [147] Zeng Jinquan, Liu Xiaojie, Li Tao, Liu Caiming, Peng Lingxi, and Sun Feixian. A self-adaptive negative selection algorithm used for anomaly detection. *Progress in natural Science*, 19(2):261–266, 2009.
- [148] Asghar Ghasemi, Saleh Zahediasl, et al. Normality tests for statistical analysis: a guide for non-statisticians. *International journal of endocrinology and metabolism*, 10(2):486–489, 2012.
- [149] Chien-Chuan Lin and Ming-Shi Wang. *Particle Filter for Depth Evaluation of Networking Intrusion Detection Using Coloured Petri Nets*. INTECH Open Access Publisher, 2010.

- [150] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522. IEEE, 2009.
- [151] Jing Xu and Christian R Shelton. Intrusion detection using continuous time bayesian networks. *arXiv preprint arXiv:1401.3851*, 2014.
- [152] Alexander Tartakovsky, Igor Nikiforov, and Michele Basseville. *Sequential analysis: Hypothesis testing and changepoint detection*. CRC Press, 2014.
- [153] Aditya Oza, Kevin Ross, Richard M Low, and Mark Stamp. Http attack detection using n-gram analysis. *Computers & Security*, 45:242–254, 2014.
- [154] Srinivas Krishnan, Teryl Taylor, Fabian Monroe, and John McHugh. Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE, 2013.
- [155] Nasser Abouzakhar and Abu Bakar. A chi-square testing-based intrusion detection model. In *Procs 4th International Conference on Cybercrime Forensics Education & Training*, 2010.
- [156] Juan Luis Santos et al. Application of adversarial risk testing to anomaly-based network intrusion detection systems. *Journal of Socioeconomic Engineering*, (2):31–40, 2014.
- [157] Hesham Altwaijry. Bayesian based intrusion detection system. In *IAENG Transactions on Engineering Technologies*, pages 29–44. Springer, 2013.
- [158] Liyuan Xiao, Yetian Chen, and Carl K Chang. Bayesian model averaging of bayesian network classifiers for intrusion detection. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 128–133. IEEE, 2014.

- [159] Xiaoyan Han, Liancheng Xu, Min Ren, and Weiping Gu. A naive bayesian network intrusion detection algorithm based on principal component analysis. In *Information Technology in Medicine and Education (ITME), 2015 7th International Conference on*, pages 325–328. IEEE, 2015.
- [160] Luca Scrucca, Michael Fop, T Brendan Murphy, and Adrian E Raftery. mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R Journal*, 8(1):289, 2016.
- [161] Wentao Fan, Nizar Bouguila, and Djemel Ziou. Unsupervised anomaly intrusion detection via localized bayesian feature selection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1032–1037. IEEE, 2011.
- [162] Wentao Fan, Nizar Bouguila, and Djemel Ziou. Variational learning for finite dirichlet mixture models and applications. *IEEE transactions on neural networks and learning systems*, 23(5):762–774, 2012.
- [163] Nicola Greggio. Learning anomalies in idss by means of multivariate finite mixture models. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 251–258. IEEE, 2013.
- [164] Christian Gruhl, Bernhard Sick, Arno Wacker, Sven Tomforde, and Jörg Hähner. A building block for awareness in technical systems: Online novelty detection and reaction with an application in intrusion detection. In *Awareness Science and Technology (iCAST), 2015 IEEE 7th International Conference on*, pages 194–200. IEEE, 2015.
- [165] Niandong Liao, Shengfeng Tian, and Tinghua Wang. Network forensics based on fuzzy logic and expert system. *Computer Communications*, 32(17):1881–1892, 2009.
- [166] Sergiu Nedevschi, Rolf Schmidt, Thorsten Graf, Radu Danescu, Dan Frentiu, Tiberiu Marita, Florin Oniga, and Ciprian Pocol. 3d lane detection system

- based on stereovision. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 161–166. IEEE, 2004.
- [167] Xue Yuan, Yifei Meng, and Xueye Wei. A method of location the vehicle windshield region for vehicle occupant detection system. In *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, volume 1, pages 712–715. IEEE, 2012.
- [168] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. Dos and ddos in named data networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, July 2013.
- [169] Satomi Honda, Yuki Unno, Koji Maruhashi, Masahiko Takenaka, and Satoru Torii. Topase: Detection of brute force attacks used disciplined ips from ids log. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1361–1364. IEEE, 2015.
- [170] Gao Feng He, Tao Zhang, Yuan Yuan Ma, and Jia Xuan Fei. Protecting user’s privacy from browser-based attacks. In *Applied Mechanics and Materials*, volume 631, pages 941–945. Trans Tech Publ, 2014.
- [171] Dove Chiu, Shih-Hao Weng, and Joseph Chiu. Backdoor use in targeted attacks. *A Trend Micro Research Paper*, 2017.
- [172] Sílvia Farraposo, Laurent Gallon, and Philippe Owezarski. Network security and dos attacks. *Feb-2005. http://www.cert.org/reports/dist_workshop.pdf*, 2005.
- [173] Prasanta Gogoi, Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Packet and flow based network intrusion dataset. In *International Conference on Contemporary Computing*, pages 322–334. Springer, 2012.
- [174] Manaf Gharaibeh and Christos Papadopoulos. The darpa-2009 intrusion detection dataset report, 2014.

- [175] The ground truth of darpa-2009 dataset. May 2016.
- [176] The tcptrace tool. April 2017.
- [177] The business intelligence development studio tool. April 2017.
- [178] Wenke Lee and Salvatore J Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)*, 3(4):227–261, 2000.
- [179] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4:119–128, 2012.
- [180] Seong Soo Kim and AL Reddy. Statistical techniques for detecting traffic anomalies through packet header data. *IEEE/ACM Transactions on Networking (TON)*, 16(3):562–575, 2008.
- [181] Ashwin Tamilarasan, Srinivas Mukkamala, Andrew H Sung, and Krishna Yendrapalli. Feature ranking and selection for intrusion detection using artificial neural networks and statistical methods. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 4754–4761. IEEE, 2006.
- [182] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [183] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.
- [184] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

- [185] Samaneh Rastegari, Philip Hingston, and Chiou-Peng Lam. Evolving statistical rulesets for network intrusion detection. *Applied Soft Computing*, 33:348–359, 2015.
- [186] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.
- [187] Dewan Md Farid, Li Zhang, Chowdhury Mofizur Rahman, M Alamgir Hosain, and Rebecca Strachan. Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert Systems with Applications*, 41(4):1937–1946, 2014.
- [188] Yacine Bouzida and Frederic Cuppens. Neural networks vs. decision trees for intrusion detection. In *IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM)*, volume 28, page 29, 2006.
- [189] Christo Panchev, Petar Dobrev, and James Nicholson. Detecting port scans against mobile devices with neural networks and decision trees. In *International Conference on Engineering Applications of Neural Networks*, pages 175–182. Springer, 2014.
- [190] Amuthan Prabakar Muniyandi, R Rajeswari, and R Rajaram. Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm. *Procedia Engineering*, 30:174–182, 2012.
- [191] Joffroy Beauquier and Yongjie Hu. Intrusion detection based on distance combination. *CESSE07, Venice, Italy, World Acacemy of Sciences, WAS*, 2007.
- [192] Heba F Eid, Aboul Ella Hassanien, Tai-hoon Kim, and Soumya Banerjee. Linear correlation-based feature selection for network intrusion detection model. In *Advances in Security of Information and Communication Networks*, pages 240–248. Springer, 2013.
- [193] The ixia pefectstorm one tool. April 2017.

- [194] The cve website. April 2017.
- [195] Jon Davis and Shane Magrath. A survey of cyber ranges and testbeds. Technical report, DTIC Document, 2013.
- [196] The accs website. April 2017.
- [197] The tcpdump tool. April 2017.
- [198] The argus tool. April 2017.
- [199] The bro-ids tool. April 2017.
- [200] The mysql cluster cge technology. April 2017.
- [201] The hadoop technologies. April 2017.
- [202] Taeshik Shon and Jongsub Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.
- [203] Zubair M Fadlullah, Tarik Taleb, Athanasios V Vasilakos, Mohsen Guizani, and Nei Kato. Dtrab: Combating against attacks on encrypted protocols through traffic-feature analysis. *IEEE/ACM Transactions on Networking (TON)*, 18(4):1234–1247, 2010.
- [204] Stuart Staniford, James A Hoagland, and Joseph M McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1-2):105–136, 2002.
- [205] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarshi Nanda, Ren Ping Liu, and Jiankun Hu. Detection of denial-of-service attacks based on computer vision techniques. *IEEE transactions on computers*, 64(9):2519–2533, 2015.
- [206] SS Chapade, KU Pandey, and DS Bhade. Securing cloud servers against flooding based ddos attacks. In *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*, pages 524–528. IEEE, 2013.

- [207] Debasish Das, Utpal Sharma, and DK Bhattacharyya. Detection of http flooding attacks in multiple scenarios. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, pages 517–522. ACM, 2011.
- [208] Kowsik Guruswamy and Siyang Yang. Detection and prevention of encapsulated network attacks using an intermediate device, September 14 2010. US Patent 7,797,411.
- [209] Amir Houmansadr and Nikita Borisov. Swirl: A scalable watermark to detect correlated network flows. In *NDSS*, 2011.
- [210] Tianbo Lu, Rui Guo, Lingling Zhao, and Yang Li. A systematic review of network flow watermarking in anonymity systems. *International Journal of Security and Its Applications*, 10(3):129–138, 2016.
- [211] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer, 2009.
- [212] R Ravinder Reddy, Y Ramadevi, and KVN Sunitha. Hybridized data technique for evaluate the anomaly. In *Advanced Computing (IACC), 2016 IEEE 6th International Conference on*, pages 685–688. IEEE, 2016.
- [213] Maher Salem and Ulrich Buehler. Mining techniques in network security to enhance intrusion detection systems. *arXiv preprint arXiv:1212.2414*, 2012.
- [214] Dan Hu, Xianchuan Yu, and Jiayin Wang. Statistical inference in rough set theory based on kolmogorov-smirnov goodness-of-fit test. *IEEE Transactions on Fuzzy Systems*, 2016.
- [215] Ana Justel, Daniel Peña, and Rubén Zamar. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997.
- [216] Selcuk Korkmaz, Dincer Goksuluk, and Gokmen Zararsiz. Mvn: an r package for assessing multivariate normality. *The R Journal*, 6(2):151–162, 2014.

- [217] Hae-Young Kim. Statistical notes for clinical researchers: assessing normal distribution (2) using skewness and kurtosis. *Restorative dentistry & endodontics*, 38(1):52–54, 2013.
- [218] Daniel T Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.
- [219] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.
- [220] The critical values of the kolmogorov smirnov test. April 2017.
- [221] The spss tool. April 2017.
- [222] Hal R Varian. Big data: New tricks for econometrics. *The Journal of Economic Perspectives*, 28(2):3–27, 2014.
- [223] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.
- [224] Asha Gowda Karegowda, AS Manjunath, and MA Jayaram. Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.
- [225] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.
- [226] Mohammed M Alani. *Guide to OSI and TCP/IP models*. Springer, 2014.
- [227] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567–581, 2013.

- [228] Yan Hu, Dah-Ming Chiu, and John CS Lui. Adaptive flow aggregation-a new solution for robust flow monitoring under security attacks. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 424–435. IEEE, 2006.
- [229] William G Cochran. *Sampling techniques*. John Wiley & Sons, 2007.
- [230] Darren R Kerr and Barry L Bruins. Network flow switching and flow data export, June 5 2001. US Patent 6,243,667.
- [231] Netflow feature documentation. January 2017.
- [232] Nick Duffield. Sampling for passive internet measurement: A review. *Statistical Science*, pages 472–498, 2004.
- [233] sflow collectors. January 2017.
- [234] Mohiuddin Ahmed, Abdun Naser Mahmood, and Michael J Maher. A novel approach for network traffic summarization. In *International Conference on Scalable Information Systems*, pages 51–60. Springer, 2014.
- [235] Malcolm Rieke, James Sebastian Dennis, and Shane Robert Thorson. Systems and methods for network data flow aggregation, December 18 2015. US Patent App. 14/974,378.
- [236] The mysql aggregation functions. April 2017.
- [237] Milosz Marian Hulboj and Ryszard Erazm Jurga. Packet sampling and network monitoring. 2007.
- [238] Valentín Carela-Español, Pere Barlet-Ros, Albert Cabellos-Aparicio, and Josep Solé-Pareta. Analysis of the impact of sampling on netflow traffic classification. *Computer Networks*, 55(5):1083–1099, 2011.
- [239] Yu Zhang, FANG BinXing, and LUO Hao. Identifying high-rate flows based on sequential sampling. *IEICE TRANSACTIONS on Information and Systems*, 93(5):1162–1174, 2010.

- [240] Myungjin Lee, Mohammad Hajjat, Ramana Rao Kompella, and Sanjay Rao. Relsamp: Preserving application structure in sampled flow measurements. In *INFOCOM, 2011 Proceedings IEEE*, pages 2354–2362. IEEE, 2011.
- [241] Sajad Shirali-Shahreza and Yashar Ganjali. Flexam: flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 167–168. ACM, 2013.
- [242] Taejin Ha, Sunghwan Kim, Namwon An, Jargalsaikhan Narantuya, Chiwook Jeong, JongWon Kim, and Hyuk Lim. Suspicious traffic sampling for intrusion detection in software-defined networks. *Computer Networks*, 109:172–182, 2016.
- [243] Jason Nikolai and Yong Wang. Hypervisor-based cloud intrusion detection system. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 989–993. IEEE, 2014.
- [244] Nour Moustafa and Jill Slay. The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems. In *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2015 4th International Workshop on*, pages 25–31. IEEE, 2015.
- [245] Juliana Freire, Bharat Kumar, and Daniel Lieuwen. Webviews: accessing personalized web content and services. In *Proceedings of the 10th international conference on World Wide Web*, pages 576–586. ACM, 2001.
- [246] Jim Conallen. Modeling web application architectures with uml. *Communications of the ACM*, 42(10):63–70, 1999.
- [247] Surya Kumar Kovvali, Charles Boyle, Ravi Valmikam, and Krishnan Ramakrishnan. Hierarchical device type recognition, caching control & enhanced cdn communication in a wireless mobile network, July 19 2012. US Patent App. 13/183,777.

- [248] Stephane Bortzmeyer. Dns query name minimisation to improve privacy. 2016.
- [249] Arkadiusz Jestratjew and Andrzej Kwiecien. Performance of http protocol in networked control systems. *IEEE Transactions on Industrial Informatics*, 9(1):271–276, 2013.
- [250] Samuel Marchal, Xiuyan Jiang, Radu State, and Thomas Engel. A big data architecture for large scale security monitoring. In *Big data (BigData Congress), 2014 IEEE international congress on*, pages 56–63. IEEE, 2014.
- [251] Alfonso Valdes and Steven Cheung. Intrusion monitoring in process control systems. In *System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on*, pages 1–7. IEEE, 2009.
- [252] Robert Koch and Mario Golling. Architecture for evaluating and correlating nids in real-world networks. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–20. IEEE, 2013.
- [253] Mohamed Nassar, Bechara al Bouna, and Qutaibah Malluhi. Secure outsourcing of network flow data analysis. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 431–432. IEEE, 2013.
- [254] Dinkar Sitaram, Manish Sharma, Mariyah Zain, Ankita Sastry, and Rishika Todi. Intrusion detection system for high volume and high velocity packet streams: A clustering approach. *International Journal of Innovation, Management and Technology*, 4(5):480, 2013.
- [255] PP Anjali and A Binu. Network traffic analysis: Hadoop pig vs typical mapreduce. *arXiv preprint arXiv:1312.5469*, 2013.
- [256] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261. ACM, 2003.

- [257] Jose Fonseca, Marco Vieira, and Henrique Madeira. Evaluation of web security mechanisms using vulnerability & attack injection. *IEEE Transactions on Dependable and Secure Computing*, 11(5):440–453, 2014.
- [258] Johannes Dahse and Thorsten Holz. Static detection of second-order vulnerabilities in web applications. In *USENIX Security*, pages 989–1003, 2014.
- [259] Mohssen Mohammed and Al-Sakib Khan Pathan. *Automatic defense against zero-day polymorphic worms in communication networks*. CRC Press, 2013.
- [260] Carol Fung and Raouf Boutaba. *Intrusion Detection Networks: A Key to Collaborative Security*. CRC Press, 2013.
- [261] Steven J Templeton and Karl E Levitt. Detecting spoofed packets. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 164–175. IEEE, 2003.
- [262] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93, 2014.
- [263] Jasmin Kevric, Samed Jukic, and Abdulhamit Subasi. An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, pages 1–8, 2016.
- [264] Andreas Janecek, Wilfried Gansterer, Michael Demel, and Gerhard Ecker. On the relationship between feature selection and classification accuracy. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pages 90–105, 2008.
- [265] Nour Moustafa and Jill Slay. A hybrid feature selection for network intrusion detection systems: Central points. 2015.
- [266] Yuh-Jye Lee, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Anomaly detection via online oversampling principal component analysis. *IEEE transactions on knowledge and data engineering*, 25(7):1460–1470, 2013.

- [267] Dayu Yang and Hairong Qi. A network intrusion detection method using independent component analysis. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.
- [268] John H Gennari, Pat Langley, and Doug Fisher. Models of incremental concept formation. *Artificial intelligence*, 40(1-3):11–61, 1989.
- [269] Atif Ahmad, Sean B Maynard, and Sangseo Park. Information security strategies: towards an organizational multi-strategy perspective. *Journal of Intelligent Manufacturing*, 25(2):357–370, 2014.
- [270] Richard Heady, George F Luger, Arthur Maccabe, and Mark Servilla. *The architecture of a network level intrusion detection system*. University of New Mexico. Department of Computer Science. College of Engineering, 1990.
- [271] P Jongsuebsuk, N Wattanapongsakorn, and C Charnsripinyo. Real-time intrusion detection with fuzzy genetic algorithm. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*, pages 1–6. IEEE, 2013.
- [272] Dumidu Wijayasekara, Ondrej Linda, Milos Manic, and Craig Rieger. Mining building energy management system data using fuzzy anomaly detection and linguistic descriptions. *IEEE Transactions on Industrial Informatics*, 10(3):1829–1840, 2014.
- [273] Wentao Ma, Hua Qu, and Jihong Zhao. Estimator with forgetting factor of correntropy and recursive algorithm for traffic network prediction. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 490–494. IEEE, 2013.
- [274] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

- [275] Michael D Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588, 1995.
- [276] B Wagle. Multivariate beta distribution and a test for multivariate normality. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 511–516, 1968.
- [277] Arjun K Gupta and Saralees Nadarajah. *Handbook of beta distribution and its applications*. CRC press, 2004.
- [278] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on pattern analysis and machine intelligence*, 24(3):381–396, 2002.
- [279] Zhanyu Ma and Arne Leijon. Beta mixture models and the application to image classification. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2045–2048. IEEE, 2009.
- [280] Josef Stoer and Roland Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.
- [281] Takao Asano, Tetsuo Asano, and Hiroshi Imai. Partitioning a polygonal region into trapezoids. *Journal of the ACM (JACM)*, 33(2):290–312, 1986.
- [282] Peter J Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, 2011.
- [283] Tarek Elgamal, Maysam Yabandeh, Ashraf Aboulnaga, Waleed Mustafa, and Mohamed Hefeeda. spca: Scalable principal component analysis for big data on distributed platforms. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 79–91. ACM, 2015.
- [284] Gholam Reza Zargar, Tania Baghaie, et al. Category-based intrusion detection using pca. *Journal of Information Security*, 3(04):259, 2012.

- [285] Eduardo De la Hoz, Emiro De La Hoz, Andrés Ortiz, Julio Ortega, and Beatriz Prieto. Pca filtering and probabilistic som for network intrusion detection. *Neurocomputing*, 164:71–81, 2015.
- [286] Gholam Reza Zargar and Peyman Kabiri. Identification of effective network features for probing attack detection. In *Networked Digital Technologies, 2009. NDT'09. First International Conference on*, pages 392–397. IEEE, 2009.
- [287] Thanassis Giannetsos and Tassos Dimitriou. Spy-sense: spyware tool for executing stealthy exploits against sensor networks. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 7–12. ACM, 2013.
- [288] Chih-Fong Tsai and Chia-Ying Lin. A triangle area based nearest neighbors approach to intrusion detection. *Pattern recognition*, 43(1):222–229, 2010.
- [289] Praneet Saurabh and Bhupendra Verma. An efficient proactive artificial immune system based anomaly detection and prevention system. *Expert Systems with Applications*, 60:311–320, 2016.
- [290] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarsi Nanda, and Ren Ping Liu. Denial-of-service attack detection based on multivariate correlation analysis. In *International Conference on Neural Information Processing*, pages 756–765. Springer, 2011.

Appendix A

Protocols of UNSW-NB15

dataset

The UNSW-NB15 dataset comprises a wide range of common protocols, including TCP, UDP and ICMP, and the services listed in Table A.1 with their numbers of records.

Descriptions of these protocols and services are provided below.

- **TCP (Transmission Control Protocol)**: is a basic protocol in the Internet protocol suite which is a standard for defining how to establish and maintain a network conversation via application programs regarding the exchange of network data.
- **UDP (User Datagram Protocol)**: is an alternative communication protocol to TCP used primarily to construct low-latency and loss-tolerating connections between programs on the Internet. Both UDP and TCP run on

Table A.1: UNSW-NB15 services

Service	Number of records
DHCP	172
DNS	781668
FTP	49090
HTTP	206273
IRC	31
POP3	1533
RADIUS	40
SMTP	81644
SNMP	113
SSH	47160
SSL	142

top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP and TCP/IP.

- **ICMP (Internet Control Message Protocol)**: is an error-reporting protocol network device, such as a router, used to create error messages to send to the source IP address when network problems prevent the delivery of IP packets which indicate that a gateway to the Internet, router, service or host cannot be reached for packet delivery.
- **DHCP (Dynamic Host Configuration Protocol)**: is a client-server protocol which automatically provides an IP host with its IP address and other related configuration information, for example, the subnet mask and default gateway.
- **DNS (Domain Name System)**: is a hierarchical decentralised naming system for computers, services or any resource related to the Internet or a network.
- **FTP (File Transfer Protocol)**: is a standard network protocol used to transfer computer files between a client and server on a network.
- **HTTP (Hypertext Transfer Protocol)**: is an application protocol for distributed, collaborative, hypermedia information systems and is the basic means of data communication for the World Wide Web.
- **IRC (Internet Relay Chat)**: is an application-layer protocol that enables communication in the form of text via chatting which works on a client-server networking model.
- **POP3 (Post Office Protocol)**: is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mails from a remote server over a TCP/IP connection.
- **RADIUS (Remote Authentication Dial-In User Service)**: is a network protocol which provides centralised authentication, authorisation and

accounting (AAA) management for users who connect to and use a network service.

- **SMTP (Simple Mail Transfer Protocol):** is an Internet standard for electronic mail transmission.
- **SNMP (Simple Network Management Protocol):** is an Internet-standard protocol for collecting and organising information about managed devices on IP networks and for modifying that information to change a device's behaviour.
- **SSH (Secure Shell):** is a cryptographic network protocol for securely operating network services over an unsecured network.
- **SSL (Secure Sockets Layer):** is the standard security technology for creating an encrypted link between a web server and browser.

Appendix B

Features of NSL-KDD dataset

This appendix discusses the features of the NSL-KDD dataset used to evaluate the mechanisms developed in this thesis. There are 41 features, 9 intrinsic, 9 time, 10 statistical and 13 content, with a class label connected to each record, which are described in the following table, respectively.

Table B.1: NSL-KDD features

No.	Name	Type	Description
Intrinsic features			
1	duration	Float	Length of record connection in seconds
2	protocol_type	Nominal	Protocol type, such as TCP, UDP and ICMP
3	service	Nominal	Network service at destination, such as HTTP and FTP
4	src_bytes	Float	Number of data bytes from source to destination
5	dst bytes	Float	Number of data bytes from destination to source
6	flag	Nominal	Normal or error status of record connection
7	Land	Binary	If source and destination of IP addresses and ports are equal, value 1, otherwise 0
8	wrong fragment	Integer	Number of ‘wrong’ fragments
9	urgent	Integer	Number of urgent packets
Time features			
10	count	Integer	Number of record connections to same host as current connection in previous two seconds
11	serror_rate	Float	Percentage of same host connections with ‘SYN’ errors
12	rerror_rate	Float	Percentage of same host connections with ‘REJ’ errors
13	same_srv_rate	Float	Percentage of same host connections to same service

14	diff_srv_rate	Float	Percentage of same host connections to different services
15	srv_count	Integer	Number of record connections to same service as current connection in previous two seconds
16	srv_serror_rate	Float	Percentage of same service connections with 'SYN' errors
17	srv_rerror_rate	Float	Percentage of same service connections with 'REJ' errors
18	srv_diff_host_rate	Float	Percentage of same service connections to different hosts
Statistical features			
19	dst_host_count	Integer	Number of record connections to same host in previous 100 connections
20	dst_host_serror_rate	Float	Percentage of record connections with 'SYN' errors
21	dst_host_rerror_rate	Float	Percentage of record connections with 'REJ' errors
22	dst_host_same_srv_rate	Float	Percentage of record connections to same service
23	dst_host_diff_srv_rate	Float	Percentage of records with same host connections to different services
24	dst_host_srv_count	Float	Number of record connections to same service in previous 100 connections
25	dst_host_srv_serror_rate	Float	Percentage of same service connections with 'SYN' errors
26	dst_host_srv_rerror_rate	Float	Percentage of same service connections with 'REJ' errors
27	dst_host_srv_diff_host_rate	Float	Percentage of same service connections to different hosts
28	dst_host_same_src_port_rate	Float	Percentage of connections from same source port
Content features			
29	hot	Float	Host indicator, e.g., access to system directories, and creation and execution of programs
30	num_failed_logins	Integer	Number of failed login attempts
31	logged_in	Binary	1 if successfully logged in, 0 otherwise
32	num_compromised	Integer	Number of compromised states on destination host (e.g., file/path 'not found' errors and 'jump to' instructions)
33	root_shell	Binary	1 if root shell obtained, 0 otherwise
34	su_attempted	Binary	1 if 'su root' command attempted, 0 otherwise

35	num_root	Integer	Number of ‘root’ accesses
36	num_file_creations	Integer	Number of file creation operations
37	num_shells	Integer	Number of shell prompts
38	num access files	Float	Number of operations on access control files
39	num_outbound_cmds	Float	Number of outbound commands in ftp session
40	is host login	Binary	1 if login belongs to ‘host’ list, 0 otherwise
41	is_guest_login	Binary	1 if login ‘guest’ login, 0 otherwise