DETECTION OF DIFFERENT TYPES OF DISTRIBUTED DENIAL OF SERVICE ATTACKS USING MULTIPLE FEATURES OF ENTROPY AND SEQUENTIAL PROBABILITY RATIO TEST

By

AL MAFRACHI BASHEER HUSHAM ALI

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirement for the Degree of Doctor of Philosophy

June 2024

FK 2024 12

DEDICATION

TO
My father
My mother
My brothers
My family
(Baraa, Ibrahim and Sireen)

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

DETECTION OF DIFFERENT TYPES OF DISTRIBUTED DENIAL OF SERVICE ATTACKS USING MULTIPLE FEATURES OF ENTROPY AND SEQUENTIAL PROBABILITY RATIO TEST

By

AL MAFRACHI BASHEER HUSHAM ALI

June 2024

Chairman : Associate Professor Nasri bin Sulaiman, PhD
Faculty : Engineering

Distributed Denial of Service (DDoS) attacks are among the most dangerous types of attacks. These kinds of attacks bring targeted servers down and make their services unavailable to legal users. The main problem of current entropy-based detection techniques is how can these techniques identify up-to-date DDoS attacks accurately using dynamic threshold and effective features that can be selected to decrease time complexity or cost by using effective flows or features generation tool to improve confusion metrics. The first objective of this study is to identify infected ethernet and detect various kinds of up-to-date DDoS attacks using dynamic threshold by implementing multiple features of entropy and Sequential Probabilities Ratio Test approach (E-SPRT). The second is to select relevant features to reduce time complexity or cost and improve performance of detection by implementing a new combination of machine learning techniques which are ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation approaches. The third is to analyse problems of original CICFlowMeter and validate

the effectiveness of revised version of CICFlowMeter tool on detection approach as well. Furthermore, the Defense Advanced Research Projects Agency (DARPA) 1998, DARPA2000, and Canadian Institute for Cybersecurity (CIC-DDoS2019) databases were utilizing to evaluate the implementation. In addition, ESPRT using feature selection approach with five features achieved an accuracy over 97% with a time cost 0.189ms based on using CICDDoS2019 dataset. The accuracy of ESPRT with best single features also exceeded or equalled to 99% for different kinds of DDoS attacks such as NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, and UDP. The average detection rates for identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, and UDP using best single features were consistently above 99%. However, the average detection rates for identifying UDP-lag, SYN, and TFTP were 97%, 89%, and 98%, respectively. Most of these attacks exhibited an average False Positive Rate (FPR) close to 0. The execution times of ESPRT approach were approximately 189ms, 625ms, 831ms, and 1,276ms when feature sets were 5, 10, 15, and 20, respectively. Finally, the accuracy and f-score for ESPRT using the revised version exceeded 99% when DARPA 1998 dataset used. However, when using the original version, the accuracy ranged between 33% and 38% for various features.

Keywords: DDoS Attacks, SPRT, Entropy, ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix

SDG: GOAL 9: Industry, Innovation, and Infrastructure, GOAL 11: Sustainable Cities and Communities

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

PENGESANAN PELBAGAI JENIS PENAFIAN SERANGAN PERKHIDMATAN TEREDAR MENGGUNAKAN PELBAGAI CIRI-CIRI ENTROPI DAN UJIAN NISBAH KEBARANGKALIAN BERURUTAN

Oleh

AL MAFRACHI BASHEER HUSHAM ALI

Jun 2024

Pengerusi : Profesor Madya Nasri bin Sulaiman, PhD
Fakulti : Kejuruteraan

Serangan Penafian Perkhidmatan (DDoS) Teragih adalah antara jenis serangan yang paling berbahaya. Serangan jenis ini menurunkan pelayan yang disasarkan dan menjadikan perkhidmatan mereka tidak tersedia kepada pengguna yang sah. Masalah utama teknik pengesanan berasaskan entropi semasa ialah bagaimana teknik ini boleh mengenal pasti serangan DDoS terkini dengan tepat menggunakan ambang dinamik dan ciri berkesan yang boleh dipilih untuk mengurangkan kerumitan masa atau kos dengan menggunakan aliran berkesan atau alat penjanaan ciri untuk menambah baik. metrik kekeliruan. Objektif pertama kajian ini adalah untuk mengenal pasti ethernet yang dijangkiti dan mengesan pelbagai jenis serangan DDoS terkini menggunakan ambang dinamik dengan melaksanakan pelbagai ciri entropi dan pendekatan Ujian Nisbah Kebarangkalian Berjujukan (E-SPRT). Yang kedua ialah memilih ciri yang berkaitan untuk mengurangkan kerumitan masa atau kos dan meningkatkan prestasi pengesanan dengan melaksanakan gabungan baharu teknik pembelajaran mesin iaitu ANOVA, Pengelas Pokok Tambahan, Hutan Rawak, dan

Matriks Korelasi dengan pendekatan Korelasi Pearson. Yang ketiga adalah untuk menganalisis masalah CICFlowMeter asal dan mengesahkan keberkesanan versi semakan alat CICFlowMeter pada pendekatan pengesanan juga. Tambahan pula, pangkalan data Agensi Projek Penyelidikan Lanjutan Pertahanan (DARPA) 1998, DARPA2000, dan Institut Keselamatan Siber Kanada (CIC-DDoS2019) digunakan untuk menilai pelaksanaan. Selain itu, ESPRT menggunakan pendekatan pemilihan ciri dengan lima ciri mencapai ketepatan lebih 97% dengan kos masa 0.189ms berdasarkan penggunaan dataset CICDDoS2019. Ketepatan ESPRT dengan ciri tunggal terbaik juga melebihi atau menyamai 99% untuk pelbagai jenis serangan DDoS seperti NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP dan UDP. Kadar pengesanan purata untuk mengenal pasti NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP dan UDP menggunakan ciri tunggal terbaik secara konsisten melebihi 99%. Walau bagaimanapun, kadar pengesanan purata untuk mengenal pasti UDP-lag, SYN dan TFTP masing-masing adalah 97%, 89% dan 98%. Kebanyakan serangan ini menunjukkan purata False Positive Rate (FPR) hampir kepada 0. Masa pelaksanaan pendekatan ESPRT ialah kira-kira 189ms, 625ms, 831ms dan 1,276ms apabila set ciri masing-masing ialah 5, 10, 15 dan 20. Akhirnya, ketepatan dan skor-f untuk ESPRT menggunakan versi yang disemak melebihi 99% apabila dataset DARPA 1998 digunakan. Walau bagaimanapun, apabila menggunakan versi asal, ketepatan berjulat antara 33% dan 38% untuk pelbagai ciri.

Kata Kunci: Serangan DDoS, SPRT, Entropi, ANOVA, Pengelas Pokok Tambahan, Hutan Rawak dan Matriks Korelasi

SDG: MATLAMAT 9: Industri, Inovasi dan Infrastruktur, MATLAMAT 11: Bandar dan Komuniti Lestari

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor, **Assoc. Prof. Dr. Nasri Sulaiman** for his continuous support, invaluable guidance, and patience as well as his encouragement and inspiration along this research journey without which this thesis could not be done as smoothly as it did. I am very thankful for all the tasks he produced for me. God bless him and his family.

Also, I would like to express my sincere gratitude to my supervisory committee members, **Prof. S.A.R. Al-Haddad, Assoc. Prof. Rodziah Atan, and Dr. Siti Lailatul Mohd Hassan** for his constructive suggestions during my research period and for supporting me through my studying time.

To my dear friends and peers, I am so grateful to have companions like you by my side during this journey to pursue my degree. It is because of you who made me feel warm all the time, making my life abroad complete and colourful.

Most importantly, I want to say thanks to my father, mother, wife, and brothers. They helped me out during the difficult times in life and provided me with warm encouragement.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Nasri bin Sulaiman, PhD
Associate Professor
Department of Electrical and Electronic Engineering
Universiti Putra Malaysia
(Chairman)

Syed Abdul Rahman Al-Haddad bin Syed Mohamed, PhD
Professor
Department of Computer and Communications Systems Engineering
Universiti Putra Malaysia
(Member)

Rodziah binti Atan, PhD
Associate Professor
Department of Software Engineering and Information Systems
Universiti Putra Malaysia
(Member)

Siti Lailatul binti Mohd Hassan, PhD
Senior Lecturer
Department of Electrical Engineering
Universiti Teknologi MARA
(Member)

ZALILAH MOHD SHARIFF, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 12 December 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

λ_1	Lower Bound Threshold
λ_2	Upper Bound Threshold
ACK	Acknowledgement
ANN	Artificial Neural Network
ANOVA	ANalysis of VAriance
APT	Advanced Persistent Threat
AWS	Amazon Web Services
Bps	Bit Per Seconds
Bwd	Backward Direction
CAIDA	Cooperative Association for Internet Data Analysis
CBA	Classification based on Associations Technique
CC	Cloud C
CICFlowMeter	Canadian Institute for Cybersecurity Flow Meter
CharGen	Character Generator Protocol
CLDAP	Connectionless Lightweight Directory Access Protocol
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CPS	Cyber-Physical System
CERL	Computer-based Education Research Lab
CNSS	Committee on National Security Systems
CUSUM	Cumulative Sum
CRW	Code Red Worm
CWR	Common Works Registration
DARPA MIT	Defense Advanced Research Projects Agency Massachusetts Institute of Technology

DDoS	Distributed Denial of Services Attacks
DNN	Dense Neural Networks
DNS	Domain Name System
DT	Decision Tree
ECE	Electronics and Communication Engineering
F_N	False Negative
F_P	False Positive
FDR	False Discovery Rate
FEDDM	Fast Fourier transform and Entropy detection of DDoS Method
FFSc	Feature Feature Score
FIN	Finish
FITS	Future Internet Testbed with Security
FOR	False Omission Rate
FNR	False Negative Rate
FNV-1a	Fowler–Noll–Vo function
FPR	False Positive Rate
Fwd	Forward direction
HESS	Hybrid Entropy SSAE-SVM
H-IDS	Hybrid Intrusion Detection System
HIDS	Host Intrusion Detection System
HOIC	High Orbit Ion Cannon
HOSVD	Higher-Order Singular Value Decomposition
HTM	Hierarchical Temporal Memory
HTTP	Hypertext Transfer Protocol
HULK	HTTP-Unbearable-Load-King
IAT	Inter-Arrival-Time

ICMP	Internet Control Message Protocol
ID3	Iterative Dichotomiser 3
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IP DSCP	Internet Protocol Differentiated Services Code Point
IPv4	Internet Protocol version 4
IRC	Internet Chat Relay
KDD Cup	Knowledge Discovery and Data Mining Tools Competition
KNN	K-Nearest Neighbors Algorithm
KOAD	Kernel-based Online Anomaly Detection
LDAP	Lightweight Directory Access Protocol
LOIC	Low Orbit Ion Cannon
MAC	Media Access Control address
Mpps	Million Packets Per Second
MSA	Multiple Sequence Alignment
MSAm	Means Square Among
MSSQL	Microsoft SQL Server
MSW	Means Square Within
NB	Naïve Bayes
NETAD	Network Traffic Anomaly Detection
NETBIOS	Network Basic Input/Output System
NIDS	Network Intrusion Detection System
NPV	Negative Predictive Value
NTP	Network Time Protocol
NSL-KDD	Network Security Laboratory- Knowledge Discovery in Databases

OVH	French cloud computing company of Virtual private server Host
P2P	Peer to Peer
PCC-AE-DNN	Pearson Correlation Coefficient- autoencoder algorithms-Dense Neural Networks
PHAD	Packet Header Anomaly
PHMM	Profile Hidden Markov Model
PPS	Packet Per Seconds
PPSA	Profiling and Preventing Security Attacks
PPV	Positive Predictive Value
PSH	Push
PWI	Windows Initiation rate
RDD	Resilient Distributed Datasets
RF	Random Forest
RNN	Recurrent Neural Networks
RPS	Request Per Seconds
RST	Reset of TCP
SDN	Software Defined Network
SIP	Session Initiation Protocol
SMOTE	Synthetic Minority Oversampling Technology
SNMP	Simple Network Management Protocol
SNN	Spiking Neural Network
SPRT	Sequential Probabilities Ratio Test
SSAE	Stacked Sparse Auto Encoder
SSDP	Simple Service Discovery Protocol
SSH	Secure Socket Shell
SVM	Support Vector Machine

SYN	Synchronize
T_N	True Negative
T_P	True Positive
Tbps	True Bit Per Seconds
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TNR	True Negative Rate
TPR	True Positive Rate
UDP	User Datagram Protocol
UNSW-NB15	University of New South Wales- Network intrusion detection systems database
URG	Urgent signal
VOIP	Voice Over Internet protocol
VPN	Virtual Private Network
W	Window size

CHAPTER 1

INTRODUCTION

DDoS attacks refer to Distributed Denial of Service attacks. These attacks pose a significant cybersecurity threat. These attacks are designed to disrupt the access of legitimate users to services provided by a targeted server or controller. To achieve this, cyber attackers exploit vulnerabilities in network systems, commandeering compromised machines to generate a massive volume of network packets or flows directed at a specific victim, which could be a server, controller, or device., or device. Consequently, the targeted victim experiences disruption, rendering its services inaccessible to authorized users. The consequences of successful DDoS attacks encompass financial losses and the compromise of critical data within the targeted system [1] .

DDoS attacks lead to reduction in CPU resources of victims and result in the denial of services to all users. DDoS attacks not only affect the resources of the targeted devices but also impact the network bandwidth of the targeted system. In such attack scenarios, attackers aim to consume available resources of victims and prevent legitimate users from accessing these resources.

1.1 Overview

The intensity, complexity, rates, and frequency of DDoS anomalies have been increased and reached to new records [2]. Attackers are driven to send a very large number of useless packets toward their victims to disrupt their services. These volumetric attacks can be categorized using three metrics as shown in Figure 1.1.

The first metric is bits per second (bps) that targets network connections. The second metric is packets per second (pps) that targets network devices and DNS servers. The third metric is requests of HTTPs per second (rps) which targets application servers [3], [4].

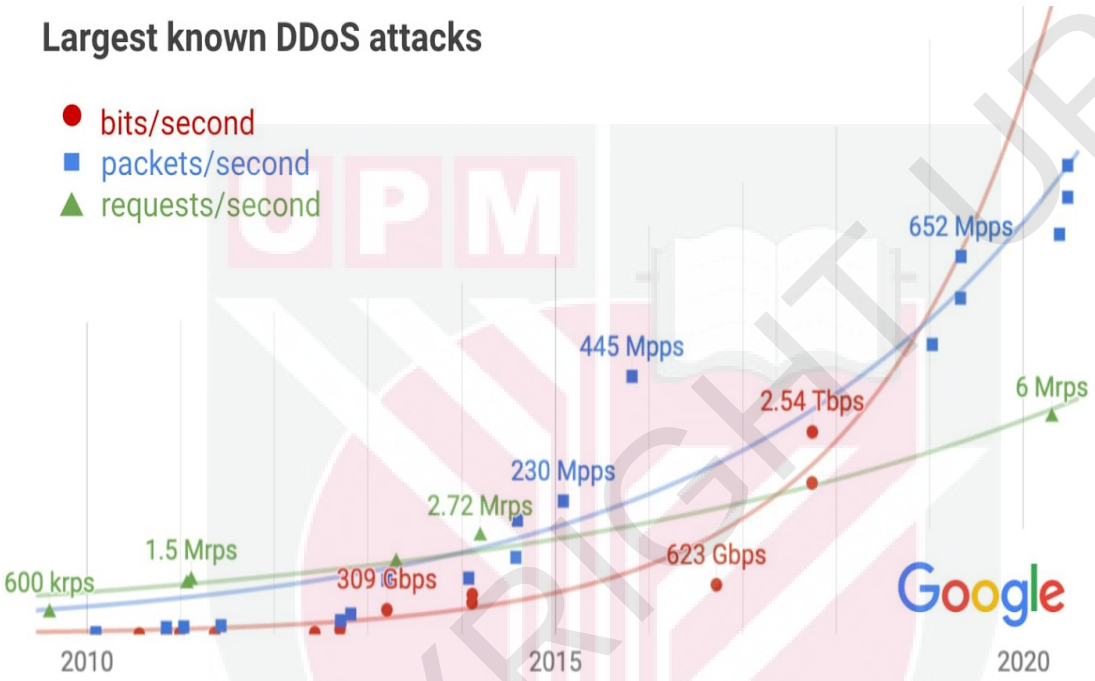


Figure 1.1 : Trends in DDoS Attack Volumes based on Different Metrics [3], [4]

The growth of DDoS across all these metrics is clear. First of all, the bps metric has been increased steadily over time. For example, Amazon Web Services (AWS) Shield, designed to protect application running on AWS against threads such as DDoS attacks, recorded notable of instances of DDoS attacks based on UDP and CLDAP reflection techniques. These attacks reached a volume of 2.3 Tbps , marking a 44% increase compared to any previously observed DDoS attacks by AWS up until Q1 of 2020. The mitigation process for such an attack spanned nearly 3 days [5].

Google cloud infrastructure detected a 2.5 Tbps DDoS attack in 2017 as shown in Figure 1.1. This attack persisted for a duration of 6 months. Attackers exploited SNMP, CLDAP, and DNS servers and manipulated them to send responses to the Google infrastructure [3]. This attack was four times larger than previous high record which is Mirai botnet. Mirai botnet is DDoS attacks that targeted multiple victims in 2016 by recruiting thousands of infected IoT devices connected to the network around the world. The count of these devices reached to nearly 600,000 and encompassed a variety of devices such as cameras, baby monitors, cars, medical equipment, DVSs, routers, headset, and more. The victims of these attacks included KrebsOnSecurity, OVH, and Dyn company. KrebsOnSecurity was one target of Mirai at high intensity, with rates reaching 620 Gbps over a span of 2 to 7 days. This attack used TCP, GRE, and HTTP protocols. The attack led to temporary shutdown of the krebs website [6]. Mirai also targeted OVH, which is French cloud computing company, generating traffic at a rate of 1.1 Tbps. The duration of attacks was between 2 to 7 days, resulting in service disruptions for OVH [4], [7]. Finally, Dyn, which is web application security company internet performance management provider, was targeted by Mirai as well. These attacks reached a traffic rate of 1.5 Tbps and ended for one day. Attackers exploited the DNS protocol server to reflect replies toward OVH. Their efforts resulted in internet outage for Dyn [8].

In 2018, GitHub servers were hit with 1.35 Tbps of traffics rates coming from around 100,000 memcached servers. These servers were originally intended to accelerate networks and websites by caching their content. However, these servers, own by businesses and organizations, were available online without protection. Attackers exploited this vulnerability by sending emails to these servers. Then, these

servers send large number of replies to victims which leads within 20 minutes of the attacks to service outages [9], [10]. Izz ad-Din al-Qassam Cyber Fighters targeted six banks in the USA with rates of 60 Gbps and continued for two days. They did that in response to a controversial movie about the prophet Muhammad "peace be upon him" that remained on a YouTube channel. This attack resulted in website crash and service outage in 2012 [11], [12]. Hongkong experienced DDoS attacks with rates of 500 Gbps in 2014 by Occupy Central. TCP and HTTPS were two vulnerable protocols involved in this attack and led to reduce system availability [4]. Spamhaus and Cloudflare companies encountered DDoS attacks, with volumes reaching 300 Gbps 400 Gbps, respectively. Spamhaus utilized DNS and TCP protocols while Cloudflare employed NTP protocol for their attacks [4].

In addition, the second metric is packets per second (pps) which targeted network layer and their protocols. Google cloud recorded an increase in the number of packets during 2020. The number of packets per second reached 690 Mpps that was generated by IoT botnet and targeted google cloud as shown in Figure 1.1. Customer VM was also targeted with 445 Mpps, occurring within a span of twenty seconds in 2015 [3], [4]. Finally, AWS documented an increase in number of packets rates during the first quarter of 2020, with 300 Mpps being recorded by using SYN flood attacks in January of 2020 [5].

Finally, the third metric is requests per second (rps) which reflects the number of HTTP requests and web application events targeted application servers. These requests may involve DDoS attacks that flood web applications with flows, making them difficult to detect using network-based detection systems. Such floods are

coming from web harvesting, web data extraction from websites, non-human flows, or account takeover bots [5]. For example, thousands of websites got infected by man-in-the-middle attack in 2014. Main attackers induced these infected websites to send a large number of HTTP requests to YouTube channel. These requests reached to 2.7 Mrps in 2014. However, these requests were increased in 2020 when Google Cloud was targeted with 6Mrps of traffic rates as shown in Figure 1.1 [3]. Unfortunately, number of DDoS malicious activities tend to be maximized in the future. Based on Cisco Annual Internet Report in 2018, the number of DDoS malicious events will be increased from almost 8 million attacks in 2018 to almost 15.4 million in 2023 [13]. Finally, Figure 1.2 illustrates escalating trend number of DDoS attacks over years.

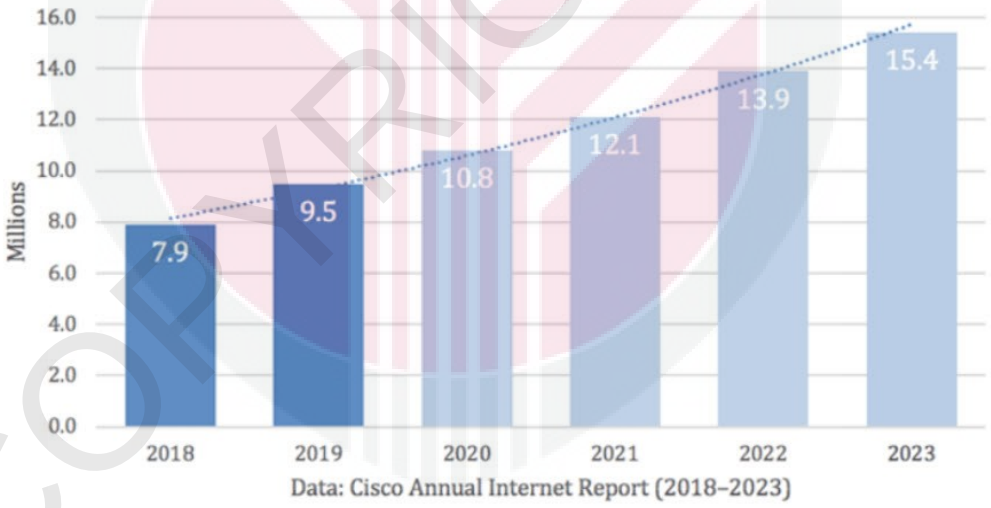


Figure 1.2 : Number of DDoS Attacks Based on Cisco Annual Internet Report (2018–2023) [13]

1.2 Techniques of DDoS Detection

Many approaches have been established to identify DDoS anomalies. In general, there are two main mechanisms to defend against DDoS. The first mechanism is

Intrusion Detection System (IDS). IDS is responsible for identifying suspicious activities that may lead to steal sensitive data from the network. port scanners, malware, or cybersecurity violations are all examples of these anomaly activities. IDS can only identify these activities and notify administrators regarding attack availability without taking further actions [14]. The second mechanism is the Intrusion Prevention System (IPS). IPS is also responsible for identifying anomalies network traffics, and it is similar to IDS. However, IPS can also take an action in response to malicious behaviors. This action may include dropping malicious flow to prevent these traffics from passing through toward targeted system [14].

Furthermore, DDoS intrusion detection system (IDS) techniques can be classified into two main groups: misuse detection and anomaly detection. Misuse detection is also called as the pattern matching technique. It assumes that a detection system or program has patterns that match attack behavior. It depends on stored traffics that are previously identified as attacks in order to detect known attacks. Misuse detection can be categorized into various types, such as signature-based detection, rule-based detection, and state-based detection [15], [16].

Moreover, unlike signature-based detection techniques, anomaly-based detection techniques rely on profiling benign patterns during the training process. The new patterns that are profiled from incoming data are then compared with the benign patterns previously stored in the database. The suspicious activities are deviated from normal patterns are considered as anomaly behavior. Statistical-based techniques, machine learning-based techniques, and deep learning-based techniques are all techniques for anomaly detection [17].

In addition, statistical models are techniques that rely on analyzing and investigating network traffics to obtain a deeper understanding of these traffics. These approaches are able to identify variables to be used as inputs and finding common connection among these inputs in order to predict output behavior. Statistcal approaches can detect attacks in early stage. Sequential Probability Ratio Test (SPRT) and entropy are some examples these effective techniques that can be to detect DDoS attack.

Likewise, entropy-based approach is used to measure the randomness. When a specific host is targeted by a large number of flows, the destination IP address appears frequently, reducing randomness and entropy. This can provide some indication about the attack availability. SPRT analyze incoming feature that were extracted from traffics and monitor switch ethernets that let these traffics passing through. SPRT is a statistical technique introduced by Waled and based on mathematical calculation. It uses two hypothesis values, H1 and H2, to differentiate between normal and compromised flows and to locate compromised interfaces. H1 is used to determine interfaces that have been injected with normal flows, while H2 is used to determine those injected with compromised flows based on feature values of flows [18]. SPRT can provide quick feedback about status of incoming traffics in early stage. Thus, combining both entropy and SPRT approaches is an effective way to detect DDoS attacks. It eliminates the dependence on using a static threshold of entropy by combining the entropy approach with the SPRT method in order to make decisions.

Finally, incoming traffics contains features that can be used as input for the detection stage. CICFlowMeter tool can be used to extract these features. Some of these

features are irrelevant and may increase execution time or cost. Thus, feature selection is an important step before detection stage in order to remove meaningless, irrelevant, and repetitive features, which in turn reduces detection errors and increases the accuracy of the identification process. Combination of four feature selection methods were used which are ANOVA, Extra Tree, Random Forest, and Pearson Correlation Coefficient to determine the important features. Then, the best group of features with the highest occurrences were chosen as inputs for the detection stage. The lower number of features selected is the shorter time required to get the results. Thus, 5 features were better than other in term of time execution. However, other number of selected features may be better when accuracy of detection is better. Finally, the feature selection process is mentioned in 4.3 section.

1.3 Current Problem of DDoS Solutions

There are many criteria can be employed to measure the effectiveness of a DDoS detection approach. First indication of perfect detection method is generating higher accuracy, f-score, and detection rate in real time. The higher accuracy can be accomplished when sensitivity or true positive rate is high, and lower probability of false alarm or false positive rate is low. Sensitivity refers to the number of compromised traffics that is correctly identified as malicious traffics while probability of false alarm means the number of compromised traffics that was incorrectly detected as normal traffics [19].

Some previous detection approaches are effective and efficient in identifying DDoS threats. On the other hands, these approaches fail to deal with complex, recent, and high-rate attacks [20], [21], [22]. Large-scale Automated DDoS detection System

[23], DDoS Network Attack Recognition and Defense [24], and Netbouncer Technology [25] are all examples of old defenses against DDoS attacks that fail to detect new and complex attacks, as stated in [26].

Some of these methods are based on using statistical methods. Many studies have been conducted using statistical-based techniques to identify DDoS attacks. These approaches provide a deeper analysis of incoming traffic in computer networks and can detect anomalies in an early stage [17]. One of these methods is the entropy-based approach, which has been widely researched. However, entropy relies on a threshold to make decisions about the presence of attacks. Depending on fixed or static value of threshold may lead to decrease the effectiveness of the detection method [27], [28]. Entropy-based methods may also rely on known attack patterns from previous datasets to make decisions, and this can reduce the ability of the detection system to identify new attacks [15].

Some other techniques, such as [18], [29], use common features that can be extracted from network traffic, such as internet protocol source address, internet protocol destination address, port source address, port destination address, or protocol name, as arguments for their detection algorithms. However, techniques that rely solely on these features might be less accurate, as attackers have evolved their methods to bypass the detection stage. These methods disregard other essential features that can be extracted from network traffic, including the total number of packets-based feature, total length of packets-based feature, flow Inter-Arrival-Time (IAT)-based feature, flags-based feature, bulk-based feature, subflow-based feature, or active and idle-based feature. By neglecting these features, these techniques might overlook

behavioral aspects of attacks and miss the opportunity to achieve high accuracy and a lower false positive rate.

Some other methods struggle to detect anomalies at an early stage due to errors in accurately determining the starting point of the detection process. Early detection of DDoS attacks avoids high processing rates that occur during the peak of an attack [29], [30], [31]. Essentially, the initial step of detection should involve identifying the infected interfaces of a switch. During an attack, these interfaces receive an excessive number of floods. These traffics are then forwarded toward the targeted system. Consequently, it is crucial to identify these interfaces to minimize the potential harm against the targeted victim and achieve early detection of DDoS attacks.

DDoS attacks involve flooding targeted system with a very large number of useless networks traffics. In its turn, extracting new features from all incoming traffics in order to analyze and examine each traffic flow will lead to consume the system resources such as storage or memory space. This will lead to increase time complexity or cost. Time complexity or cost means time needed to run the detection system. When number of incoming flows toward server increased, the time needed for handling new extracted features will also increase. Sending many numbers of ICMP, TCP, or UDP requests to a server may result in increasing the time complexity or cost of the system. This can result in the victim machine being stopped, frozen, or restarted [16]. Overwhelming sensors, path-based DoS, and reprogramming are all example of negative effects of DDoS attacks [32]. Attackers also send excessive number of packets, designed to take a long time to process, with

the intention of overloading and crashing networking devices as well such as routers, switches, and hubs [33].

Threat detection should be identified in a short time or cost. This can be done by reducing the number of inputs before applying the detection process. Some techniques utilize a large number of features, which may increase the execution time and decrease the system's performance. Using irrelevant parameters and features can consume system resources and increase computational time or cost, potentially leading to system failure [18], [29]. Another issue with current techniques is the lack of programmability. To adequately address the evolving requirements of networks, detection techniques need to be programmable. This feature can handle the increasing number of traffic instances that exhibit attack characteristics. This can be achieved by programming network switches that possess programmability capabilities [34].

1.4 Problem Statement

The problem of current detection of DDoS attacks techniques can be summarized as the following:

> *The main problem of current entropy-based detection techniques is not being able to identify up-to-date DDoS attacks and their infected interfaces with high accuracy and low false positive rate (probability of false alarm). These techniques also used static threshold in order to make decision regarding the attack availability which may lead to change the results. In addition, high time complexity or cost is another problem of these techniques which happened due to selecting irrelevant features. Finally, original CICFlowMeter tool has problems in generating flows used during detection process which*

may lead to decrease accuracy, precision, and sensitivity metrics and increase probability of false alarm and miss rate of detection models.

Therefore, the need to implement DDoS detection technique and selecting proper feature is very important to identify recent DDoS threats.

1.4.1 Research Questions

According to the previous problem statement that shows major problem of existing DDoS detection techniques, the big question that this thesis is trying to answer is the following:

How can detection techniques identify up-to-date DDoS attack accurately using dynamic threshold and effective features that can be selected to decrease time complexity or cost, and using effective flows or features generation tool to improve f-score, precision, and sensitivity metrics?

This question can be divided into three sub-questions:

1. How can a detection technique generate high accuracy, in order to identify up-to-date DDoS attack using dynamic threshold?

   This question is attempted to be answered in this thesis by analyzing and utilizing a few datasets that contain various and new types of DDoS anomalies. It can also be achieved by using tools to extract features that describe attack behavior and selecting relevant features. Furthermore, the research implements a combination of techniques based on statistical approaches to detect DDoS attacks. It eliminates the dependence on using a static threshold of entropy by combining the entropy approach with the SPRT method in order to make decisions. Finally, this leads to an improved detection process, increased accuracy, and decreased time complexity or cost.

2. What are the efficient feature selection techniques used to reduce time complexity or cost and improve performance?

   This question can be answered in this thesis by examining a few techniques that are based on a combination of machine learning techniques, namely ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation. These techniques can be utilized to select relevant features and enhance the performance of the detection process. Finally, this thesis also aims to test the effectiveness of using relevant features in reducing the time complexity or cost of detection approaches.

3. What is the most effective tool used for generating flows and features from network traffics?

   This thesis attempts to answer this question by utilizing the revised version of CICFlowMeter tool to generate bidirectional flows. The tool used to extract multiple features from each flow. These generated features describe attack behavior, which aids in the identification process. The study investigates the limitations of the original version of CICFlowMeter and assesses the effectiveness of using a revised version in the detection process in term of confusion matrix metrics such as f-score, precision, and miss-rate.

1.5 Thesis Objectives

This research has three main objectives:

1. The first objective is to identify infected Ethernet interfaces and detect various kinds of up-to-date DDoS attacks, such as Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft SQL Server (MSSQL), Network Basic Input/Output System (NETBIOS), Simple Service Discovery Protocol (SSDP), and User Datagram Protocol (UDP), using dynamic threshold by implementing multiple features of entropy and Sequential Probabilities Ratio Test approach (E-SPRT).

2. The second goal is to Select relevant features in order to reduce time complexity or cost and improve performance of detection by implementing a new combination of machine learning techniques ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation approaches.

3. The third goal is to analyze problems of original CICFlowMeter and validate the effectiveness of revised version of CICFlowMeter tool on detection approach as well. This will involve comparing the results obtained from both versions for feature extraction in order to improve accuracy, F-score, and precision in the detection approach.

1.6 Thesis Contributions

The main contribution of this thesis can be summarized as the following:

- Multiple features of the Entropy and Sequential Probabilities Ratio Test approach (E-SPRT) were implemented to determine infected Ethernet interfaces and identify various kinds of up-to-date DDoS attacks, such as Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft SQL Server (MSSQL), Network Basic Input/Output System (NETBIOS), Simple Service Discovery Protocol (SSDP), and User Datagram Protocol (UDP).

- A new combination of machine learning techniques ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation methods was implemented to select relevant features in order to reduce time complexity or cost.

- Problems of the original CICFlowMeter were analyzed, and the effectiveness of revised version of CICFlowMeter tool on detection approach was demonstrated. This will involve comparing the results obtained from both versions for feature extraction to improve accuracy, F-score, and precision in the detection approach.

1.7 Thesis Scope

The scope of the thesis is limited to using two set of datasets that are publicly available online. The first dataset is the CICDDoS2019 dataset, comprising multiple datasets captured over a span of two days in 2019 [22]. It was curated by researchers from this institute and encompasses the latest DDoS intrusions, encompassing UDP, SYN, DNS, MSSQL, SSDP, UDP-lag, TFTP, SNMP, NETBIOS, and LDAP attacks. The second dataset is DARPA (Defense Advanced Research Projects Agency) intrusion detection dataset, captured by a group of researchers from MIT (Massachusetts Institute of Technology) Lincoln Laboratory during 1998, 1999, and 2000 [35], [36].

1.8 Thesis Structure

This thesis is organized as the following:

- Chapter 1- Introduction

  This chapter introduces the problem of DDoS attacks, presenting some issues with the current proposed solutions for countering these attacks. It formulates the problem statement by outlining the main question and its sub-questions. The chapter also elaborates on the motivation behind these attacks. Lastly, it outlines the contributions of this work and provides an overview of the thesis structure.

- Chapter 2- Background and Related Work

  This chapter explains the background of DDoS attacks and how they have evolved from DoS attacks, with a significant increase in the number of affected devices compared to DoS attacks. It highlights trends in DDoS attack volumes using various metrics. Additionally, the chapter outlines how DDoS attacks can be launched and discusses the different forms of DDoS attack initiation. The classifications of DDoS attacks are also

elucidated. The chapter further depicts architectures for detecting DDoS anomalies, including centralized, hierarchical, and distributed intrusion detection systems. Finally, the chapter categorizes the current techniques for DDoS detection solutions and explains the associated challenges.

- Chapter 3- Methodology

  This chapter explains the Multi-feature ESPRT technique diagram. It illustrates the feature extraction process used in the implementation and presents the categories of features, which were classified into ten groups. It also describes the feature selection methods that were used. The chapter justifies the preprocessing techniques applied to the dataset, such as cleaning, balancing, and encoding. The SMOTE technique used to balance the imbalanced dataset is also described. Finally, the chapter concludes by presenting the detection techniques used to identify DDoS attacks.

- Chapter 4- Results and Discussions

  This chapter explains the datasets and the confusion matrix metrics used to evaluate the detection technique. It presents data preprocessing techniques carried out on the dataset, such as balancing, cleaning, and transformation. It demonstrates the results of feature selection techniques. Additionally, it shows the results of combining several features selection techniques, including ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation, on the performance of the detection process. Furthermore, this chapter presents the results of ESPRT using both all the features of CICFlowMeter and a subset of important features, based on various datasets and confusion metrics. It highlights the differences between the multi-feature ESPRT and other current techniques. Finally, the chapter showcases the effectiveness of ESPRT when using a revised version of CICFlowMeter in comparison to the original version by comparing their results.

- Chapter 5- Conclusion and Future Works

  This chapter presents the conclusion of this thesis. It provides a summary of all the chapters and highlights the contributions made in the results. Additionally, it discusses potential areas for future work.

CHAPTER 2

BACKGROUND AND RELATED WORKS

This chapter provides an overview of DDoS attacks and their evolution from DoS attacks as the number of affected devices increased. It discusses trends in DDoS attack volumes and outlines the various forms and methods of launching a DDoS attack. Additionally, it explores the classification of DDoS attacks and explains the architectures of DDoS anomaly detection systems or Intrusion Detection System (IDS) based on their control mechanisms including centralized, hierarchical, and distributed intrusion detection systems. The chapter discusses many types of approaches to mitigate and identify DDoS attacks. Generally, there are two main mechanisms that have been used to defend against DDoS activities. The first mechanism is IDS, and the second is Intrusion Prevention System (IPS). Lastly, the chapter categorizes current techniques for detecting DDoS attacks and highlights their limitations.

2.1 DoS and DDoS Background

According to the Committee on National Security Systems (CNSS), a denial of service can be defined as any activity or event targeting a specific victim and resulting in preventing that victim from carrying out their work. DoS stands for Denial of Service. The first DoS attack that happened in the world occurred in 1974. It was carried out by a teenage boy at the University of Illinois Urbana-Champaign. He executed the attack in a Computer-based Education Research Lab (CERL) located at that university [16].

This lab contained many PLATO terminals that were connected and shared for learning purposes. These terminals communicated with devices outside the network using an instruction called “external.” When this instruction was used without machines connected to the terminal, it led to the terminal stopping and needing to be restarted. The 13-year-old teenager wrote a program using this instruction and routed it simultaneously to all terminals. This resulted in freezing 31 terminals that received this program. This incident occurred during an experiment, and there was no malicious intention behind it [37].

However, the first large volume of DDoS attacks that led to financial losses was recorded in 2000 by a Canadian boy who attacked e-commerce websites. During that time, many famous internet companies were brought down for several hours. These websites included Amazon, Yahoo, CNN News, Dell Computer Corp, eBay, and ZDNet [16], [38].

2.1.1 DoS Attack Types

DoS attacks are categorized into five kinds of attacks. These categories are determined based on the location where malicious activities occur. For instance, malicious events that target network devices such as routers or switches and consume data rates are classified as network layer attacks or layer 3 attacks. Other layers may also be susceptible to DoS attacks, as shown in Figure 2.1 below [16], [32].

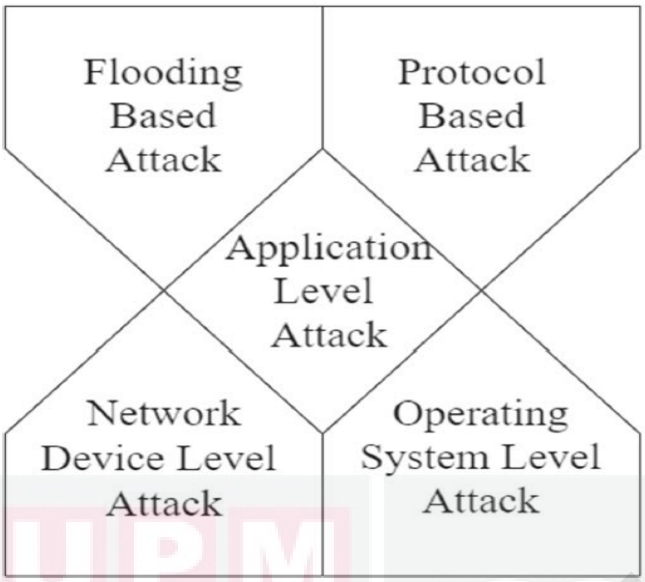


Figure 2.1 : DoS Attacks at Different Layers [16]

First of all, flooding-based attacks are a type of malicious event that involves flooding a victim with a very large number of network traffic, leading to the consumption of the victim's resources such as CPU and/or memory storage. The ping of death attack is one example of this kind of attack. It involves sending a significant number of ICMP requests toward a specific server [39].

Moreover, protocol-based attacks constitute another kind of DoS attacks. Each layer in the TCP/IP model has several protocols used to manage the transmission process among devices or layers themselves. However, these protocols have vulnerabilities that prompt attackers to engage in malicious activities. For example, Internet Protocol version 4 (IPv4) is the most common protocol used in transmission. This protocol has numerous vulnerabilities in its design. One of these vulnerabilities is that the protocol doesn't verify the correctness of the source IP address present in the IP header of a packet. Attackers exploit this weakness by generating multiple packets with forged IP source addresses. This attack is called IP spoofing, and it is

challenging to identify as it appears to originate from normal users [40]. Finally, other protocols can also be exploited based on vulnerabilities present in their designs.

In addition, application-based attacks are another form of DoS anomalies. Numerous applications operate on the devices of the targeted websites or servers. These applications may have vulnerabilities that attackers can identify. Once attackers discover these vulnerabilities, they exploit them to exhaust the resources of the targeted machine. This can result in the victim machine being stopped, frozen, or restarted [16]. Overwhelming sensors, path-based DoS, and reprogramming all serve as examples of application layer attacks [32].

Finally, operating system attacks exploit vulnerabilities within protocols designed for the operating system. Examples of operating system attacks include Teardrop and ICMP ECHO REQUEST attacks. Network device attacks represent another form of DoS attack. In these types of attacks, the attacker sends an excessive amount of traffic towards firewalls, routers, switches, or hubs [32]. Christmas tree packets serve as an example of such an attack. Attackers send numerous packets of this type, designed to take a long time to process, with the intention of overloading and crashing the router [33].

2.1.2 Transition from DoS to DDoS

With the expansion of technology, the maximum rate of information transferred across a given path and memory capacity has increased. More protective applications have been developed to address basic DoS attacks. The improvement of processor

performance and the increase in storage capacity for targeted servers have led to the minimization of DDoS impacts. Consequently, attackers have adapted their tactics due to technological advancements, giving rise to a new form of malicious activity known as Distributed Denial of Service (DDoS) attacks [16].

DoS attacks can be executed using a single device targeting a single system. However, DDoS attacks involve multiple vulnerable devices targeting a single system. These devices could number in the hundreds of thousands, coming from various networks around the world. Figure 2.2 illustrates the difference between DoS and DDoS based on their definitions [16]. Not only is the number of devices larger in DDoS attacks, but also the volume, size, intensity, and complexity of the traffic aimed at the victim are higher. For example, in Q2 of 2021, Imperva observed a 287% increase in DDoS attacks with a volume exceeding 500 Gbps, compared to Q1 of the same year [2].

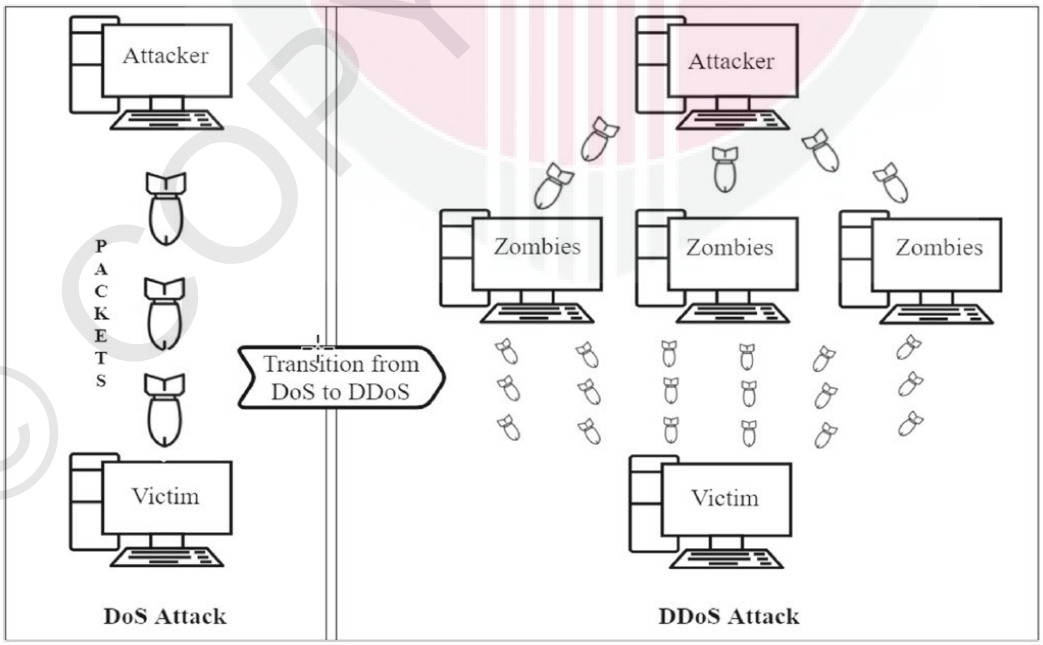


Figure 2.2 : Different between DoS and DDoS [16]

There are several differences between DDoS and DoS attacks. Firstly, a DoS attack is easy to identify because it originates from a single host. In contrast, a DDoS attack is difficult to detect as it comes from multiple devices. Secondly, DDoS attacks can be executed more quickly than DoS attacks due to their generation from multiple sources. This results in greater damage to the targeted system. Thirdly, DDoS attacks involve higher traffic volumes, flooding the targeted device with a significant amount of traffic compared to DoS attacks, which have lower traffic volumes. Fourthly, the execution methods of DDoS attacks differ from those of DoS attacks. In a DDoS attack, the attacker directs a botnet—multiple infected machines—to target a single system. On the other hand, DoS attacks utilize specific instructions to carry out the attack [13].

As mentioned earlier, DDoS stands for Distributed Denial of Services attacks. These attacks are a type of cyberattack and are highly dangerous. The objective of these attacks is to prevent legitimate users from accessing their rightful services provided by a targeted server or controller. This is achieved by recruiting vulnerable machines within the network system and compelling them to send a large number of network packets or flows towards a specific victim, such as a server, controller, or device. As a result, the targeted victim's services are disrupted and rendered unavailable to legitimate users. The success of DDoS attacks can lead to financial losses and the breach of important data belonging to the targeted system [1].

2.2 DDoS Attacks

This section discusses how to launch a DDoS attack and provides detailed explanations of strategies for performing such attacks. DDoS attacks are primarily

carried out through the use of botnets. Various forms of botnets utilized to initiate these attacks will be outlined. Additionally, this section showcases different types of DDoS attacks categorized by their attack intensity and the targets they focus on. Lastly, the section briefly lists and explains some tools used to generate DDoS attacks.

2.2.1 How to Launch DDoS Attacks

DDoS attacks can be launched using a botnet, which consists of a very large number of infected devices that are controlled by the main attacker. This main attacker is referred to as the botnet master, and it operates remotely [41], [42]. Attackers follow four primary procedures to establish a botnet network for launching DDoS attacks. These steps involve the selection of agents, compromising of bots, communication via protocols, and utilizing the botnet to carry out malicious activities [43]. These steps are illustrated in Figure 2.3.

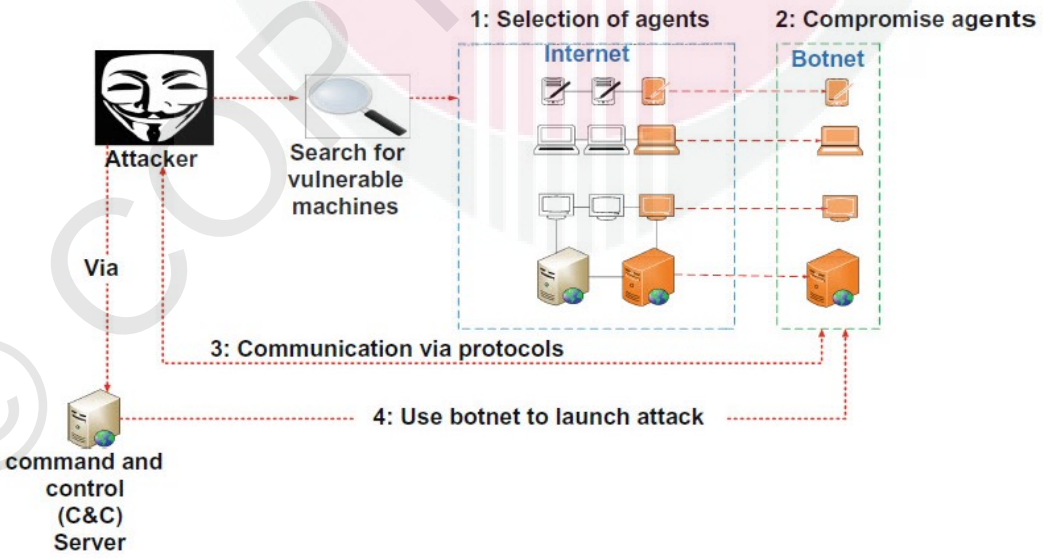


Figure 2.3 : Preparing a Botnet for a DDoS Attack [43]

Firstly, a hacker identifies potential machines within the network to serve as agents or controllers. These controllers are hacked and brought under the control of the botmaster [44]. These agents may possess significant resources that make them responsible for selecting bots [43]. These bots are also referred to as robots or zombies [45]. Secondly, agents under the control of the botmaster search for machines with vulnerabilities to serve as bots and expand the botnet. Many of these machines use Microsoft Windows as their operating system [46]. Moreover, these machines are distributed across various geographical locations, making the identification of original attacks or the botmaster challenging [47].

Furthermore, a botnet has two main goals. Firstly, it aims to rapidly and extensively increase the size of the botnet within the targeted domain by compromising a significant number of devices connected to the network. Secondly, the botnet's second objective is to collect sensitive information from its bots and transmit it back to the botmaster using well-known protocols, often over HTTP/HTTPS [48]. Several factors contribute to the strength of a botnet. A substantial number of robots within the botnet is one such factor [46]. Another factor involves the quick replacement of a detected bot with another. Network administrators struggle to identify these changes in the bot landscape and respond promptly due to various identification mechanisms [41]. The final factor pertains to the botmaster's ability to shield their bots from detection by security applications [46].

Furthermore, once vulnerable machines are identified, agents compromise or recruit them to function as zombies using either direct or indirect methods [49]. The direct approach leverages vulnerabilities present in the system and may employ malicious

software like worms, Trojan horses, backdoors, or click-and-infect mechanisms. Once any of these software types are installed, the bot connects to the botmaster and awaits further instructions. Conversely, the indirect method involves the creation of bogus websites and the dissemination of phishing emails that entice users to download and install malicious software. Upon clicking these links, victims' machines establish a connection to the botmaster [44], [49].

The third phase in preparing bots for DDoS attacks is the command and control (C&C) stage. The botmaster uses the C&C channel to communicate with their zombies, enabling the receipt of reports, transfer of malicious activities, and issuance of updated instructions. These reports are based on botmaster requests and contribute to botnet security by relaying information about infected devices [46]. Additionally, the botmaster can dispatch commands to install malicious software like backdoors, Trojan horses, and worms on targeted systems to infect them [47].

The final stage involves targeting the victim with a DDoS attack. Once the botmaster has assembled a substantial number of bots within the botnet, they direct these bots to target a specific victim by inundating the system with a vast amount of useless traffic or fabricated requests. This flood of traffic and requests is designed to consume the resources and bandwidth of the targeted system, ultimately causing a DDoS attack that can bring down the system if there is no effective defense mechanism in place [41], [42], [46], [47]. Finally, the botmaster may abandon the botnet to conceal their identity from authorities [46].

Attackers can utilize various tools and strategies to execute DDoS attacks. A severe attack results in security breaches and renders services unavailable to legitimate users. This can be achieved by overwhelming servers with an extensive volume of traffic. DDoS flooding targeting a server can be carried out through two methods: direct or indirect. Direct DDoS flooding attacks are executed by instructing a bot army to target a specific machine, as depicted on the left side of Figure 2.4 [50], [51]. Examples of this type of DDoS attack include HTTP floods, FTP attacks, HTTPS attacks, UDP attacks, TCP attacks, SYN floods, and ICMP floods [26].

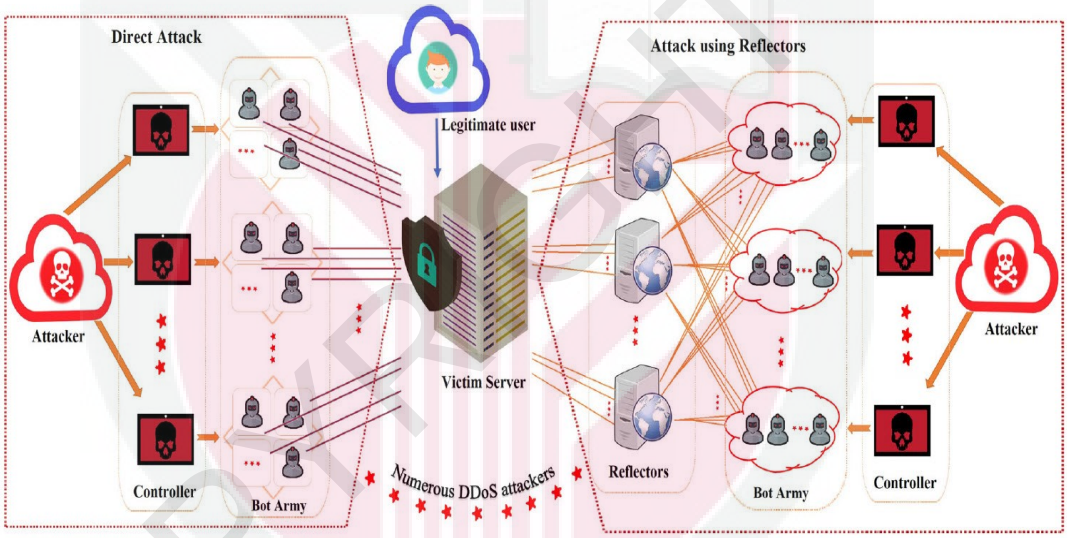


Figure 2.4 : Direct vs. Reflection DDoS Flooding Architecture [51]

Indirect flooding or amplification attacks, on the other hand, are executed when agents command the bot army to dispatch request messages to reflector servers instead of directly targeting the victim. These agents send these requests using spoofed source IP addresses to conceal their identity. Reflector servers, such as NTP servers and DNS servers, serve as examples. Subsequently, amplification servers generate a substantial volume of request messages to be reflected toward the specific

victim, effectively disrupting its operations [51], [52]. The right side of Figure 2.4 illustrates the reflection involved in DDoS flooding. Finally, Fraggle and Smurf attacks serve as examples of amplification attacks [53] .

2.2.2 Forms of DDoS Attacks Initiating

As mentioned earlier, DDoS attacks can primarily take the form of botnets. A botnet comprises compromised machines referred to as bots, which are interconnected and controlled by a hacker. The internet witnesses a significant proportion of network-based malicious activities driven by the utilization of botnets [54]. Launching a DDoS attack using bots doesn't require maximum effort for several reasons. Many of these bots are available for purchase online at low prices. Furthermore, their installation is straightforward. For hackers to initiate DDoS attacks, sending instructions to agents, responsible for distributing these instructions to the bots, is the only effort required [34].

Various forms of botnets are used to initiate DDoS attacks. Peer-to-Peer (P2P), Internet Chat Relay (IRC), Code Red Worm (CRW), and Internet-of-Things (IoT) represent well-known botnets used for this purpose. In a Peer-to-Peer (P2P) botnet, bots can communicate with each other using the P2P protocol. In this architecture, each bot can serve as a C&C server, negating the need for a single designated machine to act as such. Information updates can be easily exchanged among neighboring bots in this configuration. However, the efficiency of a DDoS attack within a centralized C&C server setup may be compromised if that particular C&C server fails to communicate with the other bots. Noteworthy examples of P2P botnets include Kademlia, Gnutella, Kelihos, and Sality [34], [54].

Furthermore, Internet Chat Relay (IRC) is a protocol that machines can utilize for real-time text messaging. The IRC botnet was developed to manage busy connections used by the IRC protocol. In this type of attack, attackers employ a C&C server to initiate a DDoS attack. Examples of IRC botnets include Agobot, Spybot, and Kaiten [26], [55].

Finally, Code Red Worm (CRW) is another botnet architecture that can be formed to initiate DDoS attacks. CRW employs worms that spread rapidly across networks and infected a very large number of machines in a short span of time. These infected machines will be as bots, targeting a specific server [26], [34]. For example, Code Red I was a worm that targeted Microsoft IIS web servers. These worms propagated through vulnerabilities of IIS server and exploited unprotected machines to create a botnet [56]. Internet-of-Things (IoT) is another kind of botnet that used to initiate DDoS attacks. Attackers focus on exploiting vulnerabilities in IoT devices such as smartphones, computers, laptops, cameras, and tablets. Finally, Mirai botnets and Hajime botnets are examples of IoT botnets [26], [55].

2.2.3 Intensities of DDoS Anomalies

DDoS attacks can be categorized into two types based on the intensity or rate of attacks: low-rate and high-rate. First of all, a low-rate DDoS attack is one type of DDoS attack. Attackers, in this kind of attack, send low rate of traffic over an extended period to overwhelm and saturate a targeted system. These attacks are very dangerous because the traffic appears similar to legitimate traffics. Detection this type of attack is challenging because it cannot be detected using standard detection applications and implementations of DDoS attack [57]. For example, attackers can

use tools like SlowDroid to carry out low-intensity attacks on vulnerable application-layer phones with limited processing capacity. This leads to overload a system by using protocols such as Slow Next [58].

In low-rate attacks, attacker can perform their activities covertly without requiring significant bandwidth. Hacker can establish several persistent connections, leading to system saturation once a threshold is surpassed. Many protocols can be used also for carrying out this attack such as TCP or HTTP [59].

However, high-rate DDoS attack is another kind of DDoS malicious activities that targeted a system with a very large number of traffics in order to overload this system with useless traffic [60]. A well-known example of a high-rate DDoS attack is the Smurf attack, which is executed by broadcasting numerous spoofed Internet Control Message Protocol (ICMP) packets or flows to devices within a network. Subsequently, these devices generate response packets directed at a specific machine within the network [61]. Attackers in this kind of attack send bursts of traffic periodically to avoid from detection and maintain a low traffic rate [62].

2.2.4 Targets of DDoS Anomalies

DDoS attacks target end network devices and links that provide services to users within the network. End devices can encompass machines or high-performance servers, which in turn offer services like banking, shopping, or education to users. Conversely, end network devices may consist of routers, switches, or gateways. In these attacks, hackers focus on the CPU, memory, and other service resources of

these devices to saturate them with DDoS attacks. This results in the disruption of these end systems, preventing legitimate users from accessing their services [63].

Network links can also fall victim to DDoS attacks. In this scenario, attackers congest these connections, overloading the available bandwidth. Consequently, the communication links between legitimate users and the targeted system are disrupted. This type of attack, which targets the links, is particularly dangerous as it can bring down all connected devices within the targeted system [26].

2.2.5 Tools of DDoS Anomalies

There is a large number of well-known tools available for generating DDoS attacks. These tools are publicly accessible online and are used by attackers. The increasing number of these online tools is indicative of the scale and complexity of these attacks. Most of these tools are simple to manage, use, and coordinate large scale of DDoS malicious. They also can run automated as well [26]. These tools can be categorized into different groups based on their purposes. For example, some tools are designed to launch DDoS attacks against specific servers, while others are meant to identify vulnerabilities in systems, applications, or protocols. There are also tools designed for network detection to identify malicious activities [64]. Some of the well-known tools of DDoS attacks are presented and explained below.

1. LOIC: This tool stands for Low Orbit Ion Cannon. It is an open-source tool, and it is publicly available on the internet. It is very simple and widely used. As soon as this tool is employed, it can initiate numerous malicious connections to a specific target using TCP, HTTP, or UDP protocol attacks. This tool does not require a specific condition to function, only the destination IP address of the targeted website needs to be known [65].

LOIC faces one main obstacle, which is the inability to hide the IP address of the hacker [64].

2. HOIC: This tool stands for High Orbit Ion Cannon. It is another open-source tool, and it is legally available for use on the internet as a stress testing tool. This tool is more dangerous than the previous tool which is LOIC. It was designed by hacktivist group Anonymous. Beginner hackers without professional experience can use this tool to carry out attacks. It requires a low number of parameters for execution and provides a fast packet controller to make malicious packets resemble normal ones. It can launch attack on 256 different targets at the same time [64]. HOIC and LOIC have some similarities [66]. For example, attackers in both tools cannot hide their IP addresses. However, there are differences between HOIC and LOIC. For example, HOIC can only be used to launch HTTP POST and GET anomalies, while LOIC can be used to launch TCP, UPD, and HTTP attacks [67]. Finally, this tool was first designed to target the United Stated Justice Department and FBI agency in 2012 [64].

3. Hping3: Hping3 is another tool that can be used for two purposes. The first purpose is generating DDoS attacks. The second goal is looking for flaws and open ports in network while testing the effectiveness of network firewall against various malicious activities [64]. In contrast to HOIC and LOIC, attackers using hping3 can hide their identity by sending traffic with spoofed IP addresses, which could either fake IP addresses or legal IP addresses belonging to other users on the network. Attackers can also adjust size of malicious packets. For example, they can fragment a packet into multiple smaller packets [68].

4. XOIC: XOIC is a simple tool used to launch DDoS attacks against a specific victim by specifying the protocol and port number. It features a graphical user interface (GUI) and is a copy of the LOIC tool. Attackers can utilize this tool to execute IRC-based DDoS attacks using various protocols such as ICMP, TCP, UDP, or HTTP. It is effective against smaller websites, but it could also be easily blocked and detected at times [64].

5. Tor's Hammer: This is another tool that can be employed to target servers. This Python-based tool utilizes the Tor network to employ fake source IP addresses. The traffic generated by this tool exhibits characteristics of normal traffic, including low size and normal rate. This enables it to bypass identification programs and consume CPU resources on the targeted system. However, it's important to note that this tool is not straightforward to use and requires technical expertise, as its GUI is quite complex [64].

6. HULK: This tool stands for HTTP Unbearable Load King. It was designed for research purposes and is used to target web servers. It is an open source, free, and user-friendly tool. This tool generates an extensive large number of HTTP GET packets that are directed towards a specific system, aiming to overwhelm and bring down the targeted system. HULK is a powerful tool because it produces dynamic traffics. These traffics have unique header that has incorrect and fabricated field. Hulk enables attackers to hide the source of the attack. Finally, e tool offers the option to halt the attack process prematurely using a safety feature [64], [69].

7. Trinity V3: This tool is used to generate various types of DDoS anomalies. Some of these attacks are UDP-based attack, while others are TCP-based. TCP RST traffic flood is one example of TCP-based attack. In this attack, all 32 bits used for the source address can be rearranged to conceal the attacker's identity [16].

2.3 Classification of DDoS Attacks

Many surveys and research studies have been conducted to present and depict DDoS attack taxonomies. For example, Mirkovic and Reiher [70] classify DDoS attack types based on various criteria, such as the degree of automation, exploitation of different weaknesses in the victim, validity of the source IP address, dynamics of attack rates, possibility of characterization, persistence of agent sets, victim types, and impact of attacks on the victim.

The degree of automation is one of these criteria, referring to how attackers initiate, infect, or recruit machines to target a specific victim. This can be further subdivided into manual, semi-automatic, or automatic methods. Exploiting different weaknesses in the victim to deny services to legitimate users is another criterion of this classification. This can be either semantic, where attackers exploit specific features or vulnerabilities in the victim's protocols or applications, or brute-force, where attackers flood the victim with an overwhelming amount of useless traffic to overload it.

Another classification criterion is the validity of the source IP address, which can be either spoofed or valid. Attack rate dynamics provide another form of classification. This can be further divided into constant rate or variable rate attacks. The possibility of characterization is another criterion in this taxonomy. Characterization can be based on protocol headers, such as IP headers and/or protocol headers, or it can be based on the actual data being used in the attack.

Mirkovic and Reiher [70] also employed the persistence of agent sets as another classification aspect in this taxonomy. Attackers can modify the set of active agent machines involved in the attack at any time. This makes the detection process difficult and hampers traceback to the original attack source. This classification can be further subdivided into constant agent sets, where there is no change in the set of machines causing the attacks, and variable agent sets, where there is a change in these machines.

Authors also considered victim types as another classification method in this taxonomy. Victim types can include applications, hosts, resources, networks, or infrastructure. Finally, the impact of attacks on the victim is another classification criterion. It can be divided into disruptive, which completely prevents legitimate users from accessing services, and degrading, which consumes a portion of the victim's resources.

Asosheh and Ramezani [71] introduced a novel classification approach for DDoS attack types based on common attacks. They categorized their taxonomy using eight characteristics: scanning methods, attack effects, attack design, level of attack vulnerability, traffic content, extent of attack distribution, automation strategy, and attack frequency or rate. The authors proposed a classification framework for DDoS defense mechanisms, which can be divided into two groups: detection and prevention. They argued that the most effective way to identify DDoS attacks is by focusing on the location of defense installation, which should be located close to the targeted system.

Bhardwaj et al. [72] proposed a classification for DDoS attack types based on a survey study of papers published from 2009 to 2015. Their classification focused on DDoS attacks within the cloud paradigm, based on four features: degree of automation, exploited weaknesses of the victim, attack rate dynamics, and impact of attacks on the victim. These features are similar to those mentioned in [70]. However, the novelty of this taxonomy was in its emphasis on analyzing parameters that contribute to effective DDoS detection. These parameters include throughput, response time, request behavior, and real-time response. Furthermore, the taxonomy

considered other parameters related to the ability to detect zero-day attacks, which are based on attackers exploiting vulnerabilities in a system before the administrator becomes aware of them.

Masdari and Jalali [73] proposed a taxonomy for DDoS attacks within the cloud computing platform. The authors explained common DDoS attacks and identified the vulnerabilities that enable these attacks to occur. Subsequently, they classified DDoS attacks based on cloud elements, such as digital versions of physical computers, enterprise-grade schedulers, web servers, virtual machine monitors, and cloud service models, including SaaS, PaaS, and IaaS. DDoS attacks in the cloud paradigm can target or disrupt various resources, including bandwidth, connectivity, information, physical components, or processes. Ultimately, the authors summarized that DDoS attacks have a more severe impact on cloud computing compared to traditional networks.

Sharafaldin et al. [22] proposed a new taxonomy for DDoS attacks that can be executed at the application layer of the TCP/IP platform. These attacks are based on the UDP and TCP protocols and are grouped into two categories: reflection-based DDoS attacks and exploitation-based attacks. Attackers in both groups of attacks use a spoofed IP source address, which is the victim's IP address, to conceal their identity. They send spoofed traffic to reflectors, and these reflectors forward responses to the targeted victim, mistaking them for legitimate requests due to the spoofed IP address. This results in overwhelming the victims with useless responses and bringing down the services provided to legitimate users.

The reflection-based attack category comprises several attacks based on TCP, UDP, or both. TCP-based attacks include SSDP and MSSQL, while UDP-based attacks encompass TFTP, NTP, and CharGen. Attacks that involve both UDP and TCP encompass PORTMAP, SNMP, NETBIOS, LDAP, and DNS attacks. On the other hand, the exploitation-based attack category includes three attack types: SYN Flood, UDP Flood, and UDP-Lag. SYN Flood, also known as neptune, is a TCP-based attack, while the other two attacks are UDP-based. This taxonomy is depicted in Figure 2.5, and the description of these attacks can be found in Table 2.1.

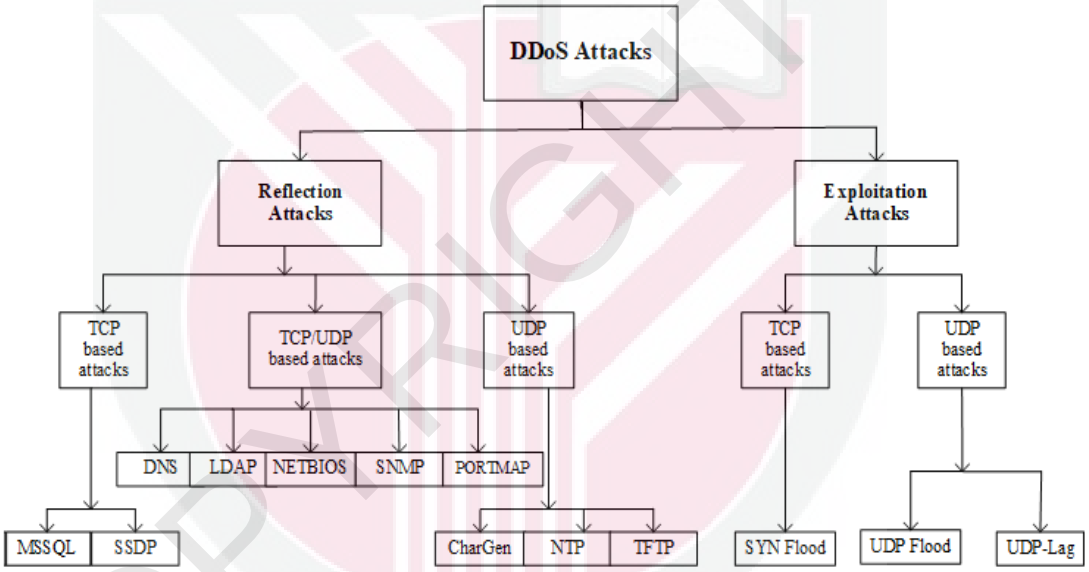


Figure 2.5 : Taxonomy for New DDoS Attacks Based on Sharafaldin et al. Proposal [22]

Table 2.1 : DDoS Attack Descriptions [21]

No.	Attack Name	Attack description	Protocol, Port number
1	MSSQL	MSSQL is based on Microsoft Server Resolution Protocol. Hackers deceive the SQL servers by sending requests which are spoofed IP address of the victim as a source. SQL servers contain number of instances which serve clients to get information that needed to establish connection with. The large number of these instances in the dataset will be used as an amplification factor toward victims. The amplification factor for MSSQL attack is 25x.	TCP, 1434
2	SSDP	SSDP is stand on Simple Service Discovery Protocol. Attackers make use of the UPnP (Universal Plug and Play) protocol to look for devices that can be used for amplification factors. Then, attackers send spoofed requests with IP address for victim as a source to all networking devices that listed in previous step. Finally, amplification factor for this attack 30x.	TCP, 1900
3	DNS	DNS is stand on Domain Name System. DNS is responsible for translating domain names into IP addresses across a DNS resolver. DNS resolver keep looking for other DNS servers if it fails find the translation in its own cache. Attackers exploit this vulnerability of DNS resolver and overwhelm targeted machine with amplified flows. Finally, amplification factor for this attack is between 28x to 54x.	TCP/UDP, 53
4	LDAP	LDAP is stand for Lightweight directory access protocol. Attackers spoofed traffic requests and exploit flaws in LDAP. They send these requests to LDAP server to generate amplification responses toward victim. Finally, amplification factor for this attack is between 46x to 55x.	TCP/UDP, 389
5	NETBIOS	NETBIOS is stand for Network Basic Input/Output System. NETBIOS is a service that can be used to share information in the network. Attackers send requests to NETBIOS servers which respond with an amplification of 2.56 to 3.58 toward target to view or alter shared info.	TCP/UDP, 137
6	SNMP	SNMP is stand for Simple Network Management Protocol. Attackers can send requests to SNMP reflectors, and these reflectors induce multiple broadband networks to produce high rate per second toward targeted machine. The amplification factor can be between 600x to 1700x.	UDP/TCP, 161
7	PORTMAP	Attackers can used port map service to send spoofed traffics to specific target. This leads to bandwidth amplification, and it could be between 7x to 28x.	UDP/TCP, 111
8	GharGen	Attackers send flows that have spoofed IP address of the victim toward devices that carry out CharGen. In its turn, these devices amplify	UDP, 19

Table 2.1 : Continued

		responses and can be reach to 358.5x of amplification factor toward victim.	
9	NTP	NTP stand for Network Time Protocol. Hackers can also overwhelm victim by exploiting NTP servers to send huge number of UDP traffic to this victim with an amplification factor of 556.9x.	UDP, 123
10	TFTP	TFTP stand for Trivial File Transfer Protocol. TFTP servers involved in this attack. Hackers send spoofed request for a file toward TFTP server, and the server will reply to targeted victim. The amplification factor can reach to 60x.	UDP, 69
11	SYN Flood	Attackers exploit the vulnerabilities in TCP three-way handshaking. They send very large number of spoofed SYN flood toward to server and leave the connection open. Then, server will forward replies (SYN/ACK) to victims and wait for (ACK) to close the connection. However, victim will not response and will be overloaded with useless packets.	TCP
12	UDP Flood	Attackers send very large number of spoofed UDP flood to random ports which lead to overload capacity of the victim to response.	UDP
13	UDP-Lag	Attackers in this attack is trying to interrupt the connection between legitimate users and specific server. These attempts can be done especially on internet gaming to slow down or break the connection between players online.	UDP

Salim et al. [20] proposed a taxonomy that groups DDoS attacks into two main types: bandwidth attacks and network resources attacks. In the ideal situation, malicious flows are dropped, while benign flows are allowed to pass through. When flows are dropped, legitimate users begin to lose their chances to connect with the targeted machines. On the other hand, attackers would exploit this phenomenon to intensify their efforts by sending a very large number of malicious flows toward victims. This leads to a decrease in CPU resources of victims and denial of services to all users.

DDoS attacks impact not only the resources of the targeted devices but also the network bandwidth of the targeted system. In these kinds of attacks, attackers aim to

consume the available resources of victims and prevent legitimate users from accessing these resources. A summary of these DDoS malicious attacks is shown in the Figure 2.6.

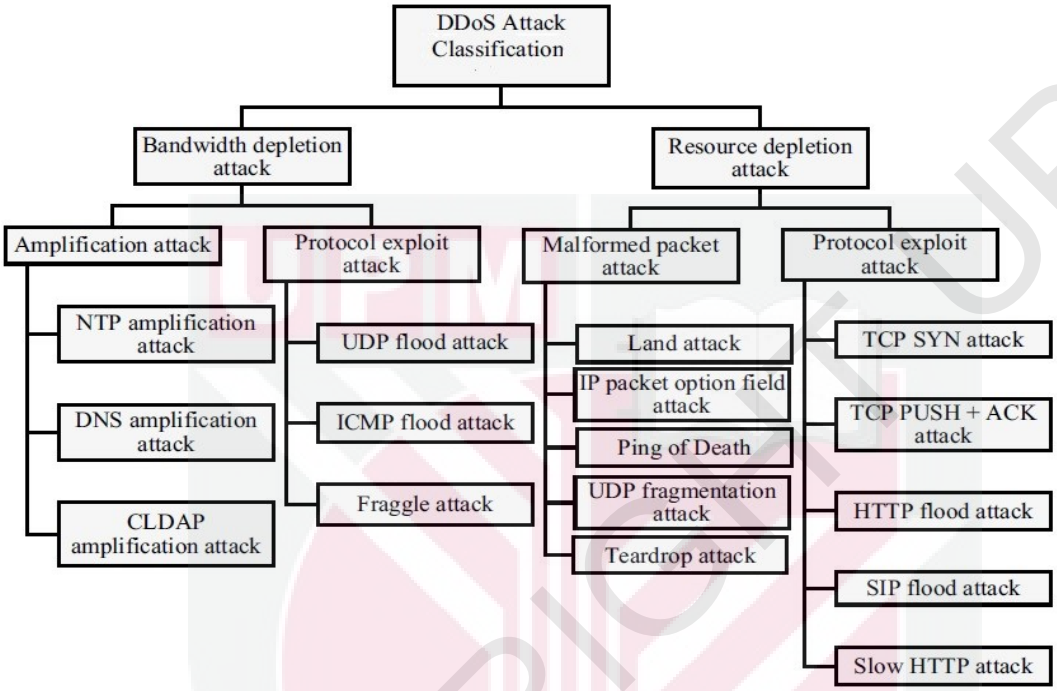


Figure 2.6 : DDoS Attacks Classification Based on Salim et al. Proposal [20]

2.3.1 Bandwidth Depletion Attack

Bandwidth depletion attack is the first main kind of DDoS attacks. This kind of attack aims to consume the server's bandwidth by utilizing malicious flows. It can be executed in two ways: either through protocol exploitation or amplification.

2.3.1.1 Protocol Exploitation

Attackers exploit vulnerabilities present in the victim's protocol system to consume their network resources. This attack can occur either in the network layer, such as the

Internet Control Message Protocol (ICMP), or in the transport layer, such as the User Datagram Protocol (UDP).

UDP flood attacks are one way that attackers can exploit to perform DDoS attacks. In this attack, intruders send the IP address of the targeted machine, the duration of the attack, and the method to execute the DDoS attack to the botmasters, which are vulnerable devices [74]. The botmasters then execute the instructions and send many UDP packets toward the victim machine with a spoofed IP address as the source to conceal the attackers' identity. The victim's machine will respond with ICMP responses to the botmasters, but the botmasters will not reply [75]. As a result, the victim's machine will slow down and crash due to the lack of response from the botmasters and the influx of a larger amount of traffic.

ICMP flood is another method that attackers can use to consume network bandwidth. In this attack, intruders send a very large number of ICMP echo traffic toward broadcast stations. They send these traffics with spoofed IP addresses to hide their identity. These broadcast stations generate ICMP echo reply traffic toward the victim device. The larger the number of broadcast stations involved in this attack, the shorter the amount of time it takes to slow down victim devices and crash them. This attack is also known as a Smurf attack [76], [77].

Fraggle attacks are the last methods that can be used by attackers, which are based on protocol exploitation in this classification. Fraggle attacks send a very large number of spoofed UDP echo traffic toward broadcast stations. The broadcast station will replicate or reflect the received traffic toward infected victims. These attacks are

similar to ICMP or Smurf attacks in sending traffic to the broadcast address. However, ICMP flood sends ICMP echo traffic while Fraggle attacks send a UDP flood to the broadcast stations [78], [79].

2.3.1.2 Amplification Attack

Attackers transmit small traffic sizes of few bytes to a reflector or amplification server. In turn, the server or reflector sends a very large amount of traffic toward the infected device in order to consume its bandwidth. NTP amplification attack, DNS amplification attack, and CLDAP amplification attack are some examples of this kind.

NTP amplification attack is based on the Network Time Protocol attack. This protocol can communicate and set the timer with the server by synchronizing the system clock. Attackers exploit the NTP UDP protocol to transmit amplification traffic toward the targeted host. First, intruders send a request packet called a MONLIST request to the NTP server. This request is very small at 64 bytes. Then, the NTP server amplifies the traffic called MONLIST or MON_GETLIST reply with a list of almost 600 devices that have communicated with the infected machine. This makes NTP a better choice for attackers to amplify their malicious [80], [81].

DNS amplification attack is based on the Domain Name System attack. Attackers send a DNS lookup echo to the DNS server using a spoofed IP to hide their identity. The DNS server responds with a reply to the victim. When attackers send a request to the DNS server, they ask for an "ANY" echo message to the DNS server. The

DNS server, in turn, will amplify the received echo and transfer as much data as possible to the victim [82], [83]

CLDAP amplification attack is based on the Connectionless Lightweight Access Protocol attack. It uses the UDP protocol to transfer a request message to the CLDAP server. The server amplifies the request and sends almost 46 to 55 times the original echo message to the infected machine with a spoofed IP [84], [85].

2.3.2 Resources Depletion Attacks

Resource depletion attacks are the second main kind of DDoS attacks in this classification. This kind of attack aims to consume the server's CPU, RAM, and sockets by sending malicious flows. It can be done in two ways: either by protocol exploitation or by amplification.

2.3.2.1 Protocol Exploitation

Attackers exploit vulnerabilities that are available in the victim's protocol system in order to consume their CPU, memory, or socket resources. Many protocols can be exploited for DDoS attacks, such as the TCP protocol (based on Transmission Control Protocol), the HTTP protocol (based on Hypertext Transfer Protocol), and the SIP protocol (based on Session Initiation Protocol).

First of all, TCP SYN attacks exploit vulnerabilities in the TCP protocol. TCP requires sending three packets between the client and server to establish a connection. These three packets are part of a three-way handshake. The first packet, known as the SYN packet, is sent from the client to the server. The server responds

with a SYN/ACK packet, and the client acknowledges with the third packet, known as the ACK packet, to confirm receipt of the second packet [86]. Attackers send spoofed SYN packets to the targeted server. The server responds with SYN/ACK and waits for an ACK packet that never arrives, causing the connection to remain incomplete. The server holds the state of the attacker's packet in its memory until the connection is either established or times out. Attackers exploit this weakness by flooding the server with a large number of SYN packets, leading to the exhaustion of server resources and forcing the server to drop SYN packets from legitimate users [87].

Moreover, the TCP PUSH and ACK attack is another type of DDoS attack related to the TCP protocol. As previously mentioned, TCP employs a three-way handshake between the client and server to establish a connection. One of these requests is the ACK packet sent from the client to the server to establish the connection between them. Attackers send a very large number of ACK or PUSH requests that do not belong to any of the open sessions listed in the server's connections. The server attempts to identify which session these incoming ACK requests belong to. This process causes the consumption of server resources, such as CPU, memory, or RAM, and disrupts normal requests [88].

In addition, the HTTP flood attack is a type of DDoS attack that involves forming bots to attack the server. Attackers attempt to discover vulnerabilities in devices within the network and increase the number of infected machines to expand the scope of DDoS attacks [88]. Identifying infected devices to be used as bots can be done through two methods: HTTP GET [86] or HTTP POST [87]. Both of these

methods consume CPU and memory resources of the server and prevent legitimate users from accessing available services.

Furthermore, the SIP Flood attack is a type of DDoS attack that targets the victim server, known as the SIP REGISTRAR, with the aim of consuming server bandwidth and resources. SIP, based on the Session Initiation Protocol, is commonly used for Voice over IP (VOIP) services. Attackers send various kinds of requests to the server, such as SIP INFO, SIP INVITE, SIP RE-INVITE, or SIP REQUEST [20], [89].

Finally, the Slow HTTP attack is another kind of DDoS attack in which attackers send small packets slowly. Attackers send a portion of the HTTP request packet and then transmit the remaining part at intervals to maintain an open session. An example of this attack is Slowloris, which aims to consume the targeted victim's resources, such as CPU and RAM. This occurs by keeping the connection between the attacker and targeted server open for as long as possible and sending a large number of traffic requests [90].

2.3.2.2 Amplification Attack

Attackers send small-sized traffic with little bytes towards a reflector or amplification server. The amplification server then reflects a significantly larger amount of traffic towards specific infected devices to consume their resources, such as CPU, memory, or socket. Land attack, IP packet option field attack, ping of death, UDP fragmentation, and teardrop attack are examples of this category.

First of all, the Land attack is one of the causes of DDoS attacks. Attackers send packets to the victim server with the same source port address and destination port address. This technique puts the packet in an infinite loop by replying to itself. This increases the CPU load and subjects the server to DDoS attacks, as demonstrated by the Nagios tool mentioned in [30].

Additionally, the IP packet option field attack is another kind of DDoS attack that depletes system resources. Attackers make the value of the option field in the packet header unpredictable and random. For instance, when attackers set the quality of service (QoS) bits to 1, they force the system to spend more time processing and handling the incoming packets. Consequently, attackers send multiple traffic of this kind to engage in malicious activities and bring down the system [20].

Furthermore, the ping of death is a dangerous attack that occurs when attackers send larger packet sizes over IP, such as ICMP echo packets. The maximum packet size is 65,535 bytes, including the payload and header size of the packet. As sending a packet larger than the maximum amount violates the IP transmission standard, the transmitted packet is fragmented into smaller packets. These packets are reassembled at the receiver side. Attackers exploit this by sending multiple oversized traffic packets to the server. This leads to incorrect fragmentation, which is dropped by the server, resulting in buffer overflow and system crash [20], [91].

Moreover, UDP fragmentation is similar to the previous attack, but attackers send packets over UDP instead of IP protocol packets, such as DNS requests [92]. This

increases the CPU load and wastes memory resources. It also leads to DDoS attacks and prevents legal users from accessing available services on the targeted server.

Finally, the teardrop attack occurs due to errors in reassembling fragmented packets. The fragment offset is a field in the IP header that helps identify the starting point of the fragmented packet. When the addition of the fragment offset and one fragmented packet differs from the addition of the next fragmented packet and the fragment offset, the packets overlap. This leads to a teardrop attack and crashes the system [93].

2.4 Detection of DDoS Anomalies Architectures

DDoS attacks have evolved due to novel ways, tools, and strategies used by attackers. Therefore, detection techniques must be robust to effectively and promptly identify these attacks. When a detection system identifies DDoS anomalies at the beginning of an attack, the targeted victim can mitigate the effects and recover early. Severe consequences may lead to various difficulties and complications. Therefore, Intrusion Detection Systems (IDSs) can be employed to identify anomalies, including DDoS attacks, by performing several steps [16]. Figure 2.7 shows the position of IDSs in the network.

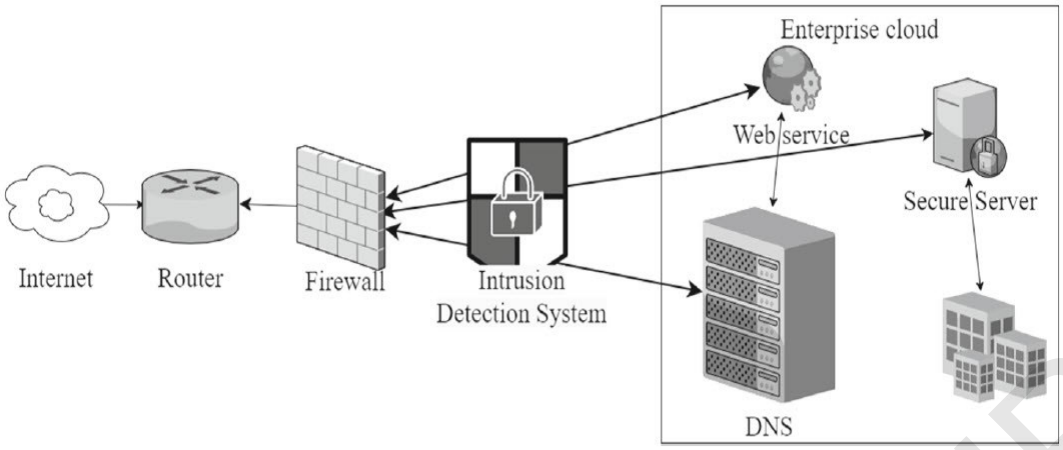


Figure 2.7 : Position of IDSs in the Network [16]

IDS can be classified into three main types based on their control mechanisms: Centralized IDS, Distributed IDS, and Hierarchical IDS are categories of IDS [93]. Each architecture has its advantages and drawbacks, and administrators can choose an appropriate design that fits their implementations.

2.4.1 Centralized IDS

In this kind of IDS architecture, one node becomes a central node that receives information from other nodes connected to it. The central node monitors network behavior and tests incoming network packets. Notifications pop up on the central node when there is suspicious behavior. Each node observes incoming packets and transfers reports to the central node in order to make a decision. Since information has been gathered and analyzed, a well-informed decision can be made using this method [94], [95].

However, attackers can bring down and hack the central node if the security application installed on the central server is weak. This is why robust security should

be installed on the central node to protect the entire network [96]. This architecture can be deployed on small networks with low scalability. Network traffic has to be transferred to the main central machine. When the number of machines increases in the network, traffic congestion may occur on the central server [95]. Figure 2.8 (a) shows the architecture of a centralized IDS.

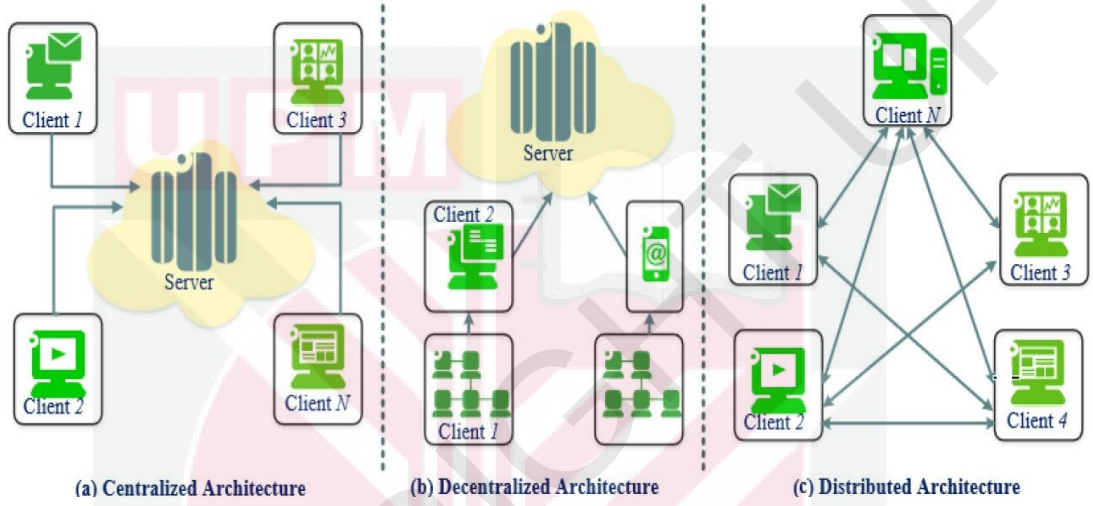


Figure 2.8 : Various Architectures for Deployment of IDS [95]

2.4.2 Hierarchical IDS

It is also called a Decentralized architecture. In this identification system, the overall architecture is divided into subsections. These levels are segmented based on multiple factors, such as the operating system used in the system, the location of the system, and/or the monitoring authorities of the system. A central system for each level collects network traffic and generates alerts when suspicious activities are detected. Each layer transfers information to the layer above it. In turn, the upper layers transfer information upward until reaching the central server for processing [94].

This method is more scalable than the previous method because there is a central server for each level, which reduces system congestion [16]. Ahmed et al. used a Decentralized IDS that merges multiple different methods. The first and second methods are rules-based concepts and decision trees. The third method is based on using a machine learning algorithm [97]. Finally, Martin and Jan also implemented a Hierarchical IDS using a knowledge model and machine learning algorithms to identify novel threats and existing attacks [98]. Figure 2.8 (b) depicts this kind of IDS detection method.

2.4.3 Distributed IDS

Distributed IDS is another type of intrusion detection system. In this identification method, the control is distributed among all machines in the network instead of using a single central server [94]. The detection method is independent, but it cannot aggregate information and reports from other systems. The drawbacks of this approach include fault tolerance, lower identification accuracy, and difficulties in load balancing [96]. By using a distributed IDS, network-based identification systems, host-based detection systems, and hybrid detection systems can be employed. Manish and Ashish suggested a Distributed IDS using the blockchain technique applied to a cloud architecture [99]. Figure 2.8 (c) shows the implementation of distributed IDS.

Intrusion detection systems can be categorized into three types based on the location of the defense system: host-based detection systems, network-based identification systems, and hybrid detection systems. First of all, when a defense system is located and deployed on a single host or local system, this is called a host-based IDS. The

detection system in the host system is responsible for examining and collecting data coming to or from that host. This data includes the behavior of the system and predefined signatures of malicious activities [96] [100]. IDS can then take action and prevent attacks based on the collected information.

Host-based IDS can be implemented in many methods. Checking the integrity of transferred files, host monitoring, and login credentials are some examples of these methods [16]. These logs can be collected from many sources. Operating systems' networks, such as Windows NT, Windows XP, Windows 2000, and/or Unix, are one source. Another source is security applications like firewalls. These logs can also be gathered from networking devices such as switches, routers, webservers, and gateways. Networking protocols can also serve as another source to collect logs, such as the FTP protocol [96]. Host IDS stores a history of system states that contain suspicious events and stays updated with operations, memory, and device artifacts [101]. These suspicious activities could include segmentation fault errors, information changes, unauthorized access, system software crashes, and/or lack of permission to use system resources [102]. However, the host IDS is not a perfect solution because it has some drawbacks, such as decreasing system performance due to high consumption of its resources [95]. The host-based IDS network structure is presented in Figure 2.9.

Network-based IDS is a group of machines or systems that are deployed in specified locations of the network. Network-based IDS is responsible for examining the entire network environment and the data communication passing through these systems [103]. All information coming in and out of the network is monitored. Hackers

might or might not have an idea regarding the availability of these systems. These systems should be installed at effective points in the network to monitor malicious activities [16]. However, network-based IDS face problems in monitoring each packet when network traffic transferred in the network is very large and massive. These challenges arise from decoding encrypted data and defining virtual private networks (VPNs) [95]. Therefore, sensors for the network should be installed and used to deal with a massive amount of information and improve IDS efficiency [96].

Many researchers have used network-based IDS to monitor the network. For example, Basant proposed a method that used stochastic gradient descent with convex Logistic Regression cost functions. They tested their proposed method on the UNSW-NB15 dataset and found that their method has a higher detection rate [104]. K. Rahul et al. reviewed all NIDS and HIDS-based methods that used deep learning techniques and neural network algorithms in their implementations [105]. Figure 2.9 shows the differences between NIDS and HIDS. Finally, a hybrid detection system combines the purposes of both NIDS and HIDS to increase the detection of malicious activities and create a more flexible structure.

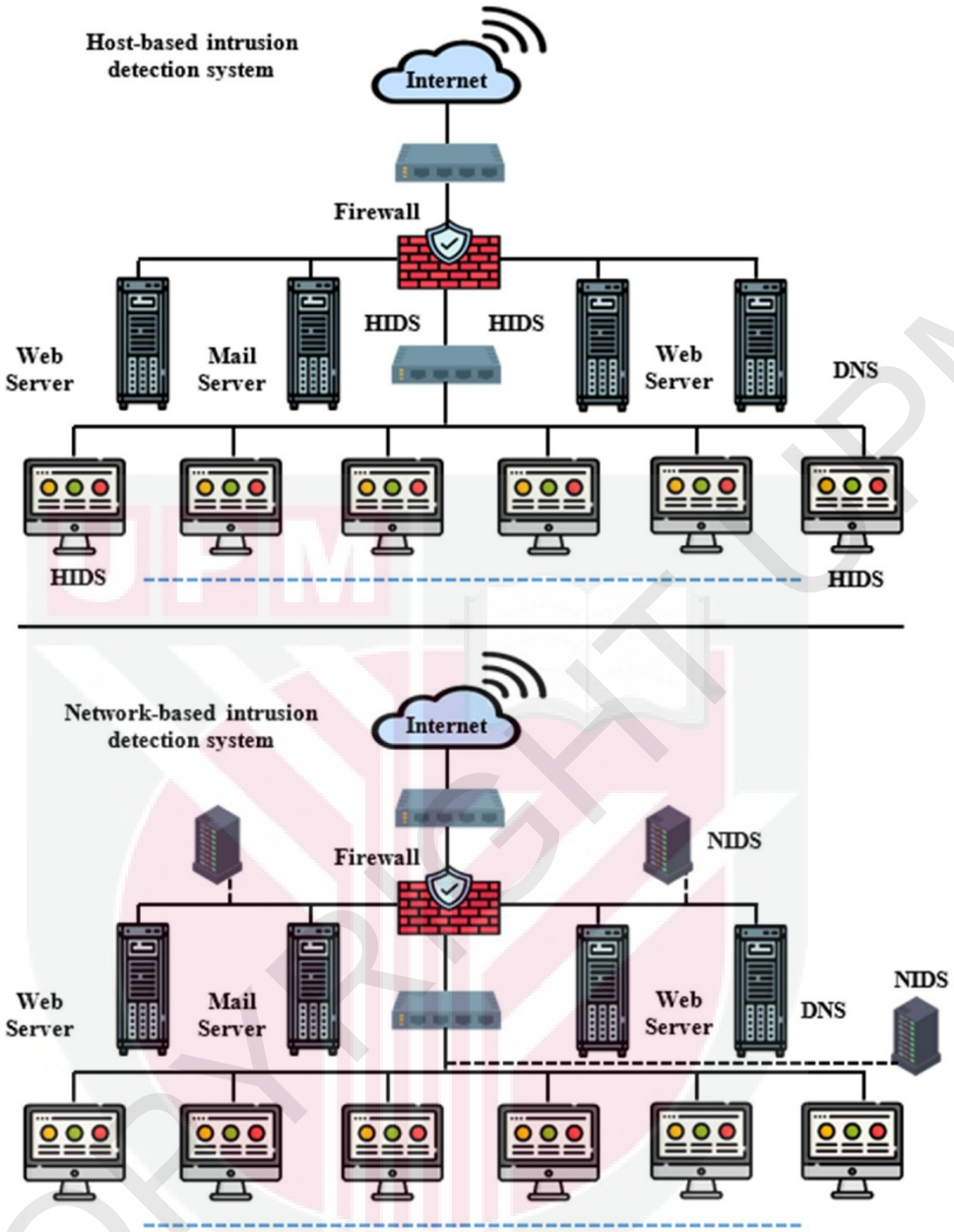


Figure 2.9 : Comparison between HIDS and NIDS [100]

2.5 Network Traffic Analysis Tools

In traffic analysis, specific functionalities are essential to accomplish particular tasks. For instance, creating a dataset involves functionalities such as collecting, extracting, and analyzing traffic data. These functionalities may sometimes be

integrated into a single tool or executed using a collection of different tools. This section examines the various network traffic analysis tools currently in use.

A flow was identified as an equivalent to a connection that streaming data and information between two points. Flow contains group of packets that have information from different layers such as network or transport layer. A flow may be a collection of packets that share similar packet fields such as IP source address, IP destination address, port source address, port destination address, protocol type, and more. These flows include packets moving in both forward and backward directions as part of a single flow.

There are many tools can be used to analyze network traffic. Some of them can also be used to capture and collect incoming packets and group these packets to form flows. Other tools have the capability to extract features from generated flows. Feature extraction is the process of transforming raw data into features that encapsulate relevant information about the data they represent. These features help to analyze incoming flows. These features may be number of packets, number of flows, direction of flows, protocol kinds, length of flows, and etc. Among these tools identified in this review, CICFlowMeter [22] , nProbe, Yet Another Flowmeter (YAF), Argus

First of all, CICFlowMeter was designed by The Canadian Institute for Cybersecurity (CIC) [22]. This tool is responsible to aggregate bidirectional flows from packets. Each flow contains a group of packets that share same properties. These flows can occur in both directions depending on packets direction which can

be either forward, from source to destination, or backward, from destination to source. It also can extract 82 different features from each generated flow. All of these features and flows are stored in Comma-Separated Values (CSV) file. Finally, it provides ability to monitor and analyze generated flows and features.

On the other hand, CICFlowMeter has several problems that impacted the performance of detection approaches. These problems primarily arose with flows based on the TCP protocol. Setup timer and closing the connection violate the TCP specification and have several implications for flow formation.

In addition, nProbe [106] is another tool that was developed by ntop company. It is an application available as an embedded system or stand alone. It uses two important and common network protocol in its design which is IPFIX and NetFlow. It can serve as flow capture, aggregation, extraction features from flows. It can store collected information on storage in order to be used later on for investigation furthermore. It is not totally free because it allows only to export twenty-five thousand flows for free.

Furthermore, Softflowd [107] is another tool used for generating flow from packets. It can also be used to analyze flows and extract features from flows. It is compatible with IPFIX and NetFlow protocol in its design. It can capture packets from the network ethernet or read captured file using the Libpcap library. Libpcap is freely C++ library to collect and filter network traffics. Although this tool does not function as a collector, it can send aggregated flows internally and summarize them for analysis. Finally, it is designed on OpenBSD and Linux.

Moreover, YAF [108] is another tool which is stands for Yet Another Flowmeter. It was designed by Computer Emergency Response Team (CERT) department which is located in Carnegie Mellon University. It is compatible with IPFIX protocol, and it can collect network traffic in real time from the interface or even offline from a Pcap file in order to form bidirectional or unidirectional flows. Finally, YAF can do deep packet inspection (DPI) which is process of capture and export network traffic in order to monitor and examine these traffics carefully [109].

Finally, there are numerous of flow generation tools with different capabilities. Most of these tools such as YAF and nProbe were built for specific target such as standard employment or accounting. This leads to handle only fixed or specific purposes, decrease flexibility, and support limited number of functions. For instances, supporting statistical operation such as average and standard deviation is typically unavailable or very limited. nProbe and Softflowd also has complex design comparing with other tools in term of CPU load. In term of average packets size and generating number of packets, YAF generate least number of packets traffics with largest size comparing with other tools mentioned above [109]. Finally, Table 2.2 shows the differences among these tools.

Table 2.2 : Differences among Some Network Traffic Analysis Tools

No.	Tools	Pros	Limitation (Cons)
1	CICFlowMeter	1. monitor and analyze generated flows and features. 2. aggregate bidirectional flows from packets. 3. extract 82 different features from each flow.	has several problems with generation of flows based on the TCP protocol.
2	nProbe	flow capture, aggregation, extraction features from flows.	1. It is not generally freely available. 2. complex design in term of CPU load.
3	Softflowd	1. capture packets from the network ethernet or read captured file using the Libpcap library. 2. extract features from flows.	1. this tool does not function as a collector. 2. complex design in term of CPU load.
4	Yet Another Flowmeter (YAF)	1. collect network traffic in real time from the interface or even offline from a Pcap. 2. deep packet inspection (DPI).	generate least number of packets traffics with largest size.

2.6 Feature Selection Techniques

Feature selection is crucial during detection of anomalies process for several reasons. It helps eliminate meaningless, irrelevant, and redundant features, thereby reducing detection errors and enhancing the accuracy of the identification process. These techniques can help to delete unimportant and repeated features. By reducing the number of features, the computational load and execution time are also minimized. Many techniques are implemented using machine learning or deep learning algorithms in order to do that. ANOVA, Random Forest, Extra Tree, and Pearson correlation are some of these techniques based on machine learning.

2.6.1 ANOVA

ANOVA stands for **an**alysis **o**f **va**riance. ANOVA is a method used to select the optimal features that lead to increase accuracy of predication and decrease execution time. It is simple, powerful, and robust to most against most of its expectations-related problems. It examines the response of two or more groups of instances of features under different circumstances that can be detected by two or more classification of instances with respect to the output. In other words, it can be used effectively when the number of instances of features differs. It can be used also to investigate the relationship and examine the difference of average between two or more groups with respect to the output [110], [111].

ANOVA employs two hypotheses, denoted as H0 and H1. H0 assumes that the variances of all groups of categorical features with respect to the output class are the same. Conversely, H1 assumes that the variances of at least one group or more of categorical features with respect to the output class are different. If the variances of all groups of a categorical feature adhere to the H0 assumption, then this feature has no effect on the detection model. However, if the variances of at least one group or more of categorical features adhere to the H1 hypothesis, then a feature has an impact on the detection approach. The principle of ANOVA revolves around the use of the F-score. The F-score is the ratio of variance among groups to variance within groups of a feature. Variance among groups is also known as mean square among (MSAm), while variance within groups is called mean square within (MSW). This can be expressed using the following equation [111]:

$$\text{F score(f)} = \frac{\text{MSAm (f)}}{\text{MSW(f)}} \tag{2.1}$$

MSAm is the ratio of the deviation of each group (gi) from the average of all groups (G) and the number of observed groups (NG). However, MSW is the ratio of the deviation of each instance (si) from the average of all instances within a group (S) and the number of instances within each group (Ns). The F-score equation is as follows [111]:

$$\text{F score (f)} = \frac{\sum_{i=1}^{f}(gi - G) \Big/ (Ng - 1)}{\sum_{i=1}^{f}(si - S) \Big/ (Ns - 1)} \quad (2.2)$$

Finally, where (f) is the number of features that need to be tested. The F-score will be calculated for each feature, and they will be ranked based on their weight. When the F-score is high, H1 is accepted for that feature because it indicates that the feature has an impact on the detection approach. However, when the F-score is low, H0 is rejected for that feature because it suggests that the feature does not have an impact on the detection approach.

2.6.2 Random Forest

Random Forest is another technique used in feature selection. It comprises multiple decision trees that are generated during the training process. The reason for using a combination of multiple trees is that it is better than using a single tree for learning algorithms. Random Forest randomly builds new subsets from the original dataset. It performs random sampling with replacement to create these new subsets. In other words, a row can occur more than once in the new dataset. Each new dataset contains the same number of instances as the original dataset. This process of generating new subsets with these specifications is called bootstrapping [112].

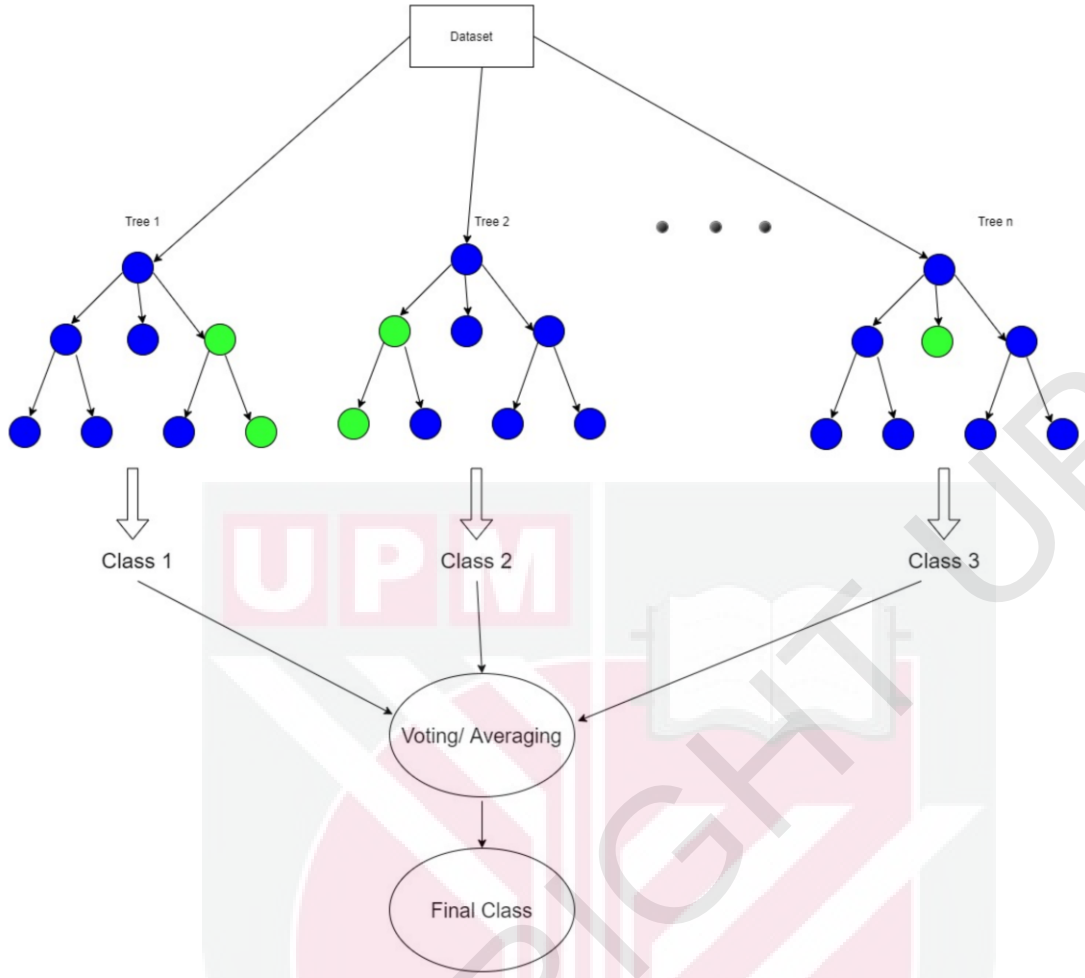


Figure 2.10 : Random Forest Steps [112]

Random subsets of features are selected for each new dataset. The combination of the randomly generated new datasets and features is used to build decision trees for training. Random Forest selects the best split at each leaf for all trees. Feature importance is calculated per decision tree by determining the decrease in impurity using entropy or the Gini method. The final feature importance score is calculated by averaging the entropy or Gini values across all trees. These values are then sorted in ascending order. Features with higher importance scores have priority in the selection process. The process of the Random Forest technique is explained in Figure 2.10. The Random Forest technique works with large datasets that have

different data types, such as categorical or numerical. On the other hand, the Random Forest technique takes a longer time because it involves a large number of trees. Finally, another potential problem with this method is that it may introduce bias [113].

2.6.3 Extra Tree

Furthermore, Extra Trees is another feature selection technique used in the implementation. Extra Trees work similarly to Random Forest in creating multiple decision trees. However, Extra Trees use the whole dataset without employing bootstrapping, which is used in Random Forest, to generate trees. It utilizes the original dataset along with random subsets of features to create these trees. Feature importance is calculated using Gini or entropy, and the final feature importance scores are combined through majority voting or arithmetic average. Finally, Extra Trees is considered better than Random Forest because it reduces overfitting and bias. It also improves the generalization performance of the algorithm and reduces execution time [114].

2.6.4 Pearson Correlation

Finally, Pearson correlation is another technique that was used for feature selection. It is employed to select the best feature based on the correlation between a specific feature and the target label. This technique measures the linear relationship between two sets of data. It generates values that range between 1 and -1. When correlation values are close to 1 or -1, it indicates that features are highly correlated with the target label. Conversely, when these values are close to 0, it suggests that features are not highly correlated with the target label. The formula for calculating the

correlation between a feature (C) and the target class (D) is as follows [115], where n is the number of features:

$$\text{Pearson Corr}\,(C, D) = \frac{n\sum CD - (\sum C)(\sum D)}{\sqrt{[n\sum C^2 - (\sum C)^2][n\sum D^2 - (\sum D)^2]}} \quad (2.3)$$

2.7 Preprocessing: SMOTE Technique

Preprocessing is an essential step before applying detection techniques. It prepares the dataset for building and training identification methods. Preprocessing addresses errors such as missing, duplicate, irrelevant, or incomplete data values, which can be introduced by machines or users. This process improves dataset consistency and reliability, resulting in higher accuracy and lower error rates in the outcomes. There are numerous data preprocessing techniques available such as cleaning, transformation, and balancing.

One of the preprocessing steps involves balancing the dataset if it is imbalanced. An imbalanced dataset occurs when the distribution of classes in the dataset is unequal. In other words, there is a significant difference in the number of instances between the normal class and the malicious class. An imbalanced dataset poses several challenges for detection and classification algorithms. During the training process, less time is allocated to learning from the minority class instances, as the focus is primarily on the majority class. This results in biased decision-making for detection algorithms, which struggle to classify the minority class instances. Another issue associated with imbalanced datasets is overfitting. This phenomenon arises when detection algorithms perform well and generate accurate results on the training dataset, but fail to perform on new testing datasets. These challenges collectively

lead to decreased accuracy and reduced performance of detection algorithms. To address these issues, the SMOTE technique, standing for Synthetic Minority Oversampling Technology, is employed. This technique was introduced by Chawla et al. as a solution to handle imbalanced data [21], [116], [117].

SMOTE is an efficient method for balancing the number of instances in both majority and minority classes. SMOTE achieves this by introducing new instances to the minority class, thereby increasing the number of instances in that class. These new instances are not duplicates of existing minority class instances; instead, they are synthetic. The generation of these synthetic instances involves a random process that occurs between instances of the minority class and their neighbors. This random generation is achieved using the k-nearest neighbors' algorithm (KKN), which assesses the proximity of instances in the characteristic space to determine similarity.

The scenario of the SMOTE technique unfolds as follows: First, the minority instances X(M) are identified from the dataset X. Second, the K nearest neighbors are determined for each instance within the minority X(M). Third, a specific group of data points, denoted as Y, is randomly selected from the K nearest neighbors, forming a vector of data (Y1, Y2, …, Yn). Fourth, a correlation formula is applied between the data in the vector (Y1, Y2, …, Yn) and the minority instances X(M). This correlation is then multiplied by random data ranging from 0 to 1 to generate new synthetic data S, as illustrated in Equation (2.4) below [117]:

$$S_n = X(M) + \text{random}(0,1) \times (Y_n - X(M)) \text{ where } n=1, 2, \ldots, N \quad (2.4)$$

The value of N can be determined by calculating the degree of imbalance between the class with low instances and the class with high instances. Finally, if the imbalance degree is not equal to the number of new synthetic data, steps two to four are repeated to generate additional synthetic data. These newly generated synthetic data points are then added to the minority dataset X(M) to balance the entire dataset X. These steps are illustrated in Figure 2.11.

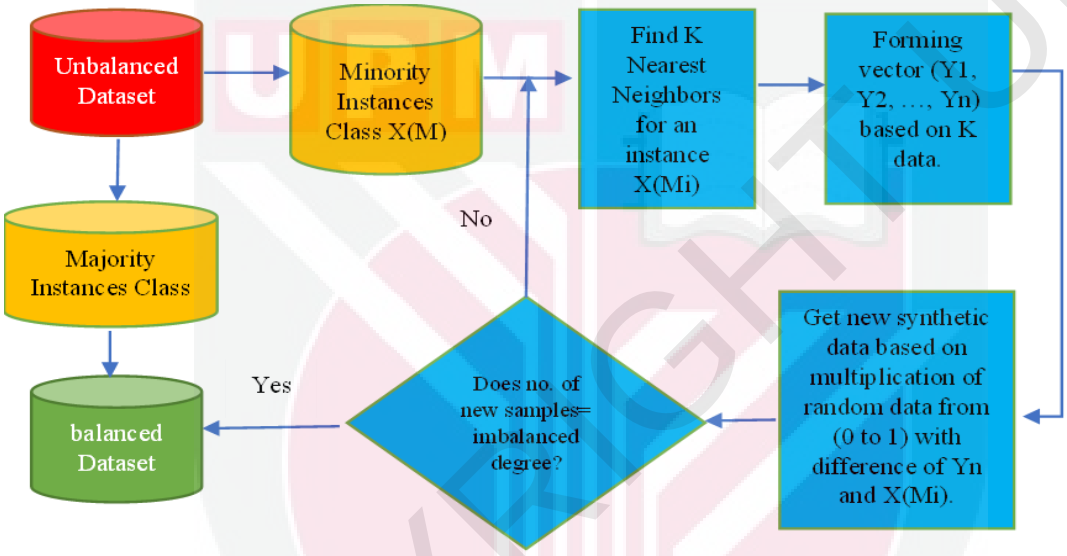


Figure 2.11 : Process of SMOTE Technique

2.7 Categories of DDoS Attacks Detection Techniques

Several methods have been developed to detect and identify DDoS attacks in their early stages. This section will discuss many types of methods designed by renowned researchers to mitigate and identify DDoS attacks. Generally, there are two main mechanisms that have been used to defend against DDoS activities. The first mechanism is Intrusion Detection System (IDS). IDS is responsible for monitoring network traffic and identifying any suspicious behaviors that could lead to the theft of important information from the network. These suspicious behaviors could

include port scanners, malware, or cybersecurity violations. The IDS is related to both software and hardware that are responsible for recording and identifying malicious activities only. IDS does not independently make decisions on what actions to take. Instead, it requires the administrator's interaction or review of the reports generated by IDS to determine the appropriate actions [14].

The second mechanism is the Intrusion Prevention System (IPS), which is similar to IDS in identifying malicious traffic. However, IPS is a control system that takes action in response to malicious behaviors. It can deny or drop malicious packets to prevent them from passing through, and it can interrupt connections between clients and servers to thwart malicious activities. These actions are accomplished by extracting traffic rules from incoming data and comparing them with previously stored rules in the dataset. If a match is found, these actions are applied [14].

IDS comprises several components to perform its function, including a sensor, analyzer, storage, and management console. Firstly, IDS has a sensor used to gather incidents related to system security. Secondly, it includes an analyzer responsible for identifying and analyzing suspicious behavior and web attacks. Thirdly, IDS provides storage to store malicious incidents and the results of the analyzer. Finally, IDS contains a management console that assists administrators in configuring the proposed IDS detection system and monitoring the analyzer's results [14].

DDoS intrusion detection system (IDS) techniques can be classified into two main groups: misuse detection and anomaly detection. Figure 2.12 provides a detailed illustration of these detection techniques.

2.7.1 Misuse Detection Techniques

Misuse detection is also known as the pattern matching technique. It assumes that a detection system or program has patterns that match attack behavior. Misuse detection identifies anomalies based on previously stored attack patterns to detect known attacks. However, misuse detection is not capable of detecting new anomalies, which is a weakness of this method [15]. Misuse detection can be categorized into various types, such as signature-based detection, rule-based detection, and state-based detection [16].

Signature-based detection is related to the information generated by both normal users and attackers in the network. It encompasses the history of executions and transactions recorded by these users, forming patterns of normal and malicious behaviors. These patterns are then compared with patterns derived from new users. Attacks and unauthorized users may be identified if there is a match between new patterns and previously recorded patterns during the identification process. However, these techniques fail to identify novel patterns because similar patterns may not be available in the database [118]. Signature-based detection can be further classified into three different kinds: Content Signatures, Context Signatures, and Attack-based Signatures [16].

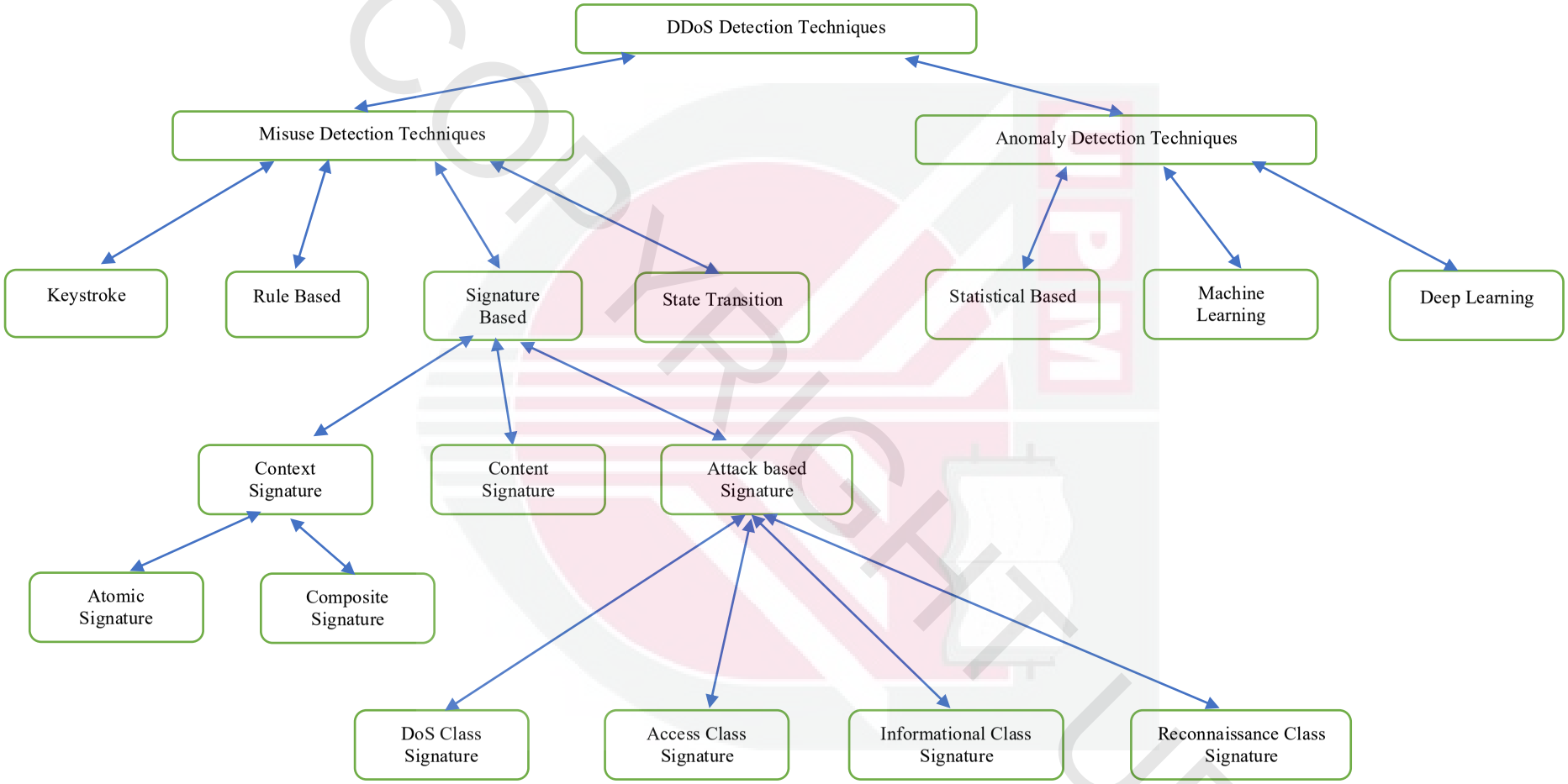


Figure 2.12 : Malicious Detection Techniques [16]

Several research studies have utilized signature-based methods to detect DDoS attacks. For instance, Khraisat et al. introduced a hybrid intrusion detection system method for detecting novel and known attacks. The system's objective is to detect signatures for known attacks, improve identification accuracy, and reduce false alarms. This method employs two machine learning algorithms: support vector machine and C5 decision tree classification. Evaluation of this approach involved different datasets and demonstrated better performance compared to AIDS and SIDS [119].

Snehi et al. conducted a comparison of various signature-based algorithms in terms of their advantages and disadvantages. They discovered that certain patterns and the number of signatures assist system managers in identifying attacks and threats. Their research also indicated that these signatures hold proportional importance for each feature of the intrusion detection system in the identification process. Ultimately, their findings suggest that nearly 80% of threats can be rapidly and effectively detected using signature-based algorithms [120].

Sajad et al. proposed a method named Suricata, which was developed using two different machine learning techniques. Suricata comprises a group of signatures employed to prevent malicious packets from targeting a victim. Metaheuristic-based feature selection is conducted through anomaly-based identification, neural network, and fuzzy logic. The malicious system's data are trained using neural networks to generate signatures. Anomaly data are analyzed using fuzzy logic to handle unknown intrusions and generate rules based on the analysis [118]. The proposed

design is illustrated in Figure 2.13. Ultimately, this method achieved 96.111% accuracy in identifying threats [118].

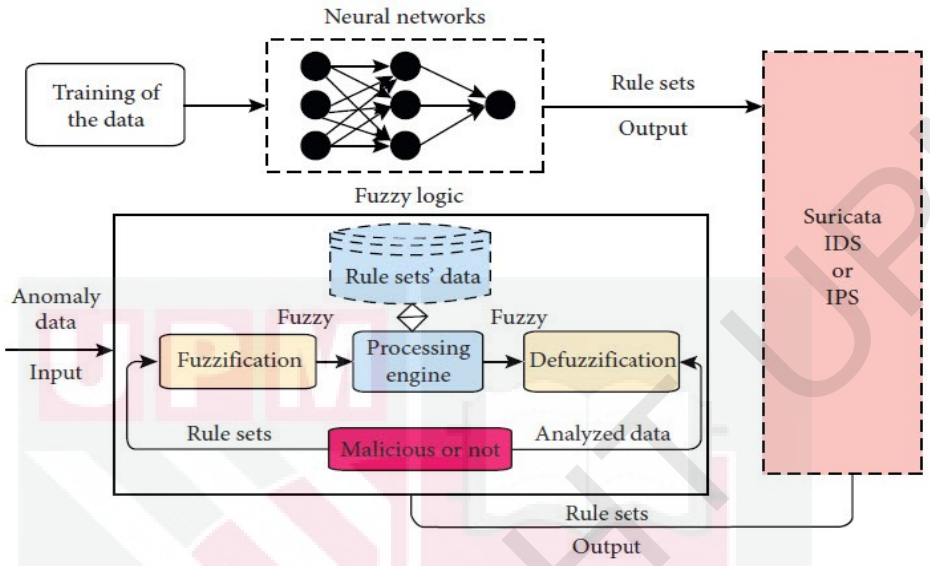


Figure 2.13 : Internal Working of the Suricata Signatures Solution for the IDS IPS [118]

Singh suggested a novel approach that utilizes misuse IDS in a hybrid system. This system combines two different algorithms. The first algorithm is packet header anomaly detection (PHAD), which is based on anomaly detection. The second algorithm involves network traffic anomaly detection (NETAD) using the Snort platform, an open-source platform implemented based on misuse detection. This system was evaluated using the DARPA dataset, and the results of the evaluation demonstrate the higher efficiency of the suggested algorithm in threat detection [121].

Rule-based detection systems are another type of misuse detection system. These rules describe threat information in the form of "if-then" statements. Administrators format these rules after analyzing malicious activities. The generated rule formats

are collected in a log file. Any new detection algorithm then compares incoming traffic with the existing data in the log file to identify suspicious behavior [16]. Joseph et al. proposed a hybrid detection method based on using deep learning for the Industrial Internet of Things (IIoT). They initially employed rule-based feature selection for the verification and training of incoming network traffic. Subsequently, they used a deep learning-based technique, specifically the deep feedforward neural network method, to detect intrusions. Their evaluation utilized two different datasets, NSL-KDD and UNSW-NB15, revealing higher accuracy and detection rates. The accuracy and detection rate were 99% when NSL-KDD was employed, while the accuracy was 98.9%, and the detection rate was 99.9% with UNSW-NB15. Ultimately, their system exhibited a lower detection rate [122].

Furthermore, state-based detection algorithms constitute another type of misuse detection. In these techniques, malicious attacks can be described as sequences of events or activities. The first state is generated when malicious behavior occurs, transitioning to another state with each subsequent attack activity. These transitions continue until an alert state is reached.

For instance, Satheesh and Ciza proposed a novel method for identifying and classifying malicious activities in the Android platform using sequence alignment states. They first decompiled Android malicious applications extracted from the dataset. They then isolated malicious methods and classes from others by encoding suspicious malicious activities, producing Multiple Sequence Alignment (MSA), and employing the Profile Hidden Markov Model (PHMM). The conversion of malicious applications into behavioral patterns is represented by PHMM, which creates and

trains the PHMM profile using three states: insertion, deletion, and Match states, as illustrated in Figure 2.14. The training profile is compared with the testing profile generated from a new dataset to find matches and detect malicious activities. Comparison is based on the log-likelihood score. Ultimately, the proposed method achieved an accuracy close to 94.5% for the detection of malware in Android [123].

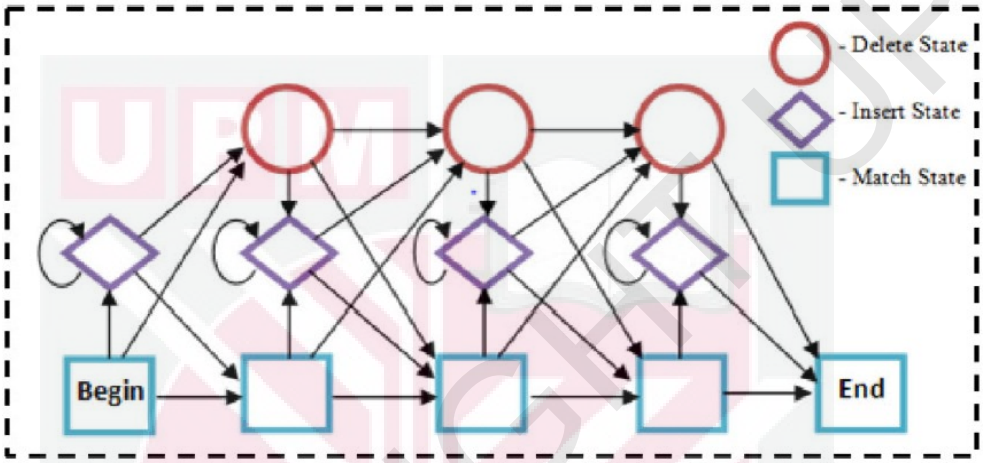


Figure 2.14 : Transition Diagram for PHMM [123]

Finally, keystroke matching is another form of misuse detection. In this type of detection, keystroke methods are based on the style of user typing during their attempts to sign into the system. How users type their characters, what kind of characters they use to log in, and how long they take to log in are all examples of these keystrokes. These keystrokes are monitored and compared to predefined keystrokes belonging to attackers to find a match.

For example, Nahid et al. proposed a new algorithm called Profiling and Preventing Security Attacks (PPSA) based on keystroke matching. PPSA aims to identify and prevent attacks before accessing cloud computing (CC), which delivers IT resources such as storage, computing power, and databases over the internet on demand. First,

they utilized the Classification Based on Associations (CBA) technique, a machine learning algorithm, to profile and predict threats. Second, they employed the keystroke method for new logins based on user properties. These properties could include attributes like age, gender, and whether the user is using a new device or not. All this information is recorded in the database after successful sign-up. Third, new keystrokes extracted from new users are compared to predefined keystrokes to identify matches. If a match is found, the user is considered an attacker; otherwise, they are classified as a legitimate user [124] .

2.7.2 Anomaly Detection Techniques

Unlike signature-based detection techniques, anomaly-based detection techniques rely on profiling benign patterns during the training process. The new patterns that are profiled from incoming data are then compared with the benign patterns previously stored in the database. Any deviations from normal patterns observed during the comparison process are considered as attack activities. These techniques can identify novel attacks as opposed to signature-based techniques, which are limited to identifying only predefined attacks [14], [15], [16], [17], [118].

Attacks can be detected either online or offline using anomaly detection. Online anomaly detection can identify threats at the beginning of malicious activities or during attack events. On the other hand, offline anomaly detection is performed after attackers have initiated their attacks. Various types of anomaly techniques have been employed in IDS implementation, including statistical-based techniques, machine learning-based techniques, and deep learning-based techniques [17], [125].

2.7.2.1 Statistical-Based Detection Techniques

Statistical models are techniques that depend on investigating and analyzing data to gain a deeper understanding of it. These methods are capable of identifying variables to be used as inputs and finding common connections among these inputs to predict outcomes. Numerous studies have employed statistical-based techniques to identify DDoS attacks. Statistical approaches offer in-depth analysis of incoming traffic in computer networks and can detect anomalies in the early stages [17]. Statistical-based detection employs various techniques for DDoS detection, such as chi-squared goodness-of-fit tests, hypothesis testing, standard deviation, entropy, covariance, and correlation matrices.

Dao et al. [126] proposed a method to detect DDoS attacks in Software-Defined Networks (SDN). SDN consists of two main parts: the controller plane and the switch plane. SDN differs from traditional networks by abstracting control from routers, switches, and physical hardware elements. It empowers a specific device, the controller, to program the entire network. This provides network administrators with the capability to make decisions and manage the entire network from a single device, the controller [52], [57], [78], [79], [83], [87], [126].

However, this controller faces security challenges, including DDoS attacks. DDoS attacks can overwhelm the controller and disrupt services for legitimate users. These attacks often involve sending a large number of packets per session towards the controller, generating fewer sessions than normal. Thus, the authors employed a statistical approach to identify DDoS attacks targeting SDN controllers. They monitored two factors: the minimum number of packets per session and the average

number of sessions for each user. These factors were compared with predefined values within the controller to make decisions. The authors simulated their method by constructing their own network. Ultimately, they found that their system effectively mitigated DDoS attacks, but encountered limitations when facing a high volume of attacks [126].

Piedrahita et al. [127] introduced an algorithm called FlowFence to mitigate DDoS attacks in SDN by calculating flow bandwidth. The FlowFence architecture comprises two main components: switches and controllers. Switches in this design are responsible for monitoring network congestion and notifying the controller of any congestion. The controller then takes action to mitigate DDoS attacks by limiting the flow transmission rate on congested switch interfaces. The controller compares the flow's transmitted bit-rate through an interface with a predefined acceptable threshold. If the incoming flow's transmitted bit-rate is lower than the threshold, the flows are considered normal without DDoS attacks.

Conversely, if the incoming flow's transmitted bit-rate exceeds the threshold, the flows are deemed malicious. The malicious transmitted flows are allocated a specified amount of bandwidth to avoid affecting normal flows. The defense process of FlowFence is illustrated in Figure 2.15. FlowFence was tested on the Future Internet Testbed with Security (FITS), a platform designed by European and Brazilian universities. The results demonstrate that FlowFence is a fast, efficient, and effective design that rapidly mitigates DDoS attacks, as stated by the authors.

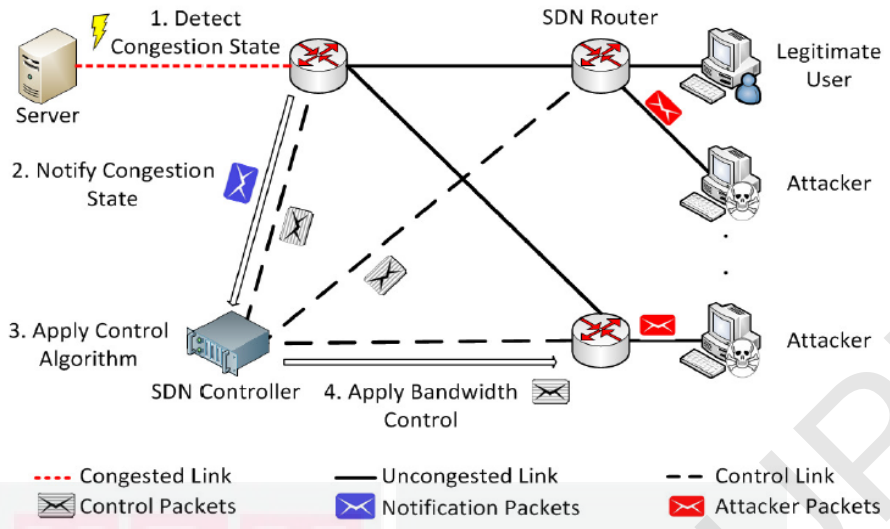


Figure 2.15 : FlowFence Defence Procedure [127]

Durner et al. [128] proposed a method to detect DDoS attacks based on a statistical approach within the context of an SDN implemented in a Cloud environment. An SDN consists of two main components: the control plane and the data plane. The authors chose to focus on protecting the data plane, particularly switches, as they are responsible for forwarding traffic towards the controller. This is significant because excessive traffic to the controller can cause congestion.

The proposed method involves creating a table within the data plane or switches, where header fields are represented as columns and hash function values of these fields are placed in rows. The 32-bit FNV-1a algorithm is used for the hash function calculations. Header fields of outgoing traffic from switches tend to change under normal conditions but remain constant during malicious activities. When the count of unchanged traffic instances increases, an OpenFlow rule is triggered to drop incoming traffic suspected to be malicious. Evaluation showed that the proposed algorithm had a low false positive rate and effectively identified malicious attacks.

However, the method was not able to detect malicious traffic that simultaneously changed all of their field values [128].

Dong et al. [18] proposed a method utilizing the Sequential Probabilities Ratio Test (SPRT) technique to identify DDoS attacks targeting SDN controllers. The researchers observed that attackers often send a large number of low flows to overload and disrupt the controller. Low flows involve a small number of packets. They categorized these flows as either normal or malicious by calculating the number of packets for each flow and comparing the results against predefined thresholds. Flows with packet counts above the threshold were considered normal; those below were flagged as malicious. The study also identified malicious interfaces generating numerous low flows using the SPRT method. The model was evaluated using the DARPA dataset and its results were compared with other approaches. The researchers found that their method exhibited higher accuracy, versatility, and speed. However, the design relied on a single feature, packet count per flow, which attackers could bypass by altering their attack behavior.

The SPRT detection ($D\_SPRT_i^s$) monitors incoming flows based on their features values (FL1, FL2, …, FLi). It also identifies the switch interface (s) that allows these flows to cross over to the targeted machine. The detection ($D\_SPRT_i^s$) is the likelihood ratio between these observations, whether they are compromised or normal flows injected into compromised interface (H2), and these observations injected into a normal interface (H1). Therefore, detection formula can be formulated as follows [18]:

$$D\_SPRT_i^s = \ln \frac{prob\ (FL_1^s, \ldots.,\ FL_i^s\ |H_2\ )}{prob\ (\ FL_1^s,\ \ldots.\ ,\ FL_i^s\ |H_1)} \tag{2.5}$$

Where (i) is the total number of flow observations. Let us assume that these observations (FL_i^s) are identically independent and distributed. Thus, the detection formula can be as follows [18]:

$$D\_SPRT_i^s = \sum_{v=1}^{i} \ln \frac{prob\ (FL_v^s|H_2\ )}{prob\ (FL_v^s\ |H_1\ )} \tag{2.6}$$

Where (v) is each value in a group of flow observations. Because (FL_i^s) can be as Bernoulli random variables, the detection will be as follow [18]:

$$prob\ (\ 0 \leq FL_i^s \leq 0.5|H_1\ ) = 1\text{- }prob\ (\ FL_i^s > 0.5|H_1) = \mu_1 \tag{2.7}$$

$$prob\ (\ 0 \leq FL_i^e \leq 0.5|H_2) = 1\text{- }prob\ (\ FL_i^s > 0.5|H_2) = \mu_2 \tag{2.8}$$

Where value of μ_2 is larger than the value of μ_1 since compromised interfaces are more likely to be injected with compromised flows to flood targeted system with DDoS attacks. Switch interfaces are more likely to be injected with infected flows when FL_i^s is between 0 to 0.5. On the other hands, switch interfaces are more likely to have normal traffics when value of FL_i^s is above 0.5. Therefore, the detection of SPRT can be shown as Equation (2.9) [18].

$$D\_SPRT_i^s = \begin{cases} D\_SPRT_{i-1}^s + \ln \dfrac{prob\ (FL_i^s|H_2\ )}{prob\ (\ FL_i^s\ |H_1)}, & 0 \leq FL_i^s \leq 0.5 \\ D\_SPRT_{i-1}^s + \ln \dfrac{prob\ (FL_i^s\ |H_2\ )}{prob\ (FL_i^s|H_1)}, & FL_i^s > 0.5 \end{cases} \tag{2.9}$$

By substituting Equations (2.7) and (2.8) in Equation (2.9), the detection equation can be rewritten as shown in Equation (2.10) [18].

$$D\_SPRT_i^s = \begin{cases} D\_SPRT_{i-1}^s + \ln\frac{\mu_2}{\mu_1}, & 0 \le FL_i^s \le 0.5 \\ D\_SPRT_{i-1}^s + \ln\frac{1-\mu_2}{1-\mu_1}, & FL_i^s > 0.5 \end{cases} \quad (2.10)$$

where $D\_SPRT_0^s=0$. The detection technique of SPRT generates two types of errors that affect the accuracy of detection. These mistakes are the false positive error λ_1 and false negative error λ_2. The False positive error λ_1 occurs when detection technique mistakenly considers normal interface H1 as a malicious interface H2. On the other hands, the false negative mistake λ_2 occurs when an infected ethernet H1 is wrongly identified as a benign ethernet. Upper bound (U) and lower bound (L) thresholds were computed as shown in Equation (2.11) in order to address these two mistakes [18].

$$\begin{cases} U = \log_2\frac{\lambda_2}{(1-\lambda_1)} \\ L = \log_2\frac{(1-\lambda_2)}{\lambda_1} \end{cases} \quad (2.11)$$

Finally, the detection result ($D\_SPRT_i^s$) for each observed flow that passes through a certain interface is checked with the upper and lower bound thresholds in order to make a decision. If the value of ($D\_SPRT_i^s$) is larger than or equal to U, then the monitored interface with its flows is considered benign, and the test will stop. However, if value of ($D\_SPRT_i^s$) is less than or equal to L, then the monitored interface with its flows is considered compromised, and the test will stop. Finally,

the detection test will continue by checking another flow observation when the above two conditions are not met.

Kousar et al. [129] implemented a DDoS detection method utilizing Apache Spark, a distributed processing framework. Unlike Hadoop, which is optimized for batch processing, Spark is designed for real-time data handling. Its efficiency arises from storing intermediate data in RAM during read/write operations, as opposed to Hadoop, which relies on slower disk-based read/write processes.

The implementation comprises two main phases. In the first phase, real-time preprocessing of traffic headers is performed using Spark RDD (Resilient Distributed Dataset). Feature selection is carried out at this stage using the correlation matrix, and the results are stored in Spark RDD. The second phase involves training and detection using Spark MLlib, which includes support vector machines, random forests, naive Bayes, and decision trees. The detection system's design reduces the time required to identify DDoS attacks and enhances detection efficiency, owing to the utilization of Spark technology [129].

Entropy-based techniques have been widely used in various studies to detect DDoS attacks. These methods are favored for their simple mathematical calculations and accurate results [130]. Entropy is a concept used not only in network security but also in information theory, physics, ecology, economics, and other fields. Entropy measures the uncertainty, information disorder, or randomness of elements within a set of variables, and how each element differs from the others. When incoming traffic originates from multiple sources, it is considered random, resulting in high

entropy values close to 1. Conversely, when entropy approaches 0 or is low, traffic variability is less random [29].

St-Hilaire and Mousavi [29] introduced an approach utilizing entropy calculations to detect DDoS attacks in SDN networks. Incoming flows were divided into fixed-size windows based on the number of flows or time slots. Entropy was computed for the number of occurrences of each unique destination IP address to assess randomness. When a specific host is targeted by a large number of flows, the destination IP address appears frequently, reducing randomness and entropy. A threshold was used to determine the presence of DDoS threats. Entropy values below the threshold indicated a higher likelihood of a DDoS attack on the SDN controller. Conversely, entropy values above the threshold suggested normal incoming traffic. The authors achieved successful DDoS attack detection after analyzing five hundred infected traffics [29].

The feature values (Ai) for flows and their corresponding interfaces are grouped into a fixed window size (W), where (i) is the number of unique feature values. The window size can be determined based on a fixed number of flows. The number of times each feature value appears (Bi) is calculated and stored in W, as shown in Equation (2.12) [29].

$$\mathrm{W} = \{ (A_1, B_1), (A_2, B_2), (A_i, B_i) \} \tag{2.12}$$

The probability of appearance of feature values, Prob(i), is calculated by the mathematical division of (Bi) over the number of flows (n_flow) for each (W) as shown in the following formula:

$$\text{Prob(i)} = \frac{\text{Bi}}{\text{n\_flow}} \tag{2.13}$$

Finally, Shannon entropy (Ey) can be computed using on the result of previous formula as shown in the following [29]:

$$\text{Ey} = -\sum_{0}^{\text{i}} \text{prob(i)} \ln \text{prob (i)} \tag{2.14}$$

Ma and Chen [131] introduced a technique to identify DDoS anomalies using the variation of Lyapunov and Tsallis entropy. The method involved capturing incoming traffic and extracting the source and destination IP addresses from each flow. Entropy values for the source and destination IP addresses were calculated using Tsallis entropy. A comparison between the entropy values of source and destination IP addresses enabled the detection of changes in network behavior and the identification of anomalies. This was achieved by calculating the separation rate of source and destination IP addresses using the Lyapunov technique. A classification threshold was determined to make decisions and identify threats. The authors found that their method efficiently detected DDoS attacks compared to common entropy-based approaches. The approach was evaluated using the MIT dataset, yielding a sensitivity of 98.56% and a probability of false alarm of 0.24.

Hoque et al. [132] introduced a method based on statistical calculations to detect threats. They employed the Feature Feature Score (FFSc) technique to distinguish malicious traffic from normal traffic. Initially, they extracted three features from incoming traffic: the variation of the IP source address, the entropy of the IP source address, and the packet rate. The 'variation of IP source address' represents the rate

of change of source IP addresses over time. When source IP addresses change frequently, the variation of the IP source address also increases.

The 'entropy of IP source address' is computed by identifying the IP source per unit of time and then calculating the entropy value for each set of sample traffic using the Shannon entropy formula. The 'packet rate' indicates the number of traffic packets transmitted in one second. These three features were used to compute the FFSc during normal network traffic. The FFSc calculation was utilized to generate a normal profile, which was then compared to the profile generated during online analysis to detect malicious activities. The similarity score was computed by comparing the pre-computed profile for normal network traffic with the profile computed during online analysis. Attacks were detected when the similarity score exceeded the proposed threshold. The authors evaluated their method using the CAIDA 2007 and MIT datasets. Their method achieved an accuracy of 98% on the CAIDA dataset when the threshold was set to 0.05 and 81% when the threshold was set to 1. On the MIT dataset, their method produced an accuracy of 99.55% with a threshold of 0.05 and 96.6% with a threshold of 1 [132].

Meng and Guo in [133] presented a method based on a statistical approach to protect SDN controller networks from DDoS threats. They initially extracted destination IP address features from incoming flows. Subsequently, they computed the entropy for the set of destination IP addresses. The Sequential Probabilities Ratio Test (SPRT) was used to make decisions regarding the presence of threats in incoming network traffic. They utilized the Scapy tool to generate infected packets for their experimental setup. The authors found that their method demonstrated higher

accuracy in identifying slow DDoS attacks, but it was unable to recognize DDoS threats arising from flash crowds.

Valizadeh and Niar [134] implemented an approach aimed at enhancing network availability and protecting SDN controller networks from DDoS threats. Their implementation combined an entropy-based method with the packet windows initiation rate (PWI) technique. Initially, the authors computed the entropy for destination IP addresses to gauge randomness and facilitate early detection of DDoS threats. As mentioned earlier [29], the entropy value was compared with a threshold to determine the presence of an attack. However, the entropy method failed to detect DDoS attacks when attackers targeted multiple hosts. This occurred because the entropy value increased, rendering the threshold ineffective for detecting such attacks. To address this, the authors introduced the PWI approach, enabling the detection of attacks in multi-host scenarios.

Li and Wu [31] proposed a **φ**-entropy technique for identifying DDoS attacks in SDN networks. This method uses the **φ** parameter to modify input parameters based on changes in network status. This adjustment enhances the feature differences between malicious and benign traffic, making it easier to detect malicious activities at an early stage. The method relies on IP addresses as features. The Mininet platform was used to conduct the experiment, and a comparison was also performed with Shannon entropy. The Scapy tool and the Random function were utilized to generate the database. Ultimately, this method exhibited a higher detection rate compared to Shannon entropy.

Aladaileh et al. [135] proposed a dynamic threshold method as an alternative to the static threshold approach to handle the changing behavior of traffic during DDoS attacks. This dynamic thresholding enables the detection of both high and low intensities of DDoS attacks, thereby enhancing accuracy and reducing the probability of false alarms. The authors introduced a technique based on rule-based detection to aid in decision-making. They achieved this by calculating the Renyi joint entropy for source and destination IP addresses from incoming flows. The results of the Renyi calculation were then compared with the dynamic threshold to apply rule-based detection and differentiate normal traffic from malicious traffic. The proposed method was implemented and evaluated across eight simulation scenarios involving flow bandwidth, attack targets, and sources.

Özçelik and Brooks [136] suggested a detection system that combines the cumulative sum (CUSUM), wavelet, and entropy techniques. In this system, entropy of packet header fields, such as source IP addresses, is computed to measure the randomness of these fields. The second stage involves applying wavelet transformation to remove long-term variations in entropy values, thereby reducing the false positive rate. The third stage employs the CUSUM detection method to identify sudden hidden growth in background data. This approach ultimately achieves high performance and efficiency in identifying DDoS attacks.

Finally, another study conducted by Hoque et al. [137] aimed to identify DDoS threats in computer networks. The design comprises three main components: preprocessing, a security manager, and attack identification. During the preprocessing stage, three features were extracted from traffic headers, and entropy

was calculated for them. These features include the number of packets, the source IP address changing index, and the source of the IP address. The security manager is responsible for recalculating the legitimate profile and threshold to adapt to network changes based on old information and new classification logs. Both the security manager and preprocessing were implemented in software, while the defense was implemented in hardware using FPGAs to enhance performance and accelerate detection times. The defense component calculates the legitimate profile and correlations among extracted features to determine the distance between the infected profile and the legitimate profile.

This distance is then compared to fixed threshold to make a decision regarding the presence of an attack. When the threshold values are changed, the values of accuracy, FNR, and TPR are changed accordingly. This will decrease the chance to achieve optimal values for the detection process. For example, when the threshold values are 0.3 and 0.4, the accuracy will be 22% and 70% respectively. In the same way, when the threshold values are close or above to 0.5, the accuracy will be close to one hundred percent. The value of FNR and TPR will be changed as well when threshold are changed, and the best value of FNR and TPR were 0.18 and 1 respectively. Ultimately, the detection process took 354 nanoseconds to make a decision, a duration significantly shorter than the state-of-the-art methods, as indicated by the authors.

2.7.2.2 Machine Learning-Based Detection Techniques

Machine learning involves using mathematical computations and calculations to train proposed techniques with a set of samples to predict output based on new

samples. Machine learning techniques are also effective for classifying and detecting DDoS threats. For example, Koay et al. [130] introduced a method to detect both high and low intensities of DDoS attacks. They first proposed an approach to extract new features based on an entropy technique, aiming to detect different types of DDoS attacks beyond the traditional features. Subsequently, they employed a combination of three different types of machine learning algorithms for the detection stage.

They extracted two types of features: regular entropy-based features and entropy variation-based features, as illustrated in Figure 2.16. Regular entropy-based features are commonly employed in threat detection, including source IP address, destination IP address, source port address, destination port address, protocol name, source MAC address, destination MAC address, packet length, IP DSCP value, TCP sequence number, TCP window size, and TCP length. However, these traditional features have limited effectiveness in identifying DDoS attacks, being capable of detecting only certain types of DDoS threats.

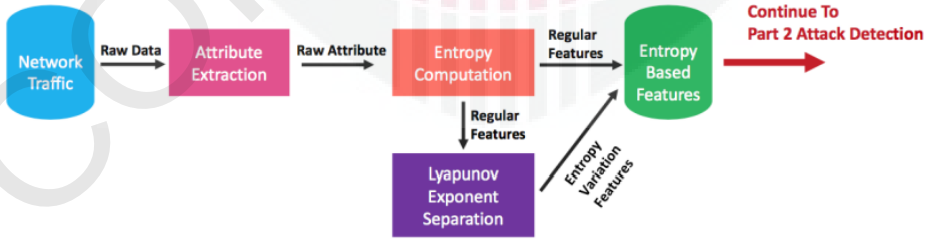


Figure 2.16 : Feature Extraction Stage by Koay et al. [130]

On the other hand, variation-based features are new and promising, as they can detect new kinds of low-intensity DDoS threats. These features are generated based on the calculated separation between two regular features (source and destination)

using the Lyapunov exponent approach to demonstrate differences between these two features. Examples of generated features include Separation IP, Separation TCP, Separation MAC, Separation Network, and Separation Port. The authors also employed the entropy approach in the calculation of each feature. Subsequently, these features were passed to a detection stage consisting of three machine learning-based detection approaches: Alternating Decision Tree, Recurrent Neural Network, and Multilayer Perceptron. Their findings indicate that their method exhibits good precision results in identifying different intensities of DDoS attacks across various datasets [130].

Maranhão et al. [138] presented a method aimed at protecting Cyber-Physical Systems (CPSs), such as sensors, communication devices, and control processing units, from DDoS attacks. The authors performed feature extraction using statistical-based techniques and fed the results into machine learning techniques to differentiate between normal and malicious traffic. Firstly, they worked on a dataset, performing tasks such as dataset splitting into training and testing subsets, label encoding, normalization, scaling, and cleaning. They also estimated multilinear ranks within the dataset. Next, they calculated the mean value for common feature extraction using Higher-Order Singular Value Decomposition (HOSVD).

Lastly, they employed the output from the previous step as input for machine learning techniques such as gradient boosting, random forest, and decision tree to discern malicious and benign traffic. The authors compared their results with other techniques using the CICDDoS2019 and CICIDS2017 datasets. Their findings show

that the proposed system offers increased error-robustness and performance. However, the method's downside is its higher computational complexity [138].

Nguyen et al. [139] proposed a DDoS detection method based on machine learning techniques. They initially extracted eight entropy-based features from incoming traffic for use as input data for classification techniques. These features include the entropy of source/destination IP addresses, source/destination port addresses, packet size, and protocol. Additionally, features like packet count, mean packet length, source/destination unique port addresses, and source/destination unique IP addresses were included. These features were fed into the next step, which involved detecting DDoS attacks.

This step consisted of two layers, both implemented using the Hierarchical Temporal Memory algorithm (HTM). This algorithm assigned attack signatures during the training phase and recognized similarities during the detection phase, making it suitable for online detection. The first layer analyzed each feature and assigned a specific label using the K-Nearest Neighbors (KNN) algorithm. The second layer generated signatures for well-known attacks and detected similarities between predefined signatures and those generated from online streams during the prediction stage. The authors utilized the MAWI and CICDDoS2019 datasets to conduct and evaluate their experiment. Ultimately, they achieved high performance in identifying most types of DDoS attacks in these datasets, except for SSDP and UDP attacks [139].

Liu et al. [28] presented the FEDDM technique, which is based on Fast Fourier Transform and Entropy detection of DDoS Method, to identify DDoS threats. The architecture consists of two stages: data management and the detection phase. The data management stage is responsible for collecting and storing data in the database. The collected data stream in this phase follows specific control conditions to be stored in the dataset. For instance, incoming data stream sets with unique source/destination IP addresses are stored in a queue. These data stream sets are removed if they are not updated, or if their count is less than 5 within a five-minute interval. If the number of data stream repetitions exceeds 5 times, the upper limit of the number of repetitions of source/destination IP addresses is checked based on data packet types. This is done to determine whether to remove or keep the first data in the queue for this destination/source IP address. When the data packet type is UDP or TFTP, the first data in the queue is removed when the upper limit of 50 stream data is reached.

However, these data streams remain stored in the queue, and calculations are performed based on these data streams to describe their behavior. The second stage involves computing features for the detection method based on the work values, utilizing information entropy and Fast Fourier Transform (FFT). The detection method relies on these features and can be implemented using the Neural Network (NN) method. The implementation of this method can be visualized in Figure 2.17. Finally, the authors utilized the CICDDoS2019 dataset to evaluate their technique, and their method demonstrated an accuracy of over 99% in identifying WebDDoS, SNMP, NetBIOS, and Syn attacks.

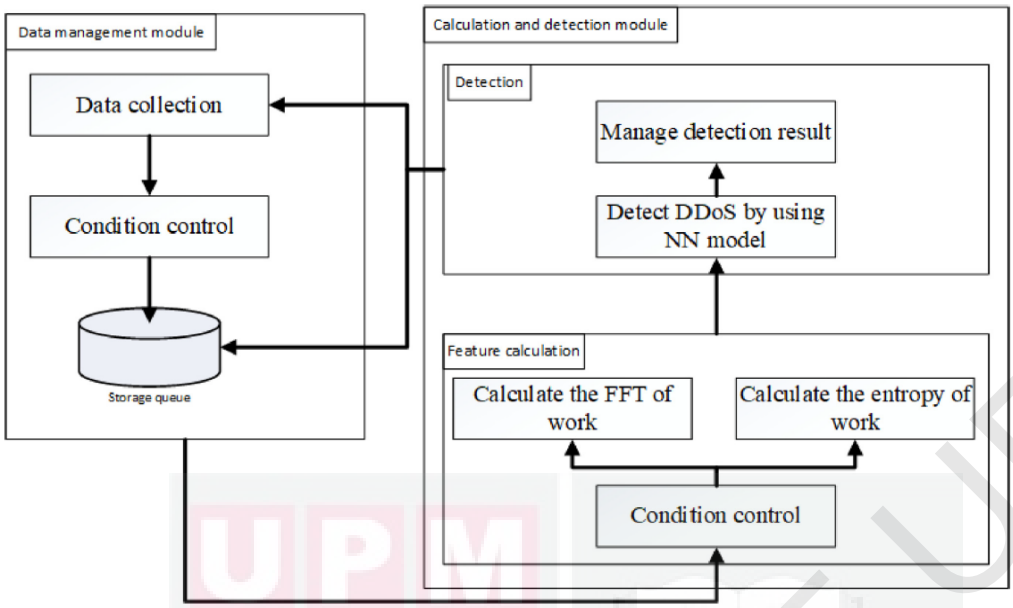


Figure 2.17 : FEDDM Approach [28]

Long and Jinsong [140] proposed HESS (Hybrid Entropy SSAE-SVM), a hybrid detection method that combines information entropy, a machine learning approach, and a deep learning approach. HESS stands for Hybrid Entropy SSAE-SVM. Initially, they calculate information entropy to detect suspicious traffic. They utilize a deep learning method known as Stacked Sparse Auto Encoder (SSAE) to select important features and reduce execution time. The OpenFlow table in the SDN network contains extensive information about incoming packets.

As a result, 19 features were extracted from this table to serve as inputs for the SSAE method. The SSAE method is an unsupervised and deterministic neural network that leverages the neural network's outcomes to reconstruct input data. The outputs of the SSAE method, which are the selected features, serve as inputs for the subsequent detection phase. This detection phase employs the Support Vector Machine (SVM), a machine learning technique based on a statistical learning model. SVM is capable

of identifying the optimal hyperplane that separates benign from malicious traffic. It is employed to verify whether the suspicious traffic identified in the entropy stage is indeed malicious. The design of the HESS approach is depicted in Figure 2.18. Ultimately, the performance of the HESS approach was tested on different datasets, achieving a detection accuracy of over 98% [140].

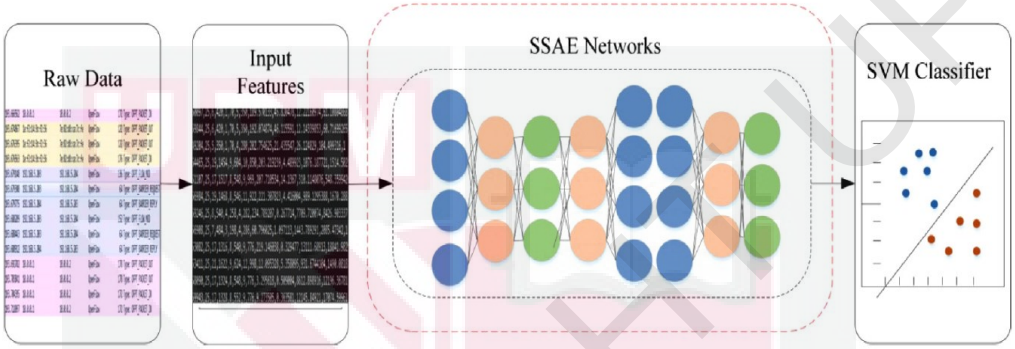


Figure 2.18 : Design for HESS Model [140]

Polat et al. [141] proposed a method based on employing four different machine learning algorithms for threat detection, along with three distinct methods for feature selection. Initially, they employed feature selection techniques to identify influential features and reduce execution time. The first feature selection method is the Relief approach, which belongs to the category of filter-based feature selection. The second method is a greedy search, involving sequential forward selection, implemented using a wrapper approach. The third method is the Lasso approach, which operates as an embedded feature selection technique. Despite the large number of input features, only 12 features were retained as output from these selection methods. These selected features were subsequently fed into the detection process. The detection stage employed four machine learning techniques: Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbors (KNN), and Artificial

Neural Network (ANN). The authors then generated their own dataset and achieved testing accuracies ranging from 91% to 98% for their technique.

Daneshgadeh et al. [142] proposed a hybrid machine learning-based approach for identifying DDoS threats and legitimate flash events. Features were extracted from generated traffic, and entropy calculations were employed to compute feature vectors. Four features were extracted: entropy of received bytes, entropy of destination IP, entropy of source IP, and time interval. These feature vectors were used as inputs for SVM and Kernel-based Online Anomaly Detection (KOAD), both of which were utilized in the detection phase. The algorithm was tested using collected threat traffic from their private organization and normal traffic from the FIFA World Cup 98 dataset. The approach demonstrated a low false alarm rate and a high true positive rate.

Sharafaldin et al. [22] proposed a model that incorporates four machine learning-based techniques: logistic regression, ID3, Naïve Bayes, and Random Forest. The CICFlowMeter was employed to generate 80 features from incoming traffic. A Random Forest Regressor was then utilized to select valuable features based on their importance. A new dataset, CICDDoS2019, was generated and used to evaluate their design. Ultimately, the ID3 model achieved precision, accuracy, and F1 scores of 78%, 65%, and 69% respectively, outperforming the other models employed.

Ye et al. [143] introduced a method based on using SVM to distinguish infected traffic from normal traffic in an SDN network. Their design consists of three components: flow table collection, characteristic extraction, and a classifier. Initially,

the SDN switch table contains information about incoming traffic directed towards the SDN controller. These details are extracted for use in this phase. The next step involves the extraction of characteristics responsible for computing six attributes used by the SVM classifier. These attributes encompass source port speed, source IP speed, flow entries speed, the ratio of paired flows, flow bytes deviation, and flow packets' standard deviation. The SVM then makes decisions based on these characteristics. The method was evaluated through simulations conducted on Floodlight and Mininet, achieving an accuracy of 95.24%.

Hofer-Schmitz et al. [144] proposed an unsupervised machine learning technique for detecting compromised traffic related to DDoS attacks, known as the Local Outlier Factor method. This technique calculates the deviation score of each sample in relation to its neighbors. A higher deviation score indicates a greater abnormality of the observed sample. The classifier was employed to detect real advanced persistent threat (APT) attacks, a type of DDoS attack characterized by being dangerous, persistent, and advanced. Features used in the classifier were extracted using the CICFlowMeter tool, yielding nearly 80 features. Feature selection was performed through correlation analysis to identify influential features and reduce execution time. The Contagio malware dataset was used to obtain APT traffic, while the CICIDS2017 dataset provided normal traffic for evaluating the performance of the Local Outlier Factor.

Nandi et al. [145] presented a DDoS detection approach employing a set of machine learning techniques and hybrid feature selection. The NSL KDD dataset was used to evaluate the detection method and test the efficiency of hybrid feature selection.

Influential features were selected using a combination of hybrid feature selection methods, including symmetrical uncertainty, chi-squared, gain ratio, Relief, and Kullback–Leibler divergence. Features were ranked and sorted in descending order based on their weight. The best fifteen features were then chosen as input for the classification approach. The dataset was discretized into binary values (0 and 1) using the Weka tool to enhance the effectiveness of detection results. Various machine learning methods, including Random Forest, Decision Table, Naïve Bayes, J48, and Bayes Net, were employed to detect malicious activities. The average detection rate achieved by this design was 99.02%.

Moreover, Gaur and Kumar [146] introduced a method for detecting DDoS anomalies in IoT devices. Four machine learning algorithms—XGBoost, Random Forest, Decision Tree, and K-Nearest Neighbors—were tested. Three different feature selection techniques, which are ANOVA, chi-square, and Extra Tree, were applied with each classifier to select relevant features. Confusion matrix results were presented for each classifier using each feature selection technique in both a cloud environment and utilizing the CICDDoS2019 dataset. This hybrid technique proved effective in quickly identifying DDoS anomalies in the IoT environment. However, optimizing parameters can help reduce overfitting and underfitting and enhance accuracy.

Khoei et al. [147] presented a technique for identifying DDoS anomalies using three different ensemble learning techniques in a smart grid environment: stacking-based, bagging-based, and boosting-based methods. CICFlowMeter features extracted from traffic were employed as inputs for these techniques. Feature selection techniques,

including tree-based and Pearson's correlation coefficient-based methods, were also applied to reduce the number of features. Various metrics were used to evaluate the performance of these classification techniques using the CICDDoS2019 dataset, with the stacking-based method outperforming the others.

2.7.2.3 Deep Learning-Based Detection Techniques

Deep learning is a form of machine learning technique that utilizes multiple layers of neural networks to emulate human thinking. These techniques require a substantial number of inputs to train the parameters of their layers, enabling them to learn features and produce accurate outputs. Deep learning methods have been employed in various tasks, such as classifying traffic, selecting important features, extracting features from traffic data, and detecting DDoS attacks.

First of all, Liu et al. [148] introduced an approach based on entropy and optimized deep learning methods to identify DDoS threats. Their implementation comprises two phases: calculating entropy for extracted features and utilizing a Convolutional Neural Network (CNN) to identify the infected port of a switch device, allowing suspicious traffic to pass through. In the first stage, the authors extracted six features from incoming traffic: source IP address, destination IP address, source port address, destination port address, packet protocol name, and packet length. From these, they selected three features, which are packet protocol, protocol length, and source address based on their effectiveness in the detection process.

Entropy was also calculated for these features, enabling the identification of suspicious traffic by comparing their values against a threshold. Subsequently, the

traffic data stream was transformed into a picture format to utilize CNN for image-based detection. Each packet was converted into a number between 0 and 255, representing a pixel in an image. The CNN model was then employed to make decisions and distinguish compromised from normal traffic. The experimental evaluation was conducted using Mininet and the POX controller, resulting in higher accuracy, precision, and recall [148].

Moreover, Yuan and Yang [149] implemented a method based on relative entropy and CNN techniques to detect DDoS attacks. Infected flows are identified by calculating the relative entropy of switch interfaces. The identified infected flows are then fed into a CNN approach for feature extraction and weight assignment to achieve optimal detection outcomes. The CNN model consists of four layers: the first layer is the convolutional layer responsible for mathematical computations on the input, producing linear output. The second layer is an activation function indicating the nonlinearity of the CNN method, represented by the Tanh function.

The third layer is the maximum pooling layer, utilizing the Max-pooling function to reduce the computational values generated by the first layer. The fourth layer is the fully connected layer, providing translation invariance to the CNN method, thereby enhancing the efficiency of the detection approach. These layers are illustrated in Figure 2.19. The authors tested their design by collecting real-time datasets using the Scapy tool, achieving an accuracy of 99.12% and a false positive rate of 3.11%.

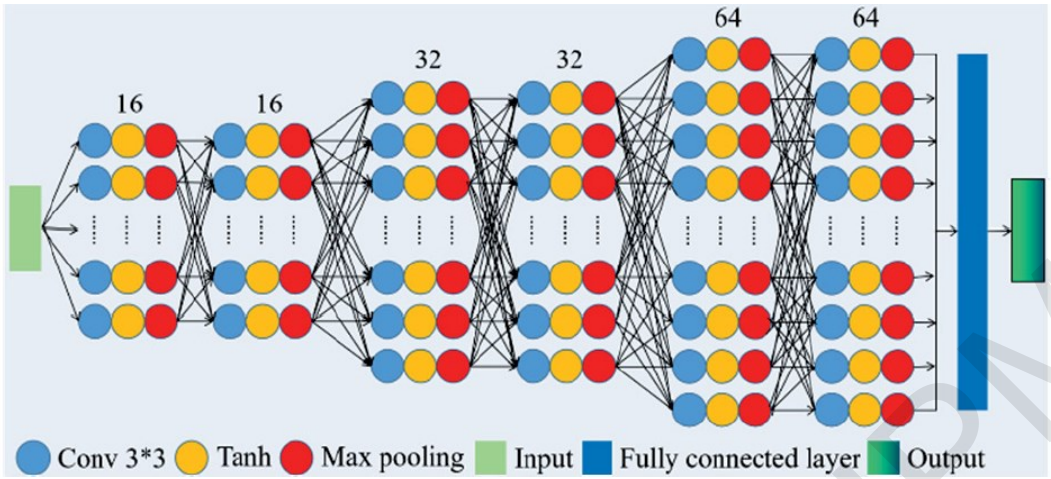


Figure 2.19 : CNN Detection Layers [149]

Likewise, Nazih et al. [150] detect DDoS threats in the Voice over IP (VoIP) system, a telephone system that employs deep learning techniques. The VoIP system utilizes the Session Initiation Protocol (SIP) for traffic transmission, but this protocol is susceptible to DDoS threats. The authors devised a two-phase design. The initial phase involves extracting features from SIP messages, accomplished by converting SIP texts into fixed-length feature vectors. This is achieved by removing all punctuation during tokenization, separating tokens with spaces to form sequences, and appending zeros to the end of each token during the padding step. The extracted features are then fed into Recurrent Neural Networks (RNNs) to identify high and low rates of DDoS threats. The proposed method was evaluated with new datasets generated for testing, revealing that the technique achieved higher accuracy across various attack scenarios.

Moreover, Chonka et al. [151] proposed a method grounded in Chaos theory and a neural network algorithm for detecting DDoS anomalies. Chaos theory was employed to compute self-similarity based on the source IP address of packets, enabling the differentiation between normal and malicious flows and providing early

attack notifications. The self-similarity results were utilized to train the neural network algorithm. Subsequently, the Lyapunov Equation was employed to predict incoming traffic and filter out infected flows. The authors tested their approach using the DARPA 2000 LLS 1.0 dataset. This method uses fixed threshold in order to make decision. Threshold values fall in the range between 0.1 to 0.9. As a results, values of TPR fall in the range between 0.88 to 0.94, and FPR fall in the range 0.05 to 0.45. Changing these values happened due to using fixed thresholds.

Sarmila and Kavin [152] proposed an unsupervised anomaly detection method for identifying DDoS threats. The process begins with data preprocessing, encompassing data cleaning to ensure accurate results. Relevant features are extracted from traffic headers, including source/destination IP addresses, source/destination port addresses, protocol type, sequence number, window size, packet size, and Ack number. The unsupervised anomaly detection relies on a heuristic clustering algorithm, tasked with handling unlabeled traffic and identifying compromised threats. This algorithm automatically generates clusters and computes similarity between clusters using the Euclidean distance to determine if a new cluster should be created. Labeling then determines whether a cluster is malicious or normal. The system was evaluated using the DARPA 2000 dataset. The experiment executed on different number of samples that were ranged from 10 to 120 packets, and the values of accuracy were changed from 41% to 95%. The values of detection rate were fall in the range 0.048 to 0.383 while FPR values were fall in the range 0.167 to 0.523.

Furthermore, Bista and Chitrakar [153] introduced a method that combines Heuristic clustering algorithms with classification methods to differentiate infected from normal traffic. Initial steps include cleaning datasets and extracting the same features mentioned in [152]. Heuristic clustering algorithms are employed to create clusters, and Naïve Bayes classification is used to classify normal flows from malicious ones. The design was tested using two datasets: DARPA 2000 and CAIDA 2007. Using DARPA 2000 dataset results in a false positive rate of 1.9% and a detection rate of 87%. The experiment executed on different number of samples that were ranged from 10 to 120 packets, and the values of accuracy were changed from 58% to 100%. TPR values were also ranged from 0.56 to 1 while FPR values were fall in range 0 to 0.352.

In addition, Cepheli et al [154] employed a hybrid intrusion detection system (H-IDS) to identify DDoS attacks. Three features were extracted from the traffic, packet length, protocol frequencies, and traffic interarrival time. Anomaly detection was performed using Gaussian mixture models, while SNORT tool was employed for signature-based detection. The outputs from both detectors were combined to make a decision and enhance the detection accuracy. The experiment was conducted using the DARPA 2000 dataset and a dataset captured by a bank in Turkey, resulting in a sensitivity of 92.1% for DARPA and 99.9% for the Turkey dataset. Finally, this method does not depend on threshold or window size to make a decision regarding attack availability.

Furthermore, Li in [155] proposed three different deep learning-based classifications to detect DDoS attacks. These approaches include Dense Neural Networks (DNN),

Pearson Correlation Coefficient (PCC), and autoencoder algorithms. CICFlowMeter was utilized to extract features from traffic headers. Secondly, the autoencoder approach was employed to select effective features before dataset training. Thirdly, three distinct models were employed to classify DDoS attacks, and compression was performed between them using confusion matrix metrics. The first model solely employed DNN as a classifier. The second model employed autoencoder to reduce the number of features and then utilized DNN as a classifier. The third model utilized Pearson Correlation Coefficient to diminish the number of features, followed by the application of an autoencoder for further feature selection. The outputs of the two feature selection methods were fed into the DNN classifier to make decisions. The experimental evaluation was conducted using the CICDDoS2019 dataset due to its inclusion of numerous and diverse DDoS threats.

Moreover, Sumathi and Karthikeyan [156] implemented a deep learning-based method to identify DDoS anomalies. Pertinent information and traffic statistics were stored in a database. Statistical techniques were employed to extract features such as averages, variances, and standard deviations. The normalization technique was used to scale feature values between 0 and 1, reducing scale variance effects. The normalized outputs were then fed into a deep learning neural network (DNN), a classifier with hidden layers designed to train on incoming traffic and identify DDoS attacks. The method was evaluated using various datasets including CONFICKER, DARPA MIT, and KDD Cup datasets, demonstrating accuracy close to 99%.

Finally, another DDoS detection method was proposed that combines deep learning and machine learning techniques. These detection methods include eXtreme

Gradient Boosting (XGBoost), Deep Neural Network (DNN), Random Forest (RF), Spiking Neural Network (SNN), and Decision Tree (DT). The aim of this research was to identify the most suitable method by constructing appropriate training models based on whether incoming traffic was benign or abnormal. Different feature selection techniques were also applied to select relevant features. Among these methods, ANOVA, Mutual Information, Extra Tree, and Chi-Square Test, Mutual Information was chosen due to its ability to yield higher results when utilized by machine learning classifiers. It selected 45 influential features out of the 82 generated from the CICDDoS2019 dataset. However, it's important to note that the feature selection technique requires a longer time to select relevant features [157].

2.8 Summary

DoS attacks are categorized into five kinds of attacks: flooding based attack, protocol-based attack, application level-based attack, network device level-based attack, and operating system level-based attack. Attackers have improved their tactics due to technological advancements, giving rise to a new form of malicious activity known as DDoS attacks. There are several differences between DDoS and DoS attacks which are mentioned in this chapter. Launching a DDoS attack involves several steps, including the recruitment of agents, compromising of bots, communication via protocols, and utilizing the botnet for malicious activities.

DDoS attacks can be categorized into two types based on the intensity or rate of attacks: low-rate and high-rate attacks. Many surveys and research studies have been conducted to present and depict DDoS attack taxonomies. One of these taxonomies classified DDoS attacks into two groups: bandwidth attacks and network resource

attacks. Other taxonomy was classified DDoS attacks into two main types: bandwidth attacks and network resources attacks.

Moreover, IDS can be classified into three main types based on their control mechanisms: Centralized IDS, Distributed IDS, and Hierarchical IDS are categories for IDS. However, DDoS intrusion detection system (IDS) techniques based on suspicious monitoring network traffics can be classified into two main groups: misuse detection and anomaly detection. Misuse detection identifies anomalies based on previously stored attack patterns to detect known attacks. Keystroke, rule based, signature based, and state transition are all kinds of misuse detection technique. On the other hand, anomaly-based detection techniques rely on profiling benign patterns during the training process. Any deviations from normal patterns observed during the comparison process are considered as potential attack activities. Various types of anomaly detection techniques have been employed in IDS implementations, including statistical-based techniques, machine learning-based techniques, and deep learning-based techniques. Finally, the pros and cons of each detection that were mentioned in this study are summarized in Table 2.3.

Current detection approaches that mentioned in the review earlier fail to deal with complex, recent, and high-rate attacks. This works implemented a combination of two approaches which are entropy and Sequential Probability Ratio Test in order to detect up-to-date attacks such as NTP, DNS, LDAP, MSSQL, NETBIOS, UDP, SSDP, SYN, SNMP, UDP-Lag, and TFTP. Entropy based approach has been widely researched. On the other hands, entropy relies on a threshold to make decisions about the presence of attacks. Depending on fixed or static value of threshold may lead to

decrease the effectiveness of the detection method. This is why this work combine entropy with ESRT approach in order to overcome problem of using fixing threshold during taking the decision.

Current findings also use common features that can be extracted from network traffic, such as internet protocol source address, internet protocol destination address, port source address, port destination address, or protocol name, as arguments for their detection algorithms. By ignoring these features, these finding might overlook behavioral aspects of attacks and miss the opportunity to achieve high accuracy and a lower false positive rate. Thus, this work uses different kinds of features that were extracted by CICFlowMeter tool such as total number of packets-based feature, total length of packets-based feature, flow Inter-Arrival-Time (IAT)-based feature, flags-based feature, bulk-based feature, subflow-based feature, or active and idle-based feature in order to improve these problem. Original version of CICFlowMeter tool has some errors in forming flows and features. Finally, revised version overcome these problems.

DDoS attacks involve flooding targeted system with a very large number of useless networks traffics. In its turn, extracting new features from all incoming traffics in order to analyze and examine each traffic flow will lead to consume the system resources such as storage or memory space. This will lead to increase time complexity or cost. Using relevant features are very important to reduce cost or complexity time. This work selects important features using four techniques based on machine learning which are ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation approaches.

Finally, three main contributions have been implemented in this thesis. First, multiple features of the Entropy and Sequential Probabilities Ratio Test approach (E-SPRT) were implemented to determine infected Ethernet interfaces and identify various kinds of up-to-date DDoS attacks. second, a new combination of machine learning techniques ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation methods was implemented to select relevant features in order to reduce time complexity or cost. Third, problems of the original CICFlowMeter were analyzed, and the effectiveness of revised version of CICFlowMeter tool on detection approach was demonstrated. This will involve comparing the results obtained from both versions for feature extraction to improve accuracy, F-score, and precision in the detection approach.

Table 2.3 : Summary of Pros and Cons of DDoS Detection Techniques

No.	Author name or paper	Approaches Name	Approach Categories	con (Advantages)	pro (Limitations)
1	Dao et al. [126]	Filtering Technique of IP source	Statistical	mitigates DDoS attacks against controller of SDN	It fails when the number of attacks is very huge.
2	Piedrahita et al. [127]	FlowFence technique	Statistical	fast, easy, and efficient, and mitigated DDoS attacks	This design can only mitigate DDoS threats without completely preventing it.
3	Durner et al. [128]	32-bit FNV-1a algorithm	Statistical	has low false positive and identify malicious attacks.	failed to detect malicious traffics that changed all of their field's values simultaneously
4	Dong et al. [18]	Sequential Probabilities Ratio Test (SPRT)	Statistical	higher accuracy, versatility, and speed in classifying flows identifying infected interface	only one feature which could be bypassed by attackers
5	Kousar et al. [129]	Apache Spark and machine learning techniques such as support vector machine, random forest, naïve base, and decision tree.	Statistical	reduced time needed to identify DDoS attacks and increased efficiency of detection.	Using only one dataset to evaluate their design and some selected features are common feature that may be bypassed by attacker.
6	St-Hilaire and Mousavi [29]	Entropy	Statistical	detect DDoS attacks after five hundred of infected traffics	Using only one feature and fixed threshold which could be bypassed by attackers
7	Ma and Chen [131]	variation of Lyapunov and Tsallis Entropy	Statistical	Detect DDoS attacks efficiently	Using common features which may lead to bypass detection techniques and generate high false positive rate

Table 2.3 : Continued

8	Hoque et al. [132]	Feature Feature Score (FFSc) technique	Statistical	Higher accuracy when threshold close to 0.	Changing threshold may affect on accuracy results
9	Meng and Guo [133]	Statistical based entropy	Statistical	higher accuracy in identifying slow DDoS attacks	not being able to recognize DDoS threats from flash crowd and using only one common feature.
10	Valizadeh and Niar [134]	combination of entropy-based method and packet windows initiation rate (PWI) technique	Statistical	Increase network avialability and detect attacks in multi-host attack.	Using only one feature which could be bypassed by attackers
11	Li and Wu [31]	**φ**-entropy technique	Statistical	higher detection rate	Threshold should be changed when network behavior changed in order to improve detection rate
12	Aladaileh et al. [135]	rule-based detection and Renyi joint entropy	Statistical	Using Dynamic threshold	Using common features which may lead to bypass detection techniques when network behavior changed.
13	Özçelik and Brooks [136]	combination of cumulative sum (CUSUM), wavelet, and entropy.	Statistical	produced high performance and efficiently in identifying DDoS attacks.	Using only one common feature which may lead to bypass detection techniques when network behavior changed.
14	Hoque et al. [137]	Real-time DDoS attack detection using FPGA	Statistical	high detection accuracy with high speed	Detect specific type of attacks
15	Maranhão et al. [138]	Using set of techniques such as gradient boosting, random forest, and decision tree	Machine Learning	results show that proposed system has higher error-robustness and performance	higher computational complexity
16	Koay et al. [130]	Decision Tree, Recurrent Neural Network, and Multilayer Perceptron	Machine Learning	identifying different intensity of DDoS attacks across datasets	Moderate accuracy and do not used feature selection approaches

Table 2.3 : Continued

17	Nguyen et al. [139]	HTM and K-nearest neighbors algorithm.	Machine Learning	High performance in identifying most DDoS attack types.	failed to detect some DDoS attacks such as SSDP and UDP attacks.
18	Liu et al. [28]	FEDDM	Machine Learning	Generate high accurcy in identifying WebDDoS, SNMP, NeTBIOS, and Syn attacks.	High false positive rate for UDP and UDPLag and low performance due to using scikit-learn library.
19	Long and Jinsong [140]	Hybrid Entropy SSAE-SVM	Machine Learning+Deep Learning	High performance in identifying most DDoS attack types.	Did not use feature selection methods to reduce execution time.
20	Polat et al. [141]	Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN).	Machine Learning	accuracy of testing their technique were between 91% to 98%	Number of features selected are changed when values of threshold changed. This may affect on performance of detection as well.
21	Daneshgadeh et al. [142]	SVM and Kernel-based Online Anomaly Detection (KOAD)	Machine Learning	Low false alarm rate and high true positive rate.	Changing threshold value leads to change performance of the detection process.
22	Sharafaldin et al. [22]	logistic regressing, ID3, Naïve bayes, and Random Forest	Machine Learning	Developing realistic DDo dataset and taxonomy	Not very high value of detection accuracy.
23	Ye et al. [143]	SVM	Machine Learning	accuracy value for design was 95.24%.	failed to detect some DDoS attacks such as ICMP attack.
24	Hofer-Schmitz et al. [144]	Local Outlier Factor	Machine Learning	it implemented to detect advanced persistent threat (APT) attack.	Selected suitable threshold need to be done in order to improve detection performace.
25	Nandi et al. [145]	Random Forest, Decision Table, Naïve	Machine Learning	the average detection rate of this design was	/

Table 2.3 : Continued

		Bayes, J48, and Bayes Net		99.02%.	
26	Gaur and Kumar [146]	XGBoost, Random Forest, Decision tree, and K-Nearest Neighbors	Machine Learning	quickly identify DDoS anomalies in IoT environment	parameters can be optimized in order to reduce overfitting and underfitting and increased accuracy.
27	Khoei et al [147]	stacking-based, bagging-based and boosting-based methods.	Machine Learning	Detect DDoS in smart grid environment.	Generate FPR.
28	Liu et al. [148]	Entropy and Convolutional Neural Network (CNN)	Deep Learning	higher accuracy, precision, and recall.	Authors used a set of parameters that need to be adjusted based on environment of network in order to get good detection results.
29	Yuan and Yang [149]	entropy and CNN	Deep Learning	detection efficiency has been improved and CPU load has been reduced.	Need to set suitable threshold to distinguish between normal and compromise flows.
30	Nazih et al. [150]	Recurrent Neural Networks (RNNs)	Deep Learning	higher accuracy for different attack scenarios.	Parallel processing for RNN is not effective in attacks detection.
31	Chonka et al. [151]	Chaos theory and neural network algorithm	Deep Learning	Not only detect infected traffics but also filter them out.	Using only one common feature which may lead to bypass detection techniques when network behavior changed.
32	Sarmila and Kavin [152]	Heuristic clustering algorithm	Deep Learning	minimum false positive rate	The detection rate was not too high.
33	Bista and Chitrakar [153]	Heuristics clustering algorithms and Naïve Bayes classification.	Deep Learning	Accuracy and Detection Rate has been improved	This approach fails when data distributions is not good.

Table 2.3 : Continued

34	Cepheli et al [154]	hybrid intrusion detection system (H-IDS)	Deep Learning	High Performance.	Since detection has a signature-based rule detector part, attacker may evade detection process.
35	Li [155]	Dense Neural Networks (DNN) and Pearson Correlation Coefficient (PCC).	Deep Learning	Can detect up-to-date attacks.	dimension reduction part is not used and cannot work in real time.
36	Sumathi and Karthikeyan [156]	DNN	Deep Learning	High accuracy and throughput.	No feature selection used in order to reduce complexity.
37	Vimal and Kumar [157]	eXtreme Gradient Boosting, Deep Neural Network, Random Forest, Spiking neural network, and Decision Tree.	Deep Learning	lower FPR values and moderate accuracy.	Higher time of training.

CHAPTER 3

METHODOLOGY

This chapter explains the diagram of the Multi-feature ESPRT technique. It illustrates the process of feature extraction used in the implementation. It presents the categories of features that were classified into ten groups. It also describes the feature selection methods that were used to select features. Furthermore, it justifies the preprocessing techniques applied to the dataset, such as cleaning, balancing, and encoding. The SMOTE technique, used to balance the imbalanced dataset, is also described. Finally, the detection techniques used to identify DDoS attacks are presented at the end.

3.1 Diagram of Detection Approach

This section explains the diagram of the Multi-feature ESPRT technique. Firstly, packets transferred in the network can be captured and stored in the Pcap file format for analysis. Packets with the same specifications are grouped together to form flows. A flow is a collection of packets that share similar packet fields such as IP source address, IP destination address, port source address, port destination address, protocol type, and more. These flows can be created using the CICFlowMeter tool. Each flow generates multiple features. The CICFlowMeter tool can also generate 82 features divided into 10 groups, as shown in Figure 3.1. These groups include identification-based features, total number of packets-based features, total length of packets per flow, flow IAT-based features, features related to flags, segment and header size-based features, bulk-based features, subflow-based features, TCP-based features, and active/idle-based features.

Furthermore, these flows with their features are all collected in CSV files. The quality of the dataset is tested before applying the detection techniques by employing preprocessing techniques on the CSV files, such as cleaning, balancing, and encoding the dataset. Dataset cleaning is used to remove inconsistent, infinite, or missing values resulting from machine or human errors. It can also be used to drop or rename unused or less powerful columns in the CSV files, such as flow ID, Fwd Header Length.1, Source IP address, Destination IP address, Source Port address, Destination Port address, Timestamp, unnamed:0, Inbound, and SimillarHTTP. Downsizing is another form of cleaning. Changing the data type of values, such as changing from float64, integer64, object, or string to more suitable forms, is an example of downsizing technique. This helps make the dataset more reliable and consistent, ultimately leading to improved accuracy. Another technique used is label encoding, which is used to convert categorical names into numbers. This conversion is done to render these variables in a machine-readable format. For instance, attack class names can be changed to 1, and benign class names can be changed to 0.

Moreover, datasets may be imbalanced, occurring when the distribution of normal and malicious instances in the dataset is unequal. This imbalance leads to issues such as overfitting and bias, ultimately reducing the performance and accuracy of detection algorithms. To address this, the SMOTE technique was used to balance the dataset. SMOTE achieves this by introducing new instances to the minority class, thereby increasing the number of instances in that class. These new instances are not duplicates of existing minority class instances; instead, they are synthetic. Therefore, the number of instances in benign class will be equal to that in malicious class.

Furthermore, feature selection techniques were employed to eliminate repeated, irrelevant, and unimportant features. Four feature selection techniques were used to reduce computational load and execution time. These techniques include ANOVA, Random Forest, Extra Tree, and Pearson correlation. The top ten features were selected from each method. In other words, the most 10 important features that were selected by each method. The results were then combined and sorted based on the number of occurrences of selected features. Ultimately, the ten best features from this combination were selected for use in the detection stage, as depicted in Figure 3.2, which illustrates the pseudocode of the implementation.

Finally, flows and the interfaces that permit their passage are grouped based on a fixed number of flows known as the window size. For each window, the probability of each unique feature value is calculated. The Shannon entropy is then computed based on these probability values. The entropy values serve as input for the subsequent stage, which involves the Sequential Probability Ratio Test (SPRT) to make decisions about specific flows and their associated interfaces based on the tested features, as shown in Figure 3.1.

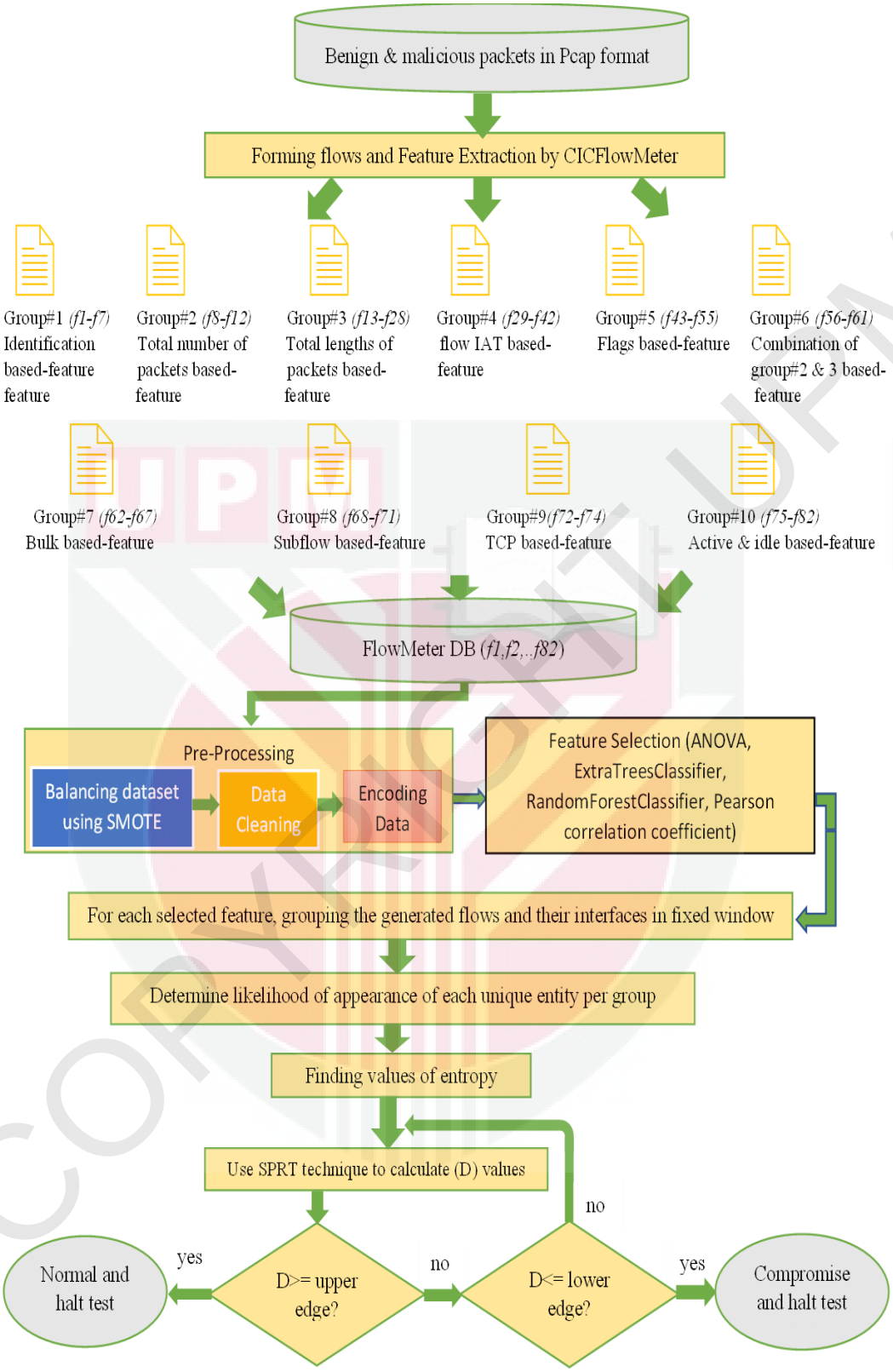


Figure 3.1 : Diagram of Multi-Features of ESPRT Detection Approach

```
Initialized Database_flows=null; // CSV file (flows in rows and features in columns)
Initialized features, Selected_features, combined_features= null;
Initialized features_ANOVA, features_ExtraTree, features_RandomForest, features_Pearson= null;
Initialized multimap_features=null;
Initialized Window_size= any number;
Initialized N=0; //number of appearances of each unique value
Initialized prob, Entropy, SPRT=0;
 Initialized Z0=0.33, Z1=0.6; // by experiment

 Database_flows =Use CICFlowMeter tool to generate bidirectional flows with multiple features;

While (Database_flows.HasMoreColumns){
    features= fetch columns from the Database_flows;
}
features_ANOVA= sort the best 10 features based on their importance using ANOVA method;
features_ExtraTree= sort the best 10 features based on their importance using ExtraTreeClassifier;
features_RandomForest= sort the best 10 features based on their importance using RandomForest;
features_Pearson= sort the best 10 features based on their correlation with the class output;
combined_features=combined results of features selection methods;

While (combined_features.HasMoreFeatures){
  N(Selected_features)++; //number of appearances of each unique feature
}

 For ( i=1 to 10){ // choose the best 10 features
    multimap_features.put (i, N(Selected_features));
 }

 While (Database_flows.HasMoreFlows){
    While (multimap_features.HasMoreFeatures){
        N(flow)++; //number of appearances of each unique flow
        If (number of flows =window_size)
            Prob=N(flow)/ window_size;
            Entropy= -prob* log2 prob;
                If ((Entropy >= 0) && (Entropy <= 0.5))
                    SPRT= SPRT+ ln(Z1/Z0);
                ElseIF (Entropy>0)
                    SPRT= SPRT+ ln((1-Z1)/(1-Z0));
                EndIF
                    If (SPRT >=upper edge)
                        Print.out ("flow is normal");
                    ElseIF (SPRT<=lower edge)
                        Print.out ("flow is compromise");
                    EndIF;
        EndIf;
       Clear SPRT, Entropy, prob, N;

    }
 }
```

Figure 3.2 : Pseudocode of Multi Feature ESPRT Technique

3.2 Feature Extraction

Data transferred over the internet can be captured and stored in Pcap format for further analysis. These data, transferred from one device to another, are referred to as packets. These packets can either be legitimate or malicious. A network packet consists of two parts: the header and the payload. The payload holds the information that needs to be moved from one place to another, such as images, files, text, or videos. Conversely, the header contains descriptions about the packet, such as source IP address, source port address, destination IP address, destination port address, flags, frame number, protocol type, and more. A flow is a group of packets that share the same field values. For instance, these fields could include source/destination IP addresses, source/destination port addresses, or protocol types for TCP or UDP-based protocols. However, for the ICMP protocol, these fields might consist of only the protocol type, source IP address, and destination IP address.

CICFlowMeter [158] is a valuable tool that serves two purposes. The first purpose is generating bidirectional flows, which can occur in both directions either forward, from source to destination, or backward, from destination to source. The second purpose is generating features. CICFlowMeter can generate nearly 82 features for each unique flow, based on specific calculations. These features can be categorized into 10 groups.

The first category is Identification-based features, containing features that identify flows. These include determining the direction a flow is moving, when a flow was captured, and the duration of the flow. The second category pertains to total number of packets-based features, encompassing the total number of packets in both the

forward and reverse directions. Another group focuses on the total length of packets per flow in either forward or backward directions, providing metrics like maximum, minimum, average, variance, and standard deviation of total packet lengths. There's also a group of Flow IAT-based features, which relate to the time between two packets in both directions, encompassing metrics such as total, maximum, minimum, average, and standard deviation times.

Another group involves Flags features-based features, which encompass attributes related to flags, such as FIN, SYN, RST, PSH, ACK, URG, CWR, and ECE. Additionally, there are groups like Segment and Header size-based features, Bulk-based features, and Subflow-based features, all generated by CICFlowMeter. Moreover, TCP-based features and Active/Idle-based features comprise other groups. Finally, the descriptions of these feature groups are provided in Table 3.1. In this manner, CICFlowMeter is employed to create flows and extract multiple features for each unique flow.

Table 3.1 : CICFlowMeter Group Features

Group #	Group name	Feature # & name	Description of feature
Group#1	Identification based feature	Source IP (f1)	Source address IP
		Destination IP (f2)	Destination address IP
		Source Port (f3)	Source address port
		Destination port (f4)	Destination port address
		Protocol (f5)	Protocol name such as TCP, UDP,…
		Timestamp (f6)	Time when flow was captured
		Flow duration (f7)	Flow duration in microseconds
Group#2	Total number of packets-based feature	Total Fwd Packets (f8)	Total number of packets in a forward direction for the observed flow.
		Total Bwd Packets (f9)	Total number of packets in a backward direction for the observed flow.
		Flow Packets/s (f10)	Total number of packets in a forward and backward direction for the observed flow over the flow duration in second.
		Fwd Packets/s (f11)	Total number of packets in a forward direction for the observed flow over the flow duration in second.
		Bwd Packets/s (f12)	Total number of packets in a backward direction for the observed flow over the flow duration in second.
Group#3	Total length of packets (the payload of app. protocol in bytes) based feature	Total Length of Fwd Packets (f13)	Total length of packets (the payload of app. protocol in bytes) in a forward direction
		Total Length of Bwd Packets (f14)	Total length of packets (the payload of app. protocol in bytes) in a backward direction
		Fwd Packet Length Max (f15)	Max length of a packet (the payload of app. protocol in bytes) in a forward direction
		Fwd Packet Length Min (f16)	Min length of a packet (the payload of app. protocol in bytes) in a forward direction
		Fwd Packet Length Mean (f17)	Mean of length of packets (the payload of app. protocol in bytes) in a forward direction
		Fwd Packet Length Std (f18)	Std of length of packets (the payload of app. protocol in bytes) in a forward direction
		Bwd Packet Length Max (f19)	Max length of a packet (the payload of app. protocol in bytes) in a backward direction
		Bwd Packet Length Min (f20)	Min length of a packet (the payload of app. protocol in bytes) in a backward direction
		Bwd Packet Length Mean (f21)	Mean of length of packets (the payload of app. protocol in bytes) in a backward direction
		Bwd Packet Length Std (f22)	Std of length of packets (the payload of app. protocol in bytes) in a backward direction

Table 3.1 : Continued

		Flow Bytes/s (f23)	Total length of packets (the payload of app. protocol in bytes) in a forward and backward direction over the flow duration in second.
		Min Packet Length (f24)	Min of length of packets (the payload of app. protocol in bytes) in a forward and backward direction.
		Max Packet Length (f25)	Max of length of packets (the payload of app. protocol in bytes) in a forward and backward direction.
		Packet Length Mean (f26)	Mean of the Total length of packets (the payload of app. protocol in bytes) in a forward and backward direction.
		Packet Length Std (f27)	Std of the Total length of packets (the payload of app. protocol in bytes) in a forward and backward direction.
		Packet Length Variance (f28)	Variance of the Total length of packets (the payload of app. protocol in bytes) in a forward and backward direction.
Group#4	Flow IAT based feature	Flow IAT Mean (f29)	Mean of Inter-Arrival-Time (IAT) between two packets sent in the flow
		Flow IAT Std (f30)	Standard deviation of Inter-Arrival-Time (IAT) between two packets sent in the flow
		Flow IAT Max (f31)	Maximum of Inter-Arrival-Time (IAT) between two packets sent in the flow
		Flow IAT Min (f32)	Minimum of Inter-Arrival-Time (IAT) between two packets sent in the flow
		Fwd IAT Total (f33)	Total of Inter-Arrival-Time (IAT) between two packets sent in the forward direction
		Fwd IAT Mean (f34)	Mean time between two packets sent in the forward direction
		Fwd IAT Std (f35)	std time between two packets sent in the forward direction
		Fwd IAT Max (f36)	max time between two packets sent in the forward direction
		Fwd IAT Min (f37)	min time between two packets sent in the forward direction
		Bwd IAT Total (f38)	Total time between two packets sent in the backward direction
		Bwd IAT Mean (f39)	Mean time between two packets sent in the backward direction
		Bwd IAT Std (f40)	Std time between two packets sent in the backward direction
		Bwd IAT Max (f41)	Max time between two packets sent in the backward direction
		Bwd IAT Min (f42)	Minimum time between two packets sent in the backward direction

Table 3.1 : Continued

Group#5	Flags based feature	Fwd PSH flags (f43)	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
		Bwd PSH Flags (f44)	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
		Fwd URG Flags (f45)	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
		Bwd URG Flags (f46)	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
		FIN Flag Count (f47)	Number of packets with FIN
		SYN Flag Count (f48)	Number of packets with SYN
		RST Flag Count (f49)	Number of packets with RST
		PSH Flag Count (f50)	Number of packets with PUSH
		ACK Flag Count (f51)	Number of packets with ACK
		URG Flag Count (f52)	Number of packets with URG
		CWR Flag Count (f53)	Number of packets with CWR
		ECE Flag Count (f54)	Number of packets with ECE
		Up/down ratio (f55)	Download and upload ratio
Groups#6	Feature based on combination of Group#2 and 3 (segment and header size) based feature	Average Packet Size (f56)	Average size of packet =(f13+f14)/(f8+f9)
		Avg Fwd Segment Size (f57)	Average size observed in the forward direction= f13/f8
		Avg Bwd Segment Size (f58)	Average size observed in the backward direction=f14/f9
		Min_seg_size_forward (f59)	Minimum segment size observed in the forward direction
		Fwd Header Length (f60)	Total bytes used for headers in the forward direction
		Bwd Header Length (f61)	Total bytes used for headers in the backward direction
Group#7	Bulk based feature	Fwd Bytes/Bulk Avg (f62)	Average number of bytes bulk rate in the forward direction
		Fwd Packet/Bulk Avg (f63)	Average number of packets bulk rate in the forward direction
		Fwd Bulk Rate Avg (f64)	Average number of bulk rate in the forward direction

Table 3.1 : Continued

		Bwd Bytes/Bulk Avg (f65)	Average number of bytes bulk rate in the backward direction
		Bwd Packet/Bulk Avg (f66)	Average number of packets bulk rate in the backward direction
		Bwd Bulk Rate Avg (f67)	Average number of bulk rate in the backward direction
Group#8	Subflow based feature	Subflow Fwd Packets (f68)	The average number of packets in a sub flow in the forward direction
		Subflow Fwd Bytes (f69)	The average number of bytes in a sub flow in the forward direction
		Subflow Bwd Packets (f70)	The average number of packets in a sub flow in the backward direction
		Subflow Bwd Bytes (f71)	The average number of bytes in a sub flow in the backward direction
Group#9	TCP based features	Fwd Init Win bytes (f72)	The total number of bytes sent in initial window in the forward direction
		Bwd Init Win bytes (f73)	The total number of bytes sent in initial window in the backward direction
		Fwd Act Data Pkts (f74)	Count of packets with at least 1 byte of TCP data payload in the forward direction
Group#10	Active and Idle based features	Active Min (f75)	Minimum time a flow was active before becoming idle
		Active Mean (f76)	Mean time a flow was active before becoming idle
		Active Max (f77)	Maximum time a flow was active before becoming idle
		Active Std (f78)	Standard deviation time a flow was active before becoming idle
		Idle Min (f79)	Minimum time a flow was idle before becoming active
		Idle Mean (f80)	Mean time a flow was idle before becoming active
		Idle Max (f81)	Maximum time a flow was idle before becoming active
		Idle Std (f82)	Standard deviation time a flow was idle before becoming active

3.3 Revised Version of CICFlowMeter

CICFlowMeter was employed to generate bidirectional flows and multiple features, nearly 82 for each flow. These features were categorized into 10 groups, as previously shown in Table 3.1. The initial version of CICFlowMeter, proposed by Sharafaldin et al. [22] had several issues that impacted the performance of detection approaches utilizing CICFlowMeter-generated features as shown in figure 3.3. These issues primarily arose with flows based on the TCP protocol. According to the TCP specification [159], a TCP flow is constituted by grouping packets with identical specifications to facilitate communication between two parties. The TCP connection is established through the exchange of three-way handshake packets between the two sides. The sender initiates by sending a SYN packet to the receiver, which responds with a SYN/ACK packet. The sender then completes the handshake with an ACK packet, marking the opening of the connection for data transfer.

TCP connections are closed through two methods. The first is initiated by the sender sending a FIN packet to indicate transmission termination on the sender's side. The receiver responds with both ACK and FIN packets, signifying termination on the receiver's side. An ACK packet is then returned from the sender to officially close the connection. The second method involves sending an RST packet from one side to terminate the flow's connection.

However, the flows of TCP generated by the original CICFlowMeter are terminated in two ways. The first occurs when the setup timer expires, exceeding 120 seconds. Alternatively, the exchange of FIN and ACK packets in one direction, either from sender to receiver or vice versa, is considered another way to close the connection.

These two conditions are crucial for forming a flow, and without them, a flow cannot be established. Yet, these conditions violate the TCP specification and have several implications for flow formation.

First, when the proposed time limit expires without a flow being formed, the remaining TCP packets belonging to that flow will establish their own separate flow. This incorrect formation process disrupts the accuracy of flow generation. Secondly, this approach omits flows that lack these specific packets, such as flows with a minimal number of packets, such as SYN attack flows. It also disregards flows that utilize RST packets for termination. Consequently, all flows involving RST packets are omitted from the flow formation process, contrary to TCP specification.

The revised edition of CICFlowMeter [160] addresses the issues present in the original version. It considers RST packets as a valid way to terminate flows, follows TCP specification by accounting for FIN and ACK packets from both sides of the connection, and ensures that flows are correctly formed.

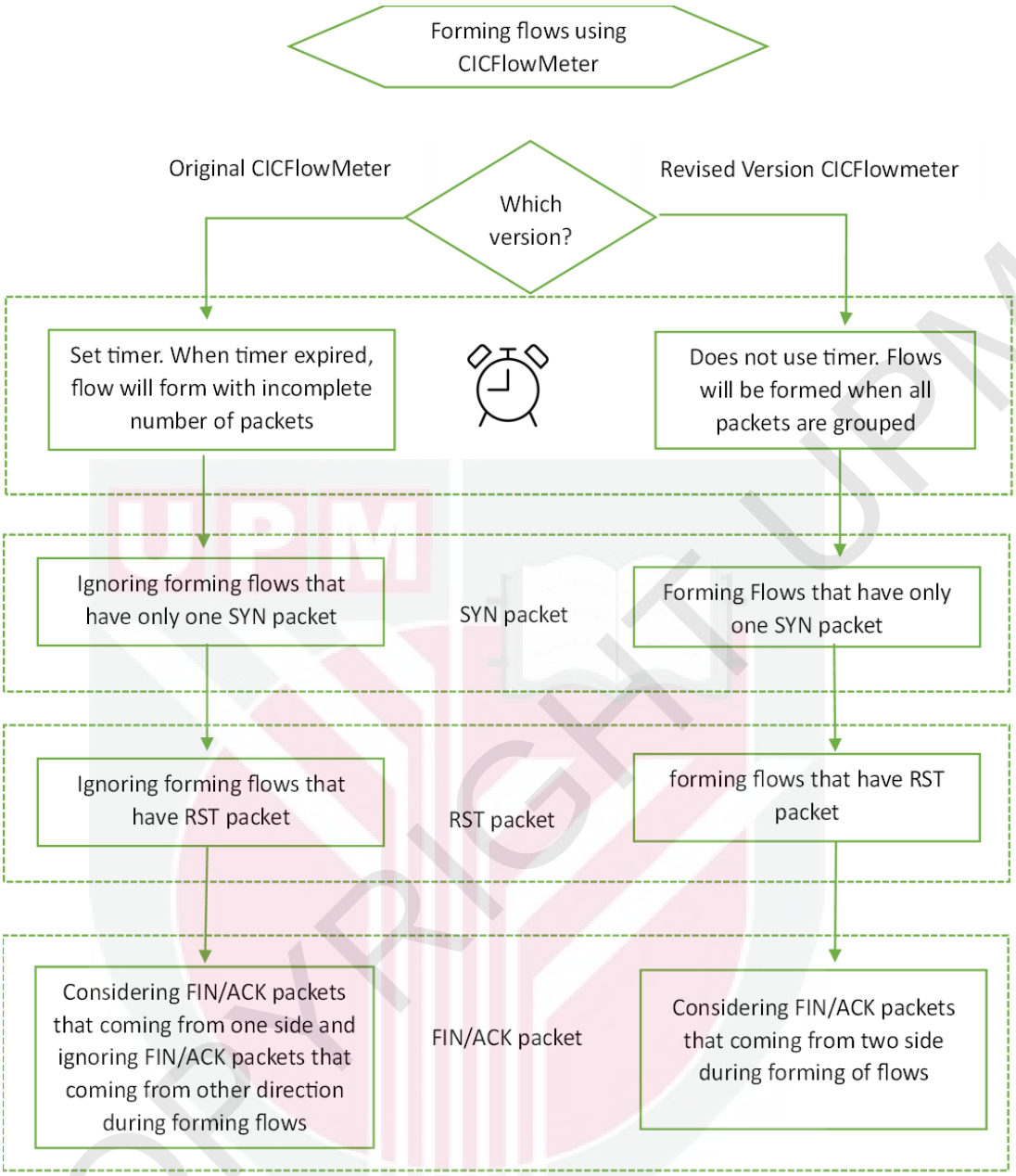


Figure 3.3 : Improvements of the Revised Version of the CICFlowMeter Tool Compared to the Original Version Digram

3.4 Feature Selection

Feature selection is an important step for various reasons. It is helpful in removing meaningless, irrelevant, and repetitive features, which in turn reduces detection errors and increases the accuracy of the identification process. The computational load and execution time are decreased when the number of features is reduced. As a

result, a combination of four selection methods was used to identify effective features. The results of the feature selections were collected. Then, the best 10 features with the highest occurrences were chosen as inputs for the next step. These methods include ANOVA, Extra Tree, Random Forest, and Pearson Correlation Coefficient.

We selected these four techniques because these techniques generated higher value of sensitivity, specificity, accuracy, and F1-score comparing with other feature selection methods. These techniques also generated lower values of probability of false alarm and miss-rate comparing with others. An experiment was conducted on six features selection techniques to show why these techniques are better than others. The experiment was presented in section (4.3).

3.5 Detection Technique

The features selected using the previous techniques are fed into this stage. Incoming flows and the interfaces through which these flows pass toward the targeted devices are gathered into groups. Each group has a fixed size known as the window size. The window size can be determined based on the number of flows or a certain time interval. For each group of flows, Shannon Entropy will be calculated for the selected features only. The results of the Entropy calculation will serve as input for the next stage, which is the SPRT technique. Finally, the SPRT takes a decision and determines whether the flows and their associated switch interfaces are normal or infected.

Current detection approaches that mentioned in the review earlier fail to deal with complex, recent, and high-rate attacks. This works implemented a combination of two approaches which are entropy and Sequential Probability Ratio Test in order to detect up-to-date attacks such as NTP, DNS, LDAP, MSSQL, NETBIOS, UDP, SSDP, SYN, SNMP, UDP-Lag, and TFTP. Entropy based approach has been widely researched. On the other hands, entropy relies on a threshold to make decisions about the presence of attacks. Depending on fixed or static value of threshold may lead to decrease the effectiveness of the detection method. This is why this work combine entropy with ESRT approach in order to overcome problem of using fixing threshold during taking the decision.

Current findings also use common features that can be extracted from network traffic, such as internet protocol source address, internet protocol destination address, port source address, port destination address, or protocol name, as arguments for their detection algorithms. By ignoring these features, these finding might overlook behavioral aspects of attacks and miss the opportunity to achieve high accuracy and a lower false positive rate. Thus, this work uses different kinds of features that were extracted by CICFlowMeter tool such as total number of packets-based feature, total length of packets-based feature, flow Inter-Arrival-Time (IAT)-based feature, flags-based feature, bulk-based feature, subflow-based feature, or active and idle-based feature in order to improve these problems. Original version of CICFlowMeter tool has some errors in forming flows and features. Finally, revised version overcome these problems.

DDoS attacks involve flooding targeted system with a very large number of useless networks traffics. In its turn, extracting new features from all incoming traffics in order to analyze and examine each traffic flow will lead to consume the system resources such as storage or memory space. This will lead to increase time complexity or cost. Using relevant features are very important to reduce cost or complexity time. This work selects important features using four techniques based on machine learning which are ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation approaches.

3.5.1 Shannon Entropy Technique

Shannon Entropy was used to measure the randomness of traffic flows based on their selected feature values. The values of entropy are high when traffic flows are random. However, when the entropy value is lower, traffic flows are not random. The randomness helps determine whether attacks are in progress or not. The feature values for flows and their corresponding interfaces are grouped into a fixed window size (W), where (i) is the number of unique feature values. The window size can be determined based on a fixed number of flows.

3.5.2 SPRT Technique

The entropy results are fed into this stage. SPRT will analysis incoming entropy values that was calculated based on extracted features of flows. SPRT will also monitor switch ethernets that let these flows passing through. The Sequential Probability Ratio Test (SPRT) is a statistical technique introduced by Waled and based on mathematical calculation. It uses two hypothesis values, H1 and H2, to differentiate between normal and compromised flows and to locate compromised

interfaces. H1 is used to determine interfaces that have been injected with normal flows, while H2 is used to determine those injected with compromised flows based on feature values of flows [18].

3.6 Dataset

Multiple datasets were utilized to test the implemented techniques. Among them was the CICDDoS2019 dataset, comprising multiple datasets captured over a span of two days. Another dataset used for analysis is the DARPA (Defense Advanced Research Projects Agency) intrusion detection dataset, captured by a group of researchers from MIT (Massachusetts Institute of Technology) Lincoln Laboratory. Both of these datasets are accessible online.

3.6.1 CICDDoS2019 Dataset

CICDDoS2019 [22] stands for Canadian Institute for Cybersecurity Distributed Denial of Service attacks. It was curated by researchers from this institute and encompasses the latest DDoS intrusions, encompassing UDP, SYN, DNS, MSSQL, SSDP, UDP-lag, TFTP, SNMP, NETBIOS, and LDAP attacks. The dataset contains benign as well as actual DDoS malicious data captured and stored in Pcap format, alongside a set of datasets stored in CSV file format. These CSV files were generated from Pcap files using CICFlowMeter, and they encompass labeled flows, each with multiple features produced by CICFlowMeter. Each flow was constructed from packets sharing common specifications, such as source/destination IP, source/destination port, and protocol type.

The testbed design responsible for generating this dataset encompasses various networking devices, including personal computers, switches, routers, and servers. The targeted computers are fortified with a firewall. Benign traffic was generated using the B-Profile system, simulating the conceptual behavior of human communications. This behavior was extracted from 25 clients, each employing different protocols such as HTTP, SSH, FTP, emails, and HTTPS [147]. Conversely, malicious traffic was generated using specialized tools and third-party applications. Details regarding the type of device, installed operating system, and IP addresses for each device are presented in Table 3.2.

Table 3.2 : Specification of Networking Devices in the Testbed [158]

Device	Operating system	IP addresses
Server	Ubuntu 16.04 (Web Server)	192.168.50.1 (first day)
		192.168.50.4 (second day)
Firewall	Fortinet	205.174.165.81
Devices (First Day)	Win 7	192.168.50.8
	Win Vista	192.168.50.5
	Win 8.1	192.168.50.6
	Win 10	192.168.50.7
Devices (Second Day)	Win 7	192.168.50.9
	Win Vista	192.168.50.6
	Win 8.1	192.168.50.7
	Win 10	192.168.50.8

The traffic compiled in the dataset was monitored and captured from the testbed design over a span of two days. The first day was January 12, 2018, spanning from 10:35 to 17:15. During this period, the victim was subjected to a range of DDoS attacks by the attacker, including NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, UDP, UDP-Lag, SYN, and TFTP attacks. The second day was March 11, 2018, during which the attacker targeted the victim with various DDoS attacks between 9:43 and 17:35. The DDoS attack types that targeted the victim on the second day

included PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN attacks. Table 3.3 below showcases the initiation and conclusion times, along with the count of infected and normal flows for each attack.

Table 3.3 : Details of Attack Types for CICDDoS2019 Dataset

Days	Attack database	Attack times	Infected Flow count in dataset	Normal flow count in dataset
(January 12th)	NTP	10:35 - 10:45	1,202,642	14,365
	DNS	10:52 - 11:05	5,071,011	3,402
	LDAP	11:22 - 11:32	2,179,930	1,612
	MSSQL	11:36 - 11:45	4,522,492	2,006
	NetBIOS	11:50 - 12:00	4,093,279	1,707
	SSDP	12:27 - 12:37	2,610,611	763
	UDP	12:45 - 13:09	3,134,645	2,157
	UDP-Lag	13:11 - 13:15	366,900	3,705
	SYN	13:29 - 13:34	1,582,289	392
	TFTP	13:35 - 17:15	20,082,580	25,247
(March 11th)	PortScan	9:43 - 9:51	186,960	4,734
	NetBIOS	10:00 - 10:09	3,454,578	1,321
	LDAP	10:21 - 10:30	2,108,110	5,124
	MSSQL	10:33 - 10:42	5,772,992	2,794
	UDP	10:53 - 11:03	3,779,072	3,134
	UDP-Lag	11:14 - 11:24	721,097	4,068
	SYN	11:28 - 17:35	4,284,751	35,790

3.6.2 DARPA Dataset

DARPA [35], [36], standing for Defense Advanced Research Projects Agency, was a research initiative carried out by MIT's (Massachusetts Institute of Technology) Lincoln Laboratory's Cyber Systems and Technology Group in collaboration with the Air Force Research Laboratory. The project's objective was to assess intrusion detection systems based on their accuracy and false positive error rates. The evaluation system was designed to be straightforward and has since garnered interest from numerous researchers in the realm of network and cyber security. In the years

1998, 1999, and 2000, various datasets containing different types of intrusions, including DDoS attacks, were generated and captured.

In addition, the 1998 intrusion detection evaluation dataset [35] consists of two categories of datasets: those generated from real-time tests and off-line tests. These generated datasets were further divided into two sets: training and testing datasets. The training data was amassed in Pcap format and stored in Tcpdump files over a span of seven weeks. Conversely, the testing data was collected over two weeks. Notable examples of DDoS attacks generated in 1998 include Smurf, Neptune (SYN flood), Pod, Teardrop, and Satan.

Among these datasets, the 'Friday.tcpdump' and 'Thursday.tcpdump' files from the fifth week of training data were selected for evaluation. The 'Friday.tcpdump' dataset encompasses (1,253,312) network traffic packets, grouped into (256,055) flows. A substantial portion of these flows consisted of TCP flows, amounting to (253,397). The remaining flows were divided between UDP flows, totaling (1,596), and ICMP flows, totaling (1,054). Attackers often employed a significant number of low flows to overwhelm the target system and cause its failure. Within the 'Friday.tcpdump' dataset, the attackers initiated an influx of low flows at (17:27:07), as depicted in Figure 3.4. They launched a Neptune attack, leading to a surge of SYN flows directed towards the victim. The attack persisted with a Smurf attack starting at (18:00:15). This attack exploits vulnerabilities in the IP and ICMP protocols.

Random Forest Steps

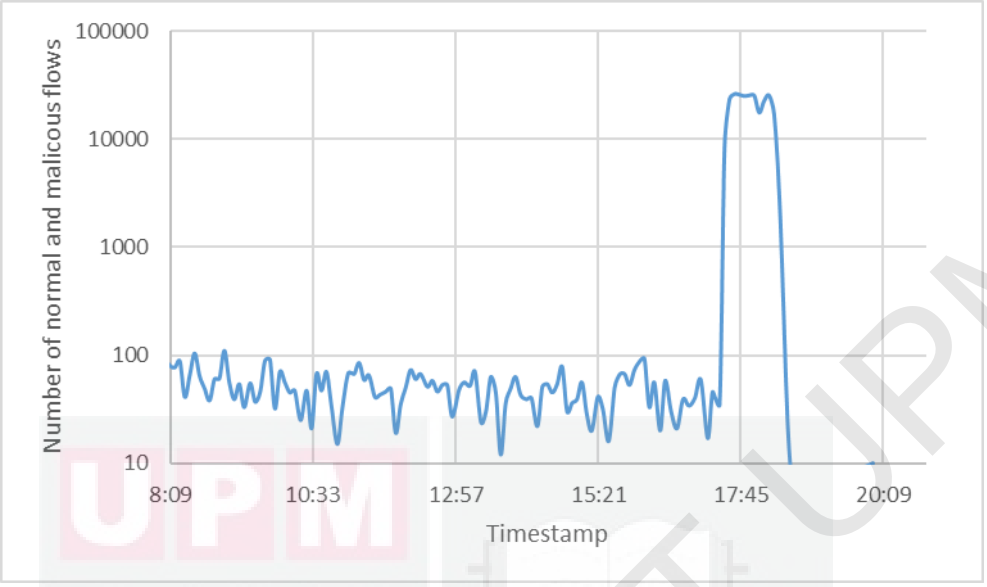


Figure 3.4 : Number of Benign and Malicious Flows per Timestamp for Friday.tcpdump Dataset

Moreover, the 'Thursday.tcpdump' dataset comprises (2,319,408) packets distributed among (219,606) distinct flows. Out of these flows, (211,606) were based on TCP, (929) were rooted in ICMP, and the rest were UDP-based. During the dataset capture, the attacker initiated numerous low flows at different intervals, as visualized in Figure 3.5. For instance, the Neptune attack generated a considerable number of low flows at various times such as (8:16), (9:33), (10:05), and (11:23). Another attack type evident in this dataset was the Smurf attack, which triggered a large number of flows in a short timeframe, commencing at (11:57), as shown in Figure 3.5.

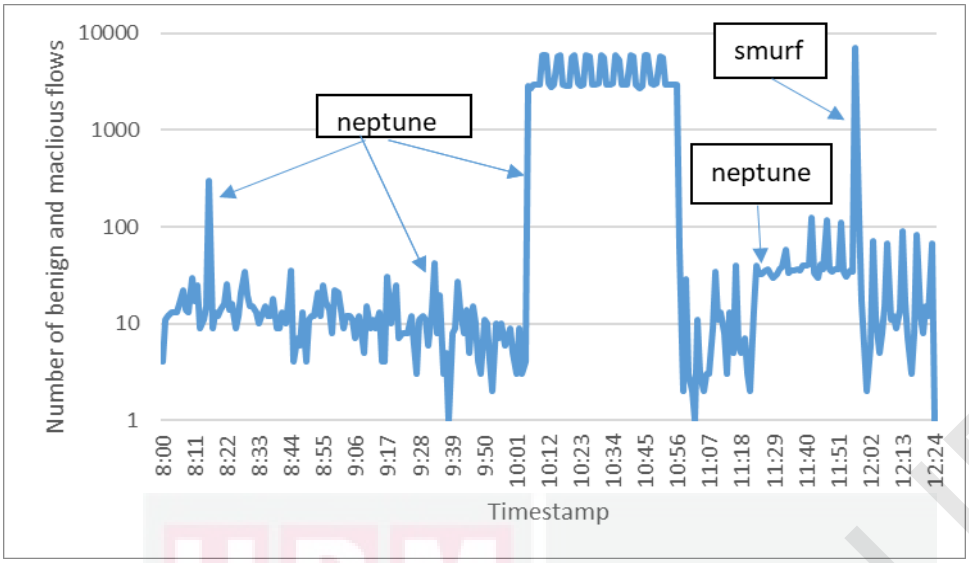


Figure 3.5 : Number of all Flows per Timestamp for Thursday.tcpdump Dataset

Finally, the 1999 intrusion detection evaluation dataset [35] comprises three weeks of training data and three weeks of test data. While the second week of the training data encompasses different types of attacks, the remaining weeks of the test data are considered normal. The 2000 intrusion detection evaluation dataset [36] represents an updated version of the 1999 dataset and encompasses five stages of intrusions. In the final stage, the attacker initiated a DDoS attack by gaining access to the victim's system via telnet. Subsequently, the attacker targeted the victim with an extensive number of spoofed IP source flows, approximately (73,700), as depicted in Figure 3.6. This attack lasted for a mere five seconds, during which flows were directed towards random TCP ports on the victim's system with the intention of overloading it. This dataset consists of 649,787 packets, which are classified into 103,040 flows. Among these flows, 82,755 were TCP-based, 20,251 were UDP-based, and the remaining were ICMP-based.

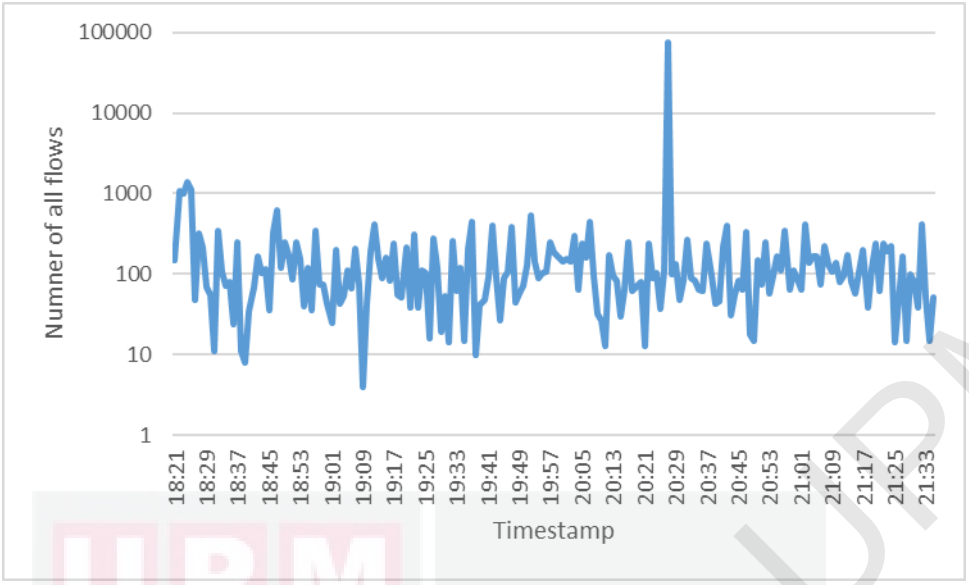


Figure 3.6 : Number of all Flows per Timestamp for 2000 Dataset

CHAPTER 4

RESULTS AND DISCUSSION

This chapter outlines the hardware configuration employed for the experiment. It also elucidates the datasets and confusion matrix metrics used to evaluate the detection technique. It presents the impact of data preprocessing on the dataset, including cleaning, balancing, and transforming the dataset. Furthermore, it describes the results of feature selection techniques. The chapter illustrates the effect of selecting the best 10 features using a combination of feature selection techniques like ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation on the performance of the detection process. Moreover, it showcases the results of ESPRT using both the complete set of features from CICFlowMeter and the selected features from feature selection techniques, across various datasets, and utilizing significant confusion metrics. This chapter highlights the distinctions between the multi-feature ESPRT and other state-of-the-art techniques. Lastly, it demonstrates the efficacy of ESPRT when comparing the revised version of CICFlowMeter with the original version, by juxtaposing their respective results.

4.1 Hardware Configuration

The specifications of the machine used for the experiment are as follows: The operating system was Windows 10 Pro, version 21H2, and it was a 64-bit operating system with an OS build of 19044.2604. The processor used was an Intel(R) Core(TM) i7-7500U, with a speed range of 2.70 GHz to 2.90 GHz. This processor was an x64-based processor. Moreover, the RAM size was 16.0 GB, and the

graphics processing unit utilized was an Intel(R) HD Graphics 620, with a total graphics memory size of 8253MB. Finally, the BIOS Information type was Intel Video BIOS.

4.2 Data Preprocessing

Preprocessing is a crucial step before the detection technique process. It readies the dataset for building and training identification techniques. Preprocessing eliminates errors such as missing, duplicate, irrelevant, or incomplete data values caused by machines or normal users. This process enhances dataset consistency and reliability, leading to increased accuracy and decreased error rates in the results. Various data preprocessing techniques are available, and some of them are listed below.

4.2.1 Cleaning Dataset

The dataset can be cleaned by selecting the most effective features or columns during the identification process and removing less powerful columns or features during the detection technique from the original dataset. Some of these features serve as metadata for a particular dataset. Metadata can provide informative details about a dataset, including comments, personal information, headers, network details, and file characteristics. For instance, features such as source IP address (f1), destination IP address (f2), source IP port address (f3), destination port address (f4), and timestamp (f6) have been excluded from the dataset. These features do not reflect attack behavior; they solely provide dataset details. Experimental evidence shows that these features are highly susceptible to shortcut learning.

Several flag-based category features were also excluded because their behavior during benign conditions mirrors their behavior in malicious statuses. Examples of features based on flags, such as Bwd PSH Flags (f44), Fwd URG Flags (f45), Bwd URG Flags (f46), FIN Flag Count (f47), PSH Flag Count (f50), and ECE Flag Count (f54), were removed from all attack datasets. SYN Flag Count (f48) was excluded from all attack datasets of January 12th of CICDDoS2019, except for the SYN, UDPLag, TFTP, and NTP attacks dataset, as this feature generates different behavior in malicious flows compared to normal flows. The bulk-based feature group (f62 to f67) was also removed for the same reason, as shown in Table 4.1. Finally, the remaining features were used in the detection process.

Additionally, removing ineffective samples or rows from the dataset can improve the detection technique process. Some rows may contain spaces, NaN, null, or infinity values, which usually occur when data is incorrectly inputted. In other words, mistakes or errors in handling the information can generate incorrect, incomplete, or inaccurate values. The number of samples or instances removed for each attack dataset is provided in Table 4.1. For instance, 36,403 samples were omitted from the UDPLag attack dataset, and 10,623 instances containing NaN or infinite values were eliminated from the SNMP attack dataset. Therefore, removing these values is crucial to ensure the reliability and accuracy of the detection technique.

Table 4.1 : Details of January 12th of CICDDoS2019 Dataset after and before Data Cleaning

#	Attack dataset names	# Features before cleaning	# Features after cleaning	Name of features removed	# Samples before cleaning	# Samples after cleaning
1	NETBIOS	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	4,094,986	3,965,133
2	SSDP	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	2,611,374	2,569,324
3	UDP	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	3,136,802	3,096,129
4	DNS	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	5,074,413	4,912,019
5	LADP	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	2,181,542	2,142,892
6	MSSQL	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	4,524,498	4,398,032
7	NTP	82	65	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f62, f63, f64, f65, f66, f67.	1,217,007	1,209,961
8	UDPLag	82	65	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f62, f63, f64, f65, f66, f67.	370,605	334,202
9	SYN	82	65	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f62, f63, f64, f65, f66, f67.	1,582,681	1,380,364
10	TFTP_Mini	82	65	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f62, f63, f64, f65, f66, f67.	1,048,575	1,028,917
11	SNMP	82	64	f1, f2, f3, f4, f6, f44, f45, f46, f47, f50, f54, f48, f62, f63, f64, f65, f66, f67.	5,161,377	5,150,754

4.2.2 Balancing Dataset

When the number of malicious instances is greater than the number of benign instances in a given dataset, or vice versa, the dataset is referred to as an unbalanced dataset. An unbalanced dataset introduces bias towards the majority class and tends to overlook the minority class during the detection technique. It also leads to the aforementioned overfitting problems. The January 12th datasets within the CICDDoS2019 dataset exhibit this issue, as depicted in Figure 4.1. For instance, in the NETBIOS dataset, the number of malicious instances is (3,963,446), whereas the number of benign instances is a mere (1,687). Similarly, there exists a substantial disparity between the majority instances, which total (2,568,569), and the minority instances, which are only (755), as shown in the figure below.

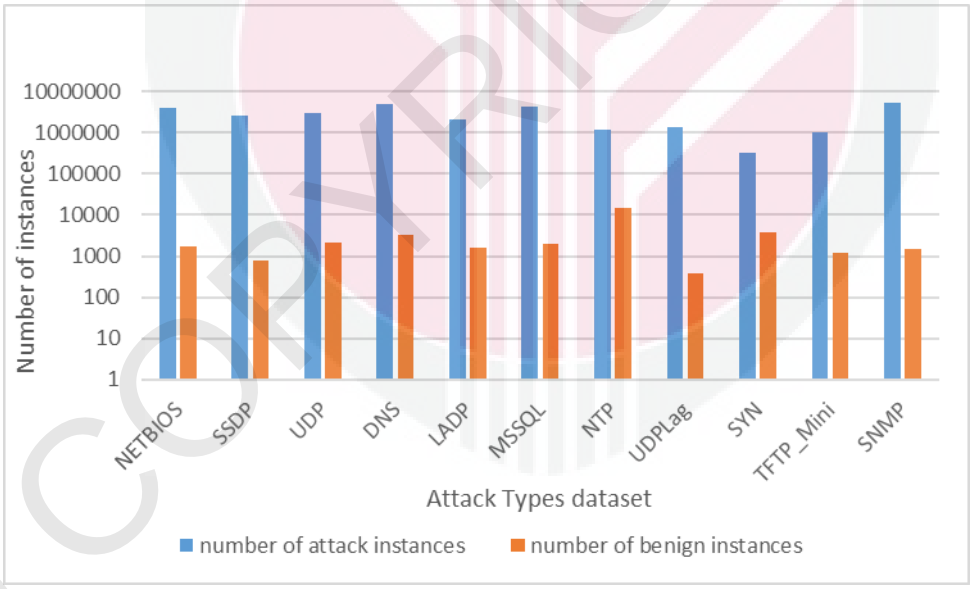


Figure 4.1 : Unbalanced Datasets for First Day of CICDDoS2019

The SMOTE technique is employed to address this problem by equalizing the number of malicious instances with the number of benign instances through the addition of new synthetic instances to the minority class, as illustrated in Figure 4.2.

These new instances are generated randomly between instances of the smaller class and their neighbors. For instance, applying the SMOTE technique to the UDP attack dataset results in an equal number of majority and minority instances, which totals (3,094,002), as shown in Figure 4.2. In another case, the SMOTE technique ensures parity between the number of malicious instances and the number of benign instances, which stands at (330,079) for the SYN attack dataset, as displayed in Figure 4.2. Ultimately, a balanced dataset resolves the overfitting and bias issues.

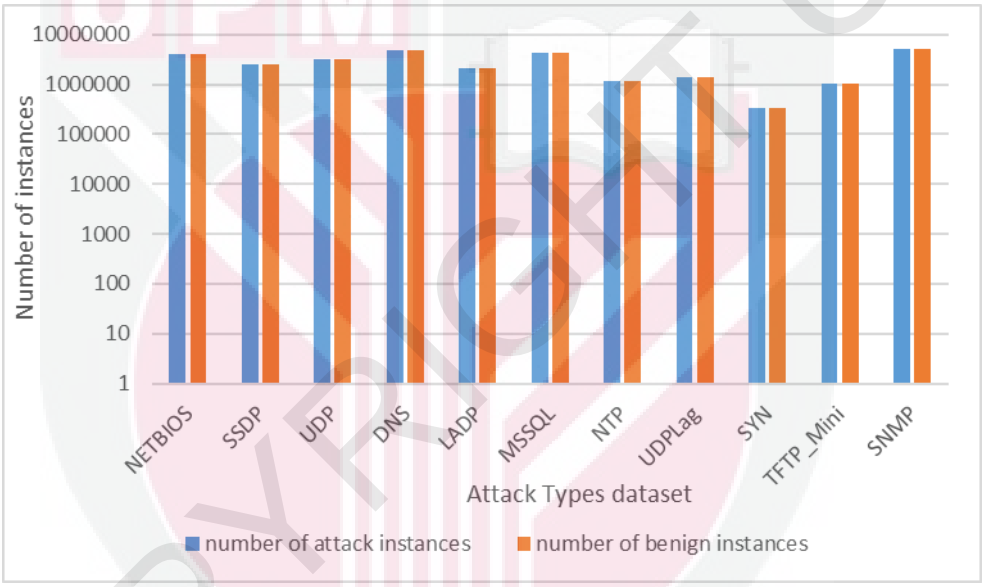


Figure 4.2 : Balanced Datasets for First Day of CICDDoS2019

4.2.3 Transformation

The dataset comprises various types of data features, some of which are based on nominal values. These values use numbers as identifiers to represent specific attributes. Since feature selection techniques are grounded in machine learning and cannot directly handle nominal data, these values are converted into numerical or decimal forms to make them compatible with these techniques. As an example, the Protocol feature contains protocol names stored as values, such as TCP, UDP,

ICMP, and others, within its cells. These protocol names are translated into numbers that have been assigned by the Internet Assigned Numbers Authority (IANA).

For instance, the TCP protocol is translated into the value (6), while the UDP protocol is represented as (17). Similarly, ICMP protocol is indicated by (1), and other protocols follow the same assigning methodology. Likewise, the output labels are transformed into numerical equivalents using the Label Encoding technique, replacing their nominal values. This technique assigns a unique numerical identifier, starting from one, to each class. For instance, the attack label names such as SYN and NTP are assigned the value (1), while the benign label is designated as (0). These transformations ensure that the feature selection techniques function effectively.

4.3 Feature Selection Results

This section will present the results of the feature selection methods used to identify the best group of features from the pool of 82 features extracted from the January 12th datasets within the CICDDoS2019 dataset. The primary purpose behind employing these feature selection methods was to reduce execution time and enhance the performance of the detection method.

The basic of choosing the techniques of ANOVA, Random Forest, Extra Tree, and Pearson correlation among others was because these techniques generated higher value of sensitivity, specificity, accuracy, and F1-score comparing with other feature selection methods. These techniques also generated lower values of probability of false alarm and miss-rate comparing with others. An experiment was conducted on

six features selection techniques. ANOVA, Random Forest, Extra Tree, and Pearson correlation, Chi-square, and Mutual Information approaches were trained and tested for that purpose. A subset of CICDDoS2019 dataset was chosen to run the experiment, and this subset was NETBIOS dataset. The train set of NETBIOS dataset contains 30,533 benign and 40,500 of malicious instances. However, the test set contains 3,963,446 of malicious instances and 3,100,100 of benign instances.

The value of TPR, TNR, accuracy, and F1-score were close to 0.99 for ANOVA, Random Forest, Extra Tree, and Pearson correlation while values for these metrics for chi-square and mutual information are close to 0.8 in the training dataset as shown in Table 4.2 or even in the testing dataset as shown in Table 4.3. On the other hands, values of FPR and FNR were closed to 0.001 for ANOVA, Random Forest, Extra Tree, and Pearson correlation which is better than value of FPR and FNR which was close to 0.1 as shown in Table 4.2 and Table 4.3 below.

Table 4.2 : Training Confusion Matrix for Different Feature Selection Approaches

Confusion Metrics	Feature Selection Techniques					
	ANOVA	Random Forest	Extra Tree	Pearson correlation	Chi-square	Mutual Information
TPR	0.9996	0.9994	0.9995	0.9998	0.8634	0.8714
FPR	0.0032	0.0035	0.0029	0.0016	0.1803	0.1
TNR	0.9967	0.9964	0.9970	0.9983	0.8196	0.9
FNR	0.0003	0.0005	0.0004	0.0001	0.1365	0.1285
Accuracy	0.9984	0.9981	0.9984	0.9991	0.8446	0.8831
F1-Score	0.9986	0.9983	0.9986	0.9992	0.8638	0.8978

Table 4.3 : Testing Confusion Matrix for Different Feature Selection Approaches

Confusion Metrics	Feature Selection Techniques					
	ANOVA	Random Forest	Extra Tree	Pearson correlation	Chi-square	Mutual Information
TPR	0.9949	0.9899	0.9845	0.9830	0.8350	0.8011
FPR	0.0033	0.0098	0.02013	0.0081	0.0544	0.06715
TNR	0.9966	0.99016	0.9798	0.99186	0.9455	0.9328
FNR	0.0050	0.0100	0.0154	0.0169	0.1649	0.1988
Accuracy	0.9956	0.9900	0.9825	0.9874	0.8738	0.8442
F1-Score	0.9961	0.9911	0.9844	0.9874	0.8957	0.8737

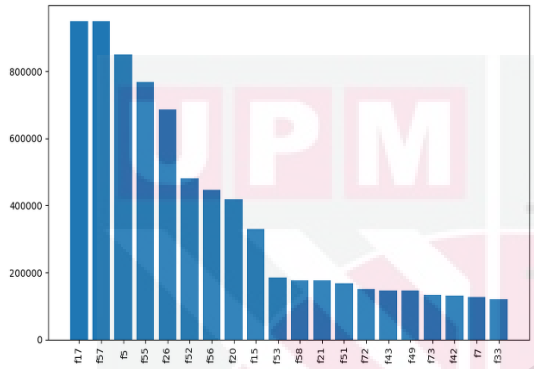
4.3.1 ANOVA Results

ANOVA was utilized to pinpoint the top 20 features among the 82 extracted via CICFlowMeter for each flow. These best features were then selected and ranked based on their respective weights. The optimal features for various attack datasets on the First Day of CICDDoS2019 are visualized in Figure 4.3. For instance, the primary feature for DNS, LDAP, MSSQL, NETBIOS, SSDP, UDP, and SNMP attack datasets was the protocol (f5), as indicated in the figure. Notably, while the protocol feature (f5) made it into the first 20 selected features for NTP and SYN, it was absent from these sets in the cases of TFTP and UDP-Lag attack datasets.

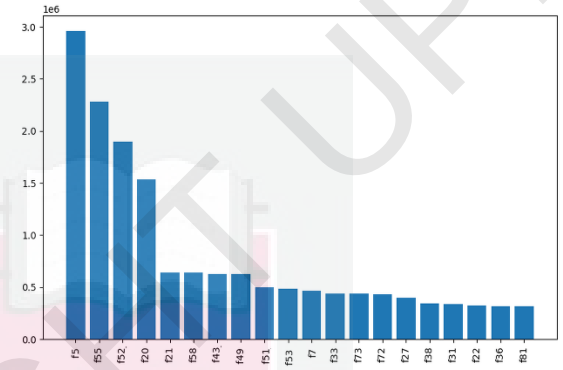
As for the second and third top features for DNS, LDAP, MSSQL, and NETBIOS, they were the up/down ratio (f55) and the URG flag (f52), respectively. Intriguingly, these two features ranked among the six highest features for NTP, SSDP, UDP, and SNMP attack datasets. Furthermore, these same two features were consistently among the best feature sets for the remaining attack datasets.

Features like Bwd Packet Length Min (f20) and Bwd Packet Length Mean (f21) formed part of the top five selected features using ANOVA for DNS, MSSQL, UDP,

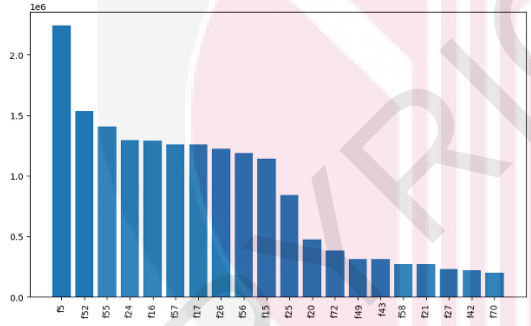
SYN, TFTP, and SNMP attack datasets. However, they were among the best 20 features for other attack datasets, such as NTP, LDAP, NETBIOS, SSDP, and UDP-Lag. Lastly, Avg Bwd Segment Size (f58), Fwd PSH flags (f43), and RST Flag Count (f49) emerged as key features selected across all attack datasets, as portrayed in Figure 4.3 (a-k) below.



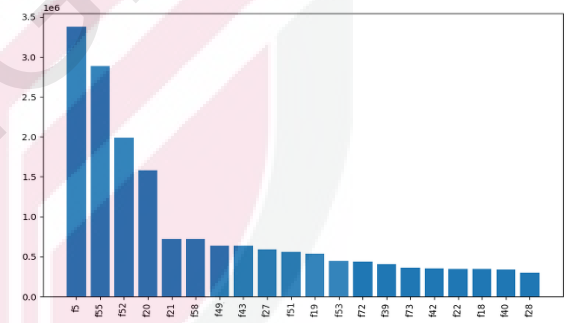
(a) NTP Attack Dataset



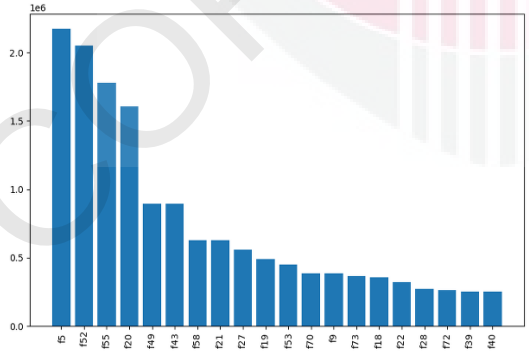
(b) DNS Attack Dataset



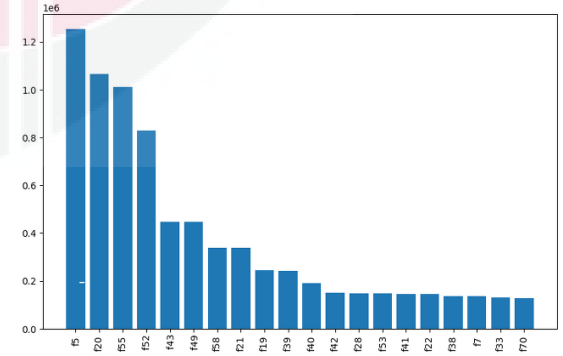
(c) LDAP Attack Dataset



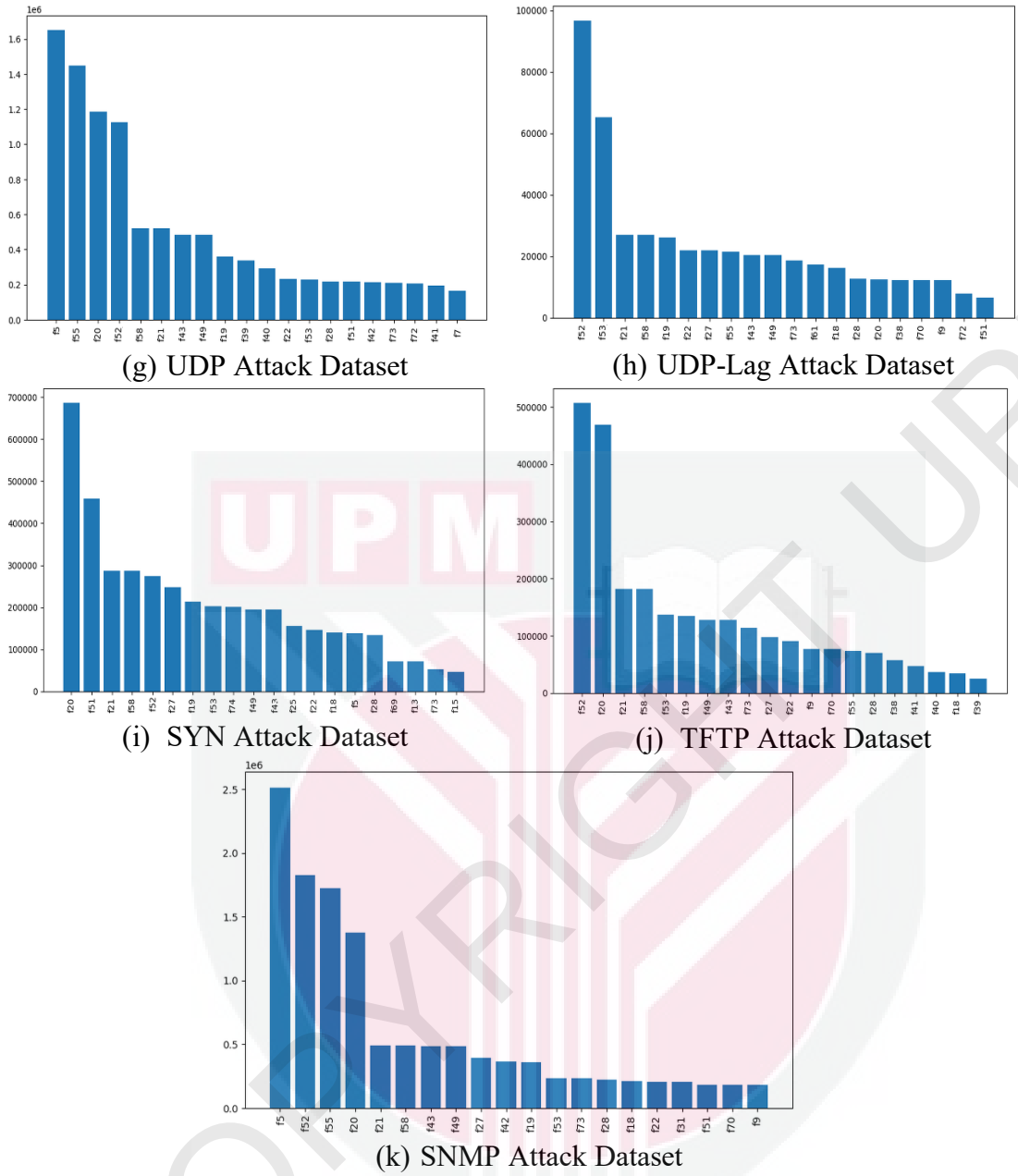
(d) MSSQL Attack Dataset



(e) NETBIOS Attack Dataset



(f) SSDP Attack Dataset



(g) UDP Attack Dataset (h) UDP-Lag Attack Dataset

(i) SYN Attack Dataset (j) TFTP Attack Dataset

(k) SNMP Attack Dataset

Figure 4.3 : Best (20) Features of Different Attack Dataset for First Day of CICDDoS2019 Using ANOVA Technique

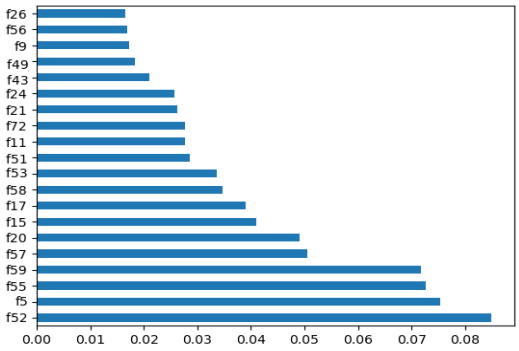
4.3.2 Extra Trees Classifier Results

The Extra Trees Classifier stands as another feature selection technique that was employed to identify the top twenty features, aiming to enhance execution speed and system performance. Among the distinct attack datatypes—NTP, DNS, MSSQL, NETBIOS, SSDP, and SNMP—the protocol feature (f5) emerged as one of the three

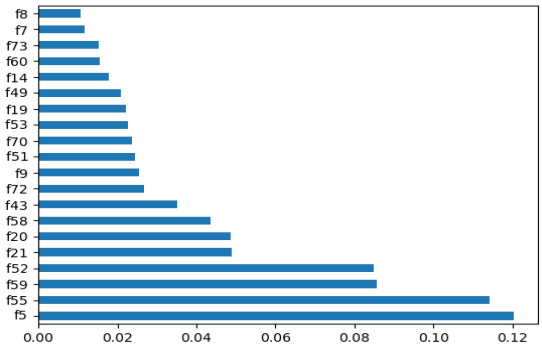
features with higher weights, selected through this method. However, for the other attack datasets, this feature was exclusively included within the set of best twenty features.

Moreover, Min_seg_size_forward (f59) was consistently recognized as the second or third best feature for UDP, SNMP, DNS, and SYN attack datasets. For NTP, NETBIOS, and SSDP attack datasets, it ranked as the fourth or fifth best feature. Notably, this f59 feature was also consistently part of the selected best twenty features for the other attack datasets, as depicted in Figure 4.4.

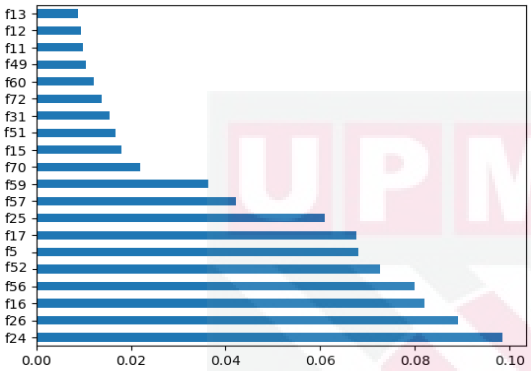
Finally, among NTP, DNS, NETBIOS, SSDP, and SNMP attack datasets, the up/down ratio (f55) was acknowledged as one of the top five selected features. Yet, its inclusion was limited to the best twenty features in the case of other datasets, as presented in the figure below. Bwd Packet Length Min (f20) secured a spot among the top five features for NETBIOS, SSDP, UDP, UDP-Lag, SYN, TFTP, and SNMP attack datasets using the Extra Tree feature selection method. However, it was omitted from the list of best twenty features for the LDAP dataset, while it retained its place in the list for the remaining attack datasets under the Extra Tree feature selection approach. This can be observed in Figure 4.4 (a-k) below.



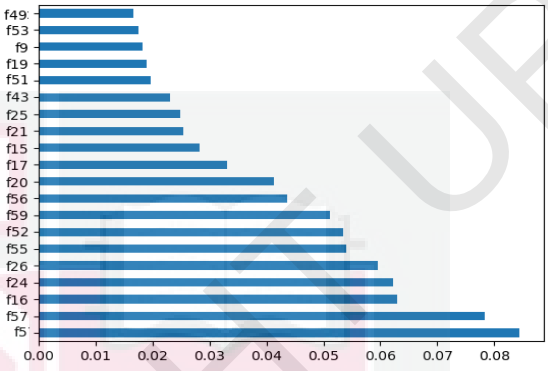
(a) NTP Attack Dataset



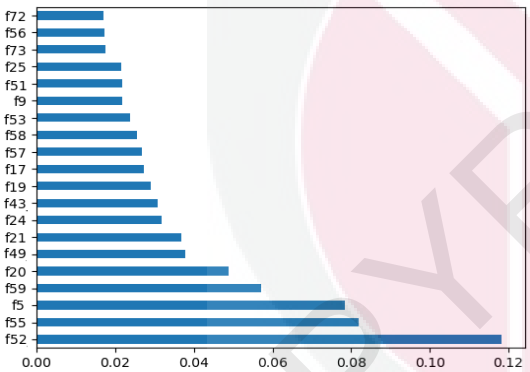
(b) DNS Attack Dataset



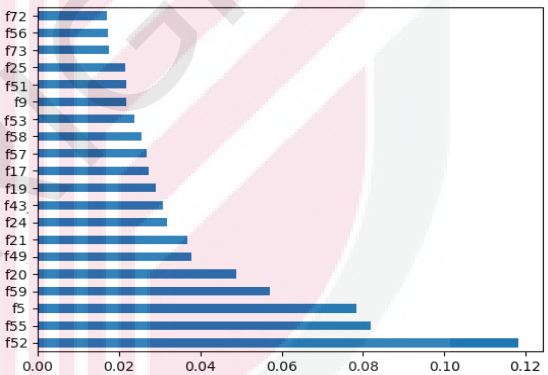
(c) LDAP Attack Dataset



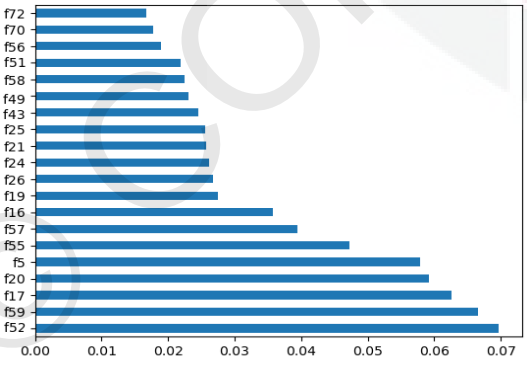
(d) MSSQL Attack Dataset



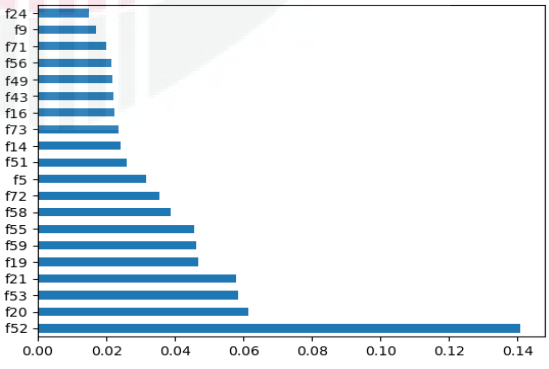
(e) NETBIOS Attack Dataset



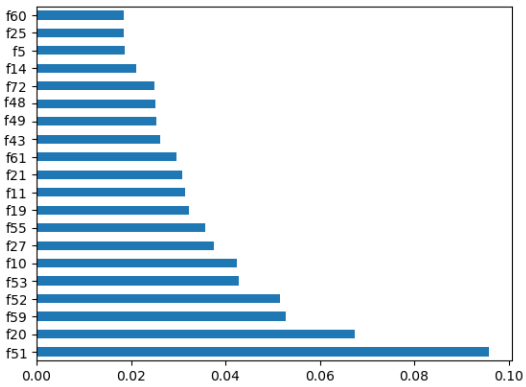
(f) SSDP Attack Dataset



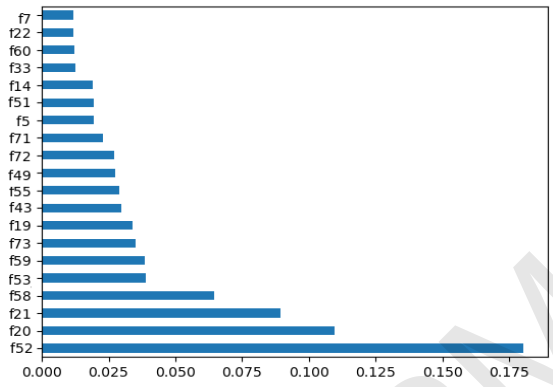
(g) UDP Attack Dataset



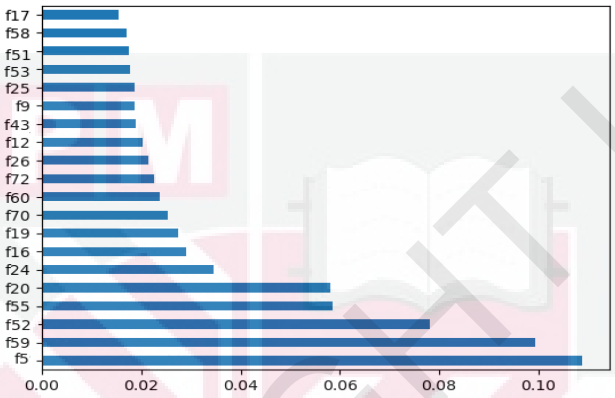
(h) UDP-Lag Attack Dataset



(i) SYN Attack Dataset



(j) TFTP Attack Dataset



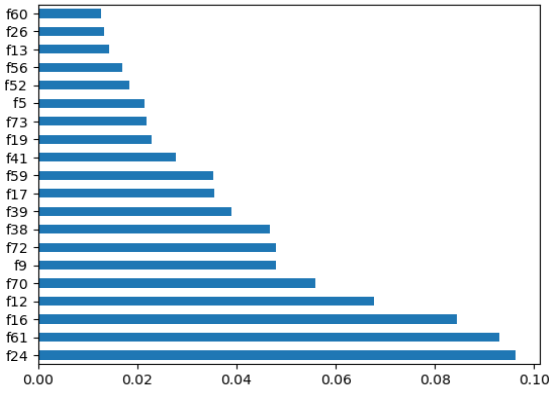
(k) SNMP Attack Dataset

Figure 4.4 : Best (20) Features of Different Attack Dataset for First Day of CICDDoS2019 Using Extra Tree Technique

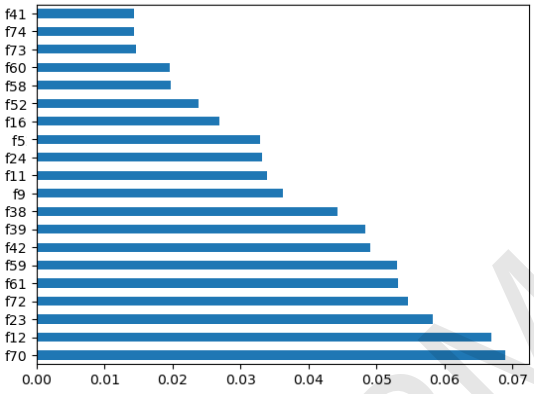
4.3.3 Random Forest Classifier Results

The Random Forest technique emerges as another feature selection method utilized in tandem with other approaches to extract impactful features. For diverse attack datasets, NTP, LDAP, MSSQL, NETBIOS, SSDP, UDP, and SNMP, Min Packet Length (f24) and Fwd Packet Length Min (f16) secured positions among the first three selected features, carrying higher weights. However, they did not make the cut for the selected twenty features in the case of UDP-Lag, SYN, and TFTP attack datasets, as presented in Figure 4.5. Another notable inclusion in the top selected twenty list for all attacks was Fwd Init Win bytes (f72).

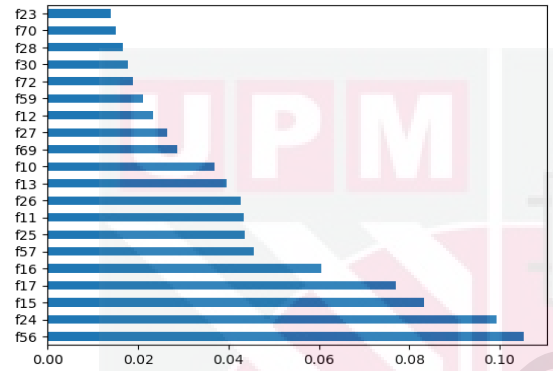
Bwd Init Win bytes (f73) is another recurrently favored feature by the Random Forest technique. It earned its spot among the most important ten features for UDP-Lag, SYN, and TFTP attack datasets, while being included in the best twenty features list for NTP, DNS, MSSQL, NETBIOS, UDP, and SNMP attack datasets, as depicted in Figure 4.5. Notably, this trend was not replicated for the LDAP and SSDP attack datasets, where this particular feature failed to make the top twenty. Finally, Average Packet Size (f56) and Avg Fwd Segment Size (f57) also found their way into the selection for specific attack datasets, NTP, MSSQL, NETBIOS, SSDP, UDP, LDAP, and SNMP. However, their inclusion was omitted for the remainder of the attacks.



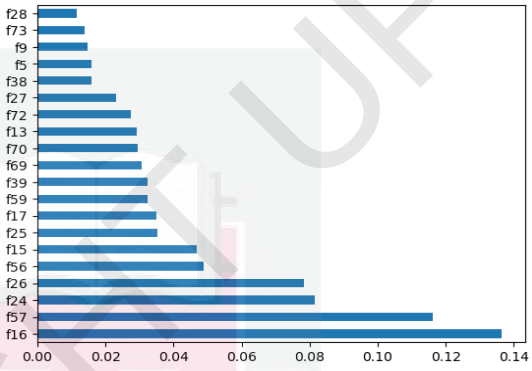
(a) NTP Attack Dataset



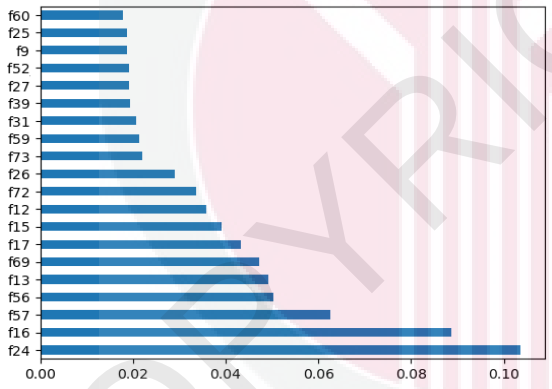
(b) DNS Attack Dataset



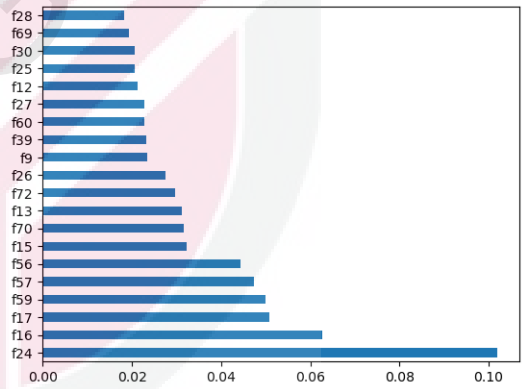
(c) LDAP Attack Dataset



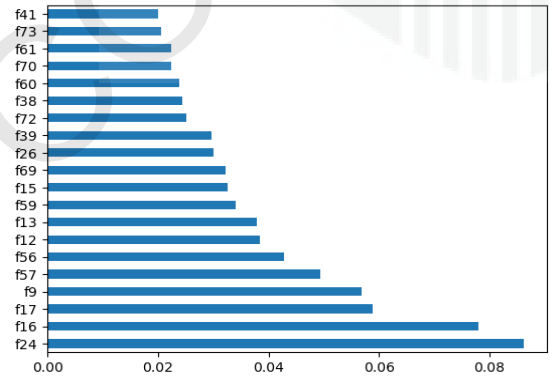
(d) MSSQL Attack Dataset



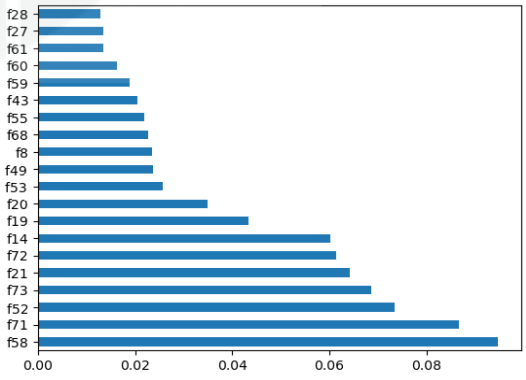
(e) NETBIOS Attack Dataset



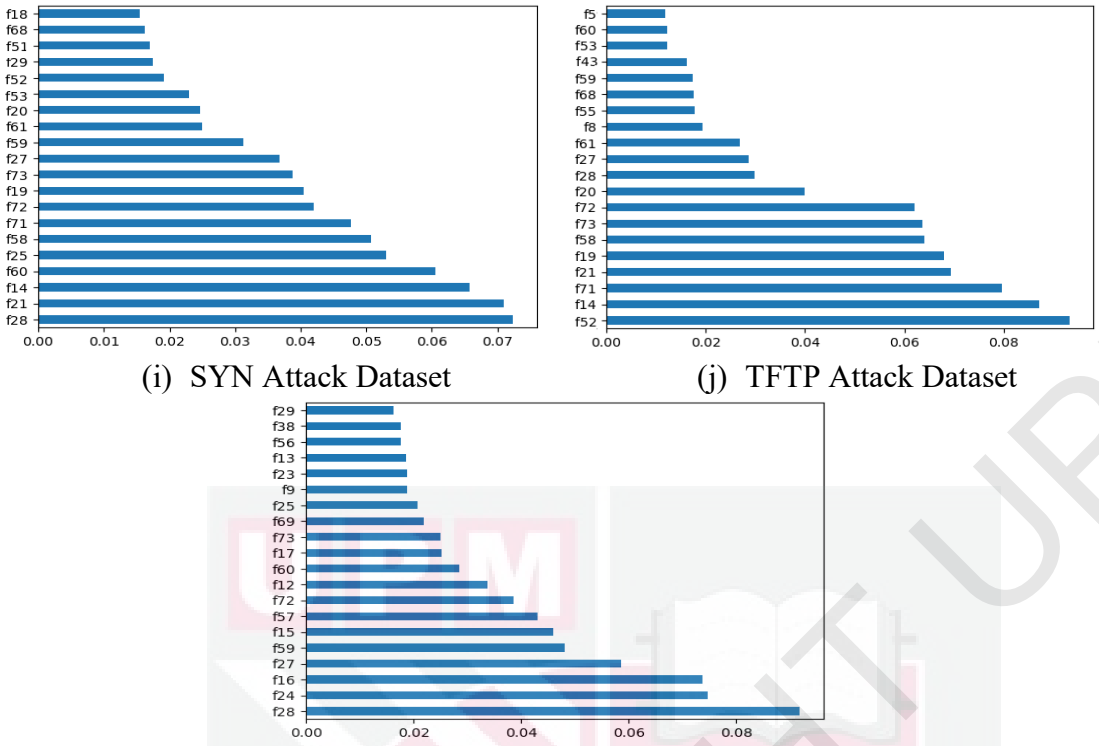
(f) SSDP Attack Dataset



(g) UDP Attack Dataset



(h) UDP-Lag Attack Dataset



(i) SYN Attack Dataset (j) TFTP Attack Dataset

(k) SNMP Attack Dataset

Figure 4.5 : Best (20) Features of Different Attack Dataset for First Day of CICDDoS2019 Using Random Forest Technique

4.3.4 Correlation Matrix with Pearson Correlation Heatmap Results

Pearson Correlation serves as a feature selection technique that ranks features based on their weights, derived from their correlation with a target label. To implement this method, a Correlation Matrix Heatmap was computed for various attack datasets on the first day of CICDDoS2019. The correlation between features and the label for the NTP attack dataset was initially computed. Illustrated in Figure 4.6, this figure showcases the correlation among the top twenty selected features and each feature individually. Weight values of the features most correlated with the label are displayed in the first row. For instance, f17 and f57, with weights of 0.66, exhibit strong correlation with the target label. Following suit, the third and fourth most

correlated features, f5 and f26, hold weights of 0.64 and 0.6, respectively. The remaining features and their respective weights are detailed in Figure 4.6.

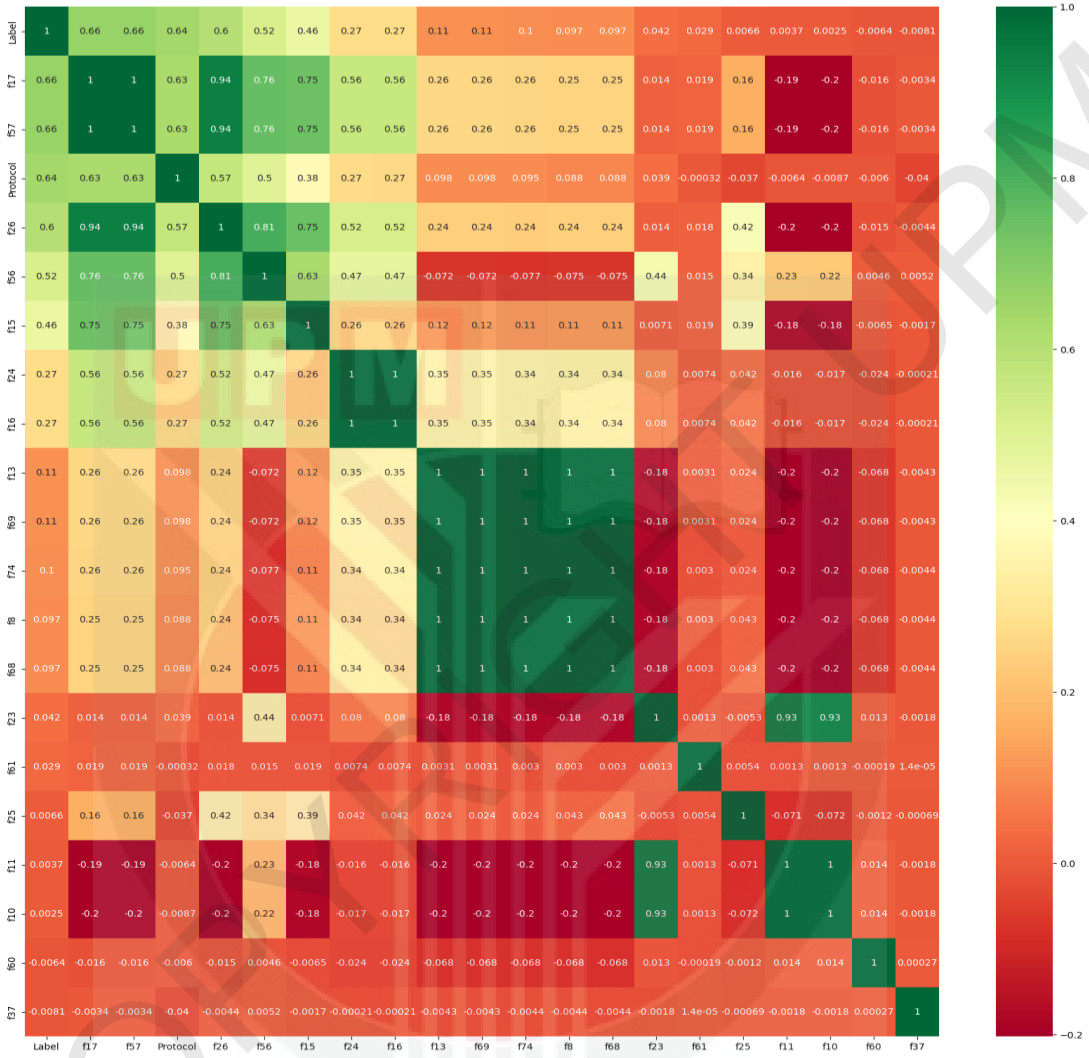


Figure 4.6 : Best (20) Features of NTP Dataset for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

Furthermore, Figure 4.7 vividly portrays the relationship between all twenty features and the potential output for the DNS attack dataset on the first day of CICDDoS2019. This heatmap delineates the upper row with the top twenty features, arranged based on their weights from left to right. Evidently, F5 assumes the mantle of the best-selected feature, boasting a significant correlation with the output label,

weighing in at 0.61. The following features—f24, f16, f17, and f57—each carry a weight of 0.12. Figure 4.7 goes on to present the remaining selected features and their corresponding weights.

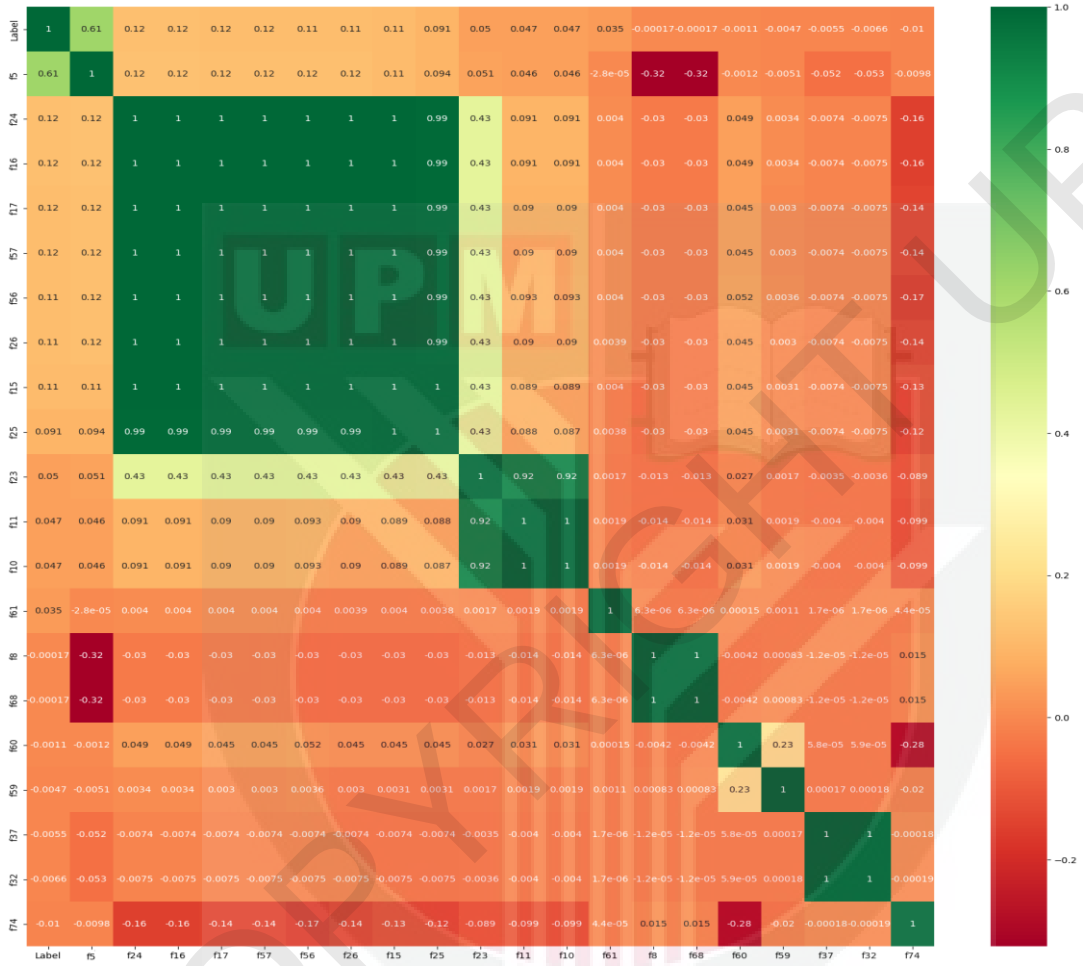


Figure 4.7 : Best (20) Features of DNS Dataset for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

Moreover, Figure 4.8 showcases the top twenty selected features for the LDAP attack dataset utilizing Pearson correlation. Notably, protocol (f5) emerges as the paramount feature, bearing a weight value of 0.71. F24, f16, f17, and f57 all share similar weight values, approaching 0.61, situating them among the upper echelon of sorted features. In the upper row, the sixth and seventh features, f26 and f56, display

a correlation weight of 0.6. A comprehensive listing of the twenty sorted selected features is encapsulated within the upper row of Figure 4.8.

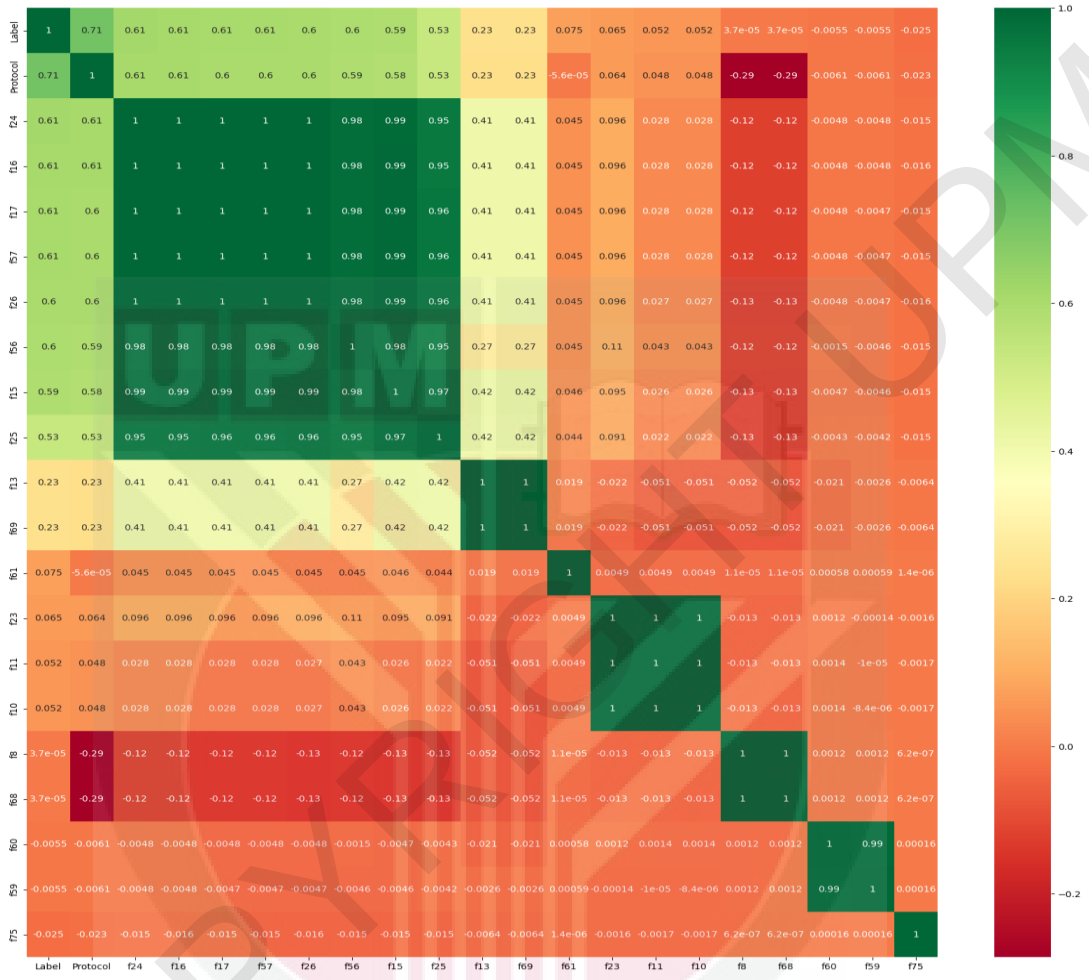


Figure 4.8 : Best (20) Features of LDAP Dataset for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

Finally, the remaining outcomes of feature selection using Pearson Correlation for the other attack datasets from the first day of CICDDoS2019 will be presented in the Appendix. Features selected from each attack dataset employing techniques including ANOVA, Extra Tree, Random Forest, and Pearson Correlation, are amalgamated, and the frequency of each unique feature is computed. Subsequently,

the most important ten features with the highest frequency are earmarked as inputs for the subsequent stage—the detection method.

4.4 Confusion Matrix Metrics

Confusion Matrix [19], also known as an error matrix, is a tabular representation that illustrates the performance of a detection system. It constitutes a distinctive form of table, termed a contingency table, possessing two dimensions: "actual" and "predicted". Both dimensions encompass the same set of "classes". Each entry in the contingency table signifies a combination of a class and a dimension. This table is constructed based on four fundamental metrics: True Positive (T_P), True Negative (T_N), False Positive (F_P), and False Negative (F_N). All other metrics within the confusion matrix are derived from these core metrics. Primarily, the True Positive Rate (TPR), also referred to as sensitivity, recall, or hit rate, denotes the proportion of correctly identified malicious flow traffic among all actual malicious flow traffic. Sensitivity can be calculated using the following equation:

$$\text{Sensitivity} = \frac{T\_P}{T\_P + F\_N} \tag{4.1}$$

Moreover, the True Negative Rate (TNR), alternatively known as specificity or selectivity, can be calculated from the four primary metrics. It denotes the proportion of accurately predicted normal flow traffic among all actual normal flow traffic. It can be computed using Equation (4.2). The False Positive Rate (FPR) signifies the proportion of incorrectly predicted normal flow traffic by the detection algorithm, which is also termed the probability of false alarm or fall-out. It can be calculated using Equation (4.3). Similarly, the False Negative Rate (FNR), often referred to as

the Miss Rate, represents the proportion of erroneously identified compromised traffic as normal traffic. It can be computed using Equation (4.4) below. Notably, higher sensitivity and specificity values and lower miss rate and probability of false alarm values denote greater accuracy of the detection system.

$$\text{Specificity} = \frac{T\_N}{T\_N + F\_P} \tag{4.2}$$

$$\text{Probability of False Alarm} = \frac{F\_P}{F\_P + T\_N} \tag{4.3}$$

$$\text{Miss Rate} = \frac{F\_N}{T\_P + F\_N} \tag{4.4}$$

Furthermore, the Positive Predictive Value (PPV) or precision signifies the proportion of normal flow results correctly identified as true positives by the detection algorithm. Conversely, the Negative Predictive Value (NPV) denotes the proportion of infected flow results correctly identified as true negatives by the identification algorithm. Elevated values of precision and NPV indicate a higher accuracy of the detection system. These metrics are computed as shown in Equation (4.5) and Equation (4.6).

$$\text{precision} = \frac{T\_P}{T\_P + F\_P} \tag{4.5}$$

$$\text{NPV} = \frac{T\_N}{F\_N + T\_N} \tag{4.6}$$

In addition, the False Discovery Rate (FDR) denotes the ratio of falsely identified normal flows as compromised flows by the detection algorithm, computed using Equation (4.7). The False Omission Rate (FOR) represents the ratio of incorrectly

identified compromised flows as normal flows by the detection algorithm, computed using Equation (4.8). Lower FDR and FOR values signify improved accuracy of the detection system.

$$FDR=\frac{F\_P}{T\_P+F\_P} \tag{4.7}$$

$$FOR=\frac{F\_N}{F\_N+T\_N} \tag{4.8}$$

Finally, while Accuracy is a widely used metric to assess detection system performance, it might not always present a comprehensive overview, particularly when dealing with imbalanced datasets. In such scenarios, the F1-score can offer a more comprehensive metric by considering both precision and recall. A higher F1-score indicates a better equilibrium between precision and recall, ultimately signifying improved performance of the detection system. The ensuing equations are utilized to calculate Accuracy and F1-score, respectively:

$$Accuracy=\frac{T\_P+T\_N}{T\_P+T\_N+F\_N+F\_P} \tag{4.9}$$

$$F_1\_score=\frac{2 * T\_P}{2 * T\_P+F\_P+F\_N} \tag{4.10}$$

4.5 Parameters Used in Implementation

Several parameters were used in the implementation, including λ_2 and λ_1, which contributed to the calculation of upper and lower bound thresholds in Equation (3.14) in order to bypass false alarm and miss rate error, mentioned in Chapter 3. The values of λ_2 and λ_1 were selected to be between 0.01 and 0.05 and were

computed through experimentation based on the minimum number of flow observations needed to detect malicious flow. Figure 4.9 depicts the effect of the number of flow observations on choosing the best values for λ_1 and λ_2. When the value of λ_1 is 0.01 and the values of λ_2 fall within the range of 0.01 and 0.05, the minimum number of observed compromised flows is 8. However, the lowest number of compromised flows is 7 when value of λ_1 is 0.02, and the values of λ_2 fall within the range of 0.01 and 0.05. Finally, the lowest number of observed flows generated from this calculation was 5 when the value of λ_1 is 0.02 and values of λ_2 fall within the range of 0.01 to 0.05, as shown in Figure 4.9.

The other parameter used in the implementation was the window size. The multi-feature of ESPRT divided incoming traffic flows into collections of a fixed size called the window size. This window size can be determined based on a fixed timestamp or a fixed number of received flows. In the implementation, a fixed number of flows was used to determine the window size, which varies for each data trace or dataset. It can be determined through experimentation based on detection accuracy. For example, the range of window sizes for the DARPA dataset that generated high accuracy falls within 5 to 120 flows. However, the best number of flows for CICDDoS2019 was above 300 flows. Therefore, the best window size can be determined experimentally.

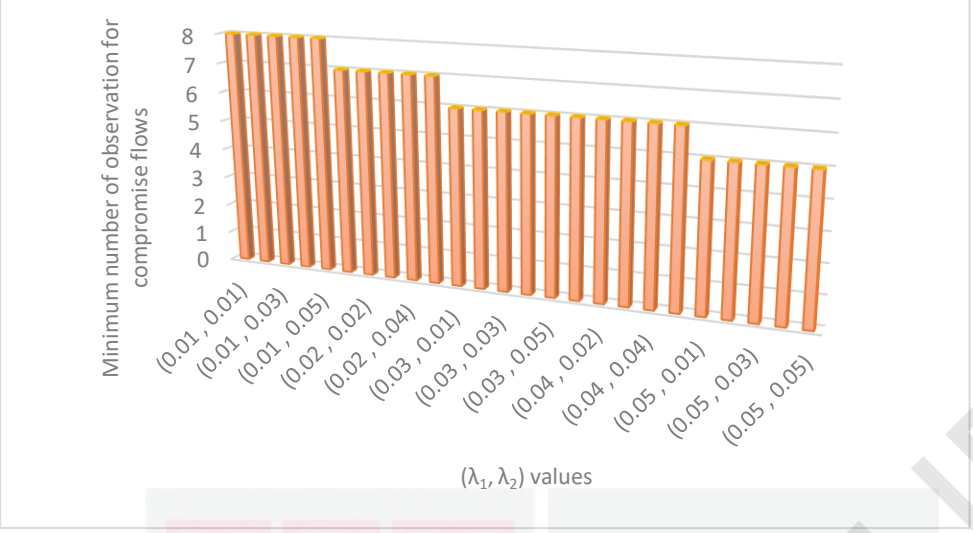


Figure 4.9 : Choosing Best Values for λ_1 and λ_2

Finally, time complexity or cost means time needed to run the detection system. When number of incoming flows toward server increased, the time needed for handling new extracted features will also increase. Sending many numbers of ICMP, TCP, or UDP requests to a server may result in increasing the time complexity or cost of the system. This can result in the victim machine being stopped, frozen, or restarted. It represents program's running time. It can be determined by calculating the difference between the ending time and starting time.

4.6 ESPRT Results Using DARPA 1998 and Source IP Address Feature

Both entropy-based detection and SPRT-based detection techniques were used to identify DDoS attacks. The entropy technique depends on two factors: window size and threshold, to perform its function, while the SPRT depends only on the window size. The window size can be determined based on the number of incoming packets, while the threshold is set to be between 0.2 and 1.5. The values of the threshold can be determined based on experimentation and changing them can impact accuracy

and F-score. Friday.tcpdump Dataset, which is part of DARPA 1998 was used to examine the effectiveness of ESPRT and compare accuracy and f-score of the entropy and ESPRT techniques.

The initial metric used to highlight differences between ESPRT and entropy, utilizing source IP address as a feature, is accuracy. For instance, when the window size is set to 5 flows and the threshold is set to 0.2, the accuracy of entropy-based detection is 0.986. However, the accuracy of entropy decreases as the threshold increases. As indicated in Table 4.4, the accuracy for entropy is 0.972 when the threshold is set to 1.5. The combination of SPRT with entropy leverages entropy's advantages and circumvents the issue of using fixed thresholds. For example, with a window size of 5 flows, the accuracy of ESPRT reaches 0.987, surpassing the accuracy of entropy for various thresholds. Similarly, when the window size is set to 100 flows, the accuracy of ESPRT is 99.3%. Conversely, the accuracy of entropy detection varies with changes in the threshold values. Ultimately, the accuracy values of ESPRT for different window sizes surpass the accuracy of entropy for various threshold values, as detailed in Table 4.4.

Table 4.4 : Accuracy of ESPRT and Entropy Approaches for Different Window Size and Threshold Using DARPA 1998

Window size	Accuracy of ESPRT Detection	Accuracy of Entropy Detection				
		(thr=0.2)	(thr=0.5)	(thr=1)	(thr=1.31)	(thr=1.5)
5	0.987	0.986	0.986	0.980	0.979	0.972
10	0.990	0.986	0.989	0.984	0.980	0.978
25	0.988	0.991	0.992	0.989	0.986	0.984
50	0.995	0.987	0.994	0.992	0.990	0.989
75	0.995	0.986	0.994	0.993	0.991	0.989
100	0.993	0.984	0.994	0.993	0.992	0.990

The other metric used to demonstrate the effectiveness of ESPRT in comparison to entropy-based detection is the f-score. For example, with a window size of 10 flows, the f-score for ESPRT is 0.994. However, the f-score of entropy is either equal to the f-score of ESPRT when the threshold is equal to 0.5 or lower than 0.994 for other threshold values. The f-score values for entropy are 0.993, 0.991, 0.990, and 0.989 when the threshold values are 0.2, 1, 1.31, and 1.5 respectively. In another scenario, when the window size is set to 75 flows and source IP address is utilized as a feature, the f-score for ESPRT reaches 99.7%. Nevertheless, for entropy detection, the f-score is lower than that of ESPRT when the threshold values are 0.2, 1.31, and 1.5, but it matches the f-score of ESPRT when the threshold values are 0.5 and 1. Ultimately, for DARPA 1998 with source IP address as a feature, the f-score of ESPRT is either equal to or superior to that achieved by entropy-based detection with varying threshold values, as depicted in Table 4.5.

Table 4.5 : F-score of ESPRT and Entropy for different Window Size and Threshold Using DARPA 1998

Window size	F-score of ESPRT Detection	F-score of Entropy Detection				
		(thr=0.2)	(thr=0.5)	(thr=1)	(thr=1.31)	(thr=1.5)
5	0.993	0.993	0.993	0.990	0.989	0.986
10	0.994	0.993	0.994	0.991	0.990	0.989
25	0.994	0.995	0.996	0.994	0.993	0.991
50	0.997	0.993	0.997	0.996	0.994	0.994
75	0.997	0.993	0.997	0.996	0.995	0.994
100	0.996	0.992	0.996	0.996	0.995	0.994

4.7 Performance of ESPRT Using Each Single Feature and CICDDoS2019

Using source IP address as a feature is not enough to detect novel and different kinds of DDoS attacks. An attacker may recruit a very large number of infected machines

to target a server with a high number of useless flows using a single spoofed IP address instead of multiple spoofed IP addresses to evade the detection process. Therefore, it is important to extract other features to evaluate the performance of ESPRT. This section will present sensitivity, probability of false alarm, specificity, and miss rate for ESPRT using all the features that were extracted from CICFlowMeter.

4.7.1 Sensitivity vs. Probability of False Alarm

The relationship between sensitivity and the probability of false alarm metrics for all features of the ESPRT detection technique across all attack types in CICDDoS2019 is presented in Figure 4.10 (a-k). In general, features located in the upper-left side of each graph indicate the effectiveness of these features in the ESPRT detection process. For example, Figure 4.10 (a) shows sensitivity and probability of false alarm values for all features of ESPRT for the NTP attack dataset. Some features have sensitivity close to 1 and a probability of false alarm close to 0, such as f5, f12, f14, f15, f19, f21, f22, f25, f35, f38, f39, f40, f41, f42, f51, f52, f53, f55, f58, f61, f71, f72, f73, f75, f77, f78, f79, f80, f81, and f82. Other features have a low true positive value close to 0, such as f7, f10, f13, f16, f23, f29, f30, f31, f33, f34, f36, f56, f60, and f69.

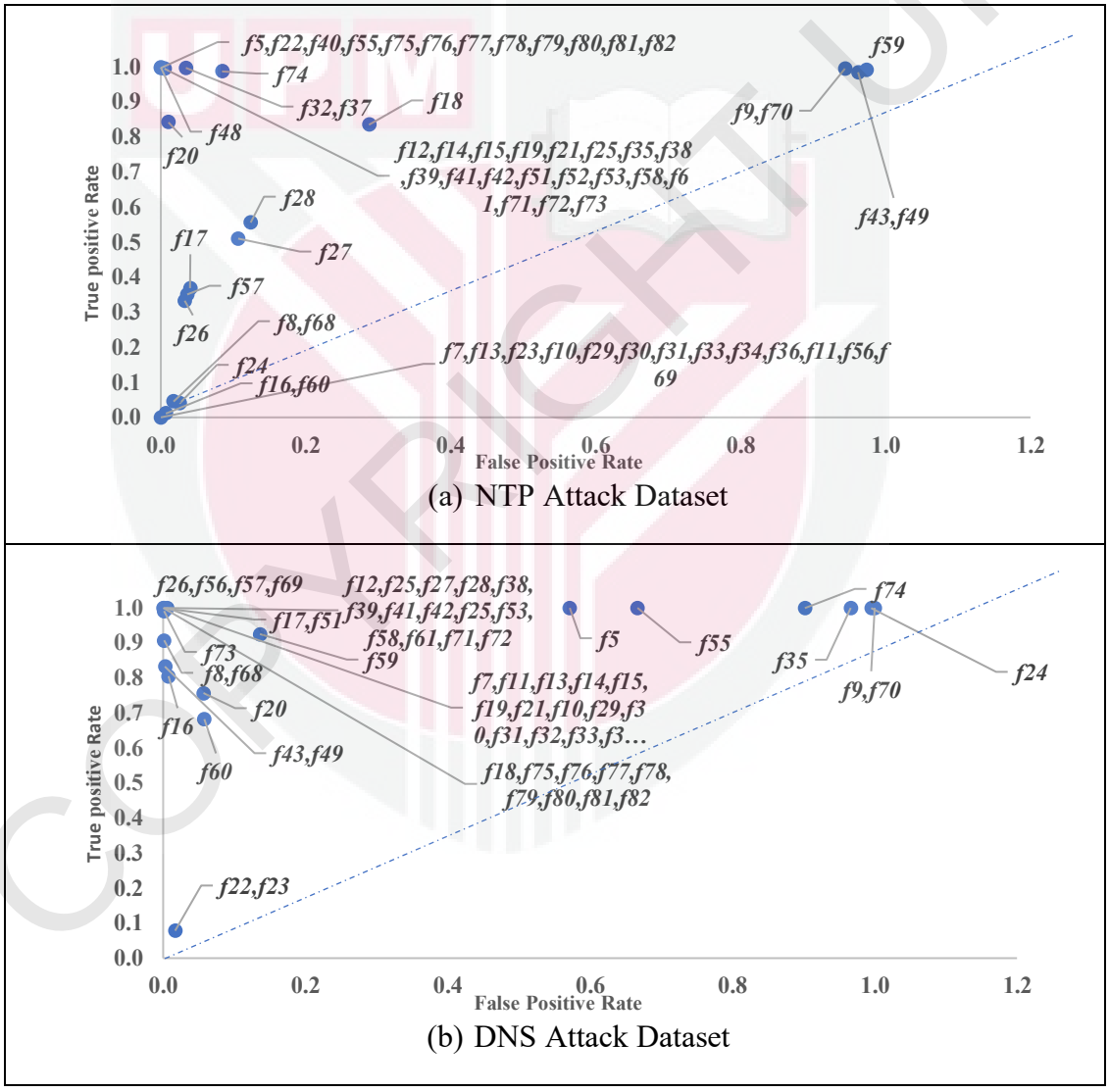
Furthermore, in the DNS attack dataset of CICDDoS2019, most of the features of ESPRT have a TPR larger than 0.68 and a low FPR. However, features F22 and F23 have a low TPR, which is close to 0, as shown in Figure 4.10 (b). Additionally, features such as f5, f9, f24, f35, f55, f70, and f74 have higher values of FPR, which can decrease the accuracy of ESPRT detection. In the LDAP attack dataset, most of

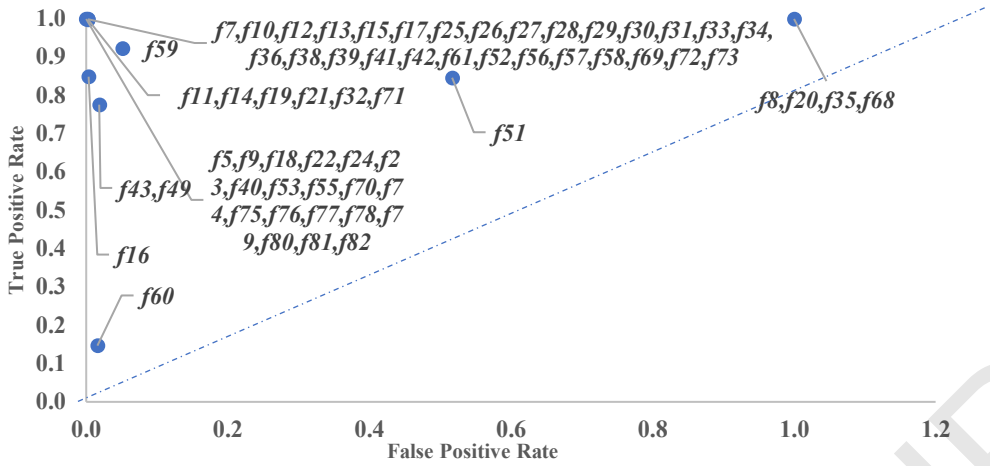
the features used for ESPRT generated high sensitivity values in the range of 0.83 to 1 and low FPR values in the range of 0.1 to 0. However, f60 has a low TPR, and f51 has an FPR of 0.5 in this dataset. Finally, some features, such as f8, f20, f35, and f68, generated high values of the probability of false alarm, close to 1, as shown in Figure 4.10 (c).

In addition, a large number of features generated a sensitivity value near 1 and a low value of FPR close to 0 in the MSSQL dataset attack. On the other hand, f60 has a TPR of 0.48, and f24 has a TPR of 0.15 for the same kind of attack dataset. However, some features such as f5, f9, f55, f70, and f74 generated a higher value of FPR, close to 1, for MSSQL attack as shown in Figure 4.10 (d). For the NETBIOS attack dataset, most of the features used in ESPRT generated a lower FPR and higher sensitivity. However, features such as f5, f9, f55, f59, f70, and f74 generated false alarm errors above 0.9 as shown in Figure 4.10 (e). For the SSDP attack dataset, almost all features have the lowest value of FPR near 0, except some features such as f5, f51, and f59. On the other side, some features such as f8, f12, f24, f23, f60, and f74 have TPR less than 0.5, and the rest are above 0.5 as shown in Figure 4.10 (f).

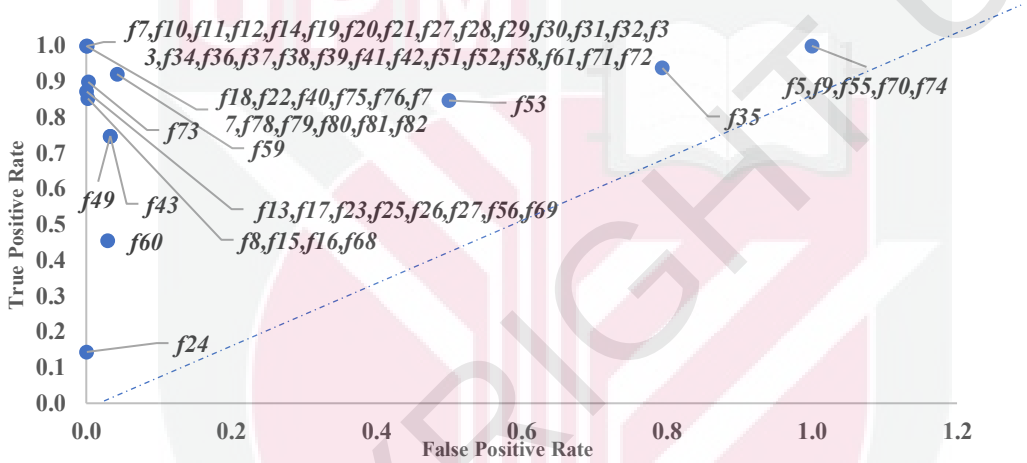
Finally, almost 25 features of ESPRT generated a very low number of TPR for the UDP attack dataset, as shown in Figure 4.10 (g). Most of these features generated a low value of FPR for this type of attack dataset, except for f5, f55, and f59. On the other hand, most of these features of ESPRT in the UDP-Lag, SYN, and TFTP attack datasets generated high sensitivity values close to 1, as shown in Figure 4.10 (g, h, i, k) respectively. The only exception was for the f61 and f73 features in the SYN attack data and for f60 and f23 for the SNMP attack dataset, which produced

TPR values smaller than 0.5. Some features in the UDP-Lag dataset attack produced an FPR larger than 0.6, such as f5, f43, f49, f59, and f74, as shown in Figure 4.10 (g). Most features have an FPR smaller than 0.3 in the SYN dataset attack, except for f5, f32, f43, f49, and f76, as shown in Figure 4.10 (h). Finally, the majority of features of the TFTP and SNMP attack datasets have a small FPR, as shown in Figure 4.10 (j, k).

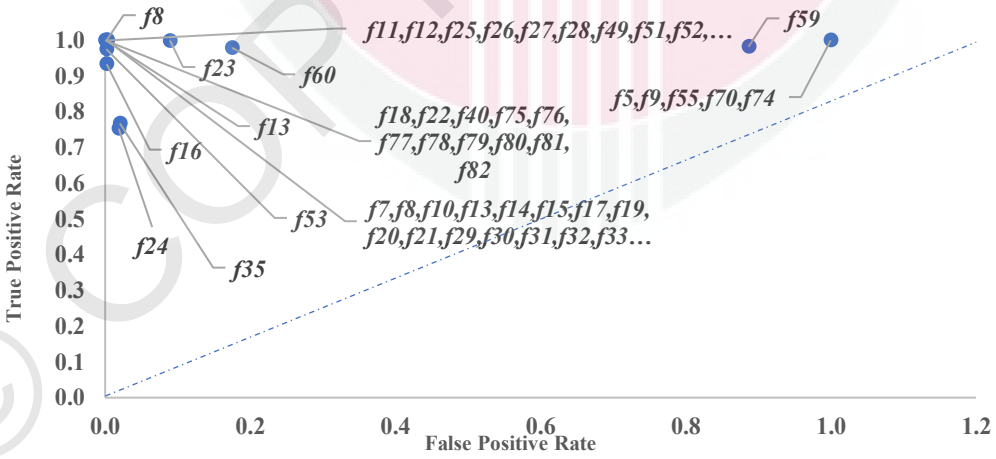




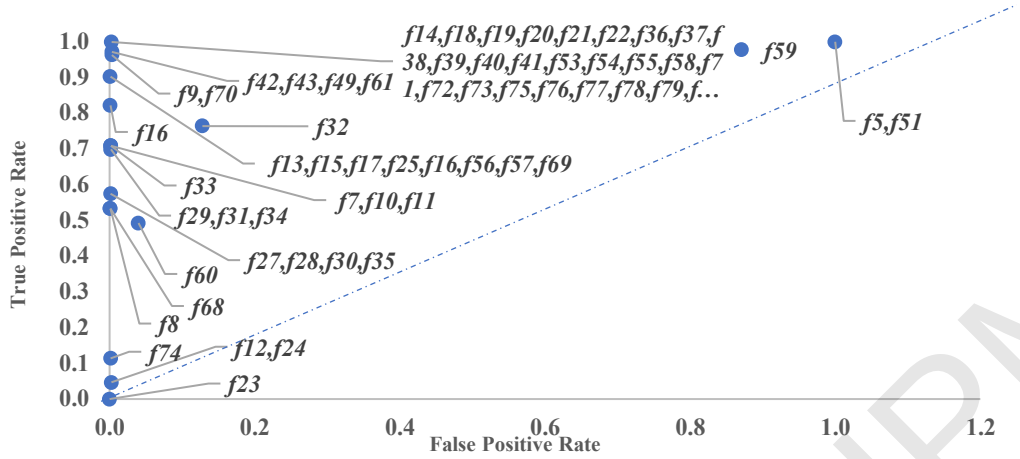
(c) LDAP Attack Dataset



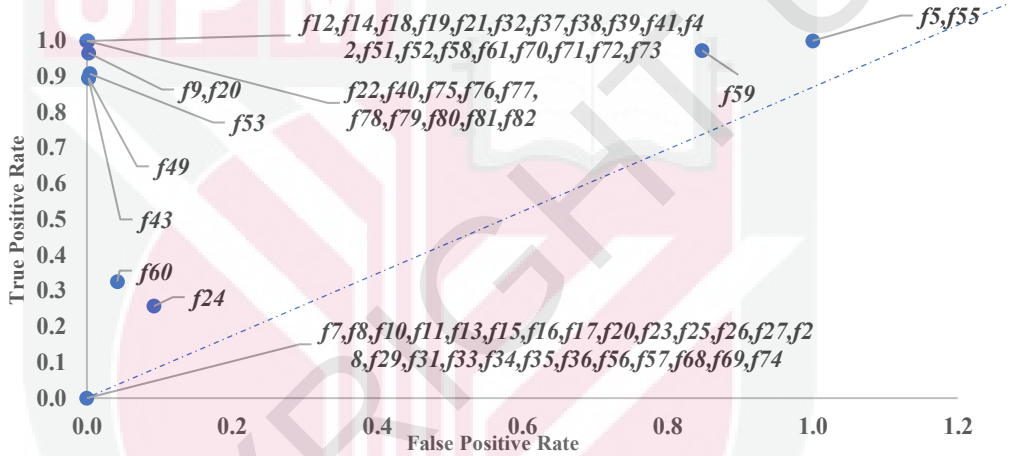
(d) MSSQL Attack Dataset



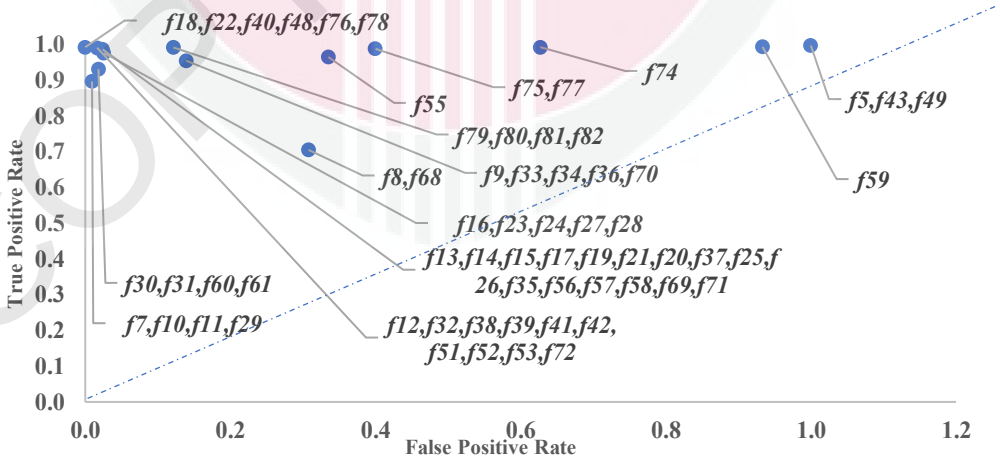
(e) NETBIOS Attack Dataset



(f) SSDP Attack Dataset



(g) UDP Attack Dataset



(h) UDP-Lag Attack Dataset

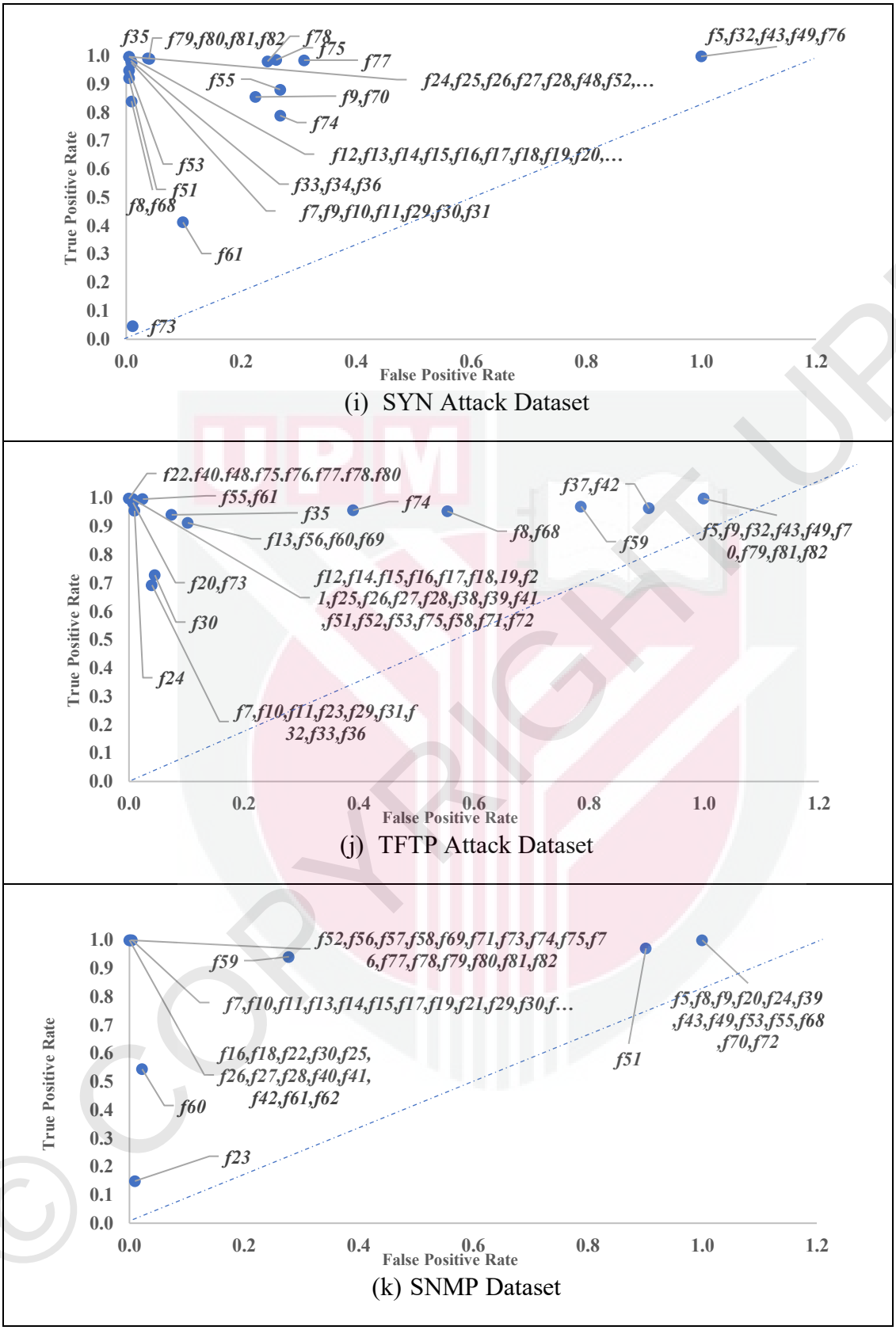


Figure 4.10 : Sensitivity (TPR) vs Probability of False Alarm (FPR) for All Features of ESPRT Using Different DDoS Attacks of CICDDoS2019 Dataset

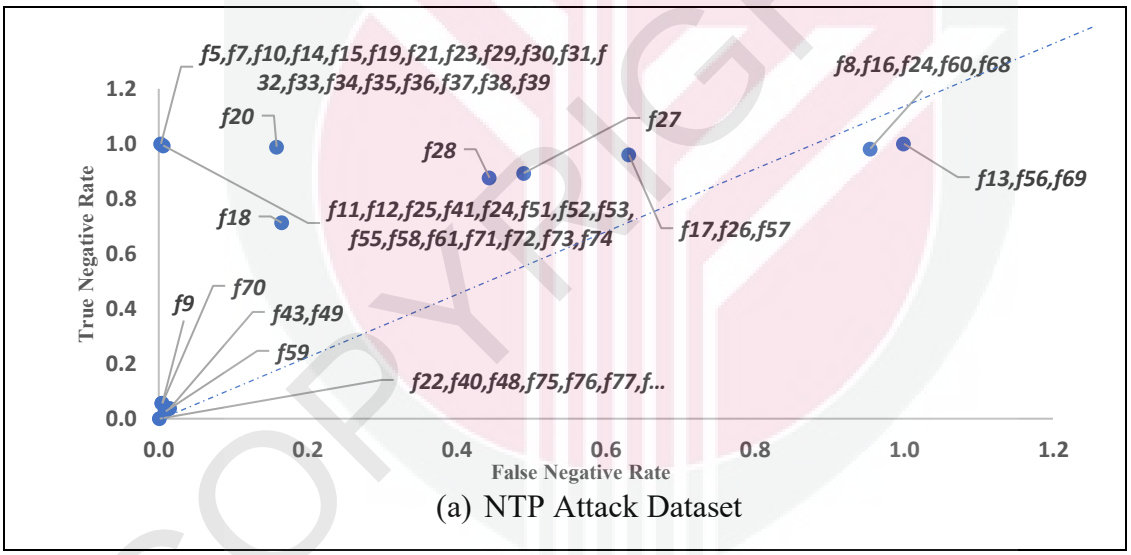
4.7.2 Specificity vs. Miss Rate

The connection between specificity and Miss Rate metrics of all features in the ESPRT identification method for different types of attacks in CICDDoS2019 is presented in Figure 4.11 (a-k). When these features are located in the upper left side of each graph, it means they are effective in the detection process. For example, most features in the NTP attack dataset have a specificity value close to 1. However, around 16 features have low NTP values close to 0 as shown in Figure 4.11 (a). Additionally, most features in the NTP attack dataset have FPR values less than 0.65, while some features have a Miss Rate value close to 1, such as f8, f13, f16, f24, f56, f60, f68, and f69, as shown in Figure 4.11 (a). Similarly, the majority of features used in the DNS attack generate a high TPR, except for f59, which generates almost 0.87, and f5 and f55, which generate almost 0.43 of TPR. Most of these features also have low FPR values less than 0.3, except for f22 and f23, which generate FPR values near 1, as shown in Figure 4.11 (b).

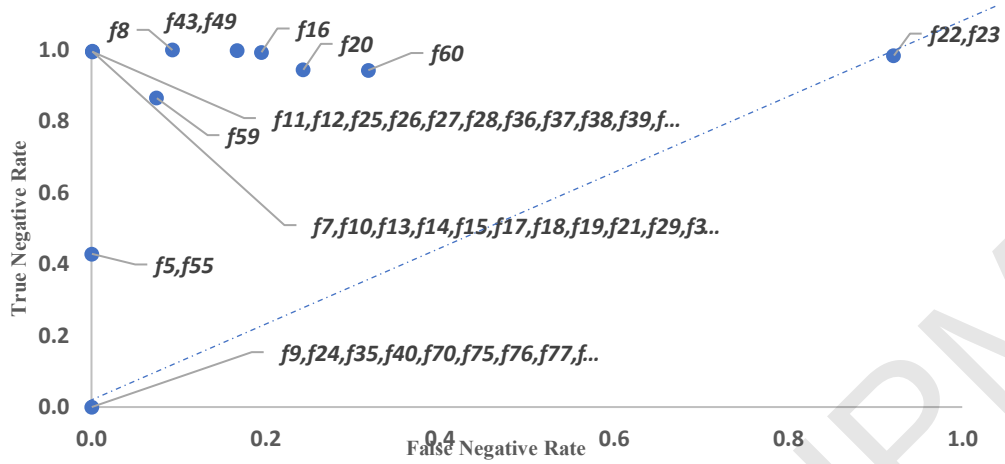
Moreover, the majority of features in the LDAP attack dataset have a miss rate of less than 0.25, except for f60 which has a miss rate close to 1. On the other hand, even though some features have a specificity close to 1, almost 20 features have a specificity close to 0, which may affect the detection accuracy if they are used alone, as shown in Figure 4.11 (c). The majority of features in the MSSQL and NETBIOS attack datasets produced low FPR, except for f24 which has an FPR of 0.9 for the MSSQL attack dataset, and f24 and f35 for the NETBIOS attack dataset produced less than or equal to 0.25 of FPR, as shown in Figure 4.11 (d) and (e). Furthermore, almost 25 features have a high FPR for the UDP dataset attack, as shown in Figure

4.11 (g), while only two features, such as f8 and f68 in UDP-Lag, have a high FPR, as shown in Figure 4.11 (h).

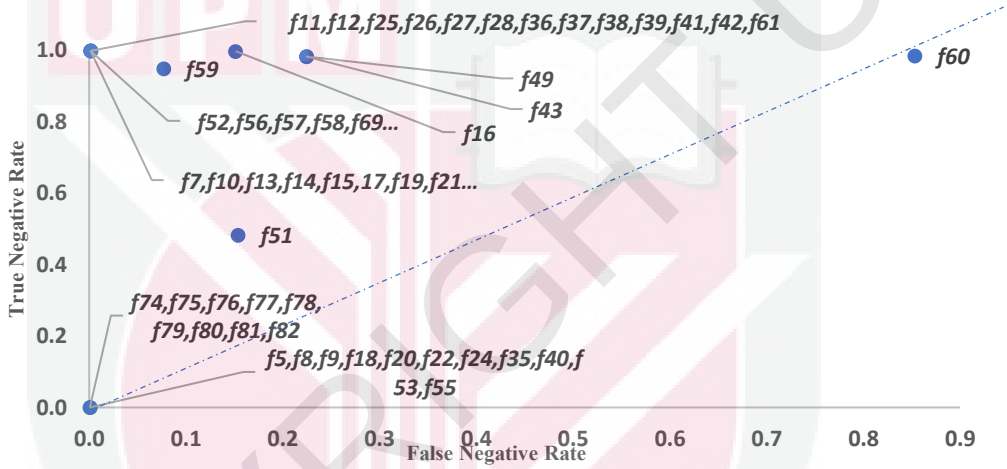
Finally, in the SYN dataset, f61 and f73 have a higher miss rate, while f23 is the only feature with a high FPR, as shown in Figure 4.11 (i) and (k) for the SNMP attack dataset. The SYN attack dataset has 10 features that generated a low TPR compared to the TFTP and SNMP attack datasets, which have 22 and 25 features, respectively, that generated low TPR, as shown in Figure 4.11 (j) and (k). Finally, the rest of the features generated a high value of specificity for these attack datasets, which would increase detection accuracy if used.



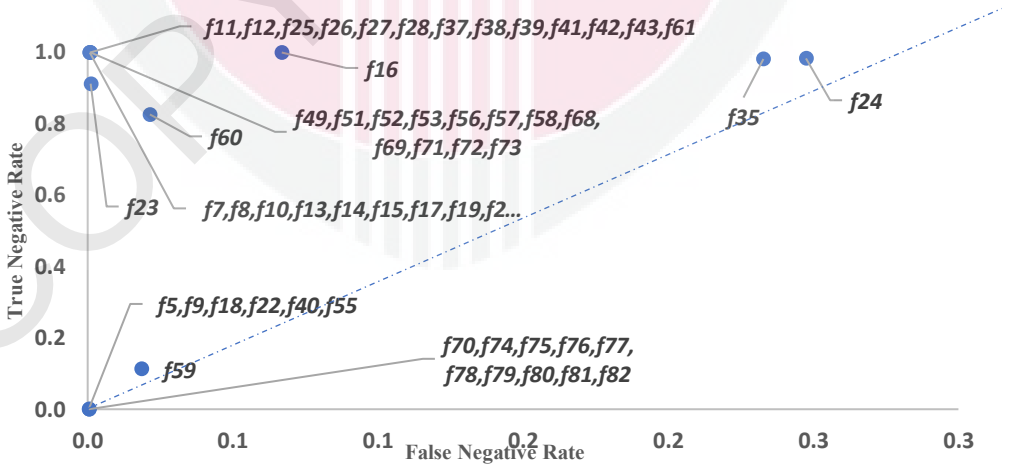
(a) NTP Attack Dataset



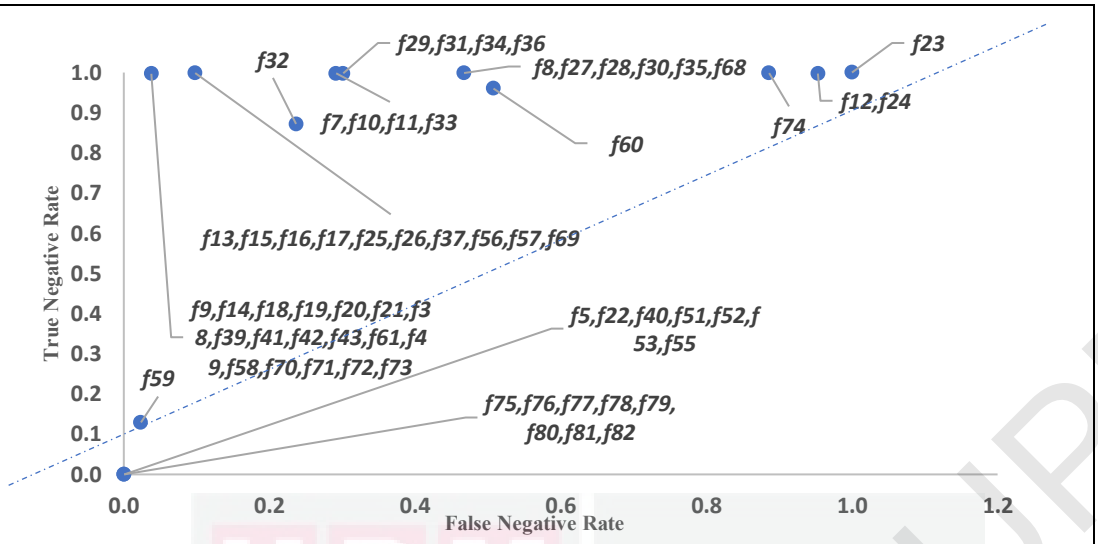
(b) DNS Attack Dataset



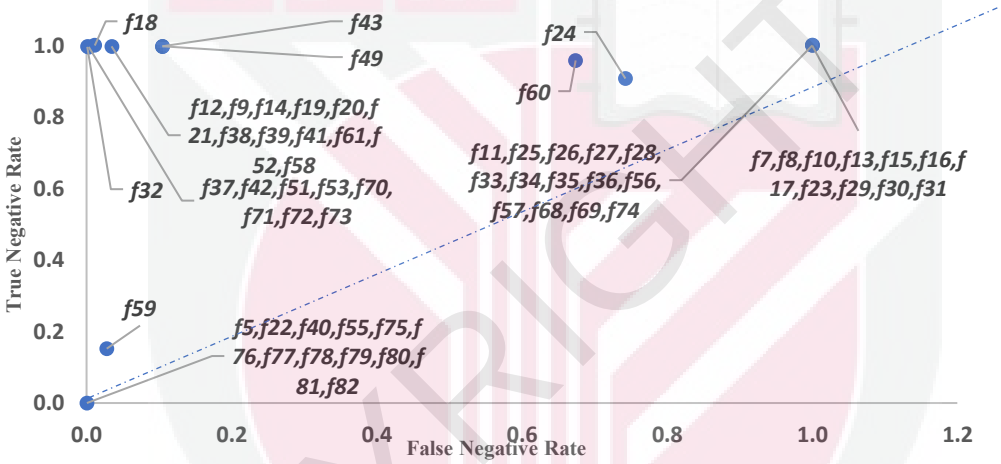
(c) LDAP Attack Dataset



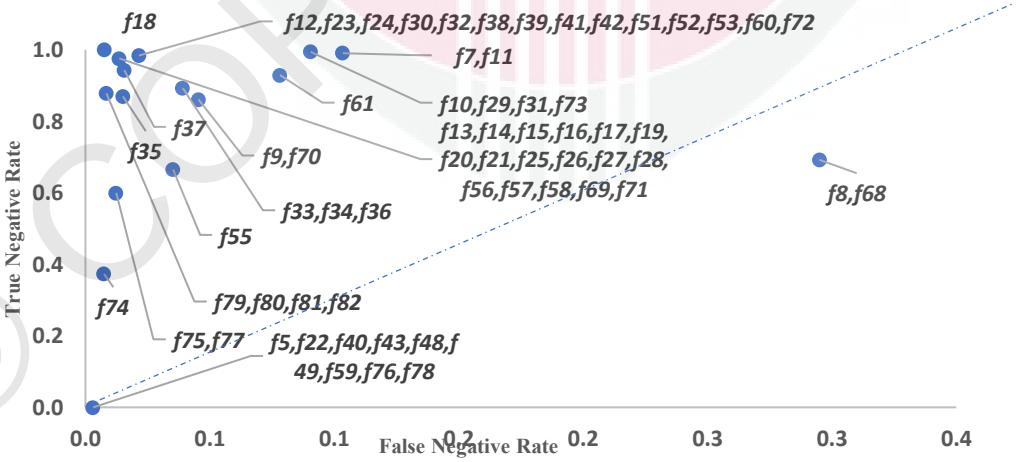
(d) NETBIOS Attack Dataset



(e) SSDP Attack Dataset



(f) UDP Attack Dataset



(g) UDP-Lag Attack Dataset

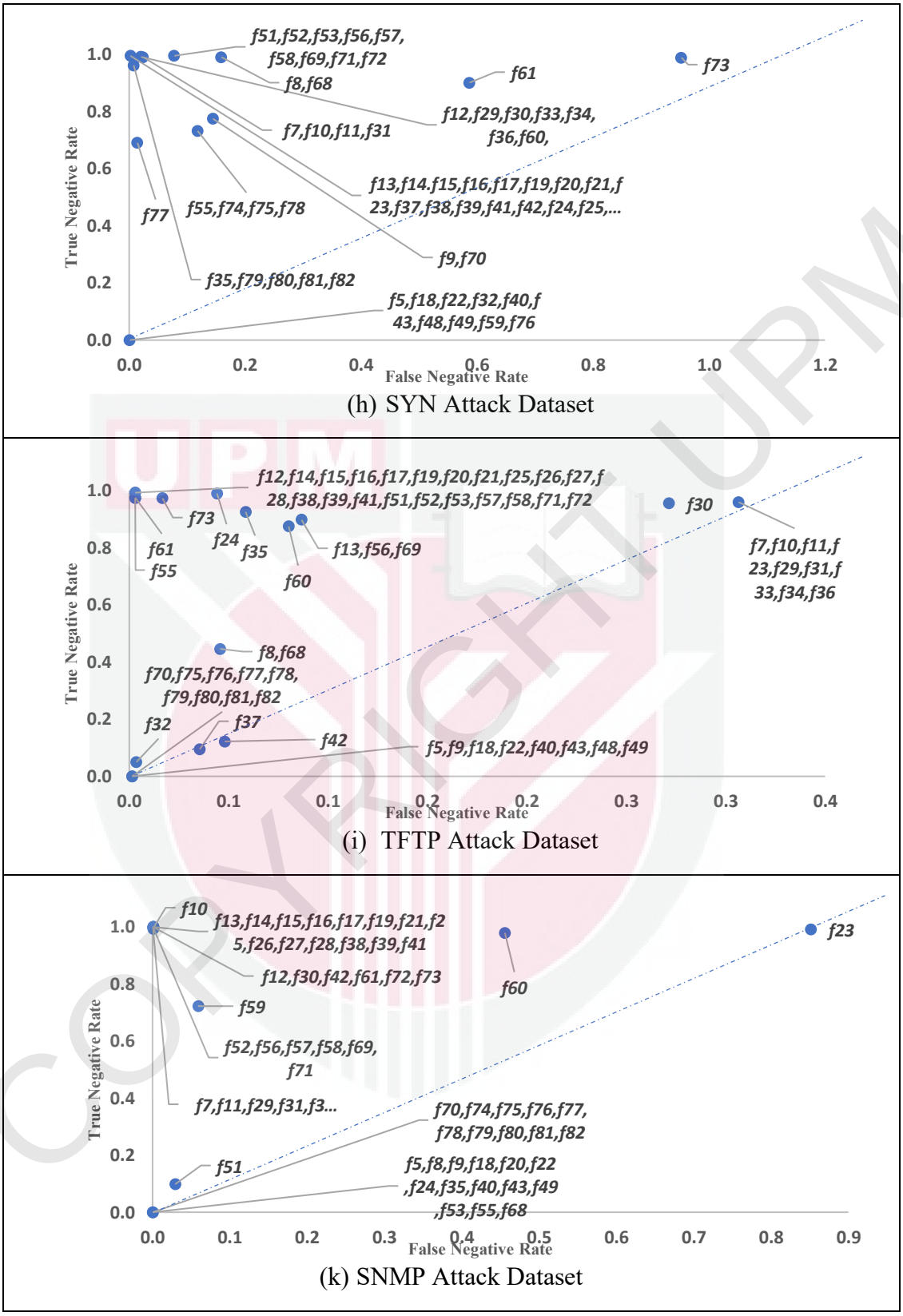


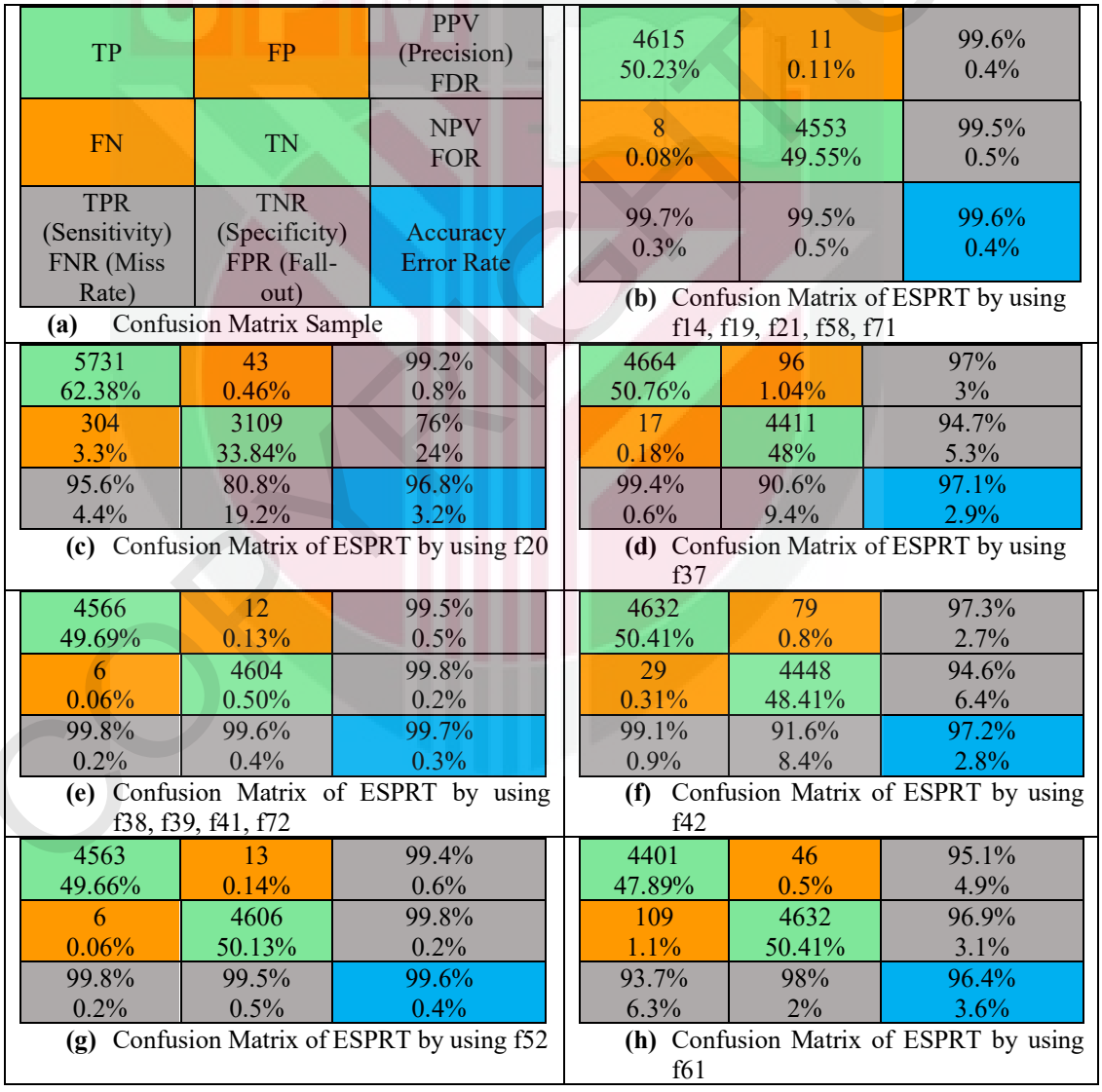
Figure 4.11 : Specificity (TNR) vs. Miss Rate (FNR) for all Features of ESPRT Using Different DDoS Attacks of CICDDoS2019 Dataset

4.8 Confusion Matrix of ESPRT Using Best of Single Feature of CICDDoS2019

Confusion matrix metrics were used to present the best features that can be used in ESPRT for all attacks of CICDDoS2019. These metrics that will be used to show the effectiveness of these features may include the four main metrics that can be used to compute the rest of the metrics. These main metrics are T_P, F_P, F_N, and T_N. The rest of the metrics are Accuracy, error rate, sensitivity (TPR), specificity (FPR), miss rate (FNR), probability of false alarm (FPR), NPV, FOR, precision (PPV), and FDR. The arrangement of these metrics is shown in Figure 4.12 (a). The first group of features that generated higher results of confusion metrics are f14, f19, f21, f58, and f71. These features generated a detection accuracy of 99.6% and an error rate of 0.4%. The sensitivity and specificity values were high for these features, at 99.7% and 99.5%, respectively. However, they also have low values of FNR and FPR, which were 0.3% and 0.5%, respectively. The precision and NPV for these features were 99.6% and 99.5%, while FDR and FOR were 0.4% and 0.5%, respectively, as shown in Figure 4.12 (b).

Furthermore, f20 and f37 generated high accuracy values, at 96.8% and 97.1%, respectively. F20 generated a TPR of 95.6% and an FPR of 4.4%, while f37 generated a TPR of 80.8% and an FPR of 19.2%. These two features produced high values of PPV and NPV and low values of FDR and FOR, as shown in Figure 4.12 (c) and (d). Moreover, f38, f39, f41, and f72 produced high values of accuracy, sensitivity, specificity, precision, and NPV, all above 99.5%. However, these features generated less than 0.5% for the rest of the metrics, as shown in Figure 4.12 (e).

Finally, f42, f52, and f61 produced high accuracy and low error rates, exactly 97.2%, 99.6%, and 96.4%, respectively, for accuracy, and 2.8%, 0.4%, and 3.6%, respectively, for error rate metrics. TPR was above 99%, and FNR was less than 0.2% for f42 and f52, while TPR was 93.7%, and FNR was 6.3% for f61. Specificity values were above 91%, and FPR values were below 8% for these features. On the other hand, NPV was above 94%, and FOR values were less than 6% for these features, as shown in Figure 4.12 (f-h).



(a) Confusion Matrix Sample

(b) Confusion Matrix of ESPRT by using f14, f19, f21, f58, f71

(c) Confusion Matrix of ESPRT by using f20

(d) Confusion Matrix of ESPRT by using f37

(e) Confusion Matrix of ESPRT by using f38, f39, f41, f72

(f) Confusion Matrix of ESPRT by using f42

(g) Confusion Matrix of ESPRT by using f52

(h) Confusion Matrix of ESPRT by using f61

Figure 4.12 : Confusion Matrix of Best Feature of ESPRT Using CICDDoS2019 Dataset

4.9 Confusion Metrics Based on Set of Selected Features of CICDDoS2019

Some ESPRT's confusion matrix metrics were calculated using the CICDDoS2019 dataset, which contains various types of DDoS attacks. A combination of feature selection techniques was used to select the best features as mentioned earlier, resulting in feature sets of 5, 10, 15, and 20. The calculated metrics included accuracy, F-score, and probability of false alarm (FPR). The accuracy of ESPRT was first evaluated using the 5 best-selected features, resulting in an accuracy rate of 93.2% for NTP. However, the accuracy of the 10, 15, and 20 feature sets were lower, at 92.5, 92.2, and 92.9, respectively, as shown in Figure 4.13.

The accuracy of ESPRT using a set of 5 features for the DNS, MSSQL, SSDP, UDP-Lag, SYN, and SNMP attack datasets was better than for other feature sets, with accuracy rates of 95.9%, 99.6%, 97.5%, 97.4%, 99.6%, and 99.6%, respectively. For LDAP and TFTP attacks, the highest accuracy was achieved using 10 features, while using 15 features for NETBIOS resulted in the highest accuracy rate of 96.7%. Finally, the 20-feature set was found to be the best for UDP attacks, with an accuracy rate of 97.3% compared to other feature sets, as shown in Figure 4.13.

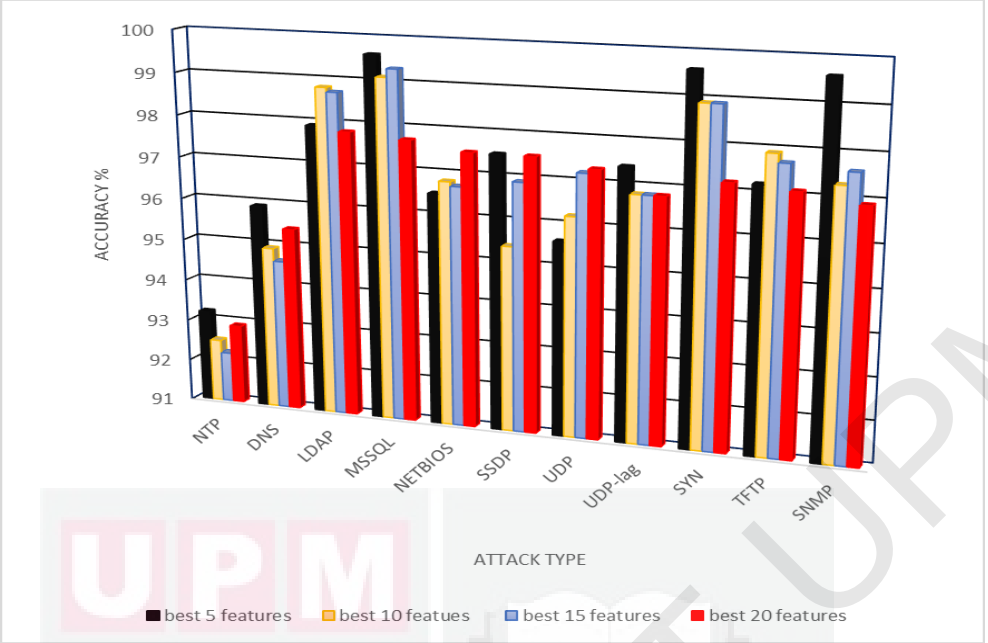


Figure 4.13 : Accuracy of ESPRT in Identifying DDoS Attacks of CICDDoS2019 Based on Different Features Selection Set

In addition, the F-scores of ESPRT were calculated for different feature sets and attack types in CICDDoS2019. The set of 5 features for DNS, MSSQL, UDP-Lag, SYN, and SNMP attack datasets resulted in the best F-score rates of 96.4%, 99.1%, 97.5%, 98%, 99.6%, and 99.7%, respectively, compared to other feature sets. On the other hand, the LDAP and TFTP attack datasets generated the best F-score values of 98.3% and 98.1%, respectively, using a 10-feature set. However, the highest F-score rates for the LDAP and TFTP attack datasets were achieved using a 15-feature set, with rates over 98%. Finally, the NTP, NETBIOS, SSDP, and UDP attack datasets achieved high F-score rates, as shown in Figure 4.14.

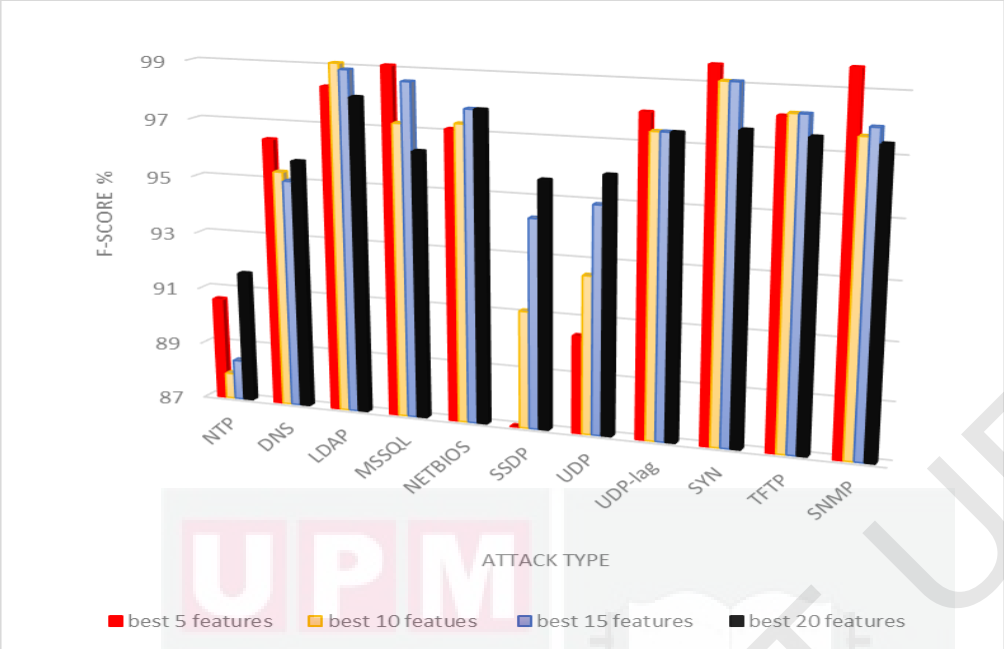


Figure 4.14 : F-score of ESPRT in Identifying DDoS Attacks of CICDDoS2019 Based on Different Features Selection Set

Finally, the probability of false alarm (FPR) for different feature sets of ESPRT using various attack datasets from CICDDoS2019 was also computed and presented in Figure 4.15. In general, ESPRT for all attack types generated very small values of FPR errors, all being less than 0.08. For specific instances, ESPRT for NTP, LDAP, MSSQL, NETBIOS, SSDP, UDP, SYN, and SNMP generated very low rates of FPR when the selected feature set was 5. These values were 0.02, 0.003, 0.0007, 0.006, 0.011, 0.004, and 0.005, respectively. This technique, based on NTP, LDAP, and SYN attack datasets, achieved lower rates of probability of false alarm when the number of selected features was 5 and 10, as shown in Figure 4.15. DNS produced an FPR of 0.01 when the number of selected features was 10. Finally, UDP-Lag had an FPR of 0.052 when the number of selected features was 20, which is the lowest value compared to other feature selected sets for the same attack dataset, as shown in Figure 4.15.

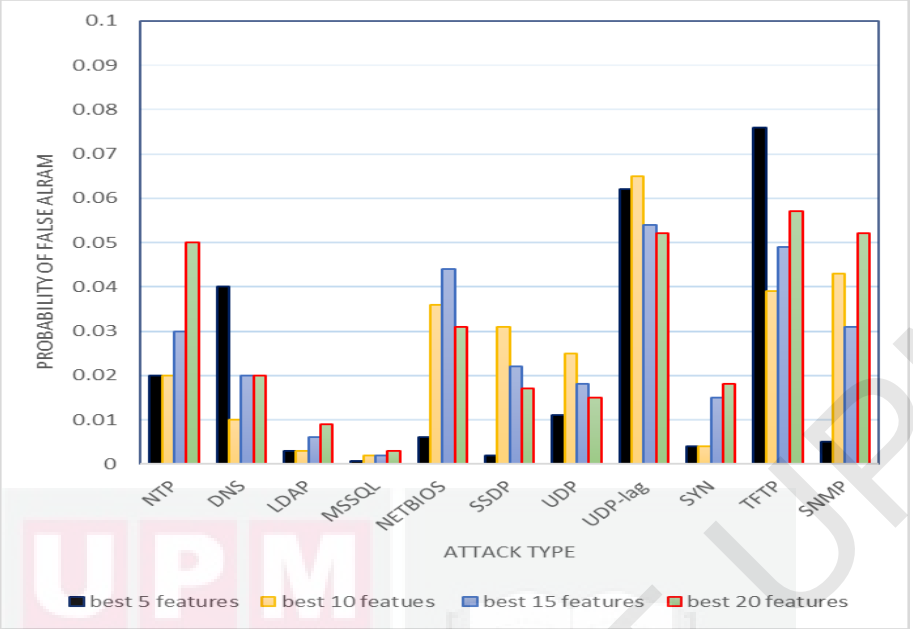


Figure 4.15 : FPR of ESPRT in Identifying DDoS Attacks of CICDDoS2019 Based on Different Features Selection Set

4.10 Time Cost of ESPRT based on Different Set of Features Selection

The time cost or performance of the multi-feature ESPRT technique was tested by calculating the execution time for different types of attacks. The computation was performed for various sets of selected features to highlight the differences between them. Figure 4.16 clearly illustrates the effectiveness of feature selection in improving the detection process. The execution time decreased for all attacks when the number of selected features decreased. The execution speed of ESPRT using five features is faster than for other selected feature sets. Similarly, the execution time of ESPRT using ten features is faster than for 15 and 20 features. Finally, the execution time of ESPRT based on 20 features was slower than for the other feature selection sets.

For instance, when the number of selected features was five, the NTP attack dataset took 29ms to process, while it required 65ms when the number of features was 10.

However, ESPRT took 185ms and 273ms when the number of selected features was 15 and 20, respectively. Moreover, the UDP-Lag attack required less time to be handled for all feature selection sets compared to other attack types, as shown in Figure 4.16. The times were approximately 12ms, 27ms, 29ms, and 27ms when the number of features was 5, 10, 15, and 20, respectively. On the other hand, the MSSQL attack dataset took longer to execute compared to others. ESPRT for this attack dataset took around 200ms, 668ms, 429ms, and 668ms when the number of features was 5, 10, 15, and 20, respectively. Finally, the execution time of multiple features of ESPRT for the other attack datasets varied, and the best attack dataset is the one with the shortest execution time.

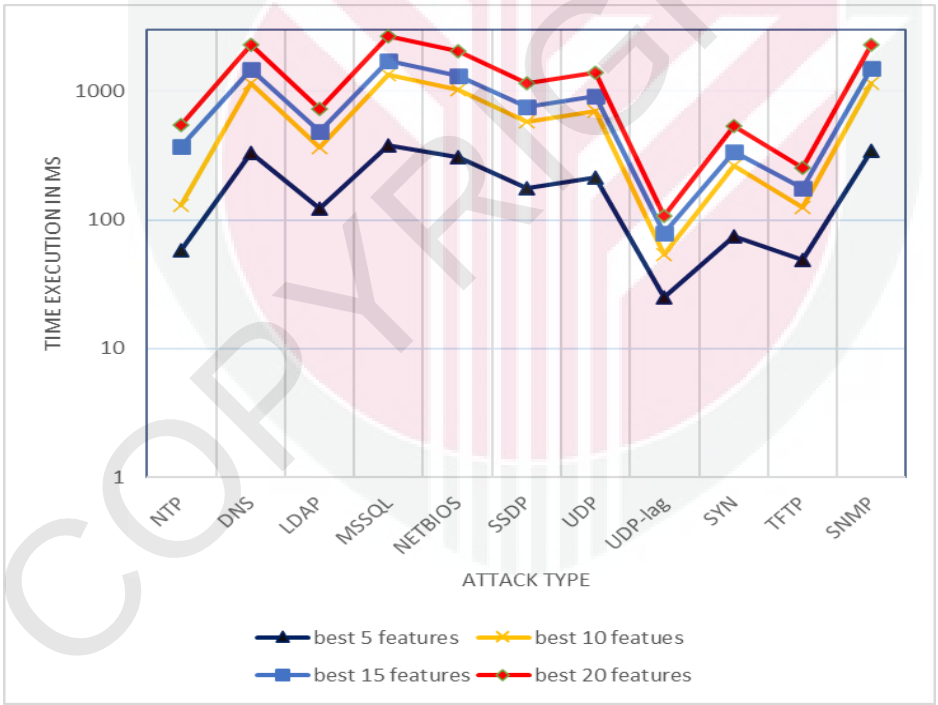


Figure 4.16 : Time Execution of ESPRT in Identifying DDoS Attacks of CICDDoS2019 Based on Different Features Selection Set

4.11 Comparison with State-of-the-Art Methods Using CICDDoS2019 Dataset

A comparison was conducted between the multiple features of ESPRT and other approaches to assess their detection effectiveness. The evaluation was performed on the CICDDoS2019 dataset, which contains various recent types of DDoS attacks. Several important metrics from the confusion matrix were employed for the comparison, including the detection rate, accuracy, and probability of false alarm rate. The results of ESPRT were compared with others based on using feature selection techniques in the first part and the best individual feature in the second part.

4.11.1 Comparison Based on Using Feature Selection Techniques

This subsection presents the results of confusion matrix metrics for ESPRT and other methods using feature selection techniques. Firstly, ESPRT generated higher metrics for most DDoS attack types in the CICDDoS2019 dataset compared to other techniques. This was achieved by using a set of ten features generated by CICFlowMeter and selected through feature selection techniques.

Table 4.6 : Multi Feature of ESPRT vs. State-of-the-Art Approaches Using CICDDoS2019 Dataset and Feature Selection Techniques

Reference	Approach name	Feature selection method	Number of features	Accuracy	F-score	TPR or Recall	FPR	Time-Cost (s)
Our approach	ESPRT	ANOVA, Random Forest, Extra Tree, and Pearson correlation	5	97.27	90.59	88.45	0.02	0.189
			10	96.78	95.7	94.91	0.02	0.625
			15	96.96	96.41	95.89	0.02	0.831
			20	96.73	96.44	95.88	0.03	1.276
			25	97.23	95.12	96.01	0.02	1.712
			30	97.12	95.33	95.55	0.02	2.148
			35	96.23	94.74	95.59	0.02	2.584
			40	96.55	94.16	97.89	0.05	3.02
			45	97.20	95.67	96.32	0.01	3.456
			50	97.17	95.16	96.44	0.02	3.892
			55	96.90	93.11	97.33	0.03	4.328
Khoei et al. [147]	bagging-based	Pearson's correlation coefficient and tree based.	21	93	-	94.8	9.5	-
	boosting-based			92.2	-	94.04	9.3	-
	stacking-based			93.4	-	96	8.9	-
Gaur and Kumar [146]	Random forest	Chi-square	35	91.55	95	92	0.01	-
	Decision tree		40	91.53	94	91	0.01	-
	KNN		40	90.41	93	90	0.04	-
	XGBoost		50	92.6	95	92	0	-
	Random forest	Extra tree	45	91.79	95	92	0	-
	Decision tree		15	91.78	94	92	0.01	-
	KNN		20	91.44	94	91	0.04	-
	XGBoost		30	92.78	94	93	0	-
	Random forest	ANOVA	55	91.91	94	94	0	-
	Decision tree		35	91.39	94	95	0.01	-
	KNN		15	91.28	94	95	0.04	-
	XGBoost		15	98.34	98	98	0	-

Table 4.6 : Continued

Sharafaldin et al. [22]	ID3	Random Forest	5	65	69	-	-	-
	Random Forest			56	62	-	-	-
	Naïve Bayes			11	5	-	-	-
	Logistic Regression			2	4	-	-	-
Gaur and Kumar [157]	Random Forest	Mutual Information	45	96.77	96	86	0	750
	XGBoost			96.67	96	85	0	1166
	DNN			96.17	82	58	0.03	590.68
	SNN			95.14	80	56	0.02	438.49
	DT			91.29	92	84	0.01	64.29

First of all, the accuracy of ESPRT fell in the range of 97.27 to 96.23 when the number of features was a multiple of 5 (from 5 to 55). These accuracy values were higher and better than those of other state-of-the-art methods. For example, Khoei et al. [147] used three different approaches: bagging-based, boosting-based, and stacking-based, combined with two other feature selection methods: Pearson's correlation coefficient and tree-based approaches. The accuracy for bagging-based, boosting-based, and stacking-based methods, with 21 selected features, were 93%, 92.2%, and 93.4% respectively, all lower than the accuracy of the ESPRT approach.

Furthermore, Gaur and Kumar [146] tested four different approaches: Random Forest, decision tree, KNN, and XGBoost, with different feature selection methods for each, such as Chi-square, Extra tree, and ANOVA. The accuracy of these approaches ranged from 90.41% to 92.6% with varying numbers of features selected using the Chi-square method, as shown in Table 4.6. However, when the Extra tree approach was used as the feature selection method, the average accuracy of these approaches was 91.94%. On the other hand, Random Forest, decision tree, and KNN achieved around 91% accuracy, while XGBoost achieved around 98% accuracy when ANOVA was the feature selection method.

Moreover, Sharafaldin et al. [22] also tested four detection methods: ID3, Random Forest, Naïve Bayes, and Logistic Regression, with Random Forest as the feature selection method. The accuracy values for these approaches were 65%, 56%, 11%, and 2% respectively, much lower than the accuracy of ESPRT, as shown in Table 4.6. Gaur and Kumar [157] used five different methods with Mutual Information as the feature selection technique. The accuracy range for these methods was between

91.29% to 96.77%, still lower or on par with the accuracy of the ESPRT method. Therefore, the accuracy of ESPRT using a combination of feature selection was higher than most others, except for XGBoost with ANOVA, which achieved an accuracy of around 98% when ANOVA was used as the feature selection technique.

Finally, the F1-score for ESPRT was better than that of other approaches when the number of selected features was 15 or 20, as shown in Table 4.6. The true positive rate was also better or close to most approaches, except for some methods such as stacking-based and XGBoost, which achieved 96% and 98% respectively, as shown in Table 4.6. The false positive rate (FPR) for ESPRT was fell in the range of 0.1 to 0.3, which is lower or similar to the FPR values of other approaches used in the comparison. Additionally, ESPRT required less execution time or complexity, leading to reduced time complexity or cost and improved performance of the detection approach compared to others as shown in Table 4.6. For example, Gaur and Kumar [157] need 1166 seconds when XGBoost was a feature selection method with 45 features compared to our method that needs 3.456 seconds using the same number of features. Therefore, multiple features of ESPRT using combination of four feature selection was better than other because it generated high accuracy and needed less time complexity compared with others.

4.11.2 Comparison Based on Using Best Signal Feature

This subsection presents the results of the confusion matrix metrics for ESPRT and other methods using specific features generated by CICFlowMeter. These features are f72, f71, f61, f58, f52, f42, f41, f39, f38, f37, f21, f19, and f14. The detection rate is the first metric used to demonstrate the effectiveness of ESPRT using these

features across different DDoS attacks. For most features of ESPRT associated with different attacks, the detection rate is higher than that of the HTM-KNN method [139] as shown in Table 4.7. The average detection rate range for ESPRT using these features was between 0.97 and 0.99, whereas the range for the HTM-KNN method was between 0 and 0.91. For instance, in identifying the UDP-Lag attack, the detection rate for HTM-KNN was 0, while ESPRT achieved rates of 0.92, 0.98, 0.99, and 0.99 using features f61, f37, f52, and f42 respectively. For ESPRT, using features f38, f39, f41, and f72 resulted in a detection rate of 0.99, while the remaining ESPRT features achieved a detection rate of 0.98, as mentioned in Table 4.7.

For most attacks, the detection rate of ESPRT features exceeded that of the PCC-AE-DNN [155], with the exception of SYN and TFTP. The detection rate range for PCC-AE-DNN was 0.56 to 0.99, while the average detection rate range for ESPRT using these features was between 0.97 and 0.99. PCC-AE-DNN exhibited a near-perfect detection rate close to 0.99 for identifying SYN and TFTP attacks, while the average detection rate for ESPRT features was 0.89 and 0.98 for identifying SYN and TFTP attacks respectively. Conversely, ESPRT achieved a detection rate higher or equal to that of FEDDM [28] which was nearly 0.99 for most attack types, except UDP-Lag, SYN, and TFTP, as indicated in Table 4.7.

Another metric used to compare ESPRT method and other methods is the false positive rate (FPR). Both the multi-selected features of ESPRT and the FEDDM approach [28] achieved low FPR values, nearly approaching zero. On average, ESPRT features had lower FPR values than FEDDM for all attack types, except for

NTP, SYN, and TFTP attacks. The differences between FEDDM and ESPRT features based on FPR are shown in Table 4.7.

Table 4.7 : Detection Rate, FPR, Accuracy of ESPRT Using Best Single Features of ESPRT vs other Approaches Based on CICDDoS2019 Dataset

Metric	Attack Types	HTM-KNN [139]	PCC-AE-DNN [155]	FEDDM [28]	Multi-features ESPRT						
					f14, f19, f21, f58, f71	f38, f39, f41, f72	f42	f52	f37	f61	Average of these features
Detection Rate	NTP	NA*	0.973	0.9956	0.9949	0.9939	0.9939	0.9939	0.9937	0.9939	0.9940
	DNS	NA	0.686	0.9979	0.9992	0.9993	0.9992	0.9992	0.9974	0.9993	0.9989
	LDAP	0.91	0.565	0.9965	0.9972	0.9986	0.9986	0.9985	0.9985	0.9986	0.9983
	MSSQL	0.77	0.949	0.9997	0.9993	0.9993	0.9993	0.9993	0.9993	0.9993	0.9993
	NETBIOS	0.87	0.994	0.9992	0.9992	0.9992	0.9992	0.9992	0.9992	0.9992	0.9992
	SSDP	0.80	0.982	0.9930	0.9988	0.9988	0.9711	0.9988	0.9988	0.9812	0.9912
	UDP	0.80	0.984	0.9999	0.9990	0.9990	0.9990	0.999	0.9968	0.999	0.9986
	UDP-lag	0	0.939	0.9975	0.9845	0.9909	0.9908	0.9908	0.9843	0.9218	0.9771
	SYN	0.90	0.998	0.9999	0.9978	0.9978	0.9935	0.9978	0.9978	0.4130	0.8996
	TFTP	NA	0.997	0.9947	0.9971	0.9970	0.9519	0.9970	0.9645	0.9970	0.9840
(FPR), probability of false alarm	NTP	NA	NA	0.0021	0.005	0.0060	0.0060	0.0055	0.0359	0.0060	0.0107
	DNS	NA	NA	0.0201	0.0018	0.0012	0.0014	0.0020	0.0060	0.0012	0.0022
	LDAP	NA	NA	0.0098	0.0028	0.0028	0.0028	0.0025	0.0025	0.0028	0.0027
	MSSQL	NA	NA	0.0078	0.0016	0.0016	0.0013	0.0013	0.0013	0.0015	0.0014
	NETBIOS	NA	NA	0.0091	0.0016	0.0016	0.0015	0.0021	0.0024	0.0022	0.0019
	SSDP	NA	NA	0.0222	0.0028	0.0028	0.0031	0.0025	0.0065	0.0030	0.0034
	UDP	NA	NA	0.1576	0.0015	0.0015	0.0031	0.0021	0.0040	0.0015	0.0022
	UDP-lag	NA	NA	0.1123	0.0292	0.0146	0.0190	0.0199	0.0568	0.0703	0.0349
	SYN	NA	NA	0	0.0047	0.0047	0.0048	0.0043	0.0060	0.0988	0.0205
	TFTP	NA	NA	0.132	0.0070	0.0070	0.8781	0.0064	0.9049	0.0241	0.3045

*NA: Not Available

The final metric used to assess the effectiveness of ESPRT using these features in comparison with other approaches is accuracy. On average, the accuracy of these ESPRT features outperformed the accuracy of PCC-AE-DNN [155] for all types of DDoS attacks, except SYN and TFTP. The average accuracy of the selected ESPRT features for identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, UDP, and UDP-Lag attacks was 99.2%, 99.8%, 99.8%, 99.9%, 99.9%, 99.4%, 99.8%, and 97.2% respectively, compared to 98.9%, 68.8%, 72.8%, 94%, 99.4%, 98.2%, 98.2%, and 93.9% for PCC-AE-DNN, as shown in Figure 4.17.

Furthermore, ESPRT based on these features also exhibited higher and superior accuracy compared to HTM-KNN for all DDoS attack types, except for the SYN attack. For instance, the accuracy of ESPRT in identifying LDAP and UDP attacks was 99.8%, while HTM-KNN achieved an accuracy of 99%. In the case of MSSQL and NETBIOS attacks, ESPRT achieved an accuracy of 99.9%, while HTM-KNN achieved 99%. Similarly, ESPRT achieved accuracies of 99.4% and 97.2% for SSDP and UDP-Lag attacks respectively, while HTM-KNN achieved accuracies of 99% and 96% for the same attacks. Finally, HTM-KNN achieved an accuracy of 99% in detecting SYN attacks, whereas ESPRT achieved an accuracy of 94.9% for the same attack type, as depicted in Figure 4.17.

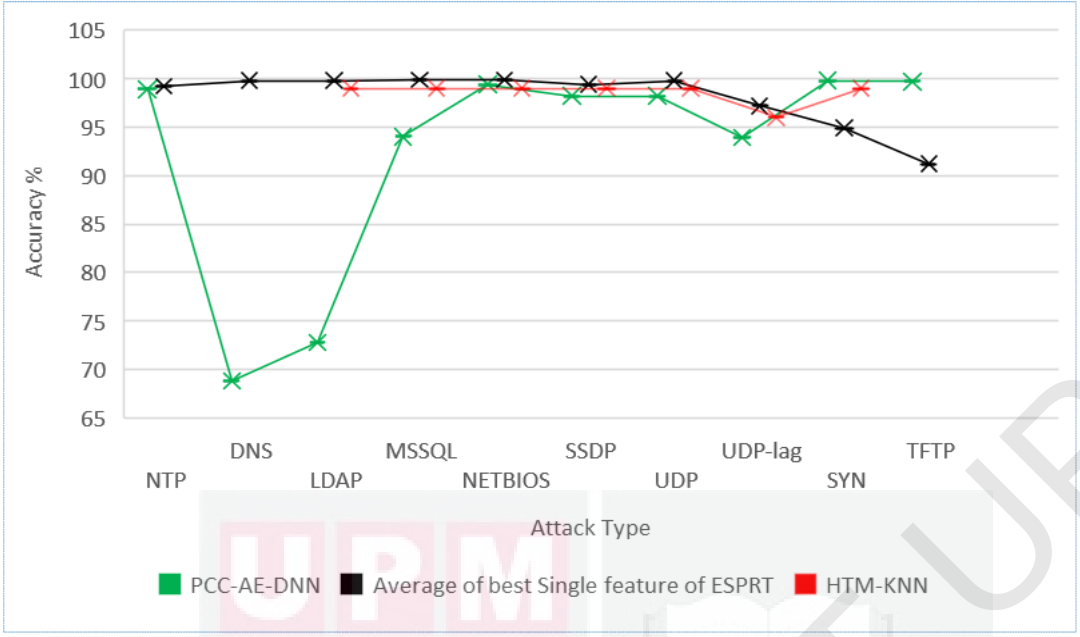


Figure 4.17 : Comparison of Average of Accuracy for best Single Features of Multi-Features ESPRT Detection and other Approaches Using CICDDoS2019 Dataset

4.12 Comparison with State-of-the-Art Methods Using DARPA 2000 Dataset

ESPRT was compared with other approaches that incorporated entropy in their implementation, using the DARPA 2000 dataset and important confusion metrics such as accuracy, detection rate, FPR, and FNR. Entropy relies on two main components to detect DDoS attacks: window size and threshold. Altering threshold values affects the results of entropy, consequently impacting the performance of the detection process. ESPRT combines both entropy (E) and Sequential Probability Ratio Test (SPRT). ESPRT addresses the challenge of fixed threshold values in entropy-based detection by employing a dynamic threshold to achieve optimal values for the detection process. In general, ESPRT demonstrated high accuracy, detection rate, and low FPR and FNR in comparison to other approaches, as depicted in Table 4.8.

For instance, when window size values ranged between 5 to 120, ESPRT achieved an average accuracy metric of 90.5%. In contrast, the average accuracy metric for the real-time-based detection [137] method was 90.3% when threshold values ranged between 0.1 to 0.9. The average accuracy for HCA with Labelling [152] was 72.42%, while it increased to 86.73% for HCA with Naïve Bayes Classification [153].

The average detection rate or TPR for ESPRT was 92.9%, while Chaos Theory [151] achieved an average of 90.7%. Conversely, TPR for HCA with Labelling [152], HCA with Naïve Base Classification [153], and H-IDS [154] approaches were 16.6%, 59.1%, and 92.1% respectively, all lower than ESPRT's detection rate. The authors of the real-time DDoS attack [137] did not provide the average TPR, only presenting the maximum value, which was close to 100%.

Finally, ESPRT achieved low FPR and FNR values, nearly approaching 0, based on the DARPA 2000 dataset. The average FPR for ESPRT was 0.207, while Chaos Theory [151] exhibited an average of 0.233. The average FPR for HCA with Labelling [152] was 0.237. Conversely, the FPR values for HCA with Naïve Bayes Classification [153] and H-IDS [154] were slightly lower than ESPRT's, at 0.119 and 0.18, respectively. Finally, ESPRT's FNR was 0.069, indicating better performance, as this error type was nearly minimized.

Table 4.8 : Comparison ESPRT with other Approaches Using DARPA 2000 Dataset

Method	Accuracy	DR or TPR	FPR	FNR	Window size	Thr
ESPRT	89.6 %to 93.2 % (Average =90.5%)	0.914 to 0.957 (Average =0.929)	0.154 to 0.270 (Average= 0.207)	0.042 to 0.078 (Average= 0.069)	Window size range between 5 to 120 packets	Dynamic threshold
Real-time DDoS attack [137]	22% to 100% (average =90.3%)	NA to 1 (They mentioned the maximum value only)	0 to NA	0.18 to NA	No need for window size in this method.	Threshold values range between 0.1 to 0.9 (22 % when thr=0.3, 70% when thr=0.4, 100% when thr=0.5 or higher).
Chaos theory [151]	NA	0.88 to 0.94 (average=0.907)	0.05 to 0.45 (average= 0.233)	NA	No need for window size in this method.	Threshold values range between 0.1 to 0.9
HCA with Labelling [152]	41% to 95% (Average=72.42%)	0.048 to 0.383 (Average= 0.166)	0.167 to 0.523 (Average = 0.237)	NA	Window size range between 10 to 120 packets	No need for Thr in this method.
HCA with Naïve Base Classification [153]	58 % to 100% (Average= 86.73)	0.560 to 1 (Average = 0.591)	0 to 0.352 (Average = 0.119)	NA	Window size range between 10 to 120 packets	No need for Thr in this method.
H-IDS [154]	NA	0.921	0.18	NA	No need for window size in this method.	No need for Thr in this method.

NA: Not available, DR: Detection rate, TPR: True positive rate,
FPR: False positive rate, FNR: False negative rate, Thr: threshold.

4.13 Comparing Original and Revised Version of CICFlowMeter

To illustrate the differences in ESPRT results when applying flows from the original and revised versions of CICFlowMeter, traffic data captured during the fifth week of the 1998 DARPA dataset was used. This dataset, depicted in Figure 4.18, contained various DDoS attacks such as neptune and smurf.

The number of flows generated by the original version amounted to 108,323, whereas the revised version produced 311,208 flows. This substantial discrepancy can be attributed to the neptune attack, where hackers inundated the targeted system with an extensive number of SYN flows. Since these SYN flows lacked FIN packets, the original flow tool disregarded them. However, the revised version accurately accounted for and generated these SYN flows.

ESPRT using revised tool is better than original tool in terms of performance. To highlight the distinctions between them, confusion metrics such as sensitivity, probability of false alarm, specificity, miss rate, and precision were utilized. The results of randomly selected features from three groups were employed to illustrate the outcomes. These groups encompass number of packets-based features, active and idle-based features, and flow IAT-based features. Generally, when employing the revised version, sensitivity, specificity, and precision of ESPRT approached 1, whereas with the original version, they fell below 0.5. Furthermore, the probability of false alarm and miss rate results were nearly 0 with the revised version, while they hovered around 0.5 using the original version across various features, as outlined in Table 4.9.

For example, when considering features f8, f10, and f11, sensitivity, specificity, and precision achieved values of 0.995, 1, and 1, respectively, using the revised version. Conversely, with the original version, these metrics yielded values of 0.489, 0.279, and 0.389. In terms of probability of false alarm and miss rate for these features, the revised version resulted in values of 0 and 0.004, while the original version exhibited values of 0.721 and 0.511, respectively, as depicted in the first row of Table 4.9.

Another example involves active and idle-based features such as f75, f77, and f78. When employing the revised version, sensitivity and precision were close to 0.99, whereas the original version produced values of 0.489 and 0.388, respectively. Regarding specificity, the revised version achieved a value of 0.975, whereas the original version yielded a value of 0.266 for these features. Probability of false alarm and miss rate for the revised version were 0.024 and 0.01, respectively. In contrast, the original version yielded values of 0.734 and 0.511 when f75, f77, and f78 were employed as features, as illustrated in the third row of Table 4.9.

Table 4.9 : Differences between Original and Revised Version of CICFlowMeter Tool by Using Active and Idle, Number of Packets, and Flow IAT-Based Feature of ESPRT Detection Using DARPA 1998

ESPRT	Sensitivity		probability of false alarm		Specificity		Miss-Rate		Precision		Group feature name
	Original	*Revised*	*Original*	*Revised*	*Original*	*Revised*	*Original*	*Revised*	*Original*	*Revised*	
Total Fwd Packets (f8), Flow Packets/s (f10), Fwd Packets/s (f11)	0.489	0.995	0.721	0	0.279	1	0.511	0.004	0.389	1	Number of Packets
Total Bwd Packets (f9), Bwd Packets/s (f12)	0.567	0.992	0.989	0	0.011	1	0.433	0.007	0.438	1	Number of Packets
Active Min (f75), Active Max (f77), Active Std (f78)	0.489	0.99	0.734	0.024	0.266	0.975	0.511	0.01	0.388	0.99	Active and Idle
Active Mean (f76), Idle Mean (f80)	0.490	0.989	0.733	0.014	0.267	0.985	0.510	0.01	0.390	0.994	Active and Idle
Idle Min (f79), Idle Max (f81), Idle Std (f82)	0.489	0.99	0.734	0.028	0.266	0.972	0.511	0.01	0.388	0.985	Active and Idle
F29, f30, f31, f32, f34, f36, f37	0.489	0.995	0.721	0	0.279	1	0.511	0.004	0.389	1	Flow IAT
F33, f35	0.487	0.99	0.721	0	0.279	1	0.513	0.009	0.389	1	Flow IAT
F38, f39, f41, f42	0.568	0.99	0.994	0	0.006	1	0.432	0.007	0.438	1	Flow IAT

Finally, considering Flow IAT-based features such as f29, f30, f31, f32, f34, f36, f37, f33, and f35, the revised version produced sensitivity, specificity, and precision values of 0.995, 1, and 1, respectively. These values contrasted with the original version, which generated metrics of 0.489, 0.279, and 0.389, as shown in the sixth and seventh rows of Table 4.9.

Other metrics of the confusion matrix were employed to highlight the differences between ESPRT using the revised and original versions, such as accuracy and F-score. First of all, the accuracy of ESPRT using the revised version was over 99% when applied to features belonging to the number of packets, active and idle, and flow IAT groups for identifying DDoS attacks within the DARPA 1998 dataset. However, the accuracy using the original version ranged from 33% to 38% for different features within these selected groups, as depicted in Figure 4.18. The reduced accuracy of the original version stemmed from problems in flow generation, including the exclusion of flows lacking FIN packets and the premature termination of flows before packets were adequately gathered in certain flows.

Another metric that can be used to demonstrate the performance contrast between ESPRT using the revised and original versions of CICFlowMeter is the F-score. The F-score of the ESPRT detection method exceeded 99% for most selected features within the number of packets, active and idle, and flow IAT groups when the revised version was used to generate TCP flows. Conversely, the F-score for this detection approach ranged from 43% to 50% when the original version tool was employed to generate flows based on the selected trace from Thursday of the fifth week of the DARPA 1998 dataset, as shown in Figure 4.18.

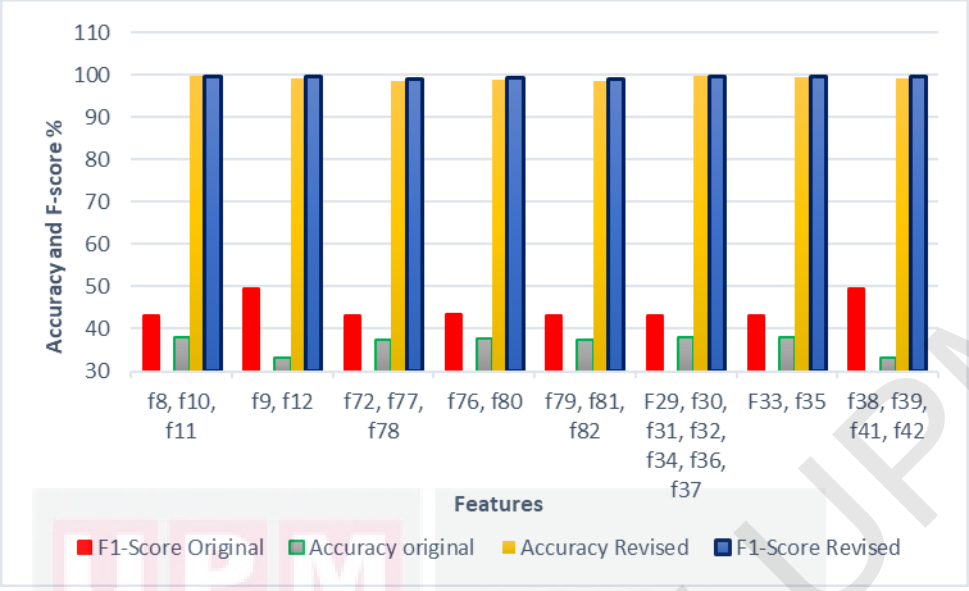


Figure 4.18 : Accuracy and F1-Score values for Active and Idle, Number of Packets, and Flow IAT-Based Feature of ESPRT Using Original and Revised CICFlowMeter Using DARPA 1998

Furthermore, the multi-feature ESPRT utilizing the revised versions of CICFlowMeter were compared with other approaches. These techniques incorporated the entropy technique in their implementations and employed the DARPA 1998 dataset to assess their performance. Crucial confusion metrics were employed in this evaluation, including precision, miss rate, accuracy, probability of false alarm, and sensitivity. ESPRT based on the revised version exhibited higher values of sensitivity, accuracy, and precision, all exceeding 99%, in comparison to other techniques, as presented in Table 4.10. However, the Signal Victim Entropy and PWI approach [134] achieved a sensitivity of 98.3%, while its multi-victim version [134] recorded 95%. In contrast, SSAE-SVM [140]. Finally, entropy-CNN [148] approach demonstrated a sensitivity, accuracy, and precision of 97%. Consequently, the multi-feature ESPRT utilizing the revised version showcased superior accuracy, sensitivity, and precision compared to other techniques, as indicated in Table 4.10.

Finally, the probability of false alarm and miss rate for the revised version of ESPRT approached zero, which was lower than that of other techniques. However, the Signal Victim Entropy and PWI approach [134] resulted in a 17% FPR and a 0% miss rate, while its multi-Victim counterpart [134] recorded a 16.6% FPR and a 33.3% miss rate. Finally, SSAE-SVM [140] produced a probability of false alarm of 4.03%. Therefore, the revised version of ESPRT exhibited lower values for probability of false alarm and miss rate, demonstrating its superiority compared to other approaches using the DARPA 1998 dataset, as outlined in Table 4.10.

Table 4.10 : Differences between Active and Idle, Number of Packets, and Flow IAT-based Feature of ESPRT using Revised CICFlowMeter and other Approaches Using DARPA 1998

Name of Approach	Sensitivity	Probability of false alarm	Accuracy	Miss Rate	Precision
ESPRT based on Revised CICFlowMeter	99.13	0.82	99.1	0.76	99.61
Signal victim of Entropy & PWI Approach [134]	98.3	17	NA*	0	NA
Multi victim of Entropy & PWI Approach [134]	95.0	16.6	NA	33.3	NA
SSAE-SVM [140]	96.4	4.03	99.0	NA	NA
Entropy-CNN [148]	97.0	NA	96.7	NA	96.7

*NA: Not Available

4.14 Summary

This chapter presents the hardware specifications used for conducting the experiments. Several datasets, such as CICDDoS2019, DARPA 2000, DARPA 1999, and DARPA 1998, were employed to evaluate the ESPRT technique. These datasets encompass various types of DDoS attacks, and they are publicly accessible online for research purposes. It is important to note that these datasets may contain errors, such as missing, duplicate, irrelevant, or incomplete data values, which can be

caused by machines or legitimate users. To enhance detection accuracy and facilitate proper feature selection techniques, preprocessing steps were applied, including cleaning, balancing, and transformation. Cleaning was employed to rectify errors, while transformation was used to convert data into numerical or decimal values. Furthermore, techniques like SMOTE were utilized to address the issue of unbalanced datasets.

The chapter also presents the results of ESPRT based on different datasets, using essential confusion metrics. When the window size was set to 100 flows and the source IP address was chosen as a feature from the Friday of the fifth week in the DARPA 1998 dataset, ESPRT achieved an accuracy of 99.3% and an f-score of 99.6%. However, since the source IP address alone is insufficient to detect novel and diverse DDoS attacks, additional features were extracted using CICFlowMeter.

ESPRT utilizing most single features exhibited higher values of accuracy, sensitivity, specificity, PPV, and NPV, all of which approached or exceeded 99%. Conversely, the miss rate, fall-out, FDR, and FOR showed low values close to 0. Some of the features that yielded these results were f14, f19, f20, f21, f37, f38, f39, f41, f42, f52, f58, f61, f71, and f72. The average detection rate of ESPRT employing these features exceeded 99% for identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, and UDP attacks. The detection rate for UDP-Lag and TFTP exceeded 98%, while it was approximately 90% for SYN attacks. The average FPR for most of these attacks was close to 0. The accuracy of ESPRT utilizing these features in identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, UDP, UDP-

Lag, SYN, and TFTP attacks was 99.2%, 99.8%, 99.8%, 99.9%, 99.9%, 99.4%, 99.8%, 97.2%, 95%, and 92%, respectively.

CICFlowMeter was employed to generate flows and extract 82 features from each flow. A combination of four feature selection techniques—ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation—were employed to choose the best features. Features selected using these techniques were combined, and the most important ten features with the highest frequency were chosen as inputs for the subsequent stage, which involved the detection method. The main objective of using feature selection methods was to reduce execution time and enhance the performance of the detection method.

When utilizing a set of 5 features for DNS, MSSQL, SSDP, UDP-Lag, SYN, and SNMP attack datasets in the CICDDoS2019 dataset, ESPRT demonstrated higher accuracy rates compared to other feature sets, achieving accuracy rates of 95.9%, 99.6%, 97.5%, 97.4%, 99.6%, and 99.6%, respectively. For LDAP and TFTP attacks, the highest accuracy for ESPRT was achieved with a set of 10 features. Moreover, ESPRT achieved its highest accuracy of 96.7% in identifying NETBIOS using a set of 15 features. Finally, a 20-feature set proved to be the most effective for UDP attacks, with an accuracy rate of 97.3% compared to other feature sets. The performance of the multi-feature ESPRT technique was assessed by calculating the execution time for various types of attacks. The execution time of ESPRT decreased for all attacks as the number of selected features decreased.

Finally, the chapter addresses the issues arising from the generation of flows in the original version of CICFlowMeter, which impact the performance of detection approaches. These problems primarily arise with flows based on the TCP protocol. Flows generated using the original version of CICFlowMeter are terminated either when the setup timer expires or when FIN and ACK packets are sent in one direction. Consequently, packets that should belong to a specific flow either merge with other flows or create new ones when the proposed time for flow formation elapses. Additionally, flows lacking FIN and ACK packets are disregarded, such as SYN attack flows. This also results in the exclusion of flows that utilize RST packets to terminate a flow, contravening TCP specifications. The revised version of CICFlowMeter addresses these issues. The accuracy and f-score for ESPRT using the revised version exceeded 99% when features belonging to the number of packets, active and idle, and flow IAT groups were utilized to identify DDoS attacks in the DARPA 1998 dataset. However, when using the original version, the accuracy ranged between 33% and 38% for various features within these selected groups.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

DDoS attacks pose a significant threat as they disrupt the availability of legitimate services on targeted servers. Attackers employ a multitude of compromised or vulnerable machines to direct an overwhelming volume of traffic towards specific servers, causing service interruptions. The aftermath of successful DDoS attacks encompasses severe financial losses for the victims. This thesis aims to address the central research question posed in Chapter 1: "How can detection techniques identify up-to-date DDoS attack accurately using dynamic threshold and effective features that can be selected to decrease time complexity or cost and using effective flows or features generation tool to improve f-score, precision, and sensitivity metrics?". This is achieved by three goals. The first objective is to identify infected Ethernet interfaces and detect various kinds of up-to-date DDoS attacks, such as Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft SQL Server (MSSQL), Network Basic Input/Output System (NETBIOS), Simple Service Discovery Protocol (SSDP), and User Datagram Protocol (UDP), using dynamic threshold by implementing multiple features of entropy and Sequential Probabilities Ratio Test approach (E-SPRT).

Incoming packets sharing similar header characteristics are grouped into flows. CICFlowMeter generates 82 features for each flow, classified into 10 groups: identification-based features, total number of packets-based features, total length of packets per flow, flow IAT-based features, features related to flags, segment and

header size-based features, bulk-based features, subflow-based features, TCP-based features, and active/idle-based features. These flows, along with their corresponding Ethernet numbers and features, are stored in a database. To enhance detection accuracy, the stored data undergoes preprocessing techniques such as cleaning, balancing, and transformation. SMOTE technique is applied to rectify unbalanced datasets by introducing synthetic instances to the minor class, aligning the number of malicious instances with that of benign ones.

Multiple datasets are employed to validate the implemented techniques, including the CICDDoS2019 dataset and the DARPA (Defense Advanced Research Projects Agency) intrusion detection dataset. The CICDDoS2019 dataset encompasses a wide range of up-to-date DDoS intrusions, including UDP, SYN, DNS, MSSQL, SSDP, UDP-lag, TFTP, SNMP, NETBIOS, and LDAP attacks. In contrast, the DARPA dataset contains diverse attacks such as Smurf and Neptune. The evaluation of the detection stage employs confusion matrix metrics and comparisons with other methods to demonstrate the effectiveness of the design.

Furthermore, since attackers exhibit various behaviors for their malicious activities, 82 features were extracted from incoming flows. Sensitivity, probability of false alarm, specificity, and miss rate were computed for ESPRT using each of these features. The confusion matrix metrics were calculated for ESPRT using the best features and the CICDDoS2019 dataset, specifically features f14, f20, f19, f21, f37, f38, f39, f41, f42, f52, f58, f61, f71, and f72. The average detection rates for identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, and UDP using these features were consistently above 99%. The average detection rates for identifying

UDP-lag, SYN, and TFTP were 97%, 89%, and 98%, respectively. Most of these attacks exhibited an average False Positive Rate (FPR) close to 0. The accuracy of the average of these ESPRT features in identifying NTP, DNS, LDAP, MSSQL, NETBIOS, SSDP, UDP, UDP-lag, SYN, and TFTP attacks was respectively 99.2%, 99.8%, 99.8%, 99.9%, 99.9%, 99.4%, 99.8%, 97.2%, 95%, and 92%.

Moreover, the accuracy values of ESPRT using feature selection techniques were also computed. ESPRT's accuracy using a set of 5 features for DNS, MSSQL, SSDP, UDP-lag, SYN, and SNMP attack datasets surpassed other feature sets, achieving accuracy rates of 95.9%, 99.6%, 97.5%, 97.4%, 99.6%, and 99.6%, respectively. For LDAP and TFTP attacks, the highest accuracy was achieved using 10 features, while 15 features were found to provide the highest accuracy rate of 96.7% in identifying NETBIOS. Additionally, a 20-feature set proved optimal for UDP attacks, yielding an accuracy rate of 97.3% compared to other feature sets.

In addition, the average accuracy of ESPRT using the DARPA 2000 dataset was 90.5% with window sizes ranging from 5 to 120. Notably, ESPRT's threshold value was dynamic, in contrast to other methods reliant on fixed thresholds, potentially affecting detection rates when attackers alter their behaviors.

Furthermore, the second goal is to select relevant features in order to reduce time complexity or cost and improve performance of detection by implementing a new combination of machine learning techniques ANOVA, Extra Trees Classifier, Random Forest, and Correlation Matrix with Pearson Correlation approaches. Feature selection techniques reduced the execution time of ESPRT; for instance,

execution times were approximately 189ms, 625ms, 831ms, and 1.276ms for feature sets of 5, 10, 15, and 20, respectively.

Finally, the third goal is to analyze problems of original CICFlowMeter and validate the effectiveness of revised version of CICFlowMeter tool on detection approach as well. This will involve comparing the results obtained from both versions for feature extraction in order to improve accuracy, F-score, and precision in the detection approach. Original version of CICFlowMeter has problems during flow generation process. These problems particularly affect TCP flows. According to the original principle, TCP flows are terminated upon timeout, leading to packets that belong to different flows merging incorrectly. Another flawed principle involves closing TCP flows when ACK and FIN packets are sent in one direction, causing the exclusion of flows with only SYN or RST packets. These problems were rectified in the revised version. The accuracy, F-score, and precision for ESPRT using the revised version consistently exceeded 99% when features from the number of packets, active/idle, and flow IAT groups were used to identify DDoS attacks in the DARPA 1998 dataset. In contrast, the accuracy, F-score, and precision using the original version ranged between 33% and 38% for various features within these selected groups.

5.2 Limitation and Future Work

The combination of the entropy and Sequential Probability Ratio Test (SPRT) approach is a promising method for identifying DDoS attacks. This approach should be extended to apply to IoT traffic, as these machines have recently become prime targets for DDoS attacks. Researchers can leverage the effectiveness of this approach to distinguish between malicious IoT traffic and legitimate traffic. Another

significant target for DDoS attacks is Software-Defined Networks (SDN), which represent a new infrastructure that enables network administrators to program network devices and servers, thereby enhancing network performance. This infrastructure has also become a goal for DDoS threats, making its implementation a potential focus for future work.

Furthermore, the experiment was conducted using an offline dataset. To expedite the detection of DDoS threats, the detection approach needs to be extended to include infected flow traffic in a real network environment. Moreover, a combination of feature selection techniques was utilized to minimize the number of selected features by choosing relevant ones. Exploring other combinations of feature selection techniques is another avenue to improve accuracy and reduce false positive rates. This thesis used an extracted set of 81 features generated by the CICFlowMeter tool, which were grouped into 10 sets. Therefore, exploring additional features may enhance the detection rate and optimize the accuracy of DDoS attack detection.

Finally, the feature selection techniques employed in this thesis were based on machine learning techniques. However, adversarial examples, artificially constructed examples designed to deceive these techniques, can lead to misjudgments in selecting relevant features. This can be achieved by inserting poisoning attacks into the training set or altering the training set. As part of future work, the detection approach and feature selection techniques should be evaluated against adversarial examples to enhance scalability and performance.

REFERENCES

[1] S. Yu, *Distributed Denial of Service Attack and Defense*, 1st ed. in SpringerBriefs in Computer Science. New York, NY: Springer New York, 2014. doi: 10.1007/978-1-4614-9491-1.

[2] Imperva, "DDoS Threat Landscape Report Q2 2022," 2022. Accessed: Aug. 28, 2022. [Online]. Available: Downloads\Documents\DDoS-Threat-Landscape-Report-Q2-2022_2.pdf

[3] D. Menscher, "Identifying and Protecting against the Largest DDoS Attacks," Google Cloud Blog. Accessed: Aug. 28, 2022. [Online]. Available: https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks

[4] R. Chaganti, B. Bhushan, and V. Ravi, "The Role of Blockchain in DDoS Attacks Mitigation: Techniques, Open Challenges and Future Directions," *arXiv:2202.03617*, pp. 1–14, Feb. 2022, doi: https://doi.org/10.48550/arXiv.2202.03617.

[5] M. Pinho, "AWS Shield Threat Landscape Review: 2020 Year-in-Review," AWS Security Blog. Accessed: Jan. 20, 2023. [Online]. Available: https://aws.amazon.com/blogs/security/aws-shield-threat-landscape-review-2020-year-in-review/

[6] A. Jawad, L. Newton, A. Matrawy, and J. Jaskolka, "A Formal Analysis of the Efficacy of Rebooting as a Countermeasure Against IoT Botnets," in *ICC 2022 - IEEE International Conference on Communications*, Seoul, Korea: IEEE, May 2022, pp. 2206–2211. doi: 10.1109/ICC45855.2022.9838865.

[7] B. Stephens, A. Shaghaghi, R. Doss, and S. S. Kanhere, "Detecting Internet of Things Bots: A Comparative Study," *IEEE Access*, vol. 9, pp. 160391–160401, 2021, doi: 10.1109/ACCESS.2021.3130714.

[8] L. Pan, R. Qiu, A. Wang, M. Yang, Y. Chen, and A. Hu, "Measuring the Resolution Resiliency of Second-Level Domain Name," in *Intelligent Computing Proceedings of the 2022 Computing Conference, Volume 3*, London, United Kingdom: Springer, Cham, Jul. 2022, pp. 742–755. doi: 10.1007/978-3-031-10467-1_45.

[9] S. Yoon and M. Kang, "DDoS Attacks Detection in the Cloud using K-Medoids Algorithm," in *International Conference on Advanced Communication Technology, ICACT*, PyeongChang Kwangwoon_Do, Korea: IEEE, Mar. 2022, pp. 91–94. doi: 10.23919/ICACT53585.2022.9728838.

[10] J. Boonchai, K. Kitchat, and S. Nonsiri, "The Classification of DDoS Attacks Using Deep Learning Techniques," in *7th International Conference on Business and Industrial Research (ICBIR)*, Seoul, Korea: IEEE, Feb. 2022, pp. 544–550. doi: 10.1109/ICBIR54589.2022.9786394.

[11] J. M. Couretas, “Cyber Policy, Doctrine, and Tactics, Techniques, and Procedures (TTPs),” in *An Introduction to Cyber Analysis and Targeting*, 1st ed., Springer Nature Switzerland: Springer, Cham, 2022, ch. Chapter 2, pp. 13–36. doi: 10.1007/978-3-030-88559-5_2.

[12] S.-Y. K. Wang and M.-L. Hsieh, “The Insecure Nature of Cyberspace,” in *Digital Robbery ATM Hacking and Implications*, 1st ed., Switzerland AG: Springer Cham, 2021, ch. Chapter 1, pp. 1–58. doi: 10.1007/978-3-030-70706-4_1.

[13] Inc. Cisco Systems, “Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper - Cisco,” 2020. Accessed: Aug. 31, 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[14] S. K. Allam and G. S. Prasad, “Prevention of DDoS Attacks and Detection on Cloud Environment—A Review and Survey,” *Advances in Intelligent Systems and Computing*, vol. 1171, pp. 463–470, 2021, doi: 10.1007/978-981-15-5400-1_47/FIGURES/1.

[15] D. Park, S. Kim, H. Kwon, D. Shin, and D. Shin, “Host-Based Intrusion Detection Model Using Siamese Network,” *IEEE Access*, vol. 9, pp. 76614–76623, 2021, doi: 10.1109/ACCESS.2021.3082160.

[16] A. Verma, R. Saha, N. Kumar, G. Kumar, and Tai-Hoon-Kim, “A Detailed Survey of Denial of Service for IoT and Multimedia Systems: Past, Present and Futuristic Development,” *Multimedia Tools and Applications* , vol. 81, no. 14, pp. 19879–19944, May 2022, doi: 10.1007/S11042-021-11859-Z.

[17] M. Nooribakhsh and M. Mollamotalebi, “A Review on Statistical Approaches for Anomaly Detection in DDoS Attacks,” *Information Security Journal: A Global Perspective*, vol. 29, no. 3, pp. 118–133, Feb. 2020, doi: 10.1080/19393555.2020.1717019.

[18] P. Dong, X. Du, H. Zhang, and T. Xu, “A Detection Method for a Novel DDoS Attack against SDN Controllers by Vast New Low-Traffic Flows,” in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia: IEEE, May 2016, pp. 1–6. doi: 10.1109/ICC.2016.7510992.

[19] T. Fawcett, “An introduction to ROC Analysis,” *Pattern Rec. Let.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/J.PATREC.2005.10.010.

[20] M. M. Salim, S. Rathore, and J. H. Park, “Distributed Denial of Service Attacks and its Defenses in IoT: a Survey,” *Journal of Supercomputing*, vol. 76, no. 7, pp. 5320–5363, Jul. 2019, doi: 10.1007/S11227-019-02945-Z.

[21] R. K. Batchu and H. Seetha, "A Generalized Machine Learning Model for DDoS Attacks Detection using Hybrid Feature Selection and Hyperparameter Tuning," *Computer Networks*, vol. 200, no. 108498, pp. 1–8, Dec. 2021, doi: 10.1016/J.COMNET.2021.108498.

[22] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in *International Carnahan Conference on Security Technology*, Chennai, India: IEEE, Oct. 2019, pp. 1–8. doi: 10.1109/CCST.2019.8888419.

[23] V. Sekar, N. Duffield, O. Spatscheck, J. van der Merwe, and H. Zhang, "LADS: Large-Scale Automated DDoS Detection System," in *USENIX Advanced Computing System Association*, Boston MA: USENIX Association, May 2006, pp. 1–14. doi: 10.5555/1267359.1267375.

[24] J. Mirkovic and P. Reiher, "D-WARD: a Source-End Defense against Flooding Denial-of-Service Attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 3, pp. 216–232, Sep. 2005, doi: 10.1109/TDSC.2005.35.

[25] R. Thomas, H. Zhu, T. Huck, and T. Johnson, "NetBouncer: Client-Legitimacy-based High-Performance DDoS Filtering," in *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2003*, Washington, DC, USA: Institute of Electrical and Electronics Engineers Inc., Apr. 2003, pp. 1–12. doi: 10.1109/DISCEX.2003.1194939.

[26] G. A. Lucky, "Distributed Network Monitoring for Distributed Denial of Service Attacks Detection and Prevention," *Ph.D. dissertation, Electronics and Computer Science Dept., University of Liverpool, Liverpool*, pp. 1–141, Nov. 2021, doi: 10.17638/03143256.

[27] A. Banitalebi Dehkordi, M. R. Soltanaghaei, and F. Z. Boroujeni, "The DDoS Attacks Detection through Machine Learning and Statistical Methods in SDN," *Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, Mar. 2021, doi: 10.1007/S11227-020-03323-W/METRICS.

[28] Z. Liu, C. Hu, and C. Shan, "Riemannian Manifold on Stream Data: Fourier Transform and Entropy-based DDoS Attacks Detection Method," *Comput Secur*, vol. 109, no. 102392, pp. 1–15, Oct. 2021, doi: 10.1016/J.COSE.2021.102392.

[29] S. M. Mousavi and M. St-Hilaire, "Early Detection of DDoS Attacks against SDN Controllers," in *2015 International Conference on Computing, Networking and Communications, ICNC*, Garden Grove, CA, USA: IEEE, Mar. 2015, pp. 77–81. doi: 10.1109/ICCNC.2015.7069319.

[30] C.-L. Chen and J.-M. Chen, "Use of MARKOV Chain for Early Detecting DDoS Attacks," *International Journal of Network Security & its Applications (IJNSA)*, vol. 13, no. 4, pp. 1–11, 2021, doi: 10.5121/ijnsa.2021.13401.

[31] R. Li and B. Wu, “Early Detection of DDoS based on φ-Entropy in SDN Networks,” in *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, Chongqing, China: IEEE, May 2020, pp. 731–735. doi: 10.1109/ITNEC48623.2020.9084885.

[32] S. Salim and L. Oughdir, “CNN-LSTM Based Approach for Dos Attacks Detection in Wireless Sensor Networks,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, pp. 835–842, 2022, doi: 10.14569/IJACSA.2022.0130497.

[33] S. Mishra and P. S. Chatterjee, “A systematic survey on DDoS Attack and Data Confidentiality Issue on Cloud Servers,” in *19th OITS International Conference on Information Technology (OCIT)*, Bhubaneswar, India: IEEE, Mar. 2022, pp. 273–278. doi: 10.1109/OCIT53463.2021.00062.

[34] Koay and Abigail, “Detecting High and Low Intensity Distributed Denial of Service (DDoS) Attacks,” *Ph.D. dissertation, Networking and Communications Dept., Victoria University of Wellington, Wellington, New Zealand*, pp. 1–168, 2019, Accessed: Sep. 10, 2022. [Online]. Available: http://researcharchive.vuw.ac.nz/handle/10063/8069

[35] R. P. Lippmann *et al.*, “Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation,” in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*, Hilton Head, SC, USA: IEEE, Aug. 2000. doi: 10.1109/DISCEX.2000.821506.

[36] “MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation.” Accessed: Mar. 17, 2023. [Online]. Available: https://archive.ll.mit.edu/ideval/data/2000/LLS_DDOS_2.0.2.html

[37] N. S. Vishnu, R. Singh Batth, and G. Singh, “Denial of Service: Types, Techniques, Defence Mechanisms and Safe Guards,” in *Proceedings of 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* , Dubai, United Arab Emirates: IEEE, Feb. 2019, pp. 695–700. doi: 10.1109/ICCIKE47802.2019.9004388.

[38] A. Verma, M. Arif, and Mohd. S. Husain, “Analysis of DDoS Attack Detection and Prevention in Cloud Environment: A Review,” *International Journal of Advanced Research in Computer Science*, vol. 9, no. 0, pp. 107–113, Jul. 2018, doi: 10.26483/IJARCS.V9I0.6147.

[39] P. V. Sontakke and N. B. Chopade, “Impact and Analysis of Denial-of-Service Attack on an Autonomous Vehicle Test Bed Setup,” in *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems* , Springer, Singapore, Mar. 2022, pp. 221–236. doi: https://doi.org/10.1007/978-981-16-7330-6_17.

[40] P. Veeraraghavan, D. Hanna, and E. Pardede, "NAT++: An Efficient Micro-Nat Architecture for Solving IP-Spoofing Attacks in a Corporate Network," *Electronics (Switzerland)*, vol. 9, no. 1510, pp. 1–21, Sep. 2020, doi: 10.3390/electronics9091510.

[41] H. Wang, H. He, W. Zhang, W. Liu, P. Liu, and A. Javadpour, "Using Honeypots to Model Botnet Attacks on the Internet of Medical Things," *Computers and Electrical Engineering*, vol. 102, no. c, Sep. 2022, doi: https://doi.org/10.1016/j.compeleceng.2022.108212.

[42] V. A. Memos and K. E. Psannis, "NFV-Based Scheme for Effective Protection against Bot Attacks in AI-Enabled IoT," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 91–95, May 2022, doi: 10.1109/IOTM.001.2100175.

[43] M. A. Alarqan, Z. F. Zaaba, and A. Almomani, "Detection Mechanisms of DDoS Attack in Cloud Computing Environment: A Survey," in *(eds) Advances in Cyber Security. ACeS 2019. Communications in Computer and Information Science*, Singapore: Springer, Jan. 2020, pp. 138–152. doi: https://doi.org/10.1007/978-981-15-2693-0_10.

[44] I. Winkler and A. T. Gomes, "Adversary Infrastructure," in *Advanced Persistent Security: A Cyberwarfare Approach to Implementing Adaptive Enterprise Protection, Detection, and Reaction Strategies*, Cambridge: Syngress, 2016, ch. Chapter 7, pp. 67–79. doi: 10.1016/B978-0-12-809316-0.00007-5.

[45] M. M. Alani, "BotStop : Packet-based Efficient and Explainable IoT Botnet Detection using Machine Learning," *Comput Commun*, vol. 193, pp. 53–62, Sep. 2022, doi: 10.1016/J.COMCOM.2022.06.039.

[46] W. N. H. Ibrahim *et al.*, "Multilayer Framework for Botnet Detection Using Machine Learning Algorithms," *IEEE Access*, vol. 9, pp. 48753–48768, 2021, doi: 10.1109/ACCESS.2021.3060778.

[47] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo, "Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation," *IEEE Trans Dependable Secure Comput*, pp. 1–15, Jun. 2022, doi: 10.1109/TDSC.2022.3183361.

[48] C. Chapman, "Introduction to practical security and performance testing," in *Network Performance and Security*, Syngress, 2016, ch. Chapter 1, pp. 1–14. doi: https://doi.org/10.1016/B978-0-12-803584-9.00001-9.

[49] C. Timm and R. Perez, "Social Networking Infrastructure Attacks," in *Seven Deadliest Social Network Attacks*, Syngress, 2010, ch. Chapter 1, pp. 1–22. doi: 10.1016/B978-1-59749-545-5.00001-X.

[50] M. Anagnostopoulos, S. Lagos, and G. Kambourakis, “Large-Scale Empirical Evaluation of DNS and SSDP Amplification attacks,” *Journal of Information Security and Applications*, vol. 66, pp. 1–17, May 2022, doi: 10.1016/j.jisa.2022.103168.

[51] A. Prasad and S. Chandra, “VMFCVD: An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning,” *Arab J Sci Eng*, vol. 47, no. 8, pp. 9965–9983, Aug. 2022, doi: 10.1007/S13369-021-06484-9/TABLES/12.

[52] Y. Cui *et al.*, “Towards DDoS Detection Mechanisms in Software-Defined Networking,” *Journal of Network and Computer Applications*, vol. 190, pp. 1–37, Sep. 2021, doi: 10.1016/J.JNCA.2021.103156.

[53] B. Bouyeddou, F. Harrou, Y. Sun, and B. Kadri, “Detection of Smurf Flooding Attacks using Kullback-Leibler-based Scheme,” in *2018 4th International Conference on Computer and Technology Applications (ICCTA)*, Istanbul, Turkey: IEEE, Jun. 2018, pp. 11–15. doi: 10.1109/CATA.2018.8398647.

[54] S. Wong, Y. Shen, S. My, and M. A. Al-Shareeda, “Review of Advanced Monitoring Mechanisms in Peer-to-Peer (P2P) Botnets,” in *8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, Mosul, Iraq: IEEE, Feb. 2022, pp. 312–317. doi: 10.48550/arxiv.2207.12936.

[55] S. Hosseini, A. E. Nezhad, and H. Seilani, “Botnet Detection using Negative Selection Algorithm, Convolution Neural Network and Classification Methods,” *Evolving Systems*, vol. 13, no. 1, pp. 101–115, Jan. 2021, doi: 10.1007/S12530-020-09362-1.

[56] G. Shanmugavadivel, S. Gopinath, K. Gowtham, and M. Mugesh, “Detecting Communication Network Anomalies and Intrusions,” in *Proceedings - 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India: IEEE, Jun. 2022, pp. 845–850. doi: 10.1109/ICICCS53718.2022.9788371.

[57] A. A. Alashhab *et al.*, “A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks,” *Symmetry (Basel)*, vol. 14, no. 8, pp. 1–30, Jul. 2022, doi: 10.3390/SYM14081563.

[58] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, “Designing and Modeling the Slow Next DoS Attack,” in *(eds) International Joint Conference (CISIS). Advances in Intelligent Systems and Computing, vol 369*, Burgos, Spain: Springer Verlag, Jan. 2015, pp. 249–259. doi: https://doi.org/10.1007/978-3-319-19713-5_22.

[59] V. Savchenko *et al.*, "Detection of Slow DDoS Attacks Based on Time Delay Forecasting," in *Proceedings of the 3rd International Conference on Information Security and Information Technologies (ISecIT 2021) co-located with 1st International Forum "Digital Reality" (DRForum)*, Odesa, Ukraine: CEUR Workshop Proceedings, 2021, pp. 38–46. Accessed: Sep. 06, 2022. [Online]. Available: http://ceur-ws.org/Vol-3200/paper6.pdf

[60] S. Kukreti, S. K. Modgil, N. Gehlot, and V. Kumar, "DDoS Attack using SYN Flooding: A Case Study," in *9th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India: IEEE, May 2022, pp. 323–329. doi: 10.23919/INDIACOM54597.2022.9763108.

[61] K. S. Kumavat and J. Gomes, "Survey of Detection Techniques for DDoS Attacks," in *3rd International Conference on Intelligent Engineering and Management (ICIEM)*, London, United Kingdom: IEEE, Aug. 2022, pp. 657–663. doi: 10.1109/ICIEM54221.2022.9853064.

[62] E. Dai, D. Huang, and L. Zhang, "Low-Rate Denial-of-Service Attack Detection: Defense Strategy based on Spectral Estimation for CV-QKD," *Photonics 2022, Vol. 9, Page 365*, vol. 9, no. 6, pp. 1–14, May 2022, doi: 10.3390/PHOTONICS9060365.

[63] W. Guo, H. Qiu, Z. Liu, J. Zhu, and Q. Wang, "The Evaluation of DDoS Attack Effect based on Neural Network," *Security and Communication Networks*, vol. 2022, pp. 1–16, Apr. 2022, doi: 10.1155/2022/5166323.

[64] A. Singh and B. B. Gupta, "Distributed Denial-of-Service (DDoS) Attacks and Defense Mechanisms in Various Web-Enabled Computing Platforms: Issues, Challenges, and Future Research Directions," in *International Journal on Semantic Web & Information Systems*, PA, United States: IGI Global, Apr. 2022, pp. 1–43. doi: 10.4018/IJSWIS.297143.

[65] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the Development of a Realistic Multidimensional IoT Profiling Dataset," in *19th Annual International Conference on Privacy, Security & Trust (PST)*, Fredericton, NB, Canada: IEEE, Aug. 2022, pp. 1–11. doi: 10.1109/PST55820.2022.9851966.

[66] D. R. Sandkühler, "Exploring the Methods and Goals of Students who DDoS Educational Facilities," *Essay (Bachelor), University of Twente, Enschede*, pp. 1–12, 2020, Accessed: Sep. 12, 2022. [Online]. Available: https://purl.utwente.nl/essays/81847

[67] C. Kemp, C. Calvert, and T. M. Khoshgoftaar, "Detecting Slow Application-Layer DoS Attacks with PCA," in *IEEE 22nd International Conference on Information Reuse and Integration for Data Science, IRI 2021*, Las Vegas, NV, USA: IEEE, Aug. 2021, pp. 176–183. doi: https://doi.org/10.1109/IRI51335.2021.00030.

[68] K. Hajdarevic, C. Pattinson, and I. Besic, "Improving Learning Skills in Detection of Denial of Service Attacks with Newcombe - Benford's Law using Interactive Data Extraction and Analysis," *TEM Journal*, vol. 11, no. 2, pp. 527–534, May 2022, doi: 10.18421/TEM112-05.

[69] A. Praseed and P. Santhi Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 661–685, Sep. 2018, doi: 10.1109/COMST.2018.2870658.

[70] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, Apr. 2004, doi: 10.1145/997150.997156.

[71] A. Asosheh and N. Ramezani, "A Comprehensive Taxonomy of DDOS Attacks and Defense Mechanism Applying in a Smart Classification," *WSEAS Transactions on Computers*, vol. 7, no. 4, pp. 281–290, Apr. 2008, doi: 10.5555/1457949.1457968.

[72] A. Bhardwaj, G. V. B. Subrahmanyam, V. Avasthi, H. Sastry, and S. Goundar, "DDoS Attacks, New DDoS Taxonomy and Mitigation Solutions - A survey," in *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, Paralakhemundi, India: IEEE, Jun. 2017, pp. 793–798. doi: 10.1109/SCOPES.2016.7955549.

[73] M. Masdari and M. Jalali, "A Survey and Taxonomy of DoS Attacks in Cloud Computing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3724–3751, Nov. 2016, doi: 10.1002/SEC.1539.

[74] B. S. Kiruthika Devi, V. J. Saglani, A. V. Gupta, and T. Subbulakshmi, "Classifying and Predicting DoS and DDoS Attacks on Cloud Services," in *Proceedings of the 2nd International Conference on Trends in Electronics and Informatics, ICOEI 2018*, Tirunelveli, India: IEEE, Dec. 2018, pp. 1–5. doi: 10.1109/ICOEI.2018.8553889.

[75] A. Ashok Acharya, A. K.M, and S. B. Kumar, "An Intrusion Detection System Against UDP Flood Attack and Ping of Death Attack (DDOS) in MANET," *Int J Eng Technol*, vol. 8, no. 2, pp. 1112–1115, 2016, Accessed: Aug. 19, 2023. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.enggjournals.com/ijet/docs/IJET16-08-02-126.pdf

[76] N. Gupta, A. Jain, P. Saini, and V. Gupta, "DDoS Attack Algorithm using ICMP Flood," in *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India: IEEE, Oct. 2016. Accessed: Sep. 13, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/7725026

[77] A. Saied, R. E. Overill, and T. Radzik, "Detection of Known and Unknown DDoS Attacks using Artificial Neural Networks," *Neurocomputing*, vol. 172, no. C, pp. 385–393, Jan. 2016, doi: 10.1016/J.NEUCOM.2015.04.101.

[78] T. V Phan, N. Khac Bao, and M. Park, "A Novel Hybrid Flow-based Handler with DDoS Attacks in Software-Defined Networking," in *Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Toulouse, France: IEEE, Jan. 2016. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069.

[79] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlowSIA: An Optimized Protection Scheme for Software-Defined Networks from Flooding Attacks," in *IEEE 6th International Conference on Communications and Electronics, IEEE ICCE 2016*, Ha-Long, Vietnam: IEEE, Sep. 2016, pp. 13–18. doi: 10.1109/CCE.2016.7562606.

[80] A. Zand, G. Modelo-Howard, A. Tongaonkar, S. J. Lee, C. Kruegel, and G. Vigna, "Demystifying DDoS as a Service," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 14–21, Jul. 2017, doi: 10.1109/MCOM.2017.1600980.

[81] T. Kawamura, M. Fukushi, Y. Hirano, Y. Fujita, and Y. Hamamoto, "An NTP-based Detection Module for DDoS Attacks on IoT," in *IEEE International Conference on Consumer Electronics (ICCE-TW)*, Taipei, Taiwan: IEEE, Jul. 2017, pp. 15–16. doi: 10.1109/ICCE-CHINA.2017.7990972.

[82] Y. Afek, A. Bremler-Barr, E. Cohen, S. L. Feibish, and M. Shagam, "Efficient Distinct Heavy Hitters for DNS DDoS Attack Detection," *ArXiv*, pp. 1–9, Dec. 2016, doi: 10.48550/arxiv.1612.02636.

[83] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arabian Journal for Science and Engineering 2017 42:2*, vol. 42, no. 2, pp. 425–441, Feb. 2017, doi: 10.1007/S13369-017-2414-5.

[84] B. Nagy, P. Orosz, and P. Varga, "Low-Reaction Time FPGA-based DDoS Detector," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan: IEEE, Jul. 2018, pp. 1–2. doi: 10.1109/NOMS.2018.8406124.

[85] I. Ko, D. Chambers, and E. Barrett, "A lightweight DDoS Attack Mitigation System within the ISP Domain Utilising Self-Organizing Map," in *(eds) Proceedings of the Future Technologies Conference (FTC) 2018. FTC 2018. Advances in Intelligent Systems and Computing, vol 881*, Vancouver, Canada: Springer Verlag, Nov. 2018, pp. 173–188. doi: https://doi.org/10.1007/978-3-030-02683-7_14.

[86] D. Shah and V. Kumar, "TCP SYN Cookie Vulnerability," *arXiv*, pp. 1–3, May 2018, doi: 10.48550/arxiv.1807.08026.

[87] R. Mohammadi, R. Javidan, and M. Conti, "SLICOTS: An SDN-based Lightweight Countermeasure for TCP SYN Flooding Attacks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, Jun. 2017, doi: 10.1109/TNSM.2017.2701549.

[88] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, "A Multi-Level DDoS Mitigation Framework for the Industrial Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 30–36, Feb. 2018, doi: 10.1109/MCOM.2018.1700621.

[89] M. Z. Rafique, M. A. Akbar, and M. Farooq, "Evaluating DoS Attacks against SIP-based VoIP Systems," in *GLOBECOM - IEEE Global Telecommunications Conference*, Honolulu, HI, USA: IEEE, Mar. 2010. doi: 10.1109/GLOCOM.2009.5426247.

[90] N. Muraleedharan and B. Janet, "A Deep Learning based HTTP Slow DoS Classification Approach using Flow Data," *ICT Express*, vol. 7, no. 2, pp. 210–214, Jun. 2021, doi: 10.1016/J.ICTE.2020.08.005.

[91] S. Kumar, "Security Enhancement in Mobile Ad-Hoc Network Using Novel Data Integrity Based Hash Protection Process," *Wirel Pers Commun*, vol. 123, no. 2, pp. 1059–1083, Sep. 2021, doi: 10.1007/S11277-021-09170-Z.

[92] T. Dai, H. Shulman, and M. Waidner, "Poster: Fragmentation Attacks on DNS over TCP," in *Proceedings - International Conference on Distributed Computing Systems*, DC, USA: IEEE, Oct. 2021, pp. 1124–1125. doi: 10.1109/ICDCS51616.2021.00118.

[93] D. Wang *et al.*, "Analysis of CAN Bus Encryption and Decryption Performance of Different Chips," in *Journal of Physics: Conference Series, 2021 International Conference on Computer, Remote Sensing and Aerospace (CRSA)*, Tokyo, Japan: IOP Publishing, Jul. 2021, pp. 1–10. doi: 10.1088/1742-6596/2006/1/012071.

[94] R. Sharma, C. A. Chan, and C. Leckie, "Evaluation of Centralised vs Distributed Collaborative Intrusion Detection Systems in Multi-Access Edge Computing," in *IFIP Networking Conference (Networking)*, Paris, France: IEEE, Jul. 2020. Accessed: Sep. 17, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9142719

[95] S. Agrawal *et al.*, "Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions," *ArXiv*, pp. 1–19, Jun. 2021, doi: 10.48550/arxiv.2106.09527.

[96] A. Thakkar and R. Lohiya, "A Survey on Intrusion Detection System: Feature Selection, Model, Performance Measures, Application Perspective, Challenges, and Future Research Directions," *Artif Intell Rev*, vol. 55, no. 1, pp. 453–563, Jul. 2021, doi: 10.1007/S10462-021-10037-9.

[97] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A Novel Hierarchical Intrusion Detection System based on Decision Tree and Rules-based Models," in *Proceedings - 15th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2019*, Santorini, Greece: IEEE, Aug. 2019, pp. 228–233. doi: 10.1109/DCOSS.2019.00059.

[98] M. Sarnovsky and J. Paralic, "Hierarchical Intrusion Detection Using Machine Learning and Knowledge Model," *Symmetry 2020, Vol. 12, Page 203*, vol. 12, no. 2, p. 203, Feb. 2020, doi: 10.3390/SYM12020203.

[99] M. Kumar and A. K. Singh, "Distributed Intrusion Detection System using Blockchain and Cloud Computing Infrastructure," in *Proceedings of the 4th International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India: IEEE, Jul. 2020, pp. 248–252. doi: 10.1109/ICOEI48184.2020.9142954.

[100] M. Abdel-Basset, A. Gamal, K. M. Sallam, I. Elgendi, K. Munasinghe, and A. Jamalipour, "An Optimization Model for Appraising Intrusion-Detection Systems for Network Security Communications: Applications, Challenges, and Solutions," *Sensors 2022, Vol. 22, Page 4123*, vol. 22, no. 11, p. 4123, May 2022, doi: 10.3390/S22114123.

[101] S. W. Lee *et al.*, "Towards Secure Intrusion Detection Systems using Deep Learning Techniques: Comprehensive Analysis and Review," *Journal of Network and Computer Applications*, vol. 187, pp. 1–22, Aug. 2021, doi: 10.1016/J.JNCA.2021.103111.

[102] P. Deshpande, S. C. Sharma, S. K. Peddoju, and S. Junaid, "HIDS: A Host based Intrusion Detection System for Cloud Computing Environment," *International Journal of System Assurance Engineering and Management 2014 9:3*, vol. 9, no. 3, pp. 567–576, Jun. 2014, doi: 10.1007/S13198-014-0277-7.

[103] Z. Wu, J. Wang, L. Hu, Z. Zhang, and H. Wu, "A Network Intrusion Detection Method based on Semantic Re-encoding and Deep Learning," *Journal of Network and Computer Applications*, vol. 164, pp. 1–12, Aug. 2020, doi: 10.1016/J.JNCA.2020.102688.

[104] B. Subba, "A Neural Network based NIDS Framework for Intrusion Detection in Contemporary Network Traffic," in *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, Goa, India: IEEE Computer Society, Jun. 2019. doi: 10.1109/ANTS47819.2019.9117966.

[105] K. Rahul-Vigneswaran, P. Poornachandran, and K. Soman, "A Compendium on Network and Host Based Intrusion Detection Systems," *Lecture Notes in Electrical Engineering*, vol. 601, pp. 23–30, 2020, doi: 10.1007/978-981-15-1420-3_3/COVER.

[106] "nProbe – My Site." Accessed: Jul. 30, 2024. [Online]. Available: https://www.ntop.org/products/NetFlow/nprobe/

[107] “GitHub - irino/softflowd: softflowd: A flow-based network traffic analyser capable of Cisco NetFlow data export software.” Accessed: Jul. 30, 2024. [Online]. Available: https://github.com/irino/softflowd

[108] “YAF.” Accessed: Jul. 30, 2024. [Online]. Available: https://tools.netsa.cert.org/yaf/

[109] D. Alexandra Pires Pinto, S. Doutora Eva Maia, I. Auxiliar, C. Doutora Ivone Amorim, and P. Auxiliar Convidado, “HERA: Enhancing Network Security with a New Dataset Creation Tool,” Jul. 2024, Accessed: Jul. 30, 2024. [Online]. Available: https://repositorio-aberto.up.pt/handle/10216/159426

[110] M. Sheikhan, M. Bejani, and D. Gharavian, “Modular Neural-SVM Scheme for Speech Emotion Recognition using ANOVA Feature Selection Method,” *Neural Comput Appl*, vol. 23, no. 1, pp. 215–227, Jul. 2013, doi: 10.1007/S00521-012-0814-8/METRICS.

[111] H. Ding, P. M. Feng, W. Chen, and H. Lin, “Identification of Bacteriophage Virion Proteins by the ANOVA Feature Selection and Analysis,” *Mol Biosyst*, vol. 10, no. 8, pp. 2229–2235, 2014, doi: 10.1039/C4MB00316K.

[112] V. P. Rekkas, S. Sotiroudis, P. Sarigiannidis, S. Wan, G. K. Karagiannidis, and S. K. Goudos, “Machine Learning in Beyond 5G/6G Networks—State-of-the-Art and Future Trends,” *Electronics (Basel)*, vol. 10, no. 22, pp. 1–28, Nov. 2021, doi: 10.3390/ELECTRONICS10222786.

[113] T. T. Nguyen, J. Z. Huang, and T. T. Nguyen, “Unbiased Feature Selection in Learning Random Forests for High-Dimensional Data,” *Scientific World Journal*, vol. 2015, no. 471371, pp. 1–19, Mar. 2015, doi: 10.1155/2015/471371.

[114] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely Randomized Trees,” *Mach Learn*, vol. 63, no. 1, pp. 3–42, Apr. 2006, doi: 10.1007/S10994-006-6226-1/METRICS.

[115] S. Jany Shabu *et al.*, “Research on Intrusion Detection Method Based on Pearson Correlation Coefficient Feature Selection Algorithm,” in *Journal of Physics: Conference Series, Volume 1757, International Conference on Computer Big Data and Artificial Intelligence (ICCBDAI)*, Changsha, China: IOP Publishing, Jan. 2021, pp. 1–10. doi: 10.1088/1742-6596/1757/1/012054.

[116] I. Bolodurina, A. Shukhman, D. Parfenov, A. Zhigalov, and L. Zabrodina, “Investigation of the Problem of Classifying Unbalanced Datasets in Identifying Distributed Denial of Service Attacks,” *J Phys Conf Ser*, vol. 1679, no. 4, Nov. 2020, doi: 10.1088/1742-6596/1679/4/042020.

[117] X. Tan *et al.*, “Wireless Sensor Networks Intrusion Detection Based on SMOTE and the Random Forest Algorithm,” *Sensors*, vol. 19, no. 1, pp. 1–15, Jan. 2019, doi: 10.3390/S19010203.

[118] S. Einy, C. Oz, and Y. D. Navaei, “The Anomaly- and Signature-based IDS for Network Security using Hybrid Inference Systems,” *Math Probl Eng*, vol. 2021, no. 6639714, pp. 1–10, 2021, doi: 10.1155/2021/6639714.

[119] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, “Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine,” *Electronics (Basel)*, vol. 9, no. 1, pp. 1–18, Jan. 2020, doi: 10.3390/ELECTRONICS9010173.

[120] Jyoti Snehi, Abhinav Bhandari, V. Baggan, and R. Manish Snehi, “Diverse Methods for Signature based Intrusion Detection Schemes Adopted,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 2, pp. 44–49, Jul. 2020, doi: 10.35940/IJRTE.A2791.079220.

[121] S. Singh, “A Hybrid Intrusion Detection System Design for Computer Network Security,” *International Journal of Engineering Science & Research Technology*, vol. 7, no. 4, pp. 339–343, 2018, doi: 10.5281/zenodo.1218603.

[122] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, “Intrusion Detection in Industrial Internet of Things Network-Based on Deep Learning Model with Rule-Based Feature Selection,” *Wirel Commun Mob Comput*, vol. 2021, no. 7154587, pp. 1–17, 2021, doi: 10.1155/2021/7154587.

[123] S. K. Sasidharan and C. Thomas, “ProDroid — An Android Malware Detection Framework based on Profile Hidden Markov Model,” *Pervasive Mob Comput*, vol. 72, no. 101336, pp. 1–16, Apr. 2021, doi: 10.1016/J.PMCJ.2021.101336.

[124] N. Eddermoug, M. Sadik, E. Sabir, A. Mansour, and M. Azmi, “PPSA: Profiling and Preventing Security Attacks in Cloud Computing,” in *15th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Tangier, Morocco: IEEE, Jul. 2019, pp. 415–421. doi: 10.1109/IWCMC.2019.8766621.

[125] B. B. Gupta and O. P. Badve, “Taxonomy of DoS and DDoS Attacks and Desirable Defense Mechanism in a Cloud Computing Environment,” *Neural Comput Appl*, vol. 28, no. 12, pp. 3655–3682, Apr. 2016, doi: 10.1007/S00521-016-2317-5.

[126] N. N. Dao, J. Park, M. Park, and S. Cho, “A Feasible Method to Combat against DDoS Attack in SDN network,” in *International Conference on Information Networking*, Cambodia: IEEE Computer Society, Mar. 2015, pp. 309–311. doi: 10.1109/ICOIN.2015.7057902.

[127] A. F. M. Piedrahita, S. Rueda, D. M. F. Mattos, and O. C. M. B. Duarte, “FlowFence: A Denial of Service Defense System for Software Defined Networking,” in *2015 Global Information Infrastructure and Networking Symposium (GIIS)*, Guadalajara, Mexico: IEEE, Dec. 2015, pp. 1–6. doi: 10.1109/GIIS.2015.7347185.

[128] R. Durner, C. Lorenz, M. Wiedemann, and W. Kellerer, "Detecting and Mitigating Denial of Service Attacks against the Data Plane in Software Defined Networks," in *IEEE Conference on Network Softwarization (NetSoft)*, Bologna, Italy: IEEE, Aug. 2017. doi: 10.1109/NETSOFT.2017.8004229.

[129] H. Kousar, M. M. Mulla, P. Shettar, and D. G. Narayan, "DDoS Attack Detection System using Apache Spark," in *International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India: IEEE, Jun. 2021. doi: 10.1109/ICCCI50826.2021.9457012.

[130] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A New Multi Classifier System using Entropy-based Features in DDoS Attack Detection," in *International Conference on Information Networking*, Chiang Mai, Thailand: IEEE Computer Society, Apr. 2018, pp. 162–167. doi: 10.1109/ICOIN.2018.8343104.

[131] X. Ma and Y. Chen, "DDoS Detection Method based on Chaos Analysis of Network Traffic Entropy," *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, Jan. 2014, doi: 10.1109/LCOMM.2013.112613.132275.

[132] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "A Novel Measure for Low-Rate and High-Rate DDoS Attack Detection using Multivariate Data Analysis," in *8th International Conference on Communication Systems and Networks (COMSNETS)*, Bangalore, India: IEEE, Mar. 2016. doi: 10.1109/COMSNETS.2016.7439939.

[133] L. Meng and X. Guo, "A Detection Method for DDoS Attack against SDN Controller," in *Proceedings of the 4th Annual International Conference on Material Engineering and Application (ICMEA)*, Wuhan, China: Atlantis Press, Feb. 2018, pp. 292–296. doi: 10.2991/icmea-17.2018.67.

[134] P. Valizadeh and A. Taghinezhad-Niar, "DDoS Attacks Detection in Multi-Controller Based Software Defined Network," in *8th International Conference on Web Research (ICWR)*, Tehran, Islamic Republic of Iran: IEEE, Jun. 2022, pp. 34–39. doi: 10.1109/ICWR54782.2022.9786246.

[135] M. A. ; Aladaileh *et al.*, "Renyi Joint Entropy-Based Dynamic Threshold Approach to Detect DDoS Attacks against SDN Controller with Various Traffic Rates," *Applied Sciences*, vol. 12, no. 12, pp. 1–17, Jun. 2022, doi: 10.3390/APP12126127.

[136] I. Özçelik and R. R. Brooks, "Cusum - Entropy: An Efficient Method for DDoS Attack Detection," in *4th International Istanbul Smart Grid Congress and Fair (ICSG)*, Istanbul, Turkey: IEEE, Jun. 2016. doi: 10.1109/SGCF.2016.7492429.

[137] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, "Real-time DDoS Attack Detection using FPGA," *Comput Commun*, vol. 110, pp. 48–58, Sep. 2017, doi: 10.1016/J.COMCOM.2017.05.015.

[138] J. Paulo, A. Maranhão, C. Lustosa Da Costa, E. Pignaton De Freitas, E. Javidi, and R. Timóteo De Sousa Júnior, "Error-Robust Distributed Denial of Service Attack Detection Based on an Average Common Feature Extraction Technique," *Sensors*, vol. 20, no. 20, pp. 1–21, Oct. 2020, doi: 10.3390/S20205845.

[139] M. H. Nguyen, Y.-K. Lai, and K.-P. Chang, "An Entropy-based DDoS Attack Detection and Classification with Hierarchical Temporal Memory," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Tokyo, Japan: IEEE, Feb. 2022. Accessed: May 13, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9689220

[140] Z. Long and W. Jinsong, "A Hybrid Method of Entropy and SSAE-SVM based DDoS Detection and Mitigation Mechanism in SDN," *Comput Secur*, vol. 115, no. 102604, pp. 1–13, Jan. 2022, doi: 10.1016/J.COSE.2022.102604.

[141] H. Polat, O. Polat, and A. Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," *Sustainability*, vol. 12, no. 3, pp. 1–16, Feb. 2020, doi: https://doi.org/10.3390/su12031035.

[142] S. Daneshgadeh, T. Kemmerich, T. Ahmed, and N. Baykal, "An Empirical Investigation of DDoS and Flash Event Detection Using Shannon Entropy, KOAD and SVM Combined," in *International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA: IEEE, Apr. 2019, pp. 658–662. doi: 10.1109/ICCNC.2019.8685632.

[143] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS Attack Detection Method Based on SVM in Software Defined Network," *Security and Communication Networks*, vol. 2018, no. 9804061, pp. 1–8, Apr. 2018, doi: 10.1155/2018/9804061.

[144] K. Hofer-Schmitz, U. Kleb, and B. Stojanović, "The Influences of Feature Sets on the Detection of Advanced Persistent Threats," *Electronics (Basel)*, vol. 10, no. 6, pp. 1–22, Mar. 2021, doi: 10.3390/ELECTRONICS10060704.

[145] S. Nandi, S. Phadikar, and K. Majumder, "Detection of DDoS Attack and Classification Using a Hybrid Approach," in *Third ISEA Conference on Security and Privacy (ISEA-ISAP)*, Guwahati, India: IEEE, Apr. 2020, pp. 41–47. doi: 10.1109/ISEA-ISAP49340.2020.234999.

[146] V. Gaur and R. Kumar, "Analysis of Machine Learning Classifiers for Early Detection of DDoS Attacks on IoT Devices," *Arab J Sci Eng*, vol. 47, no. 2, pp. 1353–1374, Feb. 2022, doi: 10.1007/S13369-021-05947-3/METRICS.

[147] T. T. Khoei, G. Aissou, W. C. Hu, and N. Kaabouch, "Ensemble Learning Methods for Anomaly Intrusion Detection System in Smart Grid," in *IEEE International Conference on Electro Information Technology (EIT)*, MI, USA: IEEE Computer Society, Jul. 2021, pp. 129–135. doi: 10.1109/EIT51626.2021.9491891.

[148] Y. Liu, T. Zhi, M. Shen, L. Wang, Y. Li, and M. Wan, "Software-Defined DDoS Detection with Information Entropy Analysis and Optimized Deep Learning," *Future Generation Computer Systems*, vol. 129, pp. 99–114, Apr. 2022, doi: 10.1016/J.FUTURE.2021.11.009.

[149] M. Yuan and G. Yang, "DDoS Attack Intrusion Detection based on Relative Entropy-CNN," in *6th International Conference on Information Science, Computer Technology and Transportation (ISCTT)*, Xishuangbanna, China: VDE, Mar. 2021. Accessed: Jun. 24, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9738866

[150] W. Nazih, Y. Hifny, W. S. Elkilani, H. Dhahri, and T. Abdelkader, "Countering DDoS Attacks in SIP Based VoIP Networks Using Recurrent Neural Networks," *Sensors*, vol. 20, no. 20, pp. 1–15, Oct. 2020, doi: 10.3390/S20205875.

[151] A. Chonka, J. Singh, and W. Zhou, "Chaos Theory based Detection against Network Mimicking DDoS Attacks," *IEEE Communications Letters*, vol. 13, no. 9, pp. 717–719, 2009, doi: 10.1109/LCOMM.2009.090615.

[152] S. K and K. G, "A Clustering Algorithm for Detecting DDoS Attacks in Networks," *International Journal of Recent Engineering Science (IJRES)*, vol. 1, no. 1, pp. 24–30, 2014, [Online]. Available: www.ijresonline.com

[153] S. Bista, R. Chitrakar, S. Bista, and R. Chitrakar, "DDoS Attack Detection Using Heuristics Clustering Algorithm and Naïve Bayes Classification," *Journal of Information Security*, vol. 9, no. 1, pp. 33–44, Nov. 2017, doi: 10.4236/JIS.2018.91004.

[154] Ö. Cepheli, S. Büyükçorak, and G. Karabulut Kurt, "Hybrid Intrusion Detection System for DDoS Attacks," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016, doi: 10.1155/2016/1075648.

[155] J. LI, "Detection of DDoS Attacks based on Dense Neural Networks, Autoencoders and Pearson Correlation Coefficient," *Master Thesis, Computer Science Dept., Dalhousie University, Halifax, Nova Scotia*, pp. 1–81, Apr. 2020, Accessed: May 13, 2023. [Online]. Available: https://dalspace.library.dal.ca/handle/10222/78536

[156] S. Sumathi and N. Karthikeyan, "Detection of Distributed Denial of Service using Deep Learning Neural Network," *J Ambient Intell Humaniz Comput*, vol. 12, pp. 5943–5953, May 2021, doi: https://doi.org/10.1007/s12652-020-02144-2.

[157] V. Gaur and R. Kumar, "HCTDDA: Hybrid Classification Technique for Detection of DDoS Attacks," in *5th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India: IEEE, Feb. 2021. doi: 10.1109/ISCON52037.2021.9702399.

[158] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2018, no. 1, pp. 177–200, Jan. 2018, doi: 10.13052/JSN2445-9739.2017.009.

[159] J. Postel, "Transmission Control Protocol. RFC: 793," Internet Engineering Task Force (IETF). Accessed: May 02, 2023. [Online]. Available: https://www.ietf.org/rfc/rfc793.txt

[160] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study," in *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops (SPW)*, San Francisco, CA, USA: IEEE, Jul. 2021, pp. 7–12. doi: 10.1109/SPW53761.2021.00009.

APPENDICES

Appendix A

A.1 Best Features using Pearson Correlation Heatmap

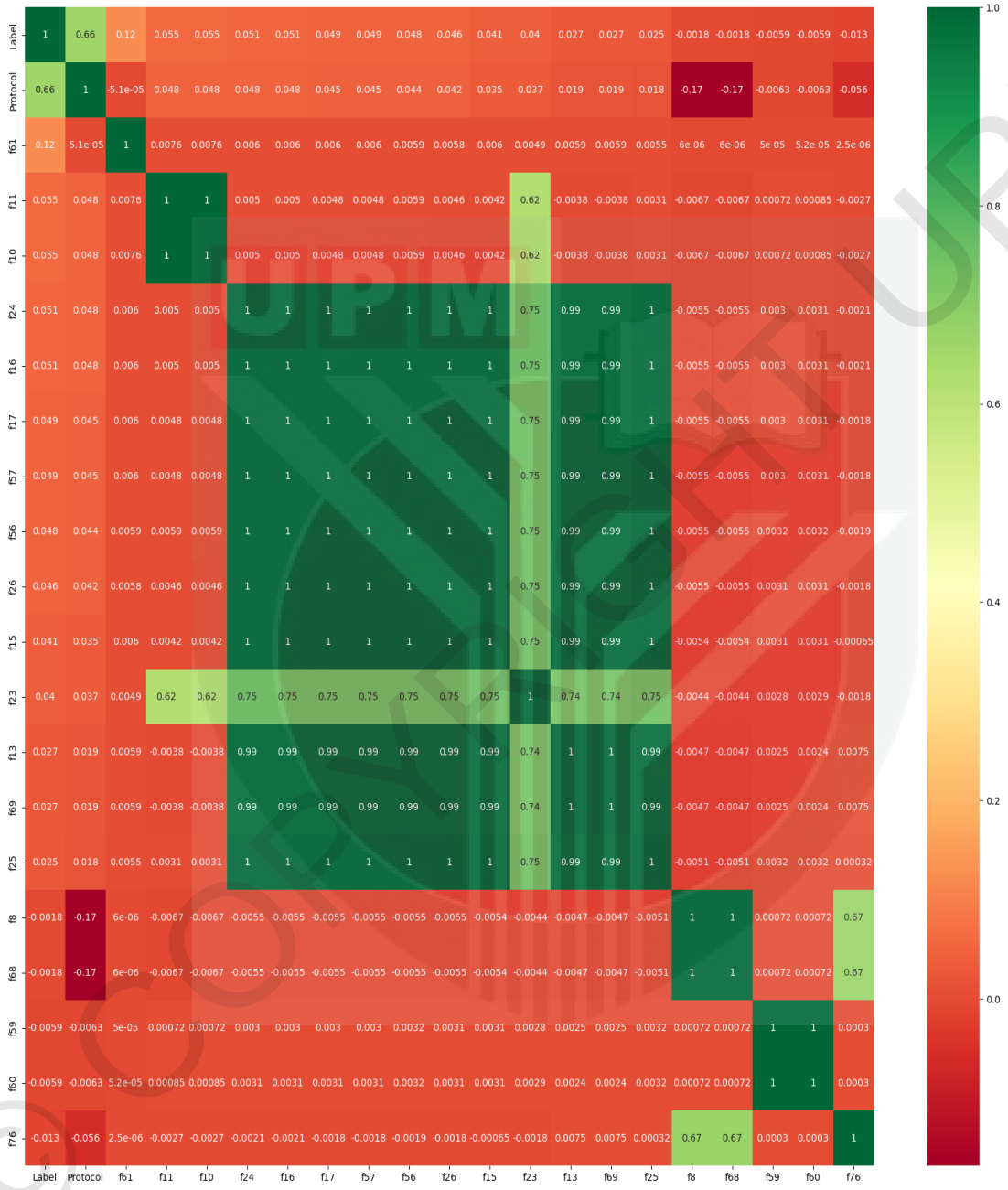


Figure A1 : Best (20) Features of MSSQL Attack for First Day of CICDDoS2019 using Pearson Correlation Heatmap

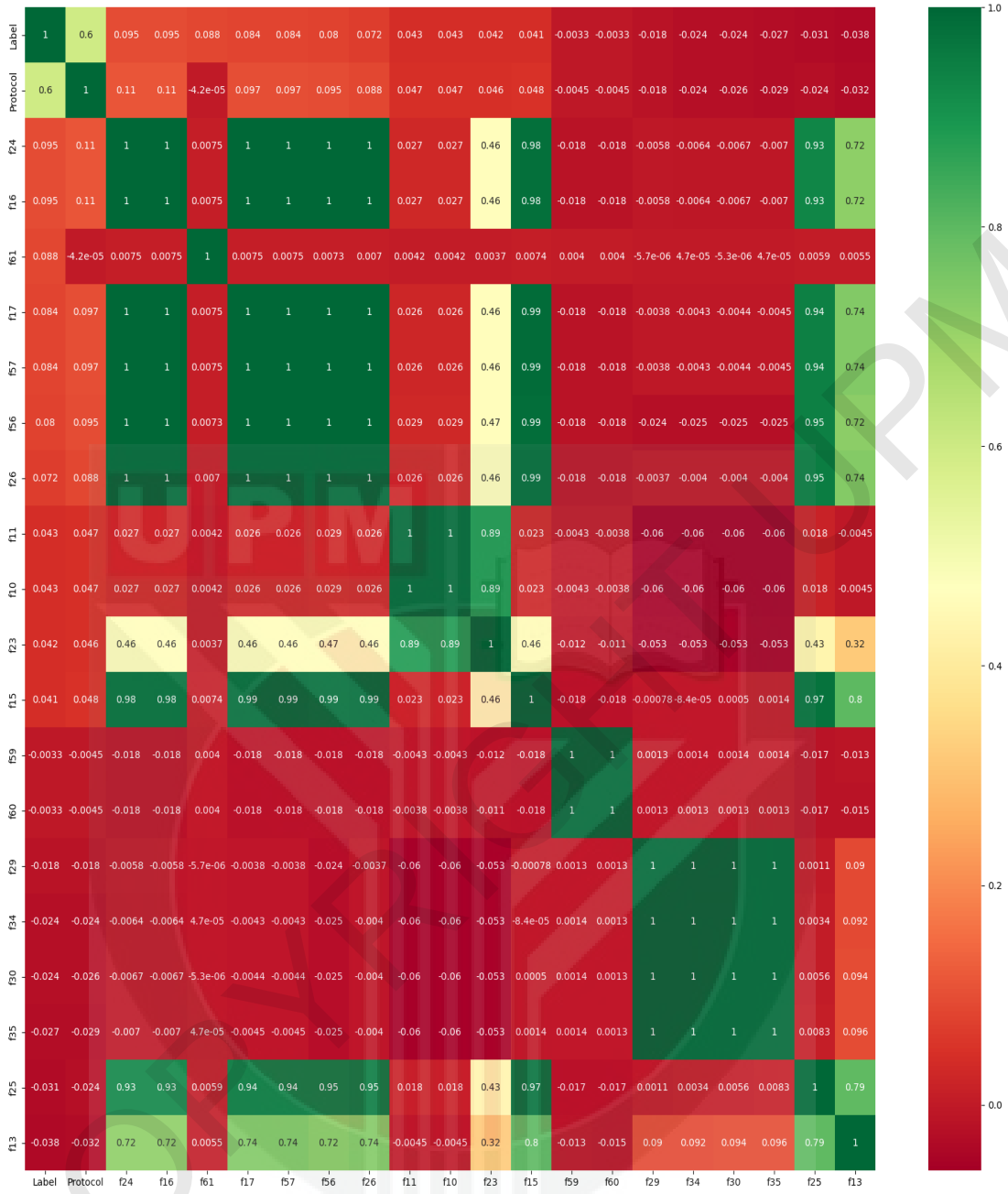


Figure A2 : Best (20) Features of NetBIOS Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

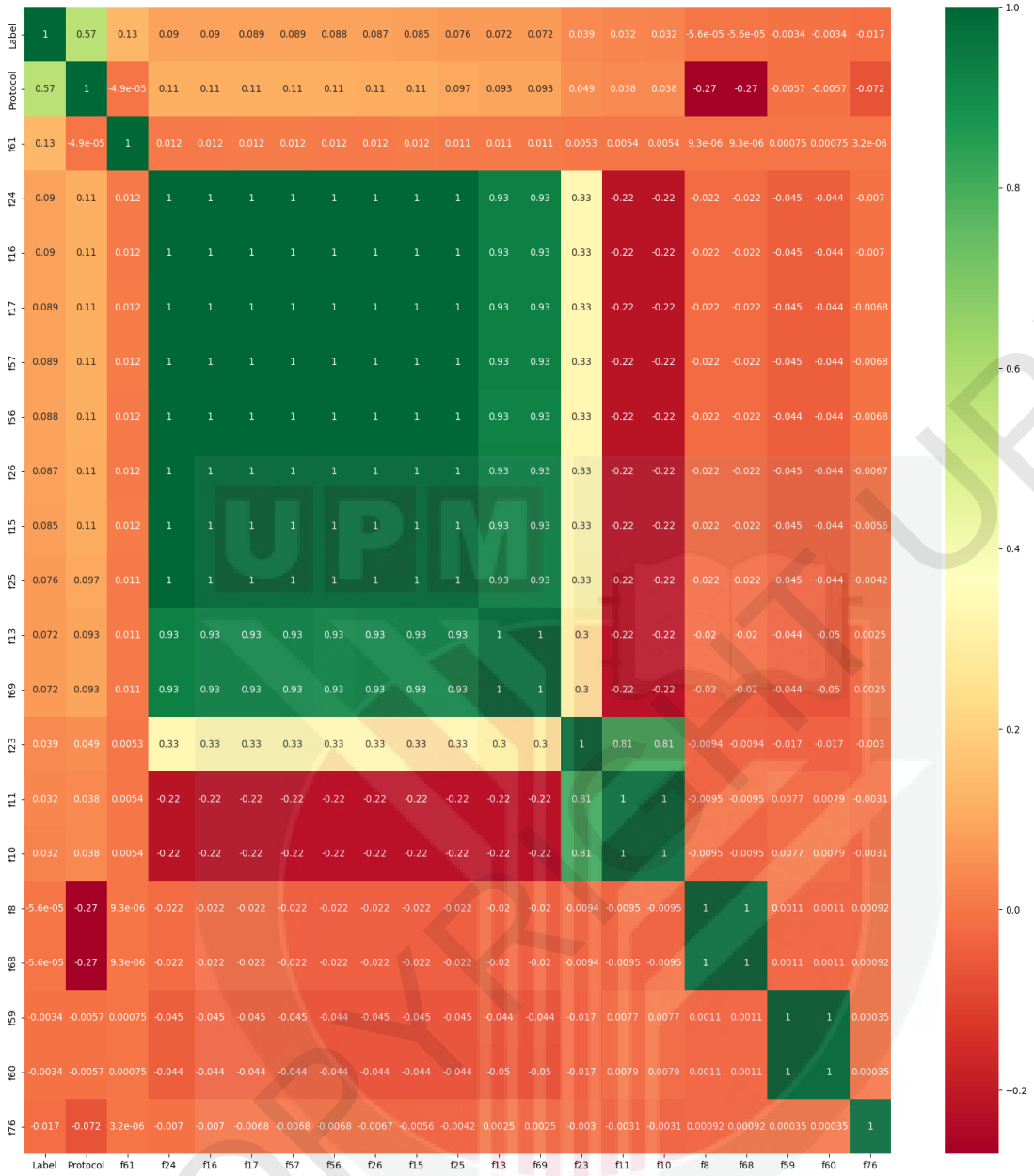


Figure A3 : Best (20) Features of SNMP Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

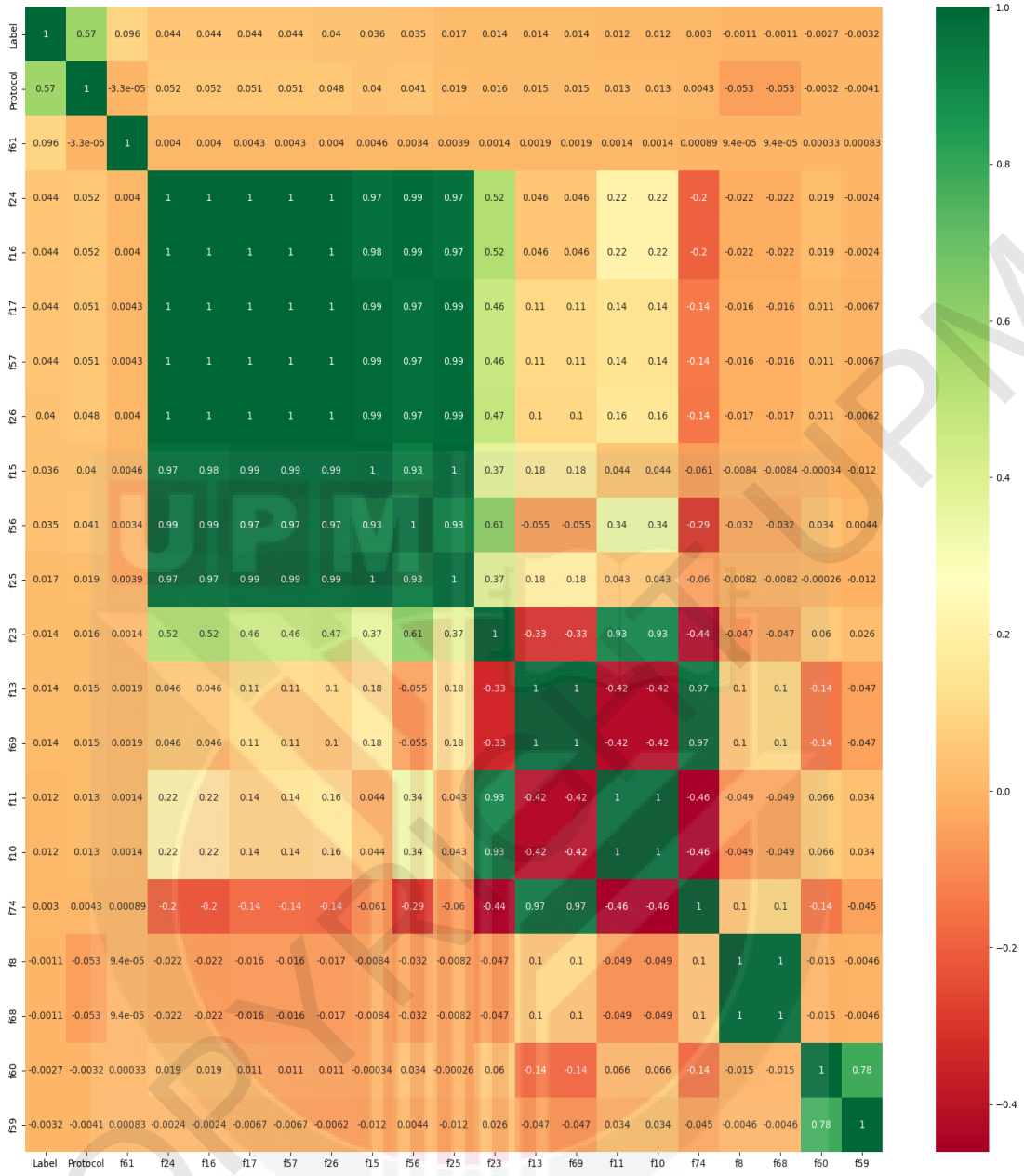


Figure A4 : Best (20) Features of SSDP Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

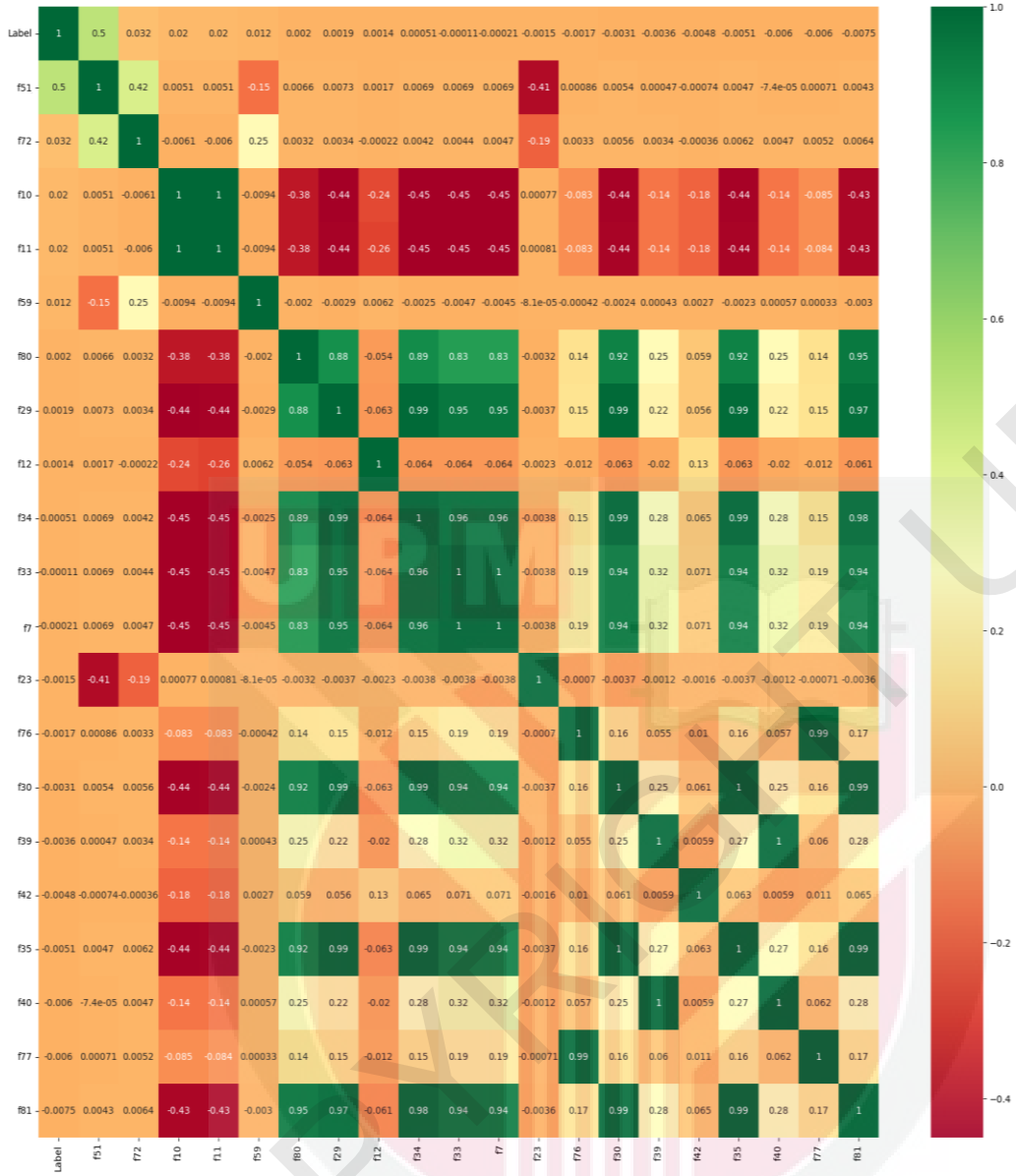


Figure A5 : Best (20) Features of SYN Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

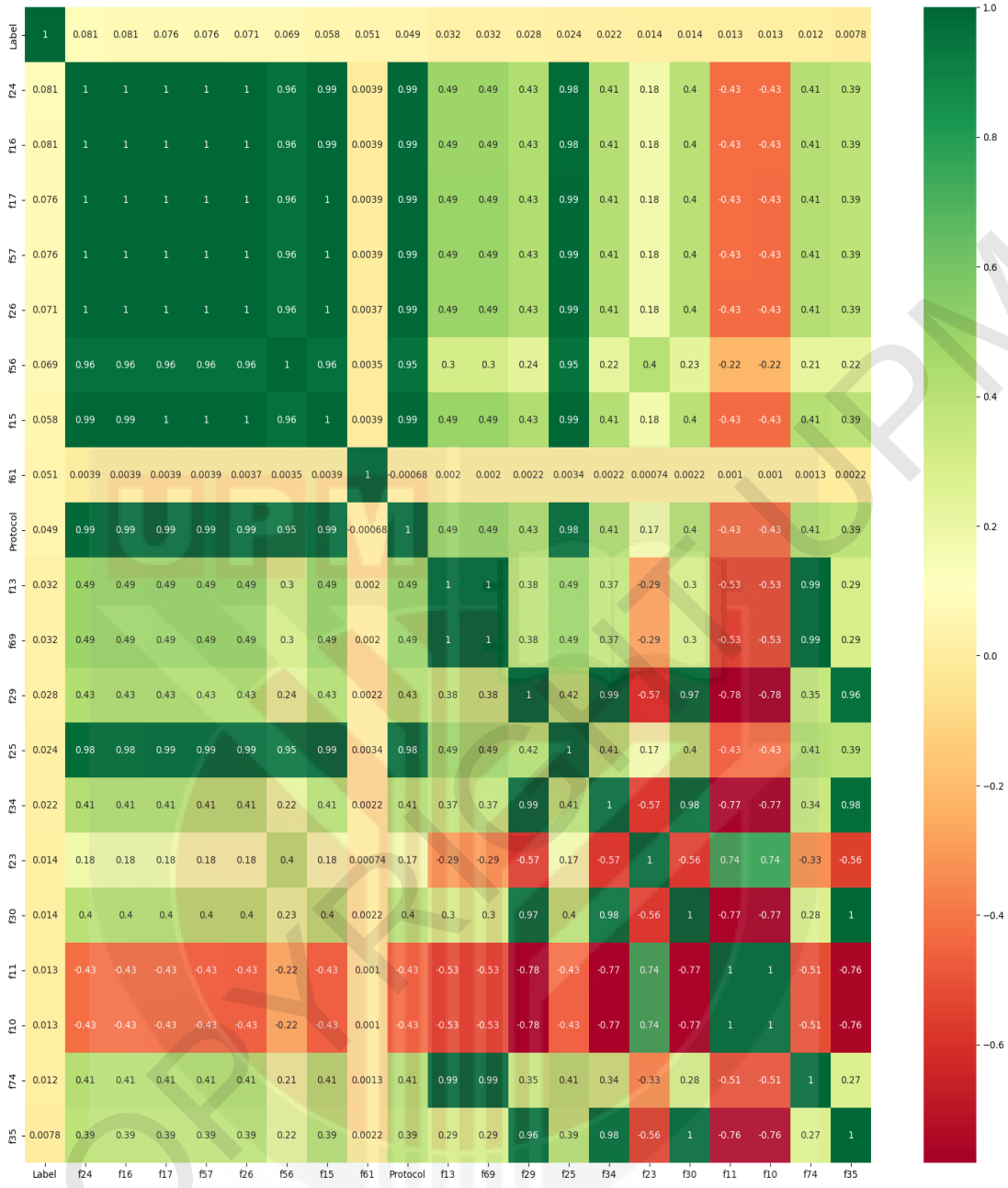


Figure A6 : Best (20) Features of TFTP Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

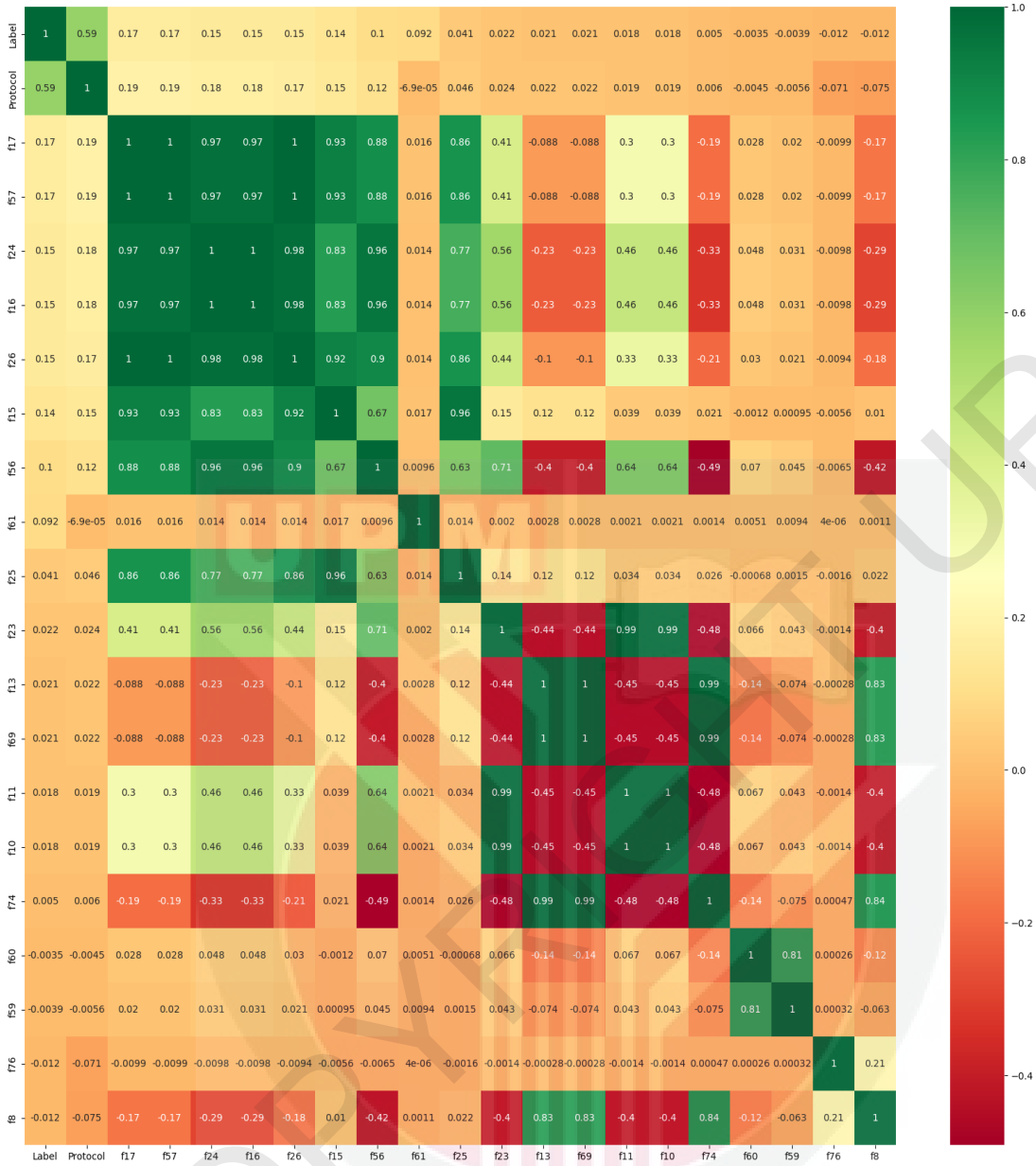


Figure A7 : Best (20) Features of UDP Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

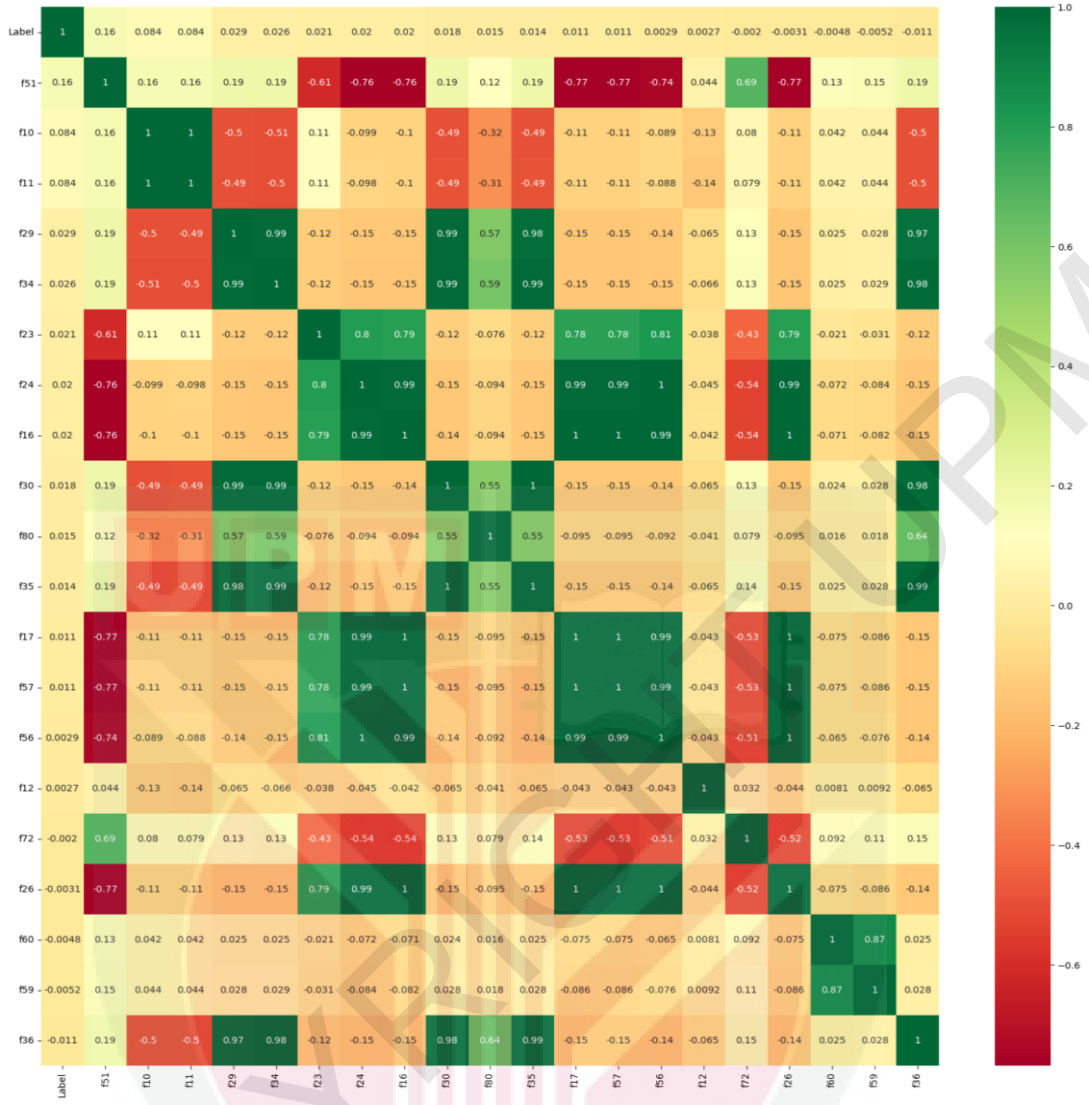


Figure A8 : Best (20) Features of UDP-Lag Attack for First Day of CICDDoS2019 Using Pearson Correlation Heatmap

Appendix B

B.1 Confusion Matrix for Set of Best Features of ESPRT Using CICDDoS2019

Table B1 : Confusion Matrix for Set of Best Features of ESPRT Using Different Attack of CICDDoS2019

Name of Attack	METRIC	Anova & ExtraTree & RandomForset &pearson (the first 5 higher rank)	Anova & ExtraTree & RandomForset &pearson (the first 10 higher rank)	Anova & ExtraTree & RandomForset &pearson (the first 15 higher rank)	Anova & ExtraTree & RandomForset &pearson (the first 20 higher rank)
NTP Attack	TP	6545	10856	17881	30901
	FP	308	526	1418	2344
	TN	12020	26006	37218	43168
	FN	1055	2468	3267	3299
	Condition positive	7600	13324	21148	34200
	Condition_negative	12328	26532	38636	45512
	Total_population	19928	39856	59784	79712
	Predicted_condition_positive	6853	11382	19299	33245
	Predicted_condition_negative	13075	28474	40485	46467
	TPR	0.861184211	0.814770339	0.845517307	0.90353801
	FPR	0.024983777	0.019825117	0.036701522	0.0515029
	TNR	0.975016223	0.980174883	0.963298478	0.9484971
	FNR	0.138815789	0.185229661	0.154482693	0.09646199
	Accuracy	0.931603774	0.924879566	0.921634551	0.92920765
	F1-score	0.90569432	0.878814863	0.884169407	0.91633183
	PPV	0.95505618	0.953786681	0.92652469	0.92949316
	FDR	0.04494382	0.046213319	0.07347531	0.07050684
	FOR	0.080688337	0.086675564	0.080696554	0.07099662

	NPV	0.919311663	0.913324436	0.919303446	0.92900338
	Prevalence	0.381372943	0.334303493	0.353740131	0.42904456
	PLR	34.46973684	41.09788335	23.03766337	17.5434394
	NLR	0.1423728	0.188976135	0.16036846	0.10169982
	DOR	242.1090047	217.4765793	143.6545776	172.502161
	Informedness_or_Youden_J_statistic	0.836200434	0.794945222	0.808815785	0.85203511
	Prevalence_threshold	0.145537246	0.13493887	0.172421029	0.19273434
	fowlkes_mallows_score	0.906906446	0.881542454	0.885094719	0.9164237
	Matthews_correlation_coefficient	0.855071207	0.830244446	0.827114954	0.85525972
	Threat_score	0.827642893	0.783826715	0.792386777	0.84558341
DNS Attack	TP	44554	85386	60906	125502
	FP	1573	1753	1595	3154
	TN	33934	69875	50197	106744
	FN	1750	6608	1837	10033
	Condition positive	46304	91994	62743	135535
	Condition_negative	35507	71628	51792	109898
	Total_population	81811	163622	114535	245433
	Predicted_condition_positive	46127	87139	62501	128656
	Predicted_condition_negative	35684	76483	52034	116777
	TPR	0.962206289	0.928169228	0.970721834	0.92597484
	FPR	0.044301124	0.02447367	0.030796262	0.028699339
	TNR	0.955698876	0.97552633	0.969203738	0.971300661
	FNR	0.037793711	0.071830772	0.029278166	0.07402516
	Accuracy	0.95938199	0.948900515	0.97003536	0.946270469
	F1-score	0.964048858	0.953325183	0.97259749	0.950085355
	PPV	0.965898498	0.979882716	0.974480408	0.975485014
	FDR	0.034101502	0.020117284	0.025519592	0.024514986
	FOR	0.049041587	0.086398285	0.03530384	0.085915891
	NPV	0.950958413	0.913601715	0.96469616	0.914084109
	Prevalence	0.565987459	0.562234907	0.547806347	0.552228103

	PLR	21.71968131	37.92521705	31.52076815	32.26467439
	NLR	0.039545627	0.073632837	0.030208475	0.076212405
	DOR	549.2309276	515.0584772	1043.441239	423.3520027
	Informedness_or_Youden_J_statistic	0.917905165	0.903695559	0.939925572	0.897275501
	Prevalence_threshold	0.176664777	0.139697089	0.151186757	0.149696112
	fowlkes_mallows_score	0.964050626	0.953675513	0.972599305	0.950407586
	Matthews_correlation_coefficient	0.917380888	0.898575491	0.939550995	0.893414003
	Threat_score	0.930592978	0.910813146	0.946656719	0.904916756
MSSQL Attack	TP	14776	22873	22873	52190
	FP	44	347	347	407
	TN	58213	122319	122319	166157
	FN	234	995	995	1047
	Condition positive	15010	23868	23868	53237
	Condition_negative	58257	122666	122666	166564
	Total_population	73267	146534	146534	219801
	Predicted_condition_positive	14820	23220	23220	52597
	Predicted_condition_negative	58447	123314	123314	167204
	TPR	0.984410393	0.958312385	0.958312385	0.980333227
	FPR	7.55E-04	0.00282882	0.00282882	0.002443505
	TNR	0.999244726	0.99717118	0.99717118	0.997556495
	FNR	0.015589607	0.041687615	0.041687615	0.019666773
	Accuracy	0.996205659	0.990841716	0.990841716	0.993384925
	F1-score	0.990680523	0.97150017	0.97150017	0.986261504
	PPV	0.997031039	0.985055986	0.985055986	0.992261916
	FDR	0.002968961	0.014944014	0.014944014	0.007738084
	FOR	0.004003627	0.008068832	0.008068832	0.006261812
	NPV	0.995996373	0.991931168	0.991931168	0.993738188
	Prevalence	0.20486713	0.162883699	0.162883699	0.242205449
	PLR	1303.381733	338.7675706	338.7675706	401.1995666
	NLR	0.01560139	0.041805877	0.041805877	0.019714947

	DOR	83542.66589	8103.348115	8103.348115	20350.02037
	Informedness_or_Youden_J_statistic	0.983655119	0.955483565	0.955483565	0.977889722
	Prevalence_threshold	0.02695245	0.051531411	0.051531411	0.047551193
	fowlkes_mallows_score	0.990700619	0.971592173	0.971592173	0.986279538
	Matthews_correlation_coefficient	0.988330156	0.966175537	0.966175537	0.981936539
	Threat_score	0.981533147	0.944579806	0.944579806	0.972895384
SSDP Attack	TP	39347	76775	94877	114477
	FP	163	1917	3635	3568
	TN	24382	51084	57675	77219
	FN	2165	2338	2350	2907
	Condition positive	41512	79113	97227	117384
	Condition_negative	24545	53001	61310	80787
	Total_population	66057	132114	158537	198171
	Predicted_condition_positive	39510	78692	98512	118045
	Predicted_condition_negative	26547	53422	60025	80126
	TPR	0.947846406	0.970447335	0.975829759	0.975235126
	FPR	0.006640864	0.036169129	0.05928886	0.044165522
	TNR	0.993359136	0.963830871	0.94071114	0.955834478
	FNR	0.052153594	0.029552665	0.024170241	0.024764874
	Accuracy	0.964757709	0.967792967	0.96224856	0.967326198
	F1-score	0.971267063	0.973036342	0.969423569	0.972497016
	PPV	0.995874462	0.975639201	0.963100942	0.969774239
	FDR	0.004125538	0.024360799	0.036899058	0.030225761
	FOR	0.081553471	0.043764741	0.039150354	0.036280358
	NPV	0.918446529	0.956235259	0.960849646	0.963719642
	Prevalence	0.628426965	0.598823743	0.613276396	0.592336921
	PLR	142.7293867	26.83081857	16.45890579	22.08136774
	NLR	0.052502254	0.030661671	0.025693584	0.025909166
	DOR	2718.538245	875.0605429	640.5842694	852.2608431
	Informedness_or_Youden_J_statistic	0.941205542	0.934278206	0.916540899	0.931069604

	Prevalence_threshold	0.077238325	0.161816295	0.197747353	0.175466869
	fowlkes_mallows_score	0.971563703	0.973039805	0.96944446	0.972500849
	Matthews_correlation_coefficient	0.927665879	0.933075559	0.920238286	0.932280954
	Threat_score	0.944139172	0.947488584	0.940661498	0.946466367
SYN Attack	TP	11472	25229	18349	43371
	FP	53	100	87	380
	TN	11443	20256	17969	24834
	FN	31	414	394	413
	Condition positive	11503	25643	18743	43784
	Condition_negative	11496	20356	18056	25214
	Total_population	22999	45999	36799	68998
	Predicted_condition_positive	11525	25329	18436	43751
	Predicted_condition_negative	11474	20670	18363	25247
	TPR	0.997305051	0.983855243	0.978978819	0.990567331
	FPR	0.004610299	0.004912556	0.004818343	0.015070992
	TNR	0.995389701	0.995087444	0.995181657	0.984929008
	FNR	0.002694949	0.016144757	0.021021181	0.009432669
	Accuracy	0.996347667	0.988825844	0.986928993	0.988506913
	F1-score	0.996352267	0.989916032	0.987062589	0.990940767
	PPV	0.995401302	0.996051956	0.995280972	0.991314484
	FDR	0.004598698	0.003948044	0.004719028	0.008685516
	FOR	0.002701761	0.020029028	0.021456189	0.016358379
	NPV	0.997298239	0.979970972	0.978543811	0.983641621
	Prevalence	0.500152181	0.557468641	0.509334493	0.634569118
	PLR	216.3211107	200.2735733	203.1774891	65.72674914
	NLR	0.002707431	0.01622446	0.021122959	0.009577004
	DOR	79899.02374	12343.92812	9618.798676	6862.975749
	Informedness_or_Youden_J_statistic	0.992694752	0.978942687	0.974160476	0.975496338
	Prevalence_threshold	0.063662401	0.065998739	0.065556429	0.10980319
	fowlkes_mallows_score	0.996352721	0.989934816	0.987096242	0.990940837

	Matthews_correlation_coefficient	0.992697146	0.977481717	0.973992615	0.975226184
	Threat_score	0.992731049	0.980033407	0.974455656	0.982044199
TFTP Attack	TP	11568	19959	19959	31539
	FP	421	559	559	966
	TN	5077	13580	13580	18681
	FN	63	160	160	201
	Condition positive	11631	20119	20119	31740
	Condition_negative	5498	14139	14139	19647
	Total_population	17129	34258	34258	51387
	Predicted_condition_positive	11989	20518	20518	32505
	Predicted_condition_negative	5140	13740	13740	18882
	TPR	0.994583441	0.992047318	0.992047318	0.993667297
	FPR	0.076573299	0.039536035	0.039536035	0.049167812
	TNR	0.923426701	0.960463965	0.960463965	0.950832188
	FNR	0.005416559	0.007952682	0.007952682	0.006332703
	Accuracy	0.971743826	0.979012202	0.979012202	0.977289976
	F1-score	0.979508891	0.982306765	0.982306765	0.981835162
	PPV	0.964884477	0.972755629	0.972755629	0.970281495
	FDR	0.035115523	0.027244371	0.027244371	0.029718505
	FOR	0.012256809	0.011644833	0.011644833	0.010645059
	NPV	0.987743191	0.988355167	0.988355167	0.989354941
	Prevalence	0.679023878	0.587278884	0.587278884	0.617665947
	PLR	12.9886455	25.09223083	25.09223083	20.20971157
	NLR	0.005865716	0.008280042	0.008280042	0.006660169
	DOR	2214.332315	3030.447451	3030.447451	3034.414156
	Informedness_or_Youden_J_statistic	0.918010141	0.952511283	0.952511283	0.944499485
	Prevalence_threshold	0.217203549	0.166411098	0.166411098	0.181966359
	fowlkes_mallows_score	0.979621418	0.982354118	0.982354118	0.981904777
	Matthews_correlation_coefficient	0.935158735	0.956801379	0.956801379	0.952037877
	Threat_score	0.95984069	0.965228746	0.965228746	0.964318474

UDP-Lag Attack	TP	5359	9924	12725	14820
	FP	177	421	470	529
	TN	2675	6055	8016	9130
	FN	40	103	244	275
	Condition positive	5399	10027	12969	15095
	Condition_negative	2852	6476	8486	9659
	Total_population	8251	16503	21455	24754
	Predicted_condition_positive	5536	10345	13195	15349
	Predicted_condition_negative	2715	6158	8260	9405
	TPR	0.992591221	0.989727735	0.981185905	0.981782047
	FPR	0.062061711	0.065009265	0.055385341	0.054767574
	TNR	0.937938289	0.934990735	0.944614659	0.945232426
	FNR	0.007408779	0.010272265	0.018814095	0.018217953
	Accuracy	0.973700158	0.968248197	0.966721044	0.967520401
	F1-score	0.980155464	0.974278421	0.972710595	0.973590855
	PPV	0.968027457	0.959304012	0.964380447	0.965535214
	FDR	0.031972543	0.040695988	0.035619553	0.034464786
	FOR	0.014732965	0.01672621	0.029539952	0.029239766
	NPV	0.985267035	0.98327379	0.970460048	0.970760234
	Prevalence	0.654344928	0.607586499	0.604474481	0.609800436
	PLR	15.99361673	15.22441048	17.71562466	17.92633798
	NLR	0.007899005	0.010986488	0.019917217	0.019273517
	DOR	2024.763418	1385.739455	889.4628532	930.1020794
	Informedness_or_Youden_J_statistic	0.93052951	0.92471847	0.925800564	0.927014473
	Prevalence_threshold	0.200031925	0.204004733	0.191975674	0.191060266
	fowlkes_mallows_score	0.980232398	0.97439714	0.972746885	0.973624742
	Matthews_correlation_coefficient	0.941843222	0.933605432	0.93030955	0.931643403
	Threat_score	0.961083214	0.949846861	0.946871047	0.948540707
UDP Attack	TP	10642	24239	24239	39834
	FP	467	1924	1924	2099

	TN	38665	74937	64627	110608
	FN	1792	2033	2030	2158
	Condition positive	12434	26272	26269	41992
	Condition_negative	39132	76861	66551	112707
	Total_population	51566	103133	92820	154699
	Predicted_condition_positive	11109	26163	26163	41933
	Predicted_condition_negative	40457	76970	66657	112766
	TPR	0.855879041	0.922617235	0.922722601	0.948609259
	FPR	0.011933967	0.025032201	0.028910159	0.018623511
	TNR	0.988066033	0.974967799	0.971089841	0.981376489
	FNR	0.144120959	0.077382765	0.077277399	0.051390741
	Accuracy	0.956192065	0.961632067	0.957401422	0.972482046
	F1-score	0.904047912	0.924535139	0.924588038	0.949276139
	PPV	0.957962013	0.926461033	0.926461033	0.949943958
	FDR	0.042037987	0.073538967	0.073538967	0.050056042
	FOR	0.044293942	0.026412888	0.030454416	0.019136974
	NPV	0.955706058	0.973587112	0.969545584	0.980863026
	Prevalence	0.241127875	0.254739026	0.283010127	0.271443254
	PLR	71.7178986	36.85721586	31.91689803	50.93611422
	NLR	0.145861667	0.07936956	0.079578012	0.052365979
	DOR	491.6843478	464.3747048	401.0768445	972.6947675
	Informedness_or_Youden_J_statistic	0.843945074	0.897585034	0.893812442	0.929985748
	Prevalence_threshold	0.105611765	0.141422424	0.150387152	0.122896109
	fowlkes_mallows_score	0.905483081	0.924537136	0.924589927	0.949276374
	Matthews_correlation_coefficient	0.878114838	0.898815746	0.894908857	0.930396276
	Threat_score	0.824897295	0.859660945	0.859752421	0.903449684
LDAP Attack	TP	21392	39203	53482	60589
	FP	49	99	304	317
	TN	13560	31342	37996	45117
	FN	687	732	1007	1041

	Condition positive	22079	39935	54489	61630
	Condition_negative	13609	31441	38300	45434
	Total_population	35688	71376	92789	107064
	Predicted_condition_positive	21441	39302	53786	60906
	Predicted_condition_negative	14247	32074	39003	46158
	TPR	0.96888446	0.981670214	0.981519206	0.983108876
	FPR	0.003600558	0.003148755	0.007937337	0.006977154
	TNR	0.996399442	0.996851245	0.992062663	0.993022846
	FNR	0.03111554	0.018329786	0.018480794	0.016891124
	Accuracy	0.979376821	0.988357431	0.98587117	0.987315998
	F1-score	0.983088235	0.989512475	0.987891942	0.988917543
	PPV	0.997714659	0.997481044	0.994347972	0.994795258
	FDR	0.002285341	0.002518956	0.005652028	0.005204742
	FOR	0.048220678	0.022822224	0.025818527	0.02255297
	NPV	0.951779322	0.977177776	0.974181473	0.97744703
	Prevalence	0.618667339	0.559501793	0.587235556	0.575637002
	PLR	269.092829	311.7645778	123.6585052	140.9040021
	NLR	0.031227978	0.018387684	0.018628656	0.017009804
	DOR	8617.043044	16955.0757	6638.079731	8283.693224
	Informedness_or_Youden_J_statistic	0.965283902	0.978521459	0.973581869	0.976131722
	Prevalence_threshold	0.057457893	0.053599598	0.082506994	0.077698272
	fowlkes_mallows_score	0.983193892	0.989544052	0.987912765	0.988934805
	Matthews_correlation_coefficient	0.957356389	0.97658823	0.971052371	0.974185064
	Threat_score	0.966738973	0.979242644	0.976073586	0.978078034
SNMP Attack	TP	51316	93304	76152	145009
	FP	188	3325	3298	3493
	TN	34149	73706	56568	106202
	FN	168	1307	1295	2759
	Condition positive	51484	94611	77447	147768
	Condition_negative	34337	77031	59866	109695

	Total_population	85821	171642	137313	257463
	Predicted_condition_positive	51504	96629	79450	148502
	Predicted_condition_negative	34317	75013	57863	108961
	TPR	0.99673685	0.986185539	0.983278887	0.98132884
	FPR	0.005475143	0.04316444	0.0550897	0.031842837
	TNR	0.994524857	0.95683556	0.9449103	0.968157163
	FNR	0.00326315	0.013814461	0.016721113	0.01867116
	Accuracy	0.995851831	0.973013598	0.966550873	0.975716899
	F1-score	0.996543287	0.975779126	0.970726018	0.978897627
	PPV	0.996349798	0.96559004	0.958489616	0.976478431
	FDR	0.003650202	0.03440996	0.041510384	0.023521569
	FOR	0.004895533	0.017423647	0.02238045	0.025320986
	NPV	0.995104467	0.982576353	0.97761955	0.974679014
	Prevalence	0.599899791	0.551211242	0.564017974	0.57393878
	PLR	182.0476236	22.84717541	17.84868826	30.8178835
	NLR	0.003281114	0.014437655	0.017695979	0.019285258
	DOR	55483.4753	1582.471394	1008.629621	1598.002178
	Informedness_or_Youden_J_statistic	0.991261707	0.943021098	0.928189187	0.949486003
	Prevalence_threshold	0.069001196	0.173014215	0.191395956	0.152639463
	fowlkes_mallows_score	0.996543305	0.975833456	0.970805132	0.978900631
	Matthews_correlation_coefficient	0.991357981	0.945590246	0.932140765	0.950321356
	Threat_score	0.993110389	0.952703807	0.943117221	0.958667469

B.2 SMOTE Technique Code

```
import numpy as np
from sklearn.neighbors import NearestNeighbors
def generate_synthetic_samples(minority_samples, n_samples, k_neighbors=5):
    """
    Generates synthetic samples using SMOTE algorithm.

    Parameters:
    - minority_samples: ndarray, samples of the minority class
    - n_samples: int, number of synthetic samples to create
    - k_neighbors: int, number of nearest neighbors to consider

    Returns:
    - synthetic_samples: ndarray, generated synthetic samples
    """
    n_minority_samples, n_features = minority_samples.shape
    synthetic_samples = np.zeros((n_samples, n_features))

    # Find the k-nearest neighbors for each minority sample
    neigh = NearestNeighbors(n_neighbors=k_neighbors)
    neigh.fit(minority_samples)
    neighbors = neigh.kneighbors(minority_samples, return_distance=False)

    for i in range(n_samples):
        # Randomly choose a minority sample
        sample_index = np.random.randint(0, n_minority_samples)
        sample = minority_samples[sample_index]

        # Randomly choose one of its k-nearest neighbors
        neighbor_index = np.random.choice(neighbors[sample_index])
        neighbor = minority_samples[neighbor_index]

        # Generate a synthetic sample along the line connecting the sample and its
neighbor
        diff = neighbor - sample
        gap = np.random.rand()
        synthetic_samples[i] = sample + gap * diff

    return synthetic_samples
```

LIST OF PUBLICATIONS

Published Papers

B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, S. L. M. Hassan and M. Alghrairi, "Identification of Distributed Denial of Services Anomalies by Using Combination of Entropy and Sequential Probabilities Ratio Test Methods," *Sensors*, vol. 21, no. 19, pp. 1-17, 2021. doi: 10.3390/s21196453.

B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, and S. L. M. Hassan, "Detection of different Types of Distributed Denial of Service Attacks using Multiple Features of Entropy and Sequential Probabilities Ratio Test," *Journal of Engineering Science and Technology*, vol. 18, no. 2, pp. 844-864, 2023.

B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, and S. L. M. Hassan, "DDoS Detection Using Active and Idle Features of Revised CICFlowMeter and Statistical Approaches," *in 4th International Conference on Advanced Science and Engineering (ICOASE),* Zakho, Iraq: IEEE, March 2023, pp. 148–153. doi: 10.1109/ICOASE56293.2022.10075591.

Accepted Papers

B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, S. L. M. Hassan, A. A. Al-Khafaji, and N.N.A. Sjarif, "Shannon Entropy based DDoS Attacks Detection using Combination of Machine Learning based Feature Importance Techniques," *AIP Conference Proceedings*, Baghdad, Iraq, 2023.