NETWORK INTRUSION DETECTION WITH PAYLOAD-BASED APPROACH

by

Süleyman Özdel

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical & Electronics Engineering
Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to thank my thesis advisors Prof. Emin Anarım and Prof. Mutlu Koca, for their patience, contributions, and valuable guidance during this thesis. I am extremely grateful for their endless support and always being ready to help with every obstacle I encountered during this journey.

In addition, I would like to thank Prof. Hakan Delic, Prof. Fatih Alagöz, and Doç. Şerif Bahtiyar for being on my thesis committee and their time.

My special thanks should be given to my friends and colleagues, Çağatay Ateş and Metehan Yıldırım, for their support all through this thesis. I appreciate their friendship and assistance.

Finally, and very importantly, I wish to thank my mother, Hülya Özdel, and my father, Veysel Özdel, for their support throughout my life.

This work is supported by the Boğaziçi University Scientific Research Projects under the Spectral and Entropy Approaches in Packet Classification, 18281.

ABSTRACT

NETWORK INTRUSION DETECTION WITH PAYLOAD-BASED APPROACH

Rapidly growing network systems become more vulnerable to threats with the improved sophistication of attack techniques. Various types of network attacks affect networks in different ways and continue to be a serious threat despite developing intrusion detection mechanisms. Early detection of network intrusions is crucial to taking precautions and reducing the damage to the system. In addition, the ability to distinguish attacker flows from legitimate ones ensures that the network continues to provide service safely to the clients. In this thesis, payload-based features that characterize network flows are proposed to provide early detection of network attacks and to identify attacker flows. Besides the features conventionally used in application classification, features based on greedy algorithm-based metrics that allow comparing defined probability distributions over different sample spaces at various lengths are also used. Moreover, features based on spectral domain analysis of payload sequences are extracted to capture the complicated patterns that are not observed in the original domain. Also, features based on discrete cosine transforms are utilized in the characterization of these network flows. These features are extracted using N-gram analysis for various N values. In the classification stage, SVM models trained with these features are used. Performance evaluation is given for publicly available IDS 2012 and IDS 2017 datasets that contain different kinds of attack traces. Early detection of network intrusions based on features extracted from the first 3 and 5 packets of a flow achieves high detection rates while detecting network intrusions early.

ÖZET

YÜK TABANLI YAKLAŞIM İLE AĞ SALDIRILARININ TESPİTİ

Hızla büyüyen ağ sistemleri, gelişmiş sofistike saldırı teknikleri ile tehditlere daha açık hale gelmektedir. Ağları farklı şekillerde etkileyen çeşitli ağ saldırıları, saldırı tespit mekanizmaları geliştirse de ciddi bir tehdit olmaya devam etmektedir. Ağ saldırılarının erken tespiti, önlem almak ve sistemin zarar görmesini azaltmak için çok önemlidir. Ayrıca, saldırgan akışlarını meşru akışlardan ayırt etme yeteneği, ağın istemcilere güvenli bir şekilde hizmet vermeye devam etmesini sağlar. Bu tezde, ağ saldırılarının erken tespitini sağlamak ve saldırgan akışları belirlemek için ağ akışlarını karakterize eden yük tabanlı özellikler önerilmiştir. Uygulama sınıflandırmasında geleneksel olarak kullanılan özniteliklerin yanı sıra, çeşitli uzunluklarda farklı örnek uzayları üzerinde tanımlanmış olasılık dağılımlarının karşılaştırılmasına olanak tanıyan açgözlü algoritma tabanlı metrik temelli öznitelikler kullanılmaktadır. Ayrıca, orijinal alanda gözlenmeyen karmaşık kalıpları yakalamak için faydalı yük dizilerinin spektral alan analizine dayalı özellikler çıkarılmıştır. Ayrıca, bu ağ akışlarının karakterizasyonunda ayrık kosinüs dönüşümüne dayalı özelliklerden yararlanılmıştır. Bu özellikler, çeşitli N değerleri için N-gram analizi kullanılarak çıkarılmıştır. Sınıflandırma aşamasında bu öznitelikler ile eğitilmiş SVM modeli kullanılır. Performans değerlendirmesi, farklı türde saldırı izlerini içeren kamuya açık IDS 2012 ve IDS 2017 veri setleri için verilmektedir. Bir akışın ilk 3 ve 5 paketinden çıkarılan özellikler, yüksek tespit oranları elde ederek ağ izinsiz girişlerinin erken tespit edilmesini sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ARP	Address Resolution Protocol
CIA	Confidentiality, Integrity, Availability
CNN	COnvolutional Neural Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DoS	Denial of Service
DDoS	Distributed Denial of Service
DNS	Domain Name System
EWMA	Exponentially Weighted Moving Average
FFT	Fast Fourier Transform
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IP	Internet Protocol Address
IRC	Internet Relay Chat
KL	Kullback-Leiber
LOIC	Low Orbit Ion Canon
P2P	Peer-To-Peer
POD	Ping of Death
PSD	Power Spectral Density
SNMP	Simple Network Management Protocol
SSH	Secure Shell Protocol
SOM	Self Organized Map
SVM	Support Vector Machines
UDP	User Datagram Protocol

1. INTRODUCTION

With the rapid growth of devices connecting to the Internet and other public or private computer systems, the frameworks that support these connections become more open to risks. The primary reason for the vulnerability is that network traffic behavior is constantly changing. In other words, because new network threats arise daily, the concept of normal and anomalous network traffic behavior varies. Due to the highly variable nature of network data, analyzing the performance of a particular algorithm to identify threats is incredibly challenging.

Computer systems and networks are threatened by viruses, malware, and cyber-attacks even though they were first used. In 2018, the number of internet users was about 3.9 billion all over the world, as shown in Figure 1.1. It is expected to increase to 5.3 billion in 2023, according to [1]. There is about a 35% increase in only 5 years, with a compound annual growth rate of 6%. Similarly, the number of devices connected to the Internet continues to increase rapidly as shown in Figure 1.1, according to Cisco (2020). Securing connections between these devices remains a challenge as the number of devices that are part of the Internet increases.

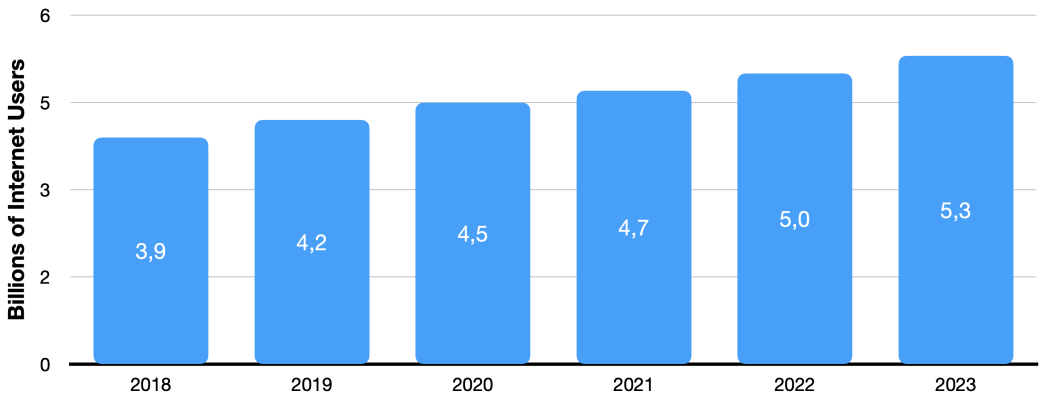

Figure 1.1. Global Internet Users [1].

The number of intrusions is also increasing significantly year after year. Numerous defense and detection mechanisms against malicious activities have been proposed in the literature to provide secure connections among network devices. However, the

exponential growth of attackers' computational resources and capabilities, which allows them to introduce sophisticated malicious actions, new challenges are created on this issue [3].

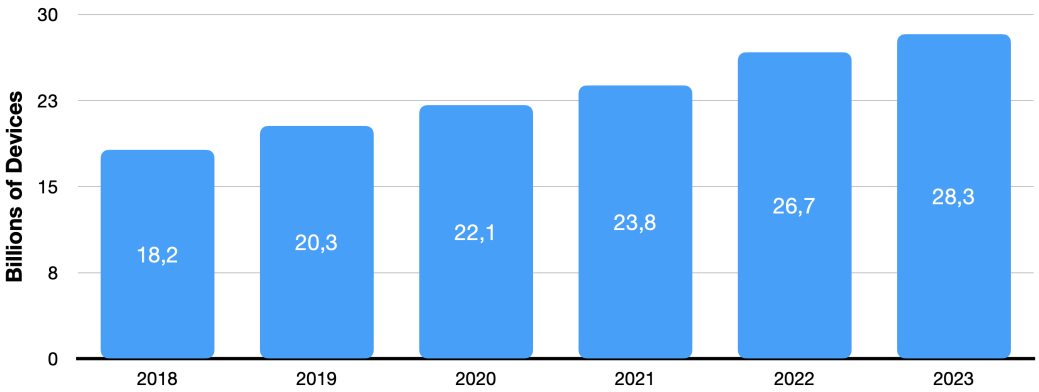

Figure 1.2. Number of Devices connected to Internet [1].

One of the primary challenges in network security is designing and implementing a detection system for intrusions that can effectively monitor network activity and identify network attacks. In order to make detection efficient, the objectives and characteristics of network intrusions should be analyzed.

1.0.1. Thesis Contribution

The primary contributions of this thesis are summarized as follows.

- We conducted a small survey covering 27 different types of network attacks with summarizing their objectives and application methods.
- Even though payload-based intrusion detection models are available in the literature, we propose novel features to characterize network flows by utilizing methods.
- A greedy algorithm-based metric previously proposed in [4] to measure the similarities of probability distributions are used in feature extraction. It makes it possible to calculate upper bound of Kullback-Leiber divergence on real data without assuming strict conditions.
- We used tools generally employed in image processing tasks to extract features. Frequency domain analysis is employed when considering payload bytes of sequen-

tial data. Related frequency domain features generally used in image classification tasks are utilized to characterize network flows.

- We applied the Discrete Cosine Transform to the payload sequence, and we used the high energy coefficients to classify network flows.
- We extend these features by applying N-gram analysis, widely used in natural language processing tasks, to capture more complex patterns.
- The classification performance of the proposed features is evaluated using detailed diversified experiments on the widely used IDS 2012 and IDS 2017 datasets in literature.

1.0.2. Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2, explanations of a large number of different network attacks are given. Also, a review of the literature is given in this section. In Chapter 3, the theoretical basis of the features depends on information theory, which is given in this section. In chapter 4, the flow-based feature extraction methods and a detailed explanation of the proposed features are given. In Chapter 5, the general architecture of the proposed intrusion detection algorithm is defined. In Chapter 6, datasets used to evaluate the algorithms are introduced, and a large number of detailed experiment results are presented. Finally, Chapter 7 concludes our thesis by summarizing the general architecture and results.

2. NETWORK INTRUSIONS AND DETECTION ALGORITHMS

An overview of the various kinds of network attacks that are widely encountered on network systems is provided in this chapter. Following that, an overview of the literature on network anomaly detection algorithms is presented.

2.1. Intrusions in Network Systems

Most people's daily activities rely on computer networks because of the Internet's rapid expansion. Network attacks try to override the network's security measures using the victim network's weaknesses. Network intrusions interrupt normal network activities by causing network equipment to fail, overloading network resources, denying legitimate users access to their services, scanning illegally, or gaining unauthorized access to the system. Additionally, an attacker may attempt to interrupt regular internet usage by exploiting gaps, defects, and configuration issues in software systems. An attack can take several forms, including a brute force, denial-of-service/denial-of-service attack, or network scanning activity. Some of the attack types are explained in this section.

2.1.1. DoS and DDoS Attacks in Network Systems

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are among the most commonly encountered attack types on network systems. As the name implies, a denial-of-service attack is an attempt by attackers to prevent users from accessing a network system, service, website, application, or other resources. Typically, the attack causes a machine to respond slowly or disables it.

A single-source attack is referred to as a denial-of-service attack. However, distributed denial-of-service assaults, which are launched at a target by several parties

orchestrated by an attacker, are significantly more prevalent today. Distributed attacks are more sophisticated, perhaps more damaging, and, in certain situations, more challenging to detect and halt for the victim.

Whether the attack is DoS or DDoS, the outcome is the same—legitimate users cannot access the services to which they are authorized. DDoS attacks are one of the most powerful ways for attackers to get around availability, the third of three important security principles called the CIA triad (confidentiality, integrity, and availability).

A Denial of Service attack aims to exhaust the resources of the victim. An attacker sent a large number of network packets to consume the resources of the victim side and make them unavailable for the legitimate hosts. There is no privilege requirement for the attacker to initiate a DoS attack. There is only one computer used in a DoS attack.

The Distributed Denial of Service attack has the same purpose as the DoS attack. The only difference is the way to make the victim unavailable. In the DDoS attack, network packets are sent by many computers controlled by the attacker, while there is only one computer used in the DoS attack. A network packet flood, generated by the numerous computers, is sent to the victim's computer to prevent the availability of the computer. DDoS attacks In these attacks, computers controlled by the attackers are frequently geographically dispersed. It makes it difficult to specify the origin of the threat and block harmful traffic. Since DDoS attacks involve client coordination, they are typically carried out via botnets. A botnet is an enormous collection of infected machines (bots) managed by the attacker. These machines are also called "zombies". These zombies can be controlled in different ways by using various agents, and DDoS attacks can be executed in a fully automatic way [5].

DDoS attacks can be categorized based on how they are executed and other properties. In literature, there are several DDoS attack taxonomies based on various aspects of DDoS attacks [2,5,6]. One of the fundamental research for the categorization is given [2]. This study examines DDoS attacks in two categories according to their

primary purpose. While some DDoS attacks are executed to consume bandwidth, others aim to consume the resources of the victim's system. The attacks in the first category create an undesired flood on the network, blocking the regular user traffic. The latter aims to make the victim's system incapable of handling service requests from regular users. The main diagram, which shows the taxonomy of the DDoS attacks according to this categorization, is given in the Figure 2.1. General descriptions of the main categories are as follows.

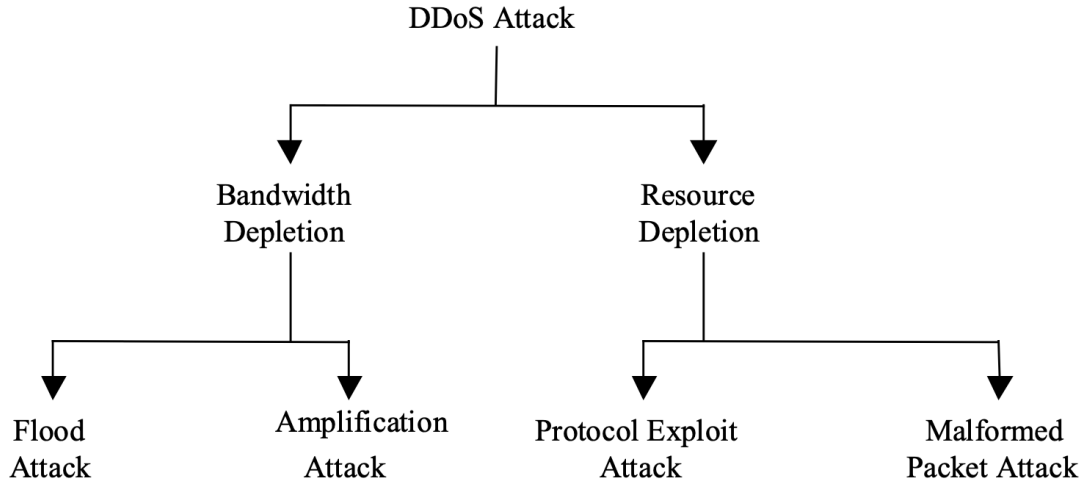

Figure 2.1. DDoS Attack Taxonomy [2].

2.1.1.1. Bandwidth Depletion Attacks.

2.1.1.1.1. Flood Attacks Zombie computers, controlled by the attackers, are utilized in the flood attacks to deliver massive network traffic to the victim's side. The victim's network bandwidth is exhausted by the large volume of traffic delivered by the zombie computers. Flood attacks are mainly named based on the used protocols, such as Internet Control Message Protocol (ICMP) flood and User Datagram Protocol (UDP) flood attacks.

2.1.1.1.2. Amplification Attacks A DDoS amplification attack uses the feature of routers to broadcast IP addresses to magnify and reflect the attack. This capability allows a sending system to specify a broadcasting IP address as the target IP address

rather than a specific IP address. It instructs the network's routers to replicate and transmit packets to all network devices in the broadcast address range. The attacker can either send the broadcast message directly or through agents to increase the volume of attacking traffic in this type of DDoS attack.

2.1.1.2. Resource Consuming Attacks.

2.1.1.2.1. Protocol Exploit Attacks Protocol exploiting attacks mainly use the weaknesses of the network protocols, both at OSI layer 3 and layer 4, to make victims' resources unavailable to legitimate users. Because of their nature, ICMP (Internet Control Message Protocol), TCP (Transport Control Protocol), and UDP (User Datagram Protocol) are generally exploited in this type of attack. The main objective is to exhaust the network's or intermediate resources' computational capabilities (such as firewalls), resulting in a denial of service attack. Protocol attacks are often measured by how many packets they send per second since they are operated at the packet level.

2.1.1.2.2. Malformed Packet Attacks An attack with malformed packets is when the attacker orders the zombies to transmit improperly constructed IP packets to the target's computer to collapse it. Malformed packet threats come in two flavors. An IP packet containing identical source and destination IP addresses can be considered as an IP address assault. This attack can cause the affected system's operating system to become confused and crash. In an IP packet options attack, a malformed packet may randomly generate the optional fields within an IP packet with quality of service bits as one, requiring the victim system to spend additional processing time analyzing the data. When multiplied by a sufficient number of agents, this attack can entirely disable the victim system's processing capability.

2.1.1.3. Current DoS/DDoS Attack Taxonomy. In recent works [7,8] and security reports [9,10], the categorization of DDoS attacks is made with a slight difference. This categorization is used more in the DDoS reports prepared by the companies in this

field. There are three main categories: Volumetric attacks, protocol attacks, and Application layer attacks. Volumetric attacks correspond to bandwidth depletion attacks, while the protocol attacks cover protocol exploit attacks in the previous taxonomy. Application layer attacks can be considered as a new category that also covers malformed packet attacks exhausting the victim's resources. It is important to note that strictly categorizing DDoS attacks is difficult because there is some overlap in these categories. Some volumetric attacks, such as ICMP flood and UDP flood, are based on vulnerabilities of these protocols, and also they are considered as protocol attacks. However, this categorization is still useful to distinguish encountered DDoS attacks. It facilitates evaluating their effects and developing detection techniques using their general characteristics. General definitions of these categories used in current literature and industry are given as follows.

2.1.1.3.1. Volumetric Attacks Volumetric attack sometimes referred to as floods, is the most prevalent sort of DDoS attack. They often flood the targeted victim's network with traffic with the intent of demanding so much bandwidth to make it impossible to access of legitimate users. Attackers frequently use botnets to increase the amount of traffic hitting a targeted system or service. This way enables attackers to conduct massive DDoS attacks that can reach volumes of hundreds of gigabits per second to terabits per second, far exceeding the capability of most organizations' networks.

2.1.1.3.2. Protocol Attacks A protocol attack is also referred to as a computational or network attack. It creates a computational load on the resources of the victim side, abusing vulnerabilities in the nature of the network protocols rather than aiming to bandwidth resources. These attacks are performed in Open Systems Interconnection layers 3 and 4. These protocols are Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), and Transport Control Protocol (TCP). The objective is to deplete the network's computing capacity or intermediary resources such as firewalls, resulting in a denial of service attack. They are often expressed in packets per second due to the fact that protocol assaults operate at the packet level.

2.1.1.3.3. Application Layer Attacks Application layer attacks sometimes referred to as OSI layer 7 attacks, are directed at web servers, online application platforms, and individual web-based apps instead of the network itself. The attacker aims to make a website or program inaccessible to users by bringing the server to a standstill. These attacks may exploit known application vulnerabilities, an application's underlying business logic, or higher-layer protocols such as Hypertext Transfer Protocol/Secure (HTTP/HTTPS) and Simple Network Management Protocol (SNMP). These assaults frequently consume less bandwidth than other forms of attacks and so do not necessarily cause a spike in traffic, making them more challenging to detect. Attacks on the application layer are quantified in terms of requests per second.

2.1.1.4. Common DoS/DDoS Attacks.

2.1.1.4.1. SYN FLood Transport Control Protocol (TCP) requires a three-way handshake to start the connection between client and server [11]. A three-way handshake starts with an SYN (synchronize) packet, which a client sends to build the connection via a website. Then, the server responds with an SYN-ACK (synchronize-acknowledge) packet and waits for the client to send an ACK (acknowledge) packet. The legitimate TCP handshake process is shown in Figure 2.2.

This volumetric attack prohibits a server from handling new connection requests. By altering the typical method, TCP connects a client to a server. In an SYN flood attack, the attacker purposefully does not transmit the ACK packet after SYN requests. The recipient system's processing and memory resources store and wait for the ACK request for this TCP SYN request until a timeout occurs. SYN Flood attacks create floods. Therefore, it can be considered as a volumetric attack. Besides that, it exploits the vulnerability of the TCP protocol, and it can also be considered as a protocol attack. IP spoofing, which modification of an internet protocol to mask the actual IP address of the sender, is used to hide zombies' identities. The zombies also utilize IP spoofing to avoid the replies back because the returning packets could also create congestion on their bandwidth.

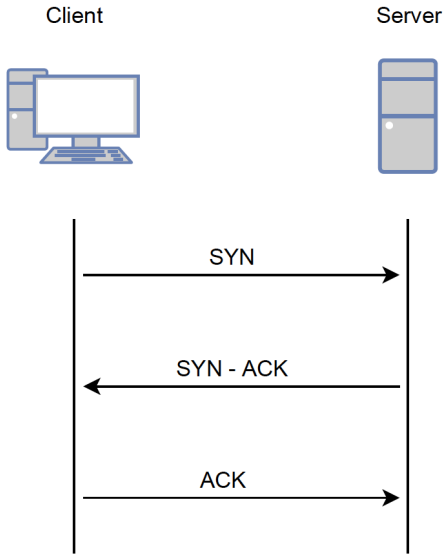

Figure 2.2. Legitimate TCP Handshake.

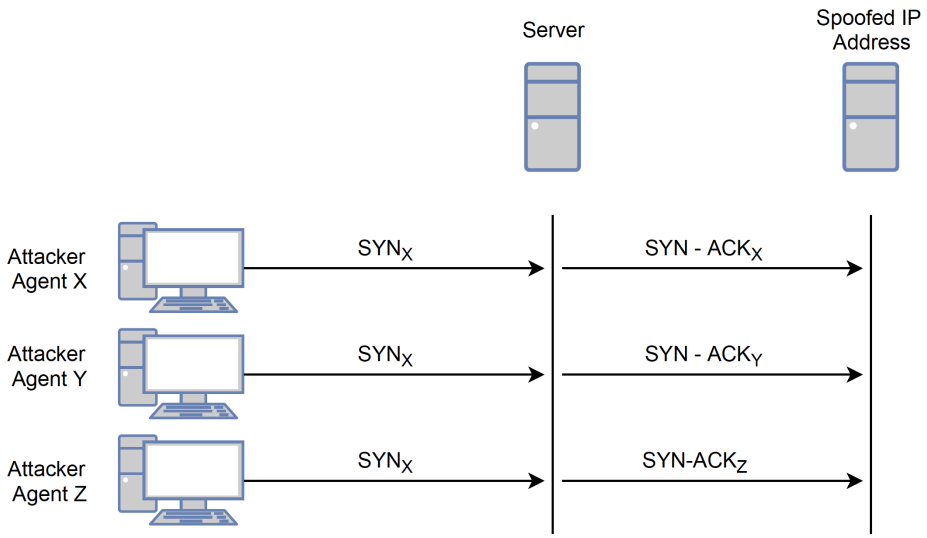

Figure 2.3. TCP SYN Attack.

2.1.1.4.2. ICMP Flood To discover network infrastructure, and calculate round-trip time between the nodes in the network, Internet Control Message Protocol (ICMP) packets are utilized. These are generally used for the management of network systems. The "Ping" process, which is generally used for controlling whether destination IP and ports are available or not, is carried out via this protocol. In these types of attacks,

bot computers are controlled by the attacker to ping the victim's server and get a reply from it. This communication could create a large volume of traffic and consume the victim's bandwidth. In this attack, IP spoofing is also utilized by the zombies to avoid from the replies back because the returning packets could also create congestion on their bandwidth. ICMP Flood attacks can be considered as both volumetric attacks and protocol attacks.

2.1.1.4.3. Smurf Attack A broadcast address is an IP address to which a computer can send a packet in order to communicate with all other computers on the same network [12]. For instance, when a new device or computer connects to a network, it makes an effort to configure an IP address. It must obtain an IP address from a remote service that distributes them. It does it by requesting a broadcast address. Any machine that sends a packet to a broadcast address effectively broadcasts the request to all hosts on the network, where hosts refer to all computers or network devices such as routers and servers. A chosen server assigns the newly connected device an IP address in response to the request. When an adversary transmits a faked ping packet to a broadcast address, the address can be misused. A ping packet is a network packet delivered between two machines to determine whether the remote system is powered on and capable of responding to network-based queries. When a host receives a ping packet, it responds with a ping-response packet to indicate that the receiving host is still alive. It is a helpful tool for auditing network connections by legitimate network administrators. When a ping packet is delivered from one device to a broadcast address, all active hosts in the network respond with ping-response packets. Smurfing is a technique in which an adversary spoofs its IP address and floods a broadcast address with ping packets. Additionally, the adversary changes the faked source IP address to that of a target machine to make the ping appear to originate from a legal target machine. When a faked ping packet is delivered to a broadcast address, all machines in the network respond with ping responses to the spoofed address. A sequence of faked ping packets used in this manner can generate a flood of ping-reply packets, rendering the legitimate target inoperable.

2.1.1.4.4. UDP Flood User Datagram Protocol (UDP) is a protocol that does not require a handshake process before the data transmission. Network data packets are just sent to the receiver's specific port, and they are processed. If there is no application working with the coming UDP packet, the receiver replies with a message that tells that this port is inaccessible. This message is sent via Internet Control Message Protocol (ICMP), and it could create a high volume of traffic on the attacker (zombie) side [13]. In order to avoid that, IP spoofing is used by the zombies to rotate this traffic to different systems and hide their identities.

2.1.1.4.5. HTTP GET and POST floods HTTP is the protocol used to connect clients to web servers on the Internet. HTTP GET is used to obtain data from a predefined resource; the HTTP POST method is used to send data to a server to create or update a resource. Without waiting for a response, attackers can simply bring down a web server by sending a continuous stream of HTTP GET or POST requests to the target. The server makes every effort to reply to each request but finally runs out of resources. These application-level attacks might be challenging to determine since the traffic seems to be valid HTTP GET and POST requests.

2.1.1.4.6. Low and Slow DDoS Attacks The purpose of Low and Slow DDoS attacks is to discreetly and secretly shut down application resources while consuming very little bandwidth. There are various forms of low and slow attack tools, all of which seek to maintain an endless monopoly on server resources. Slowloris, for example, works by creating hundreds of connection requests and maintaining each one open for as long as possible by slowly transferring data to the server before the connection times out. It utilizes so many server resources that the system finally becomes incapable of handling any new, valid requests. Because these attacks consume a negligible amount of bandwidth and occur at the application layer, where a TCP connection has already been established, the HTTP requests also seem legitimate.

2.1.1.4.7. ARP Poison Attack The address resolution protocol (ARP) connects a machine's MAC address to its IP address. The machine sends an ARP request, including its IP address, MAC address, and the destination machine's IP address. When the recipient computer receives this request, it caches the sender's IP address and MAC address. In an ARP Poison attack, the attacker modifies the cache and modifies the sender's MAC address [14]. It renders the sender's system inaccessible to any external use. Detection can be accomplished by analyzing the ARP protocol and determining whether or not the machine address is incorrect/bogus.

2.1.1.4.8. Back Attack The attacker attempts to crash the Apache web server in this attack by making requests with URLs that contain a large number of forward slashes ('/') [15]. This value can range between 6000 and 7000, and it is adequate to delay the server down momentarily. After the attack has ceased, the system may recover. This type of attack is detectable by scanning the URL and applying a 100-slash threshold.

2.1.1.4.9. Land Attack Specific older versions of TCP/IP implementations may be vulnerable to a Local Area Network Denial of Service attack [16]. Malformed faked TCP SYN, which is connection starting requests with the identical source and destination IP addresses as the packet are sent to a victim server in this attack [7]. It results in the target's machine continuing to operate while it sends replies to itself. It is an older exploit, and recent fixes should protect most systems.

2.1.1.4.10. Neptune (SYN Flood) Attack In TCP/IP implementations, the TCP server maintains a finite data structure that contains information about half-open connections [7]. If it becomes full, each connection has a timer that deletes the oldest record in the data structure to make room for subsequent connections to keep half-open connection information. Neptune utilizes this functionality to launch an assault on the TCP server. It sends an enormous number of connection requests to a TCP

server, causing the server to populate the half-connection data structure with a large number of entries. These messages are so enormous that the data structure fills up before the timer associated with the initial entry ends. Additionally, the attacker transmits IP-spoofed packets faster than the target machine can terminate the connection, resulting in memory exhaustion and system breakdown. As a result, any new valid connection is rejected, which is the primary goal of this attack.

2.1.1.4.11. Ping of Death In this form of attack, the ping command is executed and directed toward to victim system, which may exceed the IP protocol's maximum allowable size of 64K bytes [17]. The attacker sends a ping command with the maximum value allowed in the IP header's Fragment Offset field. As a result of reassembling, a massive packet larger than 64K bytes is created, which may cause the destination operating system's buffer to overflow. It may cause the operating system to slow down or even crash. POD is no longer effective, as modern operating systems are sufficiently secure to detect such attacks.

2.1.1.4.12. Smurf (ICMP flood) Attack The ICMP attack's property of replying to ICMP echo packets is abused in this attack [18]. The attacker makes ICMP echo requests to IP broadcast addresses using the victim's faked source address. Each machine that receives an ICMP request responds with an ICMP reply destined for the victim's system. It increases traffic flowing towards the victim's computer, which eventually results in a bottleneck at the victim's machine's interface.

2.1.1.4.13. Teardrop Attack This attack exploits a vulnerability in the technique for reassembling IP fragments. The reassembly function may fail when the router detects an error in the fragment offset field, such as packet data overlap. The fragment offset specifies the address of the packet's initial byte. The sum of this offset and the packet's length determines the fragment offset for subsequent fragments. Now, the attacker causes some inconsistency in the fragment offset values. For instance, if the

initial packet is 1000 bytes in length, the following fragment offset should be 1001, but the attacker will set it to 950, crippling the entire reassembly mechanism.

2.1.1.4.14. UDP Storm Attack This attack is carried out as follows: the attacker sends UDP packets to the first victim's echo port, posing as a second victim on the same network [18]. As the first victim is unaware of the faked address, it sends a reply with some data to the second victim's echo port. The second victim then responds with data to the first victim, and this loop continues until an external source of interference occurs. This traffic creates a bottleneck in the connection and may cause all genuine transmissions to slow down.

2.1.1.4.15. Peer-to-peer denial-of-service attack Peer-to-peer (P2P) file-sharing is a popular method of sharing and downloading files from other peers nowadays [7]. Attackers have developed an interest in P2P networks due to their enormous potential for malware distribution and DDoS attacks. P2P attacks are distinct from botnet attacks in that they do not interact with the clients. The attacker does not transfer traffic to the victim's network on its own but instead instructs peer-to-peer file sharing centers to disconnect from their present P2P network and connect to the victim's PC. The attacker can use this strategy to induce hundreds of thousands of machines to attack their behalf. All of these computers are connected to a file-sharing service and have their data flow redirected to the victim's computer. This massive influx (thousands of connections) devastates the victim network's connectivity, initially designed for hundreds of connections.

2.1.1.4.16. Permanent denial-of-service (PDoS) attack PDoS attacks are primarily directed at the victim's hardware system, including routers, printers, or other networking hardware. It exploits security holes in remote administrative access control. It corrupts or modifies the firmware of devices using an updated corrupted or damaged firmware image, resulting in significant damage that the hardware must be

fixed or replaced entirely. Because this is a hardware-targeted attack, it consumes fewer resources than a botnet attack. As a result, PDoS attacks are more dangerous and have attracted the attention of numerous hackers.

2.1.1.4.17. R-U-dead-yet (RUDY) attack The "R-U-dead-yet (RUDY)" attack generates traffic at a low rate, and volume [19]. As a result, typical defense measures have a tough time detecting such an attack. Numerous websites have a form for users to submit information. After submitting the information, users submit the form to the webserver. Typically, a submit request generates an HTTP POST request to the server. Typically, the client browser delivers this request to the server in one or two packets before disconnecting. However, the RUDY tool delivers genuine requests to the server via numerous packets by decompressing the data in a single packet, compelling the webserver to keep the many connections open. Each RUDY attack keeps all sessions occupied by sending continual long-form field submissions. Additionally, the attacker may send a large-sized header value, which keeps the session alive and unusable for other legal purposes.

2.1.2. Brute-Force Attacks

This attack type is a general approach used to crack passwords and get unauthorized access to hidden pages and content within a web application. It is essentially a hit-and-try attack, with the attacker eventually succeeding. Numerous tools are available for performing brute force password cracking assaults, including Hydra, Medusa, Ncrack, Metasploit modules, and Nmap NSE scripts. Additionally, there are other tools for password hash cracking, including hashcat and hashpump.

Patator is one of the essential tools used in brute-force attacks. Patator is coded in python and supports multi-core processing. It is more reliable and effective than other multi-threaded tools because it saves each response to a different log file for future analysis and supports over 30 different methods, including FTP, SSH, Telnet, and SMTP [20].

2.1.2.1. FTP-Patator. FTP is a well-known data transport protocol. It provides a connection between the hosts, which makes file transfer. There are two connections established between these hosts. While one of these connections is used to transfer data, the other is used to control the data transfer process. The control connection is made via port 21, while data transferring is conducted via port 20. The FTP server authenticates users via port 21. As a result, traffic to port 21 is examined to detect FTP brute force assaults. [21]

In this Python-based assault, the attacker attempts to guess the username and password in order to log in legitimately. In the generation of usernames and passwords automatically over a short period, the Patator tool is utilized [22].

2.1.2.2. SSH-Patator. SSH is a communication protocol that enables remote device access and management. It can be considered as the next generation of Telnet protocol. The primary distinction between telnet and SSH is that SSH employs encryption. TCP port 22 is generally utilized in SSH protocol [21].

SSH-Patator is a Python application that uses SSH to connect to a remote machine [23], similar to FTP-Patator. The attackers attempt to generate user names and passwords of generally admin accounts in a short amount of time in order to obtain remote access to control over a network or a device. The first step of the attack is to determine the hosts using SSH protocol. In this stage, generally, the server is scanned over for specific IP blocks or subnets of a network. Then, a brute force attack starts to get access to the system.

2.1.3. Port Scan

Port scanning is a technique attackers employ to scope out their target environment. Hackers perform this attack by sending packets to specified ports on a host and analyzing the responses for vulnerabilities and information about the services and versions running on the host.

Attackers must first locate hosts on the network, and then they scan the hosts' ports to detect weaknesses. Port scanning generally attempts to classify the responses into three categories. When a victim device replies with a packet, it indicates this port is open and used in communication. When there is no packet as a reply, it means that this port is closed or filtered. While there is no service listening to the port in close ports, there are firewalls or services listening to this port that only accept packets with a specific format in filtered ones. Nmap is one of the most common tools mainly to perform a port scan on a network [24].

2.2. Network Intrusion Detection Methods

Network intrusion detection algorithms are developed to detect malicious activities quickly and accurately to protect a network from damage. It has become a critical research area as Internet technologies have advanced, and intruders have developed new and sophisticated attack strategies. Even though numerous studies have been published on various network attacks, there is always a need for improved intrusion detection algorithms to prevent emerging threats.

The denial of service (DoS) attack is one of the most well-known types of network attack. The intruder's objective is to make a computer or other device inaccessible by interfering with the device's regular operation. It is accomplished by flooding the victim machine with requests, causing it incapable of responding. When an attacker employs a considerable number of infected machines to carry out the attack, it is referred to as a distributed denial of service (DDoS) attack. Different forms of DDoS attacks target specific layers. For example, HTTP flood attacks are performed on the application layer, while SYN flood attacks are on the transport layer. Attack detection and countermeasure mechanisms against these attacks are evolving day by day. An overview of existing network attack detection mechanisms is given in [25], and [26]. Signature-based attack detection algorithms can be used to detect DoS/DDoS attacks. These type of algorithms requires training data to detect an attack. One of the robust signature-based attack detection algorithms is proposed in [27]. Machine learning-based system designed to detect DoS and DDoS attacks. In this work, there are 33

extracted variables for each flow. In order to reduce the number of the used features, the proposed feature selection algorithm is applied, and eventually, 20 feature is used in the algorithm. Entropy, coefficient of variation, quantile coefficient, and rate of change, which are commonly used metrics in anomaly detection, are also utilized in the detection algorithm. The designed system is evaluated with the different recent attack datasets. These datasets are utilized to generate attacks and regular traffic on the network test environment. Statistical methods are widely used in DDoS attack detection algorithms, such as covariance analysis [28]. This article proposes a primary DDoS attack detection method that uses multivariate correlation analysis. Specifically, SYN flooding attacks are detected using covariance analysis with linear complexity. This method uses the correlations among the TCP flags, which give information about network characteristics. During the SYN flooding attack, the number of SYN flags does not match with the FIN flags, while it is generally matched in the typical case. This situation is detected with the time-series calculation of the covariance value of each pair of the TCP flag in a practical manner. Additionally to statistical features, entropy-based features are widely used in the DDoS attack detection [29]. These features reveal changes in the network when the attack is started. The main difference of this work is that a large number of entropy-based features are used to detect different types of DDoS attacks. High and low-density DDoS attacks are detected thanks to these differentiated features. In the detection phase, different machine learning algorithms are used to separate normal and abnormal behavior in the network. ISCX 2012 and 1998 Darpa datasets are used to validate the proposed algorithm. Another work that is based on entropy is given in [30]. DDoS attacks that exploit the HTTP protocol in the cloud environment are analyzed and detected in this work. The HTTP protocol is commonly used in cloud environments due to its fundamental nature. It makes it easier to access services on the cloud with less transmission cost. However, it makes vulnerable the cloud services to HTTP DDoS attacks. HTTP DDoS attacks is a significant threat to both web services and the cloud environment. It can be executed as a flood, and it can be executed with a low number of requests in a slow way to avoid detection. Another algorithm that uses the various information-theoretic metrics is proposed to detect HTTP DDoS attacks in the cloud environment [30]. The entropy of the packet

header information such as source and destination IP address, port numbers, network protocol, and the number of bytes is used to characterize network traffic. The use of the entropy is modified according to the nature of the DDoS attacks in HTTP protocol. Different machine learning classifiers such as Random-Forest, Naive Bayes, Multilayer Perceptron, Decision Tree, and K-Nearest Neighbor algorithms are tested in the classification stage. The best result is achieved with the Random-Forest algorithm on the CIDDS-001 dataset.

DDoS attack detection problems can be evaluated in separate domains. Extracted features and the collected parameters could be more effective in distinguishing DDoS attacks and the legitimate traffic on the different domains. Frequency domain analysis could be used to get more efficient solutions [31]. This work utilizes discrete Fourier transform and discrete wavelet transforms to transform time-domain features to the frequency domain. It is seen that DDoS attack samples have main energy components on low-frequency levels while the normal samples have a more uniform distribution on the frequency domain. There is no gathering on some specific frequency levels for the usual traffic. In the detection stage, two primary approaches are evaluated. The traditional thresholding is compared with the Naive Bayes classifier. It is seen that the Naive Bayes classifier gives better results than the thresholding. Regular traffic in the Boğaziçi Dataset [32] and the actual attack traffic in the booter dataset [33] are used for the performance evaluation.

The Neural Network algorithm is a tool that can be used in both feature extraction and classification stages of the anomaly detection systems. Auto-encoders and self-organizing maps are widely used to detect anomalies in network systems [34–37].Self-Organizing Maps, which summarize high dimensional data as low dimensional data, can be utilized to extract the nonlinear statistical relations between the high dimensional complex data points. The specialty of the Self-Organizing Maps can be used to detect anomalous in the network traffic [34]. In this work, network connections are characterized by using six parameters. Then the network is represented in two-dimensional space. The algorithm is tested in Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), and HyperText Transfer Protocol (HTTP) traffic. Another early

work with used neural networks is given in [35]. In this work, the neural network is utilized to detect network-based anomalies. Initially, statistical preprocessing is applied to the network data then neural network algorithms are applied. Perceptron, Backpropagation (BP), Perceptron-backpropagation-hybrid (PBH), Fuzzy ARTMAP, and Radial-based Function algorithms are applied as neural network algorithms, and they are compared in this work. Low-intensity UDP flooding attacks are detected with this scheme, and high accuracy results are obtained.

Dimension reduction methods can be utilized to extract features in classification problems. Pre-calculated features are given as an input for this dimension reduction algorithm. Then the data is represented on a different subspace which provides a more effective solution. Auto-encoder can be used to get lower-dimensional data from the higher one by exploding the nonlinear relationships among the features [37]. In this work, a convolutional auto-encoder-based network anomaly detection algorithm is proposed. The performance of this algorithm is compared with the conventional auto-encoder and principle component analysis algorithm, which is one of the most common dimension reduction algorithms. Authors perform their evaluations using NSL-KDD Dataset. Another algorithm based on autoencoder models is given in [36]. This work proposes an algorithm that depends on the two-deep learning algorithms to detect a zero-day attack that is not encountered yet. Denoising autoencoder and deep learning algorithms are utilized in the network modeling stage. Reconstruction distribution (RE) is defined as a difference between inputs and the outputs of algorithms. These algorithms are trained with the labeled data, and the reconstruction distributions are obtained for the ordinary and anomalous data samples to model the network traffic behavior. The anomaly threshold is calculated using the proposed stochastic model. It is important to remark that this threshold is determined as the boundaries of the normal data samples, and it enables to detection of zero-day attacks. NSL-KDD dataset is used in the evaluation part. The test data in the NSL-KDD dataset contains 19 different attack types than its training data, and it enables the evaluation of the algorithm's zero-day attack performance. Both algorithms give relatively close accuracy results of about 88%.

Convolutional Neural Network (CNN) is one of the important tools used in network anomaly detection systems. There are some contemporary applications of CNNs to analyze network traffic [38, 39, 39, 40] and classifies the malware on the systems [41–44].Designed Lightweight Usable CNN in DDoS Detection (LUCID) algorithm [44] exploits CNN's properties to reach high detection rates by using a small number of features on the classifier. This algorithm is proposed to detect DDoS attacks with a low computational cost in an online environment. The attack detection algorithm has mainly two parts. In the first part preprocessing of network traffic is executed. In this stage, 11 features are extracted for each packet on the network. Then the input is prepared for the main algorithm with normalization and padding. 2-D matrix is generated from the packets for each flow on the network. In the latter part, basic LUCID architecture is utilized to classify the network flows. The cross-entropy approach is used as a cost function of the binary classification problem. At the end of the algorithm, flow is labeled as normal or anomaly. The performance evaluation of this algorithm is executed with the ISCX2012, CIC2017, and CSECIC2018 datasets. In each dataset, samples are selected to create a new balanced dataset. UNB201X is utilized in the validation process.

A genetic algorithm is another tool that is used in network anomaly detection systems [45].In this work, a flow-based anomaly detection algorithm is proposed. The network characteristics are obtained via a genetic algorithm that analyzes the network flows and generates the Digital Signature of Network Segment using Flow (DSNSF). It enables to predict network behavior for coming time windows and compare the actual network traffic with the expected one. In the comparison stage, a threshold is determined using an Exponentially Weighted Moving Average (EWMA) algorithm, which is proposed in [46]. It enables the weighting of the previous observation according to passing the time over the sample. In the decision stage of the algorithm, the Fuzzy Logic scheme assigns a network sample anomaly score between 0 and 1 instead of a binary assignment. The dataset obtained from the University Campus Network contains three types of network anomaly. These are Denial of Service(DoS), Distributed denial of service (DDoS), and Flash crowd anomalies. High accuracy rates are obtained by using the proposed algorithm as 96.53%.

Network anomaly detection algorithms generally use packet headers to detect network attacks. Besides that, the payload can be analyzed to detect anomalies in the network.PAYL (PAYLoad intrusion detection system) proposed by Wang and Stolfo can be considered as one of the fundamental algorithms based on packet payloads [47].PAYL performs n-gram analysis, a widely used tool in text classification. The system identifies anomalies by integrating a one-gram analysis approach with combining payload length clustering method. The model is generated using a 1-byte sliding window to calculate relative frequency values of 1-gram sequences, which directly correspond to the bytes. There are 256 possible byte values in a payload, and a histogram of these values is generated with the calculated relative frequency values. Individual models are created using these histograms for each payload length in the training stage. It employed a simplified Mahalanobis distance metric to compare new incoming traffic to the model.

PAYL is a 1-gram approach, and it is not able to make a structural analysis of a payload because the relative positions of the bytes are ignored in this approach. To make a structural analysis evaluating bytes' relative positions, a higher-order n-gram approach (n¿1) must be used. Wang and Stolfo propose ANAGRAM to improve the performance of PAYL [48]. Anagram is based on the storage of separate n-grams observed in normal network traffic in a bloom filter while storing individual n-gram values from known malicious signatures. The payload of a new incoming packet is compared with pre-modeled n-gram values of the Bloom filter values. An alarm is generated if there is a significant difference with the normal n-gram values.

POSEIDON proposed by Bolzoni [49], a two-tier anomalous intrusion detection system based on payloads. A self-organizing map or SOM integrates the payload length and relative frequency distribution. SOMs were utilized to preprocess packet payloads in the POSEIDON architecture, whereas PAYL was employed as the base of malware detection. SOMs were used to map high-dimensional data points to a single or multiple-dimensional grid. The SOM's objective was to determine payloads comparable to a given destination address and port. Self-Organizing Maps are used to increase the sensitivity of detection.

Another work that utilizes packet payload is presented by Hubballi [50] and a score-based model is proposed to detect anomalies that maintain the n-grams from the training dataset together with their associated frequencies. A clustering technique is used to construct several frequency bins, and each bin is assigned a score. Each n-gram from the test packet discovered in a given bin will receive the bin's score. A scoring function is proposed for calculating the packet's score, which is the sum of the scores for each n-gram.

OCPAD is proposed in this work as a mechanism for detecting network packets having suspicious payload content [51]. The suggested technique combines the advantages of single-class classification with information on the frequency of short sequences. As an anomaly detector, a one-class Multinomial Naive Bayes classifier is utilized in order to identify HTTP threats. OCPAD determines a packet's degree of maliciousness by calculating the likelihood of each short sequence occurring in the content of known legitimate network packets. OCPAD estimates the likelihood range for each sequence's occurrence from each packet during the training phase. A unique and effective data structure called Probability Tree to hold the likelihood range of these sequences is proposed. It considers a brief sequence, being anomalous during the testing phase if it is not discovered in the database or if its likelihood of occurrence in a packet significantly deviates from the range determined during the training phase. It provides a class label for a test packet based on the likelihood of short anomalous sequences. Performance evaluation is done with a large dataset of 1 million HTTP packets gathered from an academic network, demonstrating that OCPAD has a high Detection Rate (up to 100

3. FUNDAMENTAL CONCEPTS IN INFORMATION THEORY

3.1. Entropy

Entropy is one of the fundamental concepts in the information theory since introduced by Claude Shannon in 1948 with the paper "A Mathematical Theory of Communication". Entropy can be defined as a measure of information in the information theory. In communication, more uncertainty in the bits means there is more information. Entropy increases as the uncertainty of a random variable increases, and it shows there is more information. When the entropy equals zero, the random variable does not contain any information because there is no uncertainty. For any discrete random variable X, Entropy $H(X)$ can be defined as

$$H(X) = -\sum_{i=1}^{n} P(X = x_i) \log \big(P(X = x_i)\big), \tag{3.1}$$

where the x_i is possible value of a random variable X which can take, $P(X = x_i)$ corresponds to the probability of random variable X is equal to x_i. $P(X = x_i)$ can be written as $P(x_i)$ and it is probability mass function which takes always positive values.

3.2. Kullback-Leiber Divergence

Kullback-Leiber(KL) divergence is known as relative entropy, which measures the distance between probability distributions of two random variables. KL divergence is also known as KL distance because it is a measure of the similarity of two probability distributions with equal length. Kullback-Leiber distance between distributions of random variables P and Q can be defined as

$$D_{KL}(P||Q) = -\sum_{\in} P(i) \log \frac{Q(i)}{P(i)}, \tag{3.2}$$

where X and Y are discrete random variables defined on the same sample space X. Equal length and the same sample space requirement come from the nature of the logarithm. A logarithm is not defined for zero. Therefore, there can be no undefined values for one of the probability distributions.

3.2.1. Basic Properties of Kullback-Leiber Divergence

One of the fundamental property of the Kullback-Leiber divergence is that KL distance never takes negative values and

$$D_{KL}(P||Q) \geq 0. \tag{3.3}$$

This inequality comes from Gibbs inequality, and it can be written as

$$-\sum_{i=1}^{n} P(x_i) \log P(x_i) \leq -\sum_{i=1}^{n} P(x_i) \log Q(x_i), \tag{3.4}$$

where P and Q are random variables which are defined on the same sample space. In other words, entropy of the random variable is always less than cross entropy with another random variable with the same sample space. The equality holds if and only if

$$P(x) = Q(x) \ \forall x. \tag{3.5}$$

It is condition for the equality in both Gibbs inequality given in (3.4) and (3.3).

Another important point is that KL distance is an non-symmetric metric and

$$D_{KL}(P||Q) \neq D_{KL}(Q||P). \tag{3.6}$$

The Kullback-Leibler distance is frequently used in different areas to compare different probability distributions. However, the requirement to have the same sample space

makes it challenging to use Kullback-Leibler in practical life. Even if it is used in real-life applications, samples with zero probability create a problem for this metric. Besides that, it is impossible to calculate the relative distance between the two probability distribution, which are defined on the entirely different sample spaces with different sizes. Vidyasagar proposed a metric for the upper bound of the Kullback-Leiber distance in 2012 [4]. The metric, based on the Greedy algorithm, is introduced in the paper as "A metric between probability distributions on finite sets of different cardinalities and applications to order reduction".

Before moving to the derivation of the metric, it is essential to define the mutual information concept.

3.3. Mutual Information

Mutual information can be defined as a measure of the relationship between two random variables with the same sample space. In other words, it measures how much unique information contains both random variables.

The formal definition is given as

$$I(X,Y) = H(X) + H(Y) - H(X,Y), \tag{3.7}$$

where $H(X,Y)$ is joint probability and it rewritten as

$$H(X,Y) = H(Y) - H(Y|X) \tag{3.8}$$

$$= H(X) - H(X|Y). \tag{3.9}$$

Mutual information is a symmetrical measure and

$$I(X,Y) = I(Y,X). \tag{3.10}$$

Mutual information can be written in another way by using the Kullback-Leiber divergence. Instead, mutual information is the KL distance between the joint probability of two random variables and the product of these variables. Then it can easily be redefined as

$$I(X,Y) = D_{KL}(P_{XY}(X,Y)||P_X(X)P|Y(Y)). \tag{3.11}$$

3.4. Derivation of the Metric - Greedy Algorithm

This metric is related to the Kullback-Leiber distance's upper bound between two random variables. To derive a metric, initially, the problem is identified.

3.4.1. Computing the Metric - Greedy

The computation of the greedy metric is started with the defining problem. The problem can be defined where ϕ and psi are random variables defined on S_n and S_m, respectively. S_n and S_m are two different sample spaces that could also have different cardinalities, m, and n. Let's define the problem for the case $n > m$. The problem is whether the combination of ψ can constitute ϕ or not. In other words, what is the optimal bin allocation in ϕ to fill ψ. In this bin-packing problem, overstuffing is also allowed. Instead, both problem definitions come to the same purpose, and they are NP-hard problems. A greedy algorithm (sometimes called "best-fit algorithm" in computer science) is generally used to solve these problems. Then, the solution can be found with this greedy algorithm. The computation procedure is given below step by step.

(i) Set $s = 1$, where s is the round counter. Define $n_s = n$, $m_s = m$, $\phi_s = \phi$, $\psi_s = \psi$.

(ii) Place each element of ϕ in the bin with the largest unused capacity. If a particular component $(\phi_s)_i$ cannot be assigned to any bin, the index i should be assigned to an overflow index set Ts.

(iii) When all elements of ϕ_s have been processed, define $I_1^{(s)}, ..., I_{m_s}^{(s)}$ as the indices from $\{1, ..., n_s\}$ which have been assigned to the various bins, and let T_s represents the set of indices which are not assigned to any bin. If $|T_s| > 1$ proceed to Step 4; otherwise proceed to Step 5.

(iv) Let $\alpha_1^{(s)}, ..., \alpha_{m_s}^{(s)}$ is the unutilized remaining capacities of the m_s bins, and define $\boldsymbol{\alpha^{(s)}} = [\alpha_1^{(s)} ... \alpha_{m_s}^{(s)}]$. Then the total capacity that is not used $c_s := \boldsymbol{\alpha^{(s)}} e_{m_s}$ satisfies

$$c_s = \sum_{j=1}^{m_s} \alpha_j^{(s)} = \sum_{i \in T_s} (\phi_s)_i. \tag{3.12}$$

Since each $(\phi_s)_i$, $i \in T_s$ does not fit into any bin, it is clear that $(\phi_s)_i > \alpha_j^{(s)}$, $\forall i, j$. In turn this implies that $|T_s| < m_s$. Next, set $n_{s+1} = m_s$, $m_{s+1} = |T_s|$, and define

$$\phi_{s+1} = \frac{1}{c_s} \boldsymbol{\alpha^s} \in S_{n_{s+1}}, \psi_{s+1} = \frac{1}{c_s} [(\phi_s)_i] \in S_{m_{s+1}}. \tag{3.13}$$

Proceed to Step 2 by increasing the counter by one.

(v) When this step is reached, $|T_s|$ is either zero or one. If $|T_s| = 0$, then it implies that ψ_s is a exact aggregation of ϕ_s. Therefore, define $V_s = 0$ and proceed as follows. If $|T_s| = 1$, then only one element of ϕ_s, denoted by the variable $(\phi_s)_k$ cannot be placed into any bin, and the remaining component $(\phi_s)_k$ must equal c_s. So let

$$\begin{aligned} v_s &= \frac{1}{c_s} \boldsymbol{\alpha^s} \in S_{m_s} \\ V_s &= c_s H(v_s) \\ U_s &= V_s + H(\phi_s) - H(\psi_s). \end{aligned} \tag{3.14}$$

Define $P_s \in S_{n_s \times m_s}$ by

$$p_i = b_j \quad if \;\; i \in I_j^s, \;\; p_k = v_s, \tag{3.15}$$

where b_j is the j^{th} unit vector with m_s components. Then V_s is the minimum value of $J_{\phi_s}(.)$, and P_s achieves that minimum. Next define $Q_s \in S_{m_s x n_s}$ by

$$Q_s = [diag(\psi_s)]^{-1}]P_s^T diag(\phi_s). \tag{3.16}$$

Then it follows that Q_s minimizes $J_{\psi_s}(.)$, and that U_s corresponds to the minimum value.

(vi) In this step, we invert all of the above steps by transposing Q_{s+1}, applying the transformation in (3.16), and embedding the resulting matrix into P_s. We also correct the cost function using (3.14). Decrease the counter s by one and recall that $m_s = n_{s+1}$. The unutilized capacity c_s which is defined in (3.12) which has been found during the forward iteration, and define

$$V_s = c_s U_{s+1}, U_s = V_s + H(\phi_s) - H(\psi_s). \tag{3.17}$$

Define $P_s \in S_{n_s x m_s}$ by

$$p_i = b_j \; if \; i \in I_j^{(s)}, p_i = i^{th} \text{ row of } Q_{s+1}. \tag{3.18}$$

If $s = 1$, halt; otherwise repeat the step.

After these steps, we compute an upper bound on the distance metric between the two probability distributions which is

$$d(\phi, \psi) \leq V_1 + U_1. \tag{3.19}$$

This metric is calculated as a function of ϕ and ψ, which are two random variables with their own sample spaces. It is important to remind that there is no restriction for the sample spaces. Computed distance will be shown as $d_{greedy}(\phi, \psi)$. The metric based on the Greedy algorithm will be used as "Greedy" for the rest of the thesis.

3.5. Maximum Possible Entropy

Entropy definition is given in Section 3.1. In this entropy definition, mainly the probabilities of the samples are utilized. However, frequencies of the samples are utilized to calculate entropy of a sequence .

Let X is defined on the set $\{x_1, x_2, ..., x_i, ..., x_n\}$. $p(x)$ is defined for $x \in X$ and $p(x_i)$ corresponds to the probability of x_i. Then, Shannon entropy for X can be defined as

$$H(X) = -\sum_{x \in X} p(x) \log(p(x)). \tag{3.20}$$

Although different logarithm of bases can be used in the calculation, base 2 and natural logarithm "ln" in which logarithm of the base is Euler's number "e" is the general selection for this base. Base convention of entropy can be easily done with using $H_b(X) = \log_b(a)[H_a(X)]$.

Expected value or average of a random variable is determined with the expectation function $E[]$ and mathematically entropy can be shown as $E[-\log(p(x))]$. It is important to emphasize that entropy is not a function of the elements of a random variable, but it is the function of the probabilities of these symbols. However, the frequency of each element is used as an estimation of the probability of the element instead of real probability values. Then, entropy for a random variable X can be calculated as

$$\hat{H}(X) = -\sum_{x \in X} f(x) \log(f(x)), \tag{3.21}$$

where $f(x)$ is corresponding frequency in a sequence for value x which belongs to previously defined for random variable X. In order to modify this formula for network packet payloads, it is required to define some basic variables.

Let S be the network packet's payload sequence which contains ASCII numbers for each byte. The length of the sequence S is represented by N, and it is equal to the length of the payload. The maximum possible value of N is equal to the maximum length of the payload in a network. Each symbol in a payload corresponds to the ASCII numbers and can be considered as an alphabet $\mathcal{A} = \{0, 1, 2, ..., 255\}$. The length of the alphabet is equal to 256 for ASCII numbers, and it can be represented with m. Entropy for this problem can be redefined as

$$\hat{H}_N(S) = -\sum_{x \in \mathcal{A}} f(x) \log(f(x)), \tag{3.22}$$

where $f(x)$ corresponds to the frequency of ASCII number x in payload. $\hat{H}_N(S)$ converges to $H(S)$ in the case the N which is the length of the sequence S (payload length of network packet) goes to $+\infty$ [52]. It is known as asymptotic equipartition property theorem. It can be proven with Weak Law of Large Numbers by Bernoulli. Asymptotic equipartition property theorem and its proof is formalized as follows.

Theorem 3.1 (The Weak Law of Large Numbers [53]). *If X is independent and identically distributed (i.i.d.) random variable with mean $\mu = E[X]$ and given $X_1, X_2, ..., X_i$ is sequence of these variables with the length N. Lets define*

$$\overline{X} = \frac{1}{N} \sum_{i}^{N} X_i, \tag{3.23}$$

and ϵ is small positive number $\epsilon > 0$. Then,

$$\lim_{N \to \infty} P(|\overline{X} - \mu| \leq \epsilon) = 0. \tag{3.24}$$

Theorem 3.2 (Asymptotic Equipartition Property Theorem [54]). *If $X_1, X_2, ..., X_N$ are independent and identically distributed (i.i.d.) random variables with probability $p(X)$ and*

$$-\frac{1}{N} \log p(X_1, X_2, ..., X_N) \to H(X) \quad \textit{in probability.} \tag{3.25}$$

Proof. Functions of independent random variables are also independent random variables. Then, $\log p(X_i)$ is identically distributed independent random variable because X_i are i.i.d.. Then using Theorem 3.1,

$$-\frac{1}{N}\ \log p(X_1, X_2, ..., X_N) = -\frac{1}{N}\sum_{i}^{N}\log(p(X_i)) \tag{3.26}$$

$$\rightarrow -E[\log(p(X))] \quad \text{in probability} \tag{3.27}$$

$$= H(X). \tag{3.28}$$

can be obtained for large values of N. Then,

$$H(x) = -\sum_{x\in X} p(x)\log p(x) = -E[\log p(x)]. \tag{3.29}$$

□

If random variable X has uniform distribution, probability $p(x) = \frac{1}{N}$ With using the Theorem 3.1 and 3.2, $\hat{H}_N(X) = H(X)$ for large values of N. It can be formalized as

$$\lim_{N\rightarrow+\infty} \hat{H}_N(X) = -f(x)\log f(x) \tag{3.30}$$

$$\lim_{N\rightarrow+\infty} \hat{H}_N(X) = -\frac{1}{N}\log p(x) \tag{3.31}$$

$$\lim_{N\rightarrow+\infty} \hat{H}_N(X) = H(X), \tag{3.32}$$

where $f(X_i)$ is frequency of random variable calculated from the occurrence of the random variable in given sequence.

Entropy is calculated for payload sequence S, and $\hat{H}_N(S)$ is obtained to modify these theorems to our case. Payload sequence contains elements from $\mathcal{A} = 0, 1, 2, ..., 255$ with length 256. There are 256 unique ASCII numbers and in the case of uniform distribution probability of each number is equal to $\frac{1}{256}$. For a sufficiently long payload

sequence, calculated entropy $\hat{H}_N(S)$ converges to $\log(256) = \log(2^8) = 8$. The approximation is still valid in the case that N is much higher than the length of alphabet 256 ($N \gg 256$).

Calculated entropy for given payload sequence S with length N, $\hat{H}_N(S)$, converges to $H(S)$ when N goes to infinity. In the network environment, there are some limitations to the payload sizes of the network packets. It mainly depends on the network protocol used in the transmission.

Monte-Carlo simulation is a mathematical approach that relies on repeatedly randomizing samples to derive final findings [55]. All steps of the Monte-Carlo simulation are repeated by computer, significantly reducing time and effort. Monte-Carlo simulation is frequently used in physical and mathematical system simulations. This method is particularly ideal for computer calculations when an accurate answer cannot be obtained using a deterministic algorithm. It can be used to create models of phenomena that have ambiguous inputs. Monte Carlo simulation will perform a large number of iterations to get maximum entropy values by varying the length of the payload within a suitable range in order to determine how the effect of length on maximum entropy. It enables us to analyze the variation of $\hat{H}_N(S)$ from $H(S)$. In this simulation, packet payloads are generated with uniform distribution for various payload lengths. Then, each case is iterated 1000 times, and the following graph is obtained. Payload length varies from 1 to $2^{16} = 65536$. However, most network packets cannot reach the upper limit and there are some limits on the calculation of the proposed features, therefore, the variation analysis is done up to 10000.

It is seen that the bias between $\hat{H}_N(S)$ and $H(S)$ is small in some regions. In these regions, $N \ll m$ and $N \gg m$. To express these region numerically, regions can be defined where $N < 32$ and $N > 4096$. It means that the gap between $\hat{H}_N(S)$ and $H(S)$ is higher to use $\hat{H}_N(S)$ without bias correction for the region $32 < N < 4096$. The maximum possible entropy encountered in a payload or network packet can be obtained by using the curve in the Figure.

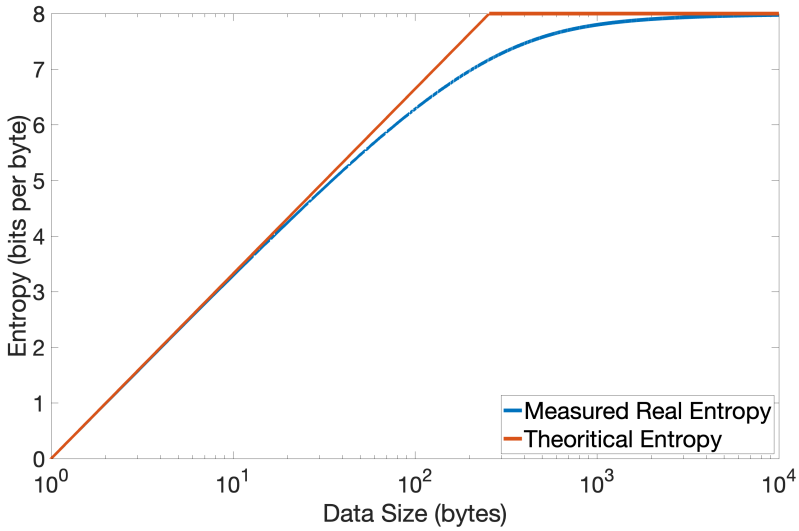

Figure 3.1. Comparison of Measured Entropy and Theoretical Entropy.

4. FLOW BASED PAYLOAD FEATURES

In a network system, each connection can be considered a flow. The flow starts with the first packet, which is sent from a specific source to a specific destination. Then, network packets with the same source IP, source port, destination IP, and destination port can be considered as a flow. Moreover, these packets must be sent with the same protocol. It means flows contain network packets with the same 5-tuple information. (Source IP, Source Port, Destination IP, Destination Port, and Protocol.) However, network packets that are sent at different times cannot be considered a flow even if they have the same 5-tuple information. In the flow definition, network packets must be sent at a pre-defined time period after the first packet is sent. It is similar to the time, which is when the connection is alive. In network intrusion systems, flows are often used to look at the characteristics of these connections because each connection is usually set up for a specific reason.

4.1. N-Gram Payload Feature Extraction

The N-gram analysis method was first introduced by Damashek in 1995 to analyze text independently from its language [56]. It is mainly developed for natural language processing (NLP) to analyze the characteristics of languages. N corresponds to the number of elements in the analyzed sequence. This element could be a character or a word in language processing. Except for zero, the value of N could be any of the natural numbers. It is referred to as a unigram when N equals one. Each sequence contains only one character or word, and the probability and other features of the sequence are evaluated. In the case of N = 2, it is called a bigram, and each sequence contains 2 elements. The probability and the other features of the sequence are utilized to analyze the whole text. Sequences are created by using a sliding window with the "N" width, and then the probability of the sequences is calculated for the entire text.

This method is also widely used in the network security area to extract the characteristics of the payload vectors [47–49].

The PAYL system, a 1-gram-based payload anomaly detector (PAYL), was proposed by Wang and Stolfo [47]. The system identifies anomalies by integrating a one-gram analysis approach based on packet payload data clustering. The system makes use of a collection of models; each model incrementally keeps the results of the 1-gram analysis for packets of the same length; hence, each payload length has a unique model. Each model maintains two data series: the mean byte frequency (i.e., relative byte frequencies that span several payloads of length l) and the standard deviation of the byte frequency for each n-gram sequence. In unigram, it corresponds to each byte value in the payload. Same variables are calculated for arriving packets in the prediction stage, and then it is checked with the model parameters and the model parameters to see whether there is any significant deviation from the model parameters. If there is, the final decision will be made for this package.

POSEIDON, which is proposed by Bolzoni, is a modified PAYL architecture-based system [49]. PAYL takes advantage of the data length variable to determine the appropriate model. In comparison, POSEIDON classifies network packets during the pre-processing stage using a neural network. The authors use Self-Organizing Maps (SOMs) [57] to create unsupervised clustering. To begin with, the SOM analyzes the entire packet payload, and the most similar neuron is given as an output. As with PAYL, this neuron network is used to figure out the frequencies of each byte and the standard deviation of the data.

As a general use of the N-Gram analysis for the payload analysis, the payload is processed using a sliding window of width N, and the number of occurrences of each n-gram sequence is collected. Let D_η be the dictionary containing all unique N-gram sequences for any given $N = \eta$ value. Length of D_η equals m, and it can be defined as

$$D_\eta = \{d_1, d_2, ...d_i, ...d_m\}, \tag{4.1}$$

where d_i corresponds to a unique sequence in the n-gram analysis. Create a new payload sequence for $N = \eta$ value from the original payload sequence using these unique elements in D_η. This new payload sequence is used in the feature extraction stage to

extract these features. While some of the proposed features use the payload sequence directly, some of them use the probability distribution of the payload sequence. To generate a probability distribution for a payload sequence with $N = \eta$, relative frequencies for each unique element in D_η are obtained.

The occurrence of the d_i can be represented with the O_i, and the relative frequency of the d_i is f_i can be obtained with the expression

$$f_i = \frac{O_i}{\sum_{k=1}^{m} O_k}. \tag{4.2}$$

PD_η refers to a probability distribution of a payload which is extracted for $N = \eta$ with m length and it can be defined as

$$PD_\eta = \{f_1, f_2, ...f_i, ...f_m\}. \tag{4.3}$$

After the generation of the probability distribution from the payload for predetermined N values, calculations for proposed features can be easily made. In this work, N-gram models is generated for $N = 1$ (unigram) , $N = 2$ (bigram) and $N = 3$ values.

The size of the probability distributions mainly depends on the unique sequences which are obtained by the N-Gram Models. In the unigram case, each byte directly corresponds to a sequence. The maximum possible length of a probability distribution is equal to the maximum possible number of unique bytes. The payload can contain values in the range 0-255 then the length of the probability distribution could be equal to 256. However, for payloads with less length or fewer unique bytes, this distribution length varies. Similarly, bigram case in which N=2, the maximum number of possible unique sequences equals $256 * 256 = 2^{16}$. In the $N = 3$ case, the number of possible unique sequences is increased to $256 * 256 * 256 = 2^{24}$. It is much higher than the unigram case, and the complexity of the process is increased. However, the obtained probability distribution of sequence is generally lower than this number because only unique sequences with a higher probability than zero are considered in the proposed

calculation methods.

N-gram analysis is used in our work to characterize network flows more effectively and to extract more information from the payload sequences. New payload sequences and corresponding probability distributions are obtained for $N = 1$, $N = 2$, and $N = 3$ values, and each feature is calculated for each case separately. It expands the feature space and enables us to catch more complex patterns in the payload sequences.

4.2. Entropy

The information is transferred with the payload in the network packets, which consists of bytes that can take values in the range of 0-255. By using these byte values for each packet, histograms can be generated. In this process, byte values can be evaluated individually or as groups of 2 and 3 in N-Gram models. In our work, byte distribution histograms are extracted using N-Gram models, which are mainly used in NLP processes. Then, probability distributions for each packet are obtained by normalizing these histograms. These distributions reflect the characteristics of the payload of the packet. It contains information about how they are produced. Different payload vectors with different byte distribution characteristics are generated for different applications and attacks. Besides that, the packets which are produced by the bots in a network attack also show some different attributes than the legitimate packets.

Entropy is one of the most important information theory metrics which measures the randomness in a probability distribution. The randomness in the attack traffic is often different than the legitimate traffic, and the randomness level of the attack flows are similar to each other. Entropy is commonly used to detect encrypted traffic in the network [58, 59]. The higher entropy levels indicate encrypted payload on the network because of randomness in the nature of the encryption. There are some proposed algorithms to perform application classification [60–62] and to detect some specific applications [63]. Besides that, payload entropy is utilized in the detection of some kind of malicious activities such as Botnets [64] and Malware in HTTP traffic [65].

Payload entropy can be easily calculated for the probability distributions and entropy value is obtained for each packet in a flow. PD_η defined in 4.3 refers to a probability distribution which is extracted for $N = \eta$ with m length. It is important to note that PD_η does not contain probability values which are equals to zero. Entropy $\hat{H}(X)$ is calculated for probability distribution PD_η and

$$\hat{H}(\mathcal{PD}_\eta) = -\sum_{i=1}^{m} f_i \log(f_i). \tag{4.4}$$

Higher entropy values are obtained for higher randomness levels in a payload distribution. While higher entropy values are taken for more uniform distributions, small entropy values are taken for more concentrated probability distributions.

4.3. Maximum Possible Entropy & Actual Entropy

Network protocols have different payload structures; however, they generally do not use specific payload lengths. Packets with TCP protocol, which is commonly used in network communication, contains a payload vector with length 0 to 1460, so payload lengths of the network packets vary significantly.To analyze network packets regardless of their length, it may be insufficient to use entropy alone to compare network packets and identify patterns. The actual entropy concept also takes into account the maximum possible entropy value of a payload vector, so the difference of the entropy from the maximum possible entropy is slightly different from the entropy concept. This metric measures difference between the actual randomness and maximum possible randomness in a payload vector. This metric was used to detect encrypted network packets by Dorfinger [59]. It is seen that this metric is also effective in extracting characteristics of the network packets and diversifying the attacker packets from the legitimate ones.

The difference between actual entropy and maximum possible entropy can be represented with $D_{E_{max}}$ and it can be calculated for probability distribution $\mathcal{PD}_\eta$ with

using

$$D_{E_{max}}(\mathcal{PD}_\eta) = \hat{H}(\mathcal{PD}_\eta) - \hat{H}_{N_{max}}, \tag{4.5}$$

where $\hat{H}(\mathcal{PD}_\eta)$ corresponds to the calculated entropy value with using generated probability distribution $\mathcal{PD}_\eta$ and $\hat{H}_{N_{max}}$ corresponds to the maximum possible entropy for the payload with length N. The derivation of the maximum possible entropy is given in Section 3.5. The pre-calculated maximum possible entropy values for payload with different lengths are used in this feature. The calculation of the maximum possible entropy values are taken with the large number of iterations.

4.4. Ratio of Printable Characters

DoS and DDoS attack tools generate their packet with different methods to avoid detection mechanisms. Various packet payload types are also generated by the attacker to consume bandwidth and the resources of the victim's side. Similarly, different tools used to perform other types of network attacks create payload data in different ways. The ratio of Printable Characters is a significant feature in order to characterize any pattern caused by both the attacker's traffic and legitimate traffic. Ratio of Printable Characters is a feature which is mainly used in the encrypted packet detection algorithms [59] and anomaly detection applications [47, 66, 67]. Variation of the ratio between the legitimate payload and malicious payload is a significant feature in characterizing network packets.

The payload is mainly encoded with the ASCII characters, which can take values 0 to 255. Only the bytes between 32 and 127 correspond to printable characters, and they are used mainly in text messages. In a randomly generated payload, the ratio of these printable characters is about $37,5\%$. The higher ratios are generally taken for the text messages in which the characters are transferred.

The calculation of this feature is simple. The ratio of the total number of occurrences of n-gram sequences with printable characters to the number of sequences that

are generated from the payload. The main point of this feature is to count sequences only full with printable characters. In the $N = 1$ case, 1-gram sequences contain only one byte, and straightly bytes corresponding to printable characters are counted. However, when N equals a number that is larger than one, each sequence contains more than one byte. In this case, only the sequences with only if all bytes correspond to the printable characters are counted to calculate this feature. This feature is represented with $R_{PrintableChar}$, and it can be formalized as

$$R_{PrintableChar} = \frac{O_{PrintableCharSeq}}{\sum_{i=1}^{m} O_i}, \tag{4.6}$$

where O_i is number of occurrence of an unique sequence S_i in the N-gram analysis and $O_{PrintableCharSeq}$ is the total number occurrences of n-gram sequences with printable characters. $O_{PrintableCharSeq}$ is calculated using

$$O_{PrintableCharSeq} = \sum_{i=1}^{m} \tau_i * O_i, \tag{4.7}$$

and

$$\tau_i = \begin{cases} 1, & \text{if} \quad 32 < b_k < 127 \quad \text{for all} \quad b_k \in d_i \\ 0, & \text{otherwise,} \end{cases} \tag{4.8}$$

where unique sequence in n-gram analysis $d_i = \{b_1, b_2, ..., b_k, ..., b_n\}$ and b_k is ASCII value of the kth byte in the n-gram sequence.

4.5. Ratio of Unique Bytes

Characterization of the payload distribution utilizing extracted histograms is quite significant in analyzing and detecting network anomalies. The ratio of the unique bytes in a payload sequence is one of the valuable features to classify applications on the network [68]. It is a crucial feature in the characterization and discrimination process of both malicious activities, and legitimate traffic [69].

The ratio of unique bytes is modified for N-gram sequence analysis, and a simple ratio of unique sequences is used as a feature. The payload is mainly encoded with the ASCII characters, which can take values 0 to 255. In the $N = 1$ case,a payload sequence can contain 256 unique sequence(same with byte in $N = 1$ case) value if its length is higher than 256. The number of possible unique characters in the payload with a smaller length than 256 is equal to the length of the payload vector. Similarly, the bound is equal to the number of possible unique sequences for the cases N is larger than 1. In order to remove the effect of the variations in the payload lengths, this ratio is calculated by considering this bound.

The ratio of the unique bytes is represented with $R_{UniqueBytes}$, and it can be mathematically defined as

$$R_{UniqueBytes} = \frac{\sum_{i=1}^{m} \kappa_i}{min(m, l_{payload})}, \tag{4.9}$$

where $l_{payload}$ corresponds to the length of the payload sequence and m equals to the size of the N-gram dictionary D_η defined in (4.1). The sequence existence indicator κ can be defined as

$$\kappa_i = \begin{cases} 1, & \text{if} \quad d_i \quad \text{exists in payload} \\ 0, & \text{otherwise,} \end{cases} \tag{4.10}$$

where d_i corresponds to the sequence in the N-gram dictionary.

4.6. Greedy Distance Between the Packet Payloads

The probability distribution of the payload vector represents the characteristic of the network packet. The byte distribution is generally based on the application, and each application has its own patterns. While some of the applications transfer encrypted data, some of them directly transmit text, voice, video, or different types of data. Each type of data shows distinctive patterns in the packet payloads. Similarly,

network attacks managed by an attacker using different tools have specific patterns in their payloads. Besides that, replicating the payload pattern of legitimate network traffic is a very challenging task for the attacker. It can be used to differentiate attack traffic from legitimate network traffic.

In addition to this, analysis of the similarity of the packet payloads in a network flow exhibits some specific characteristics based on the generation way of these packets. This analysis can be performed by comparing the probability distributions of the payloads in a flow. Kullback-Leiber divergence is one of the most used ways to compare different probability distributions. It can be considered as a measure indicating the similarity between two probability distributions. However, probability distributions generated from the packet payload may not be with the same length because the unique n-gram sequences with zero probability are not accounted for in these probability distributions. These distributions are generated for only observed N-gram sequences in the payload. The main reason is that the length of these probability distributions is significantly enormous, especially with n values larger than 1. Besides that, lots of zero probabilities create a problem for calculation in Kullback-Leiber divergence. The Greedy Distance metric is a measure that gives the upper limit of the Kullback-Leiber distance without requiring the identical length probability distributions [4]. A detailed explanation of this metric is given in Section 3.4. The greedy metric can be easily applied to practical problems to compare probability distributions. In our work, the greedy distance metric is used as a measure that shows the similarity between probability distributions.

In order to calculate this metric, there must be at least 2 packet with a payload in a flow because similarity calculation is performed for each packet pair in a flow. Suppose that, a flow is represented with

$$\mathcal{F} = \{Packet_1, Packet_2, ..., Packet_m, ...Packet_k, ...Packet_N\}. \tag{4.11}$$

The total possible number of pairs for flow $\mathcal{F}$ can be taken as the 2-combinations of N. Greedy distance computed for a pair can be represented with G_{Pair_j} and

$$G_{Pair_j} = d_{greedy}(\mathcal{PD}_{\eta,m}, \mathcal{PD}_{\eta,k}) \quad Pair_j = (Packet_m, Packet_k), \tag{4.12}$$

where $Pair_j$ is the pair of $Packet_m$ and $Packet_k$. $\mathcal{PD}_{\eta,m}$ and $\mathcal{PD}_{\eta,k}$ corresponds to the probability distributions generated with η-gram analysis for $Packet_m$ and $Packet_k$, respectively. The number of greedy values are changes according to the number of packets in the analyzed flow. Then, in order to generalize this feature for a flow and to extract as a number feature, the various statistics of the greedy distance values is used. These statistics are mean, standard deviation, minimum and maximum values. These statistics are used as a different features to characterize network flows.

Smaller greedy distance values are obtained for packets that have similar payload distributions. Large greedy distance values indicate various probability distributions for the packet pair. Byte distribution characteristics of especially forward and backward packets are varied for a different types of attacks or applications. These patterns are extracted and utilized in the classification process.

4.7. Frequency Domain Analysis of Payloads

The frequency-domain analysis approach enables the investigation of periodic patterns in the payload vector. Features that give information about the pattern are extracted by transferring information from the set of bytes in a payload vector to the frequency domain. It enables us to make analyses to characterize these flows more efficiently.

Fourier transform enables the transformation of a signal in the spatial domain to spectral-domain [70]. Fourier transform of a function $f(x)$ defined in the spatial

domain can be defined as

$$F(w) = \int_{-\infty}^{+\infty} f(t)e^{-jwt}dt, \qquad w \in (-\infty, +\infty), \tag{4.13}$$

where $j = \sqrt{-1}$ and $e^{j\theta} = \cos(\theta) + j\sin(\theta)$ and w corresponds to the angular frequency [71]. In this function, time variable is replaced with the angular frequency. It means that time spectrum transferred to frequency spectrum. Similarly , inverse of Fourier Transform can be taken with

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{jwt}dt, \qquad t \in (-\infty, +\infty). \tag{4.14}$$

Detailed definitions of the variables used in the equations are given in Table 4.1.

Table 4.1. Definitions for Fourier Transform.

Definitions	
$x(t_n)$	input signal amplitude (real or complex) at time t_n (sec)
t_n	$nT = n^{th}$ sampling instant (sec), n an integer≥ 0
T	sampling period (sec)
$X(w_k)$	spectrum of x (complex valued, at frequency w_k)
w_k	$k\Omega = k^{th}$ frequency sample (radians per second)
Ω	$\frac{2\pi}{NT}$ =radian-frequency sampling interval (rad/sec)
f_s	$\frac{1}{T}$ = sampling rate (samples/sec, or Hertz (Hz))
N	number of time samples (number of frequency samples (integer))
e	$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = 2.7182$
j	$\sqrt{-1}$, basis for complex numbers

In the Fourier Transform, integral boundaries go from $-\infty$ to $+\infty$, and it is required for continuous functions or signals. However, if there are discrete signals, it is not required. Discrete Fourier Transform (DFT) can be applied to transform discrete signal to the frequency domain. DFT can be obtained by replacing integral with infinite

boundaries with a finite sum function, and DFT is

$$X(w_k) = \sum_{n=0}^{N-1} x(t_n) e^{-jw_k t_n} \qquad k = 0, 1, 2, ..., N-1. \tag{4.15}$$

In this case, if all N elements of signal $x(t_n)$ is real, this signal can be considered as a real signal and $x \in R^N$. If $x(t_n)$ is a complex signal, it can be written as $x \in C^N$.

Inverse DFT can be defined as

$$x(t_n) = \frac{1}{N} \sum_{k=0}^{N-1} X(w_k) e^{jw_k t_n} \qquad n = 0, 1, 2, ..., N-1. \tag{4.16}$$

Payload vectors are considered as the original signal in the spatial domain in order to make frequency domain analysis of network packet payloads. Then, applying 1-D Discrete Fourier Transform, frequency domain representation of the payload vector is obtained.

DFT of a signal can be taken using The Fast Fourier Transform (FFT) method, which is a numerical approach for fast and efficient computing of the Fourier transform [72]. Discrete Fourier Transform can be computed with using Fast Fourier Transform [73–76]. By taking DFT, the payload vector is transformed into a spectral domain.

The power spectral density is a representation of a time series's power or variance as a function of frequency. Power The Spectral Density of a signal being random or periodic corresponds to its power analysis of the response in the frequency domain. It is a tool to identify hidden periodicity in data and to expose characteristic features. Power spectral density can be calculated by using the expression

$$P_{XX}(w_k) = |X(w_k)| = X(w_k)X^*(w_k), \tag{4.17}$$

where $X^*(w_k)$ is the complex conjugate of the $X(w_k)$ and $P_{XX}(w_k)$ is corresponds to power spectral density function. The power spectral density function, which is

calculated in this way, is referred to as a periodogram. There are different methods to estimate the power spectrum of a signal. The periodogram is one of the commonly used power spectrum estimators. It equals the Fourier transform of the auto-correlation of the spatial domain signal.

4.7.0.1. Mean Frequency. Mean frequency is one of the useful features to show the characteristic of a signal [77]. Mean frequency can be considered as an average frequency in the power spectrum, which is calculated by dividing the sum of the multiplication of power and frequency values for each instance by the sum of the powers [78,79]. It can be calculated by using

$$MF = \frac{\sum_{i=1}^{K} f_i P(f_i)}{\sum_{i=1}^{K} P(f_i)}, \tag{4.18}$$

where f_i refers to the frequency value of the power spectral density and P_i is corresponding power component for instance i in the spectrum. In this work, mean frequency (MF) and power at the mean frequency is used as a feature.

4.7.0.2. Peak Frequency. Peak power frequency and peak power are another useful features to characterize a signal [77]. It can be calculated using

$$PP = max(P(f_1), P(f_2), ..., P(f_i), ..., P(f_K)), \tag{4.19}$$

and Peak Power frequency is the frequency value for the maximum power at the spectrum.

4.7.0.3. Spectral Entropy. Spectral entropy is one of the features generally used in classification problems of images [80] and electrocardiogram signals (ECG) [81]. It measure of the randomness of the frequency domain representation of a signal and it can be calculated considering normalized power values at the PSD as a probability.

Probabilities can be obtained with using

$$p(f_i) = \frac{P(f_i)}{\sum_{i=1}^{K} P(f_i)}, \tag{4.20}$$

where $P(f_i)$ refers to power at the frequency f_i in the PSD and $p(f_i)$ corresponds to the probability value for frequency f_i.

With using Shannon entropy formula, spectral entropy can be calculated using

$$SE = -\sum_{i=1}^{K} p(f_i) \log(p(f_i)). \tag{4.21}$$

Spectral entropy can be used to detect deeper pattern characteristics in the payload vector.

4.7.0.4. Greedy Distance of PSD. Power spectral density can be considered as a distribution of the powers for corresponding frequency values. Different deep periodicity patterns can be observed in this distribution. Besides that, this distribution can be used to define the attribute of a payload vector. Similarly, in the spatial domain, the differences between the packet attributes in a flow can be utilized to classify these flows more efficiently. Similarity and difference in the patterns in the payload vectors can be changed from threat to threat. Similarly, it also shows different attributes for different applications, and the tools which are generally used to perform these attacks have their own patterns to execute these attacks. The difference between probability distributions on the spectral domain can be used to differentiate these threats.

The probabilities for each frequency value are obtained from power spectral density using normalized power values. The generated probability distribution for the frequency spectrum is compared with other probability distributions extracted for other packets in a flow. The greedy distance metric, which is mentioned in Section 3 is used in this comparison. Greedy values for each possible packet pair are obtained in a flow.

Greedy distance value for a pair j is represented by GF_{Pair_j} and Greedy distance value for packet pair j can be represented with GF_{Pair_j} and

$$GF_{Pair_j} = d_{greedy}(\mathcal{PD}(f)_{\eta,m}, \mathcal{PD}(f)_{\eta,k}) \quad Pair_j = (Packet_m, Packet_k). \tag{4.22}$$

A vector containing these greedy values is generated by obtaining greedy values for each pair in a flow. Statistics of this vector are used as a feature for a flow in the classification stage.

4.8. Discrete Cosine Transform Coefficients

The Discrete Cosine Transform is a mathematical technique and physical transformation that is used to transform information or signal from the time domain to the frequency domain. It is frequently used in digital signal processing applications, where it is used in converting the input sequence to real coefficients and condensing the information coefficients in the low-frequency range.

Discrete Cosine Transform was proposed by Nasir Ahmed in 1972 [82]. A discrete cosine transform (DCT) is a mathematical function that describes a finite discrete data sequence as a summation of cosine functions fluctuating at various frequencies. DCT provides a real transformation of the spatial domain to the frequency domain. It is one of the most common methods which is used, especially in signal processing and image compression.

Payload vectors contain bytes in the range from 0 to 255 like images. The techniques which are used in image processing can be applied to classify or analyze payload vectors. Discrete Cosine Transform is extensively used to compress information in the images. Components of DCT contains information that effectively summarizes the characteristic of a payload distribution. 1-D DCT is applied to the payload vector to get these coefficients. Then the first three coefficients with more power are utilized to classify legitimate and attack flows in the network.

The general equation of discrete cosine transformation from the spatial domain to the frequency domain is

$$C(n) = \sqrt{\frac{2}{K}} \sum_{k=0}^{K-1} x(k) w(n) \cos(\frac{\pi n}{2K}(2k+1)), \tag{4.23}$$

where

$$w(\theta) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } \theta = 0 \\ 1, & \text{otherwise.} \end{cases} \tag{4.24}$$

Inverse of the discrete cosine transformation is obtained with using

$$x(k) = \sqrt{\frac{2}{K}} \sum_{n=0}^{K-1} C(n) w(n) \cos\big(\frac{\pi n}{2K}(2k-1)\big), \tag{4.25}$$

where

$$w(\theta) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } \theta = 0 \\ 1, & \text{otherwise.} \end{cases} \tag{4.26}$$

The discrete cosine transform coefficients are sorted according to their energy probabilities. Energy probabilities are obtained with normalizing energies DCT coefficients.

The expression to obtain energy probability for DCT coefficient $C(n)$ is

$$p_{C(n)} = \frac{P_{C(n)}}{\sum_{i=1}^{K} P_{C(i)}}, \tag{4.27}$$

where $P_{C(n)}$ is energy of the DCT coefficient $C(n)$ and $p_{C(n)}$ refers to energy probability. Energy probabilities give information about the discrimination ability and the amount of carried information [83]. Energy probabilities is applied as a coefficient selection criteria and first three coefficient are used to characterize network payloads.

5. GENERAL FRAMEWORK OF INTRUSION DETECTION SYSTEM

A general overview of the proposed anomaly detection scheme is presented in this chapter. Network anomaly detection systems process streaming network packets and detect intrusions on the network. The proposed anomaly detection scheme consists of three main modules. These modules can be summarized as pre-processing module, feature extraction module and classification module. The Block diagram of the proposed scheme is given in Figure 5.1. Pre-processing module extract flows from the network data in the first place. Then, the payload information of a predetermined number of network packets in a flow is extracted and given as an output of this module. In the feature extraction module, byte sequences are evaluated with N-gram analysis, and new byte sequences for each N value are generated. After that, conventional and proposed features are extracted to characterize network flows. These flow features are classified with the SVM classifier, and flow is labeled according to the result of the classification module.

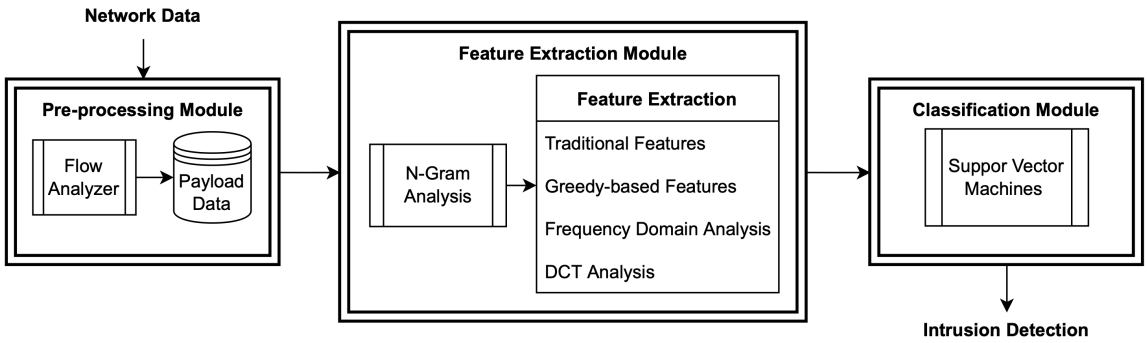

Figure 5.1. Blok Diagram of Proposed Scheme.

Section 5.1 describes pre-processing module in detail. Feature extraction module is explained in Section 5.2 and summary of the proposed features are given. In Section 5.3, detailed information is given about SVM classifier.

5.1. Data Pre-Processing

In a network system, data is carried by network packets. These packets go from their sources to their destinations. Various protocols store this information in various layers. These packets route from some network devices to their destinations. Network environment, packet capture is executed on these network devices while the network traffic is flowing. One of the most common ways to capture these network packets is by using the Wireshark tool. Wireshark is a free and open-source tool that is used in network protocol analysis, network monitoring, and troubleshooting. Wireshark is a project started by Gerald Combs in 1998 and continues with the volunteer contributions of networking specialists worldwide. It is written in the C and C++ programming languages, and it enables you to capture and analyze network packets flowing through the network interface. The captured network packets are generally stored in pcap format.

Publicly available datasets contain raw network packet data in pcap files [20, 84]. These pcap files store all network packets flowing through the network devices during the data capturing process. In these pcap files, there is no division based on the network flows. Various tools have been developed in various environments to analyze network packets using a flow-based approach, such as OpenFlow [85]. OpenFlow is a flow-based networking communication protocol that enables network devices to route network packets. Besides that, it enables us to analyze network packets on the network controller. However, publicly available datasets provide raw data in pcap format, and flow-based categorization and division operations are required to analyze network flows. In the pre-processing stage, network packets are categorized into flows using the Python library 'pcap-splitter' [86]. This library allows us to divide a pcap file into its subsets based on flows. This means this library separates a pcap file into new pcap files, and these files contain network packets that belong to only one flow. Then, these pcap files are converted to CSV files so that Matlab can read them. These CSV files contain packet header information and payload bytes in ASCII format.

The pre-processing module extracts flows from the "pcap" files of the IDS 2012 and IDS 2017, and corresponding labels are matched for the evaluation. Then, payload sequences of packets in a flow are extracted as a sequence of byte values in this module. These payload sequences are processed separately for each flow in the proposed framework.

5.2. Feature Extraction Module

The proposed system mainly analyzes the network flows to detect intrusions on the network. In this module, features are extracted for each flow by using the payload sequences of the packets in that flow. Initially, N-gram analysis is used in the generation of the new payload vectors. Different payload vectors are obtained for each N value. In this work, new payload sequences are obtained for $N = 1$, $N = 2$ and $N = 3$ values. As a second stage, defined features in Chapter 4 are extracted to characterize network flows in order to separate intrusions from legitimate ones. These features are also extracted for different parts of the flow. These parts are forward packets and backward packets. Forward packets correspond to the packets in a flow going from the source to the destination, while backward packets are replies from the destination side. Features are extracted separately for forward, backward, and all packets in a flow. This approach improves the characterization performance of the proposed features because patterns can be changed for different segments of the flows. A summary of the features is given in Table 5.1. A detailed overall list of the features is given in Table A.1.

5.3. Classification Module - Support Vector Machines (SVM)

A constructed feature vector containing a large number of features exploits network characteristics efficiently. The Support Vector Machine (SVM) algorithm is used in the detection stage. The Support Vector Machine (SVM) is one of the most effective classifiers. It can handle both linear and non-linear functions by utilizing a variety of kernels. SVM may be implemented on datasets with a greater number of features without increasing the system's complexity [87].

Table 5.1. Summary of Payload Feature List.

	Feature Names
1	Ratio of Printable Characters
2	Ratio of Unique Bytes
3	Shannon Entropy
4	Maximum - Actual Entropy
5	Greedy Distance of Payload Distributions
6	Mean Frequency
7	Peak Frequency
8	Spectral Entropy
9	Greedy Distance of Power Spectral Densities
10	Discrete Cosine Coefficients

SVM produces a class from one of two possible labels in its simplistic definition. Classification can be performed without assuming and optimizing any other parameters because SVM classifiers are non-parametric statistical machine learning tools [88]. This implies that no assumptions regarding the distribution of data are made. Support Vector Machine is a method for supervised classification, and it uses class labels in the training phase. It generates a hyperplane between the classes to classify data. It is the maximum separable boundary between distinct classes. An example of an SVM hyperplane is given in Figure 5.2.

Training data with containing n arbitrary point can be represented with a set of $(x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)$. The dimension of the x is equal to the number of features, and it can be represented as a d. y corresponds to an element indicating the class of a random sample. Representation is given as

$$(x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n) \text{ , where } x \in \Re^d, \ y \in \{-1, +1\}. \tag{5.1}$$

Equation for optimal decision plane is given as

$$(w^T \times x) + b = 0, \tag{5.2}$$

where w represents weight vector, x corresponds to input vector and b is the bias term. With using this equations and trained data points, the main condition for this hyper-plane is

$$y^i((w^T \times x) + b) \geq 1. \tag{5.3}$$

Overall classification problems solved with SVM can be represented as

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\mathbf{w}|| \\ &\text{subject to} \quad y^i((\mathbf{w}^T \times x) + b) \geq 1 \qquad \forall i = 1, \ldots, N. \end{aligned} \tag{5.4}$$

Linear separable data points can be easily classified using SVM, but it is not able to be applied to non-linear separable cases. For non-linear cases, kernels are used to transform data points from two-dimensional space to n dimensional space. This kernel trick makes SVM still applicable for non-linear cases. Kernels can be categorized into two main categories: Linear and non-linear kernels. Although the optimization for non-linear kernels is much more complex than for linear kernels, they can be used to achieve better separation between distinct data classes.

5.3.1. Kernel Trick

SVM is applicable only to linear separable problems. A kernel trick is used to adapt the SVM algorithm for various data types, and it makes SVM applicable to non-linear separable problems [89]. The kernel trick is a very effective tool because it makes it possible to solve non-linear problems by using the methods proposed for linear problems. By using only the dot product of two vectors, the original feature space is transferred to another feature space. Data attributes linearly to the mapped high dimensional feature space, although it shows non-linear behavior in its original

feature space. The kernel function can be defined as

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}). \tag{5.5}$$

Using the kernel in the SVM algorithm enables us to add dimensions to make data linearly separable. The original data samples belong to different classes and could not be separable with a linear boundary. Original data samples are transferred to another feature space by adding a new dimension by using the kernel trick. Data samples are linearly separable in the new feature space, and the SVM algorithm finds the optimum hyperplane for the classification problem. The projection of the optimal hyperplane onto the original feature space is not linear. This trick makes it possible to classify non-linearly separable data with the SVM algorithm.

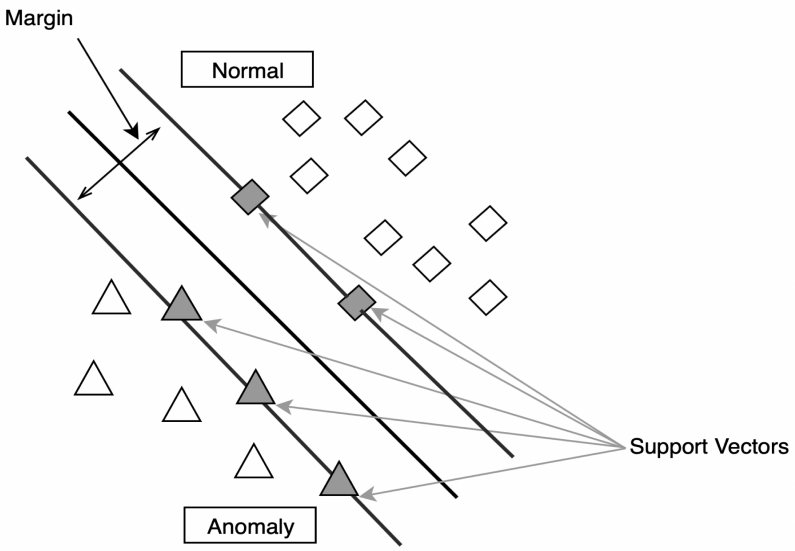

Figure 5.2. Hyperplane with Support Vector Machines.

There are 3 common nonlinear kernel types: Sigmoid Kernel, Polynomial Kernel, and RBF Kernel. They are given as follows.

5.3.1.1. Sigmoid Kernel. The Hyperbolic Tangent Kernel is also referred to as the Sigmoid Kernel or the Multilayer Perceptron (MLP) kernel. The Sigmoid Kernel originates in the field of Neural Networks, in which the bipolar sigmoid activation function is frequently applied to activate artificial neurons.

A sigmoid kernel function-based SVM model is equivalent to a two-layer perceptron neural network. Due to its origins in neural network theory, this kernel was incredibly popular for support vector machine algorithms. Additionally, it is practically applicable and shows good performance because it only requires a positive-definite condition.

The sigmoid kernel can be defined as

$$K(\mathbf{x}, \mathbf{y}) = \tanh(a_1 \mathbf{x} \cdot \mathbf{z} + a_2), \tag{5.6}$$

where a_1 and a_2 adjustable parameters. a_1 corresponds to the slope, while a_2 is a constant for interception. Generally, a_1 equals to $\frac{1}{N}$ where N is the size of the data [90].

5.3.1.2. Polynomial Kernel. The polynomial kernel is one of the most frequently used kernel functions used with Support Vector Machines(SVMs) to classify non-linear separable data instances. It represents the original variables over a feature space constructed by the polynomial functions of the original features. The general definition of polynomial kernel function is given as

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{z} + a)^d = \left(\sum_{j=1}^{N} x_i z_i + a \right)^d, \tag{5.7}$$

where $\mathbf{x}$ and $\mathbf{z}$ are vectors on the input space, and a is a parameter that enables to adjustment of the effects of the high-order and low-order components in the polynomial function. $a \geq 0$ and in the case where $a = 0$ is referred as a homogeneous kernel.

Special case for $d = 2$ is referred as quadratic kernel and it can be written as

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{z} + a)^2 = \Big(\sum_{j=1}^{N} x_i z_i + a \Big)^2 = \sum_{j=1}^{N} (x_i^2)(z_i^2) + \sum_{i=1}^{N} \sum_{j=1}^{i-1} (\sqrt{2} x_i x_j)(\sqrt{2} z_i z_j) + \sum_{j=1}^{N} (\sqrt{2a} x_j)(\sqrt{2a} z_j) + a^2. \tag{5.8}$$

Polynomial kernel is suitable for the normalized data samples. Normalization is applied for proposed features and quadratic kernel is used in our classification problem.

5.3.1.3. Radial Basis Function (RBF)/Gaussian Kernel. Radial basis function (RBF) kernels can be considered as one of the most generic forms of kernelization. With close resemblance to the Gaussian distribution, RBF kernel is one of the extensively used kernel types. The RBF kernel function measures the similarities or closeness of two vectors $\mathbf{x}$ and $\mathbf{z}$. Mathematically expression for RBF kernel can be given as

$$K(\mathbf{x}, \mathbf{y}) = \exp\big(- \frac{||\mathbf{x} - \mathbf{z}||^2}{2\sigma^2} \big), \tag{5.9}$$

where $||\mathbf{x} - \mathbf{z}||$ corresponds to L_2-norm or euclidean distance between $\mathbf{x}$ and $\mathbf{z}$ vectors. σ is variance and a hyper-parameter for this kernel.

6. EXPERIMENTS AND RESULTS

This chapter provides experiment results regarding the intrusion detection system discussed in the previous chapter. The proposed features given in Chapter 4 are implemented in the framework described in Chapter 5. Before discussing the results, a detailed dataset explanation is given in Section 6.1.

6.1. Datasets

Performance evaluation of the proposed algorithm is tested on IDS 2012 [84] and IDS 2017 [20], which are commonly used in the network anomaly detection literature. These datasets contain different types of network attacks such as DDoS, DoS, PortScan, etc.

The Canadian Cyber-Security Institute generates IDS 2012 and IDS 2017 network intrusion datasets by considering specific characteristics. These characteristics are significant because some problems usually encountered in other network attack datasets significantly impact performance evaluation. These characteristics are realistic network and traffic, labeled dataset, total interaction capture, complete capture, and diverse intrusion scenarios.

- *Realistic Network:* Network intrusions and legitimate network traffic should not exhibit any artificial behavior. The effects of the network attacks should be the same as the real network systems, and the responses of the serves against the attack should be realistic. There should not be any additional capture causing inconsistencies in the generated dataset.
- *Labelled Dataset:* Labels of the captured raw data should be available publicly. The dataset should not be required any manual labeling because it makes it harder to use the dataset.
- *Total Interaction Capture:* In the capturing process, all network traffic, including internal LANs, should be observed. The data is essential in the evaluation of the

effects of network attacks on the overall system.

- *Complete Capture:* Most of the publicly available intrusion detection datasets are heavily anonymized because of privacy concerns. Anonymization and removing raw data of the captured traffic make it impossible to use the dataset from the researchers in some cases. This dataset is captured in the controlled simulation environment, and it does not require anonymization and other operations needed because of privacy concerns.
- *Diverse Intrusion Detection Scenarios:* Dataset should contain different types of network attacks to evaluate the performance of network intrusion detection algorithms.

6.1.1. IDS 2012 Network Intrusion Dataset

IDS 2012 Network Intrusion Dataset consists of network traffic for seven days. While 3 days (Wednesday, Friday, and Saturday) contain only normal network traffic, the other 4 days contain both attack and regular traffic. In this work, days that contain only normal traffic are not used in the evaluation.

Network intrusions on Sunday, Monday, Tuesday, and Thursday are explained as follows.

- *IDS 2012 - Sunday:* An intrusion, which is called infiltrating the network from inside, is simulated on this day. The attacker gains unauthorized access from a host inside the network in this attack type. Captured data contains both attacks and legitimate activity in the network.
- *IDS 2012 - Monday:* HTTP Denial of Service attack is simulated on this day. This attack is performed with characteristics such as stealthy and low bandwidth consumption. "Slowrois" attack tool is utilized to perform HTTP DoS attacks without creating a flood in the network.
- *IDS 2012 - Tuesday:*Distributed Denial of Service attack is implemented using an Internet Relay Chat (IRC) Botnet, which is an emerging threat to the network systems. This attack is performed via the infected host on the network.

- *IDS 2012 - Thursday:* A Brute Force attack is implemented by utilizing the "brutessh" tool, which allows you to perform a brute force attack in an "SSH" environment.

The number of flows for attacks and legitimate flows for each day is summarized in the Table 6.1.

Table 6.1. IDS 2012 - Number of Flows.

	Flow Label	**Number of Flows**
Sunday	*Normal*	106705
	Infiltrating network from inside	9966
Monday	*Normal*	111160
	HTTP DoS	3130
Tuesday	*Normal*	232888
	DDoS via IRC	10995
Thursday	*Normal*	152233
	Brute Force - SSH	4753

6.1.2. IDS 2017 Network Intrusion Dataset

The IDS 2017 dataset was generated by considering the principles mentioned above. Different types of network intrusion activity are captured in the simulation environment. A labeled dataset containing raw packet data is publicly available. This dataset contains five-day network activity of anomalous and routine behaviors. While Monday's network traffic data consists of only legitimate flows, the other days contain both intrusion and regular network activity. Besides that, the number of flows with payload for network attacks on Thursday is deficient, so Monday and Thursday are not used in the evaluation. Network intrusions on Tuesday, Wednesday, and Friday are explained as follows.

- *IDS2017-Tuesday:* Brute force, being one of the most common network attacks, is simulated using the "Patator" tool. This attack is implemented using FTP and

SSH methods.

- *IDS2017-Wednesday:* Different DoS/DDoS attacks are simulated on this day. These network attacks are DoS Slowloris, DoS Slowhttptest, DoS Hulk, and DoS GoldenEye. These attacks exploit various vulnerabilities in the network and show different characteristics. On this day, the Heartbleed Port 444 attack is also implemented, but there are very small numbers of samples belonging to this attack, so it is not used in the evaluation.
- *IDS2017-Friday:* Bot, DDoS and PortScan attacks are implemented on this day. Bot attack is implemented using the python-based "Ares" tool. The DDoS attack is carried out with the Low Orbit Ion Canon (LOIC) tool, which generates floods in the TCP, UDP, and HTTP protocols. Portscan attack is performed with "Nmap", a well-known tool for this attack type.

The number of flows belonging to legitimate and anomalous network activity is given in the Table 6.2.

Table 6.2. IDS 2017 - Number of Flows.

	Flow Label	**Number of Flows**
Tuesday	*Normal*	149964
	Brute Force - Patator - FTP	3942
	Brute Force - Patator - SSH	2955
Wednesday	*Normal*	152638
	DoS GoldenEye	7367
	DoS Hulk	152318
	DoS Slowhttptest	810
	DoS Slowloris	1989
Friday	*Normal*	82782
	Bot - Ares	1153
	DDoS - LOIC	47158
	PortScan	86738

6.1.3. Dataset Preprocessing

IDS 2012 and IDS 2017 datasets contain two types of files. One is a "CSV" file that contains summary information about each flow and its label. Another one is raw network traffic data in "pcap" format. This data contains unmodified packet payload data, which enables an analysis of packet payload to detect intrusions.

Label of network flows are given in "CSV" files in these public datasets [20, 84]. In the pre-processing stage of the dataset, each raw network packet from "pcap" files is labeled using "CSV" files. In this stage, network packets are grouped based on flows and flow-id and time extracted. Flow-id generated using source IP, destination IP, source port, destination port, and protocol information. By using Flow ID and timestamp information, the label is found from the "CSV" file and matched with the network packets. However, some of these network flows labeled in the "CSV" files do not match with the network flows obtained from the raw "pcap" files. The leading cause of that time information is not the same for them in "pcap" and "CSV" files. These unlabelled flows are not used in the evaluation stage. Only the safely labeled flows are used in this stage. The given numbers of flows belong to the only safely labeled flows.

6.2. Results

The performance of the proposed algorithm is tested on IDS 2012 and IDS 2017 datasets. These datasets and pre-processing stage are explained in Section 6.1. Training and test datasets are generated in different ways to evaluate performance efficiently. In Case A, a one-class classification model is trained. In the second Case B, One-class and Multi-Class classification models are trained for each dataset separately. In this case, all attack samples belonging to IDS 2012 and IDS 2017 datasets are used separately in the training of the classification model.

6.2.1. Evaluation

Algorithm performance is evaluated with well-known performance metrics. These are accuracy, specificity, precision, recall, false-positive rate, and F1-Score. F1-Score is a metric accounting for both precision and recall, and it is better to use to evaluate overall performance. In each scenario, a table is provided with performance metrics for the features extracted from 3, 5, and 10 packages in a flow. Besides that, for detailed analysis, a confusion matrix is given for each scenario for 5 packet case. 5 packet case is selected for detailed analysis because early detection is essential in intrusion detection. The 3-packet case for some attack cases is not sufficient to characterize the attack flows; however, the 5-packet case performed well in most cases.

6.2.1.1. Performance Metrics. Well-known performance metrics are used to evaluate the accuracy of the network intrusion detection performance. These metrics are accuracy, specificity, precision, recall, false-positive rate, F1-Score, and confusion matrix.

6.2.1.1.1. Accuracy: The general accuracy of classifier is calculated by comparing the results of the intrusion detection algorithm with the real labels. The following equation describes the calculation of this metric,

$$Accuracy = \frac{(TP + TN)}{(TP + FN + FP + TN)}. \tag{6.1}$$

6.2.1.1.2. Specificity: Specificity, which is also known as True Negative Rate, is a term that refers to the ratio of samples labeled as normal, given the samples are actually normal and it can be defined as

$$Specificity = \frac{TN}{(TN + FP)}. \tag{6.2}$$

6.2.1.1.3. Precision: Precision, which is also known as a positive predictive value, refers to the ratio of detected true attack samples to all samples labeled as an attack. This metric can be formulated as

$$Precision = \frac{TP}{(TP + FP)}. \tag{6.3}$$

6.2.1.1.4. Recall: Recall which is also known as true positive rate, sensitivity or detection rate refers to the ratio of succesfully detected attack samples. It can be calculated using the following expression,

$$Recall = \frac{TP}{(TP + FN)}. \tag{6.4}$$

6.2.1.1.5. False Positive Rate: False-positive rate, which is also known as the false alarm rate, is the percentage of legitimate traffic that is incorrectly labeled as an attack. It can be formulated as

$$FalsePositiveRate = \frac{(TP + TN)}{(TP + FN + FP + TN)}. \tag{6.5}$$

6.2.1.1.6. F1-Score: F1-Score is calculated using precision and recall values. It corresponds to the harmonic mean of these values, and the F1-score aggregates both into a single metric. It can be formulated as

$$F1 - Score = 2 \times \frac{(precision \times recall)}{(precision + recall)}. \tag{6.6}$$

6.2.1.1.7. Confusion Matrix: An example of a confusion matrix is given in Figure 6.1. In the confusion matrix, rows belong to actual class labels while the columns correspond to predicted classes. Central matrix cells contain the number of flows for actual and predicted classes. The ratios given on the right side are True Positive (TP)

and False Negative (FN) rates for each class. The rates given below the table show False Predictive Values (FPV) and False Discovery Rates (FDR), respectively.

- TP = Number of attack samples predicted as an attack
- TN = Number of normal samples predicted as also normal
- FP = Number of normal samples predicted as an attack
- FN = number of attack samples predicted as normal
- TPR= Ratio of incorrectly labeled samples among actual normal samples.
- FNR= Ratio of incorrectly labeled samples among actual attack samples.
- FPR= Ratio of incorrectly labeled samples among predicted normal samples.
- FDR= Ratio of incorrectly labeled attack samples among predicted normal samples.

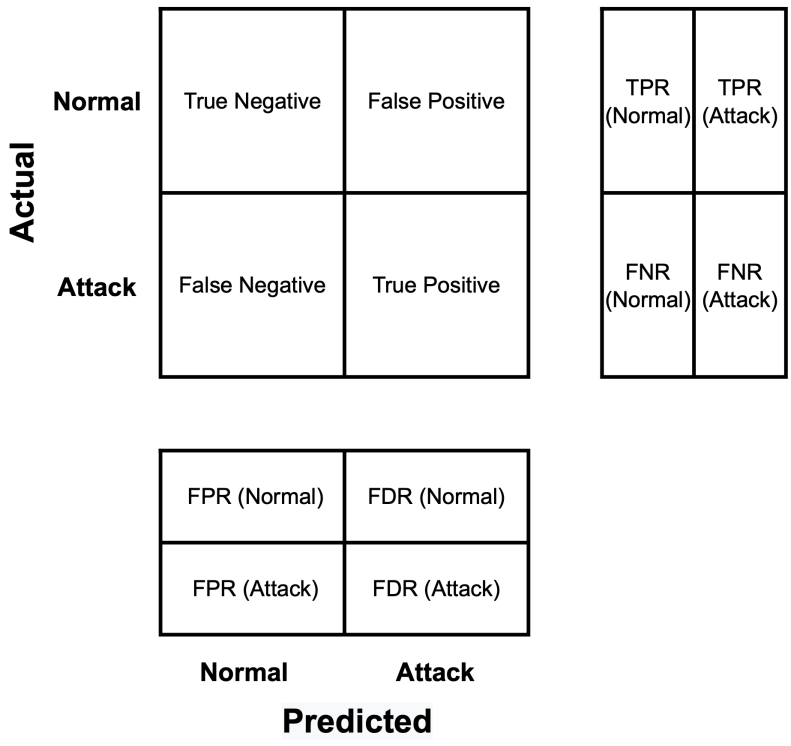

Figure 6.1. Example of Confusion Matrix.

6.2.2. Case A: Individual Attack Performance Analysis

IDS 2012 and IDS 2017 datasets are used to evaluate the performance. It can easily seen that these dataset are unbalanced dataset with varying sample numbers

from 810 to 232888 for each class from Table 6.1 and 6.2. In order to reduce the effect of unbalance distribution, each attack is evaluated one by one. Normal data is taken randomly from the day when the selected attack is simulated. The number of samples is determined according to the class with the lowest number of flows for each evaluation. For example, DoS Slowloris contains 1989 samples and the normal class on the same day contains 152638 samples from Table 6.2. In this case, 1989 samples are taken from both DoS Slowloris and Normal classes, then randomly selected 70% samples are used in training while the remaining 30% samples are used in the test phase. One Class SVM model is trained with a quadratic kernel.

6.2.2.1. Infiltrating Network from Inside - IDS 2012 - Sunday. The number of flows belonging to this attack is 9966, and the number of legitimate flows on this day equals 106705. A randomly selected 9966 flows from the normal class, and all flows of the infiltrating network from inside attack are used in the evaluation.

The proposed payload-based intrusion detection system detects infiltrating Network from Inside attack flows. Performance evaluations are given for 3, 5, and 10 packet cases in Table 6.3. The attack detection performance of the algorithm slightly increases when the number of packets used in the feature extraction increases. However, even in the 3 packet case, the F1-Score is 0.9841. In this case, 99.60% of attack flows are detected, although there are some false alarms.

Table 6.3. Infiltrating Network from Inside - IDS 2012 - Sunday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9839	0.9719	0.9726	0.9960	0.0281	0.9841
5 Packet	0.9854	0.9732	0.9739	0.9973	0.0268	0.9855
10 Packet	0.9871	0.9779	0.9783	0.9974	0.0221	0.9872

The training dataset contains a total of 13952 flows for each class, and the test dataset has 5980 flows. Figure 6.2 shows a complexity matrix containing a detailed analysis of five packet cases. In this case, 80 attack flows are predicted as normal,

corresponding to 2.7% of the total flow. 13 normal sample is predicted as an attack, and a false alarm is generated. It accounts for %0.4 percent of all legitimate flows.

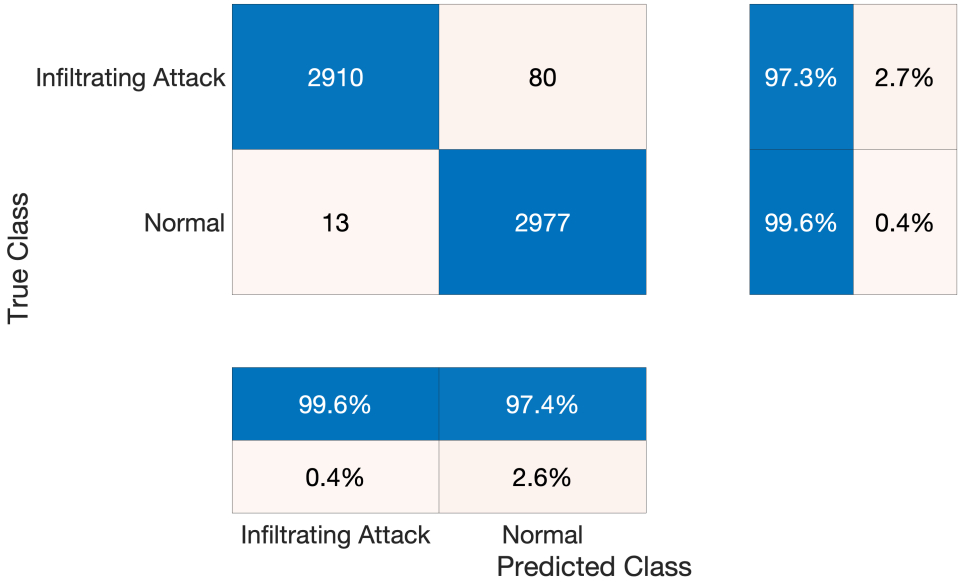

Figure 6.2. Infiltrating Network from Inside - IDS 2012 - Sunday - Confusion Matrix.

6.2.2.2. HTTP DoS - IDS 2012 - Monday. The total number of flows associated with this attack is 3130, while the total number of legitimate flows on this day is 111,160. Randomly selected 3130 flows from the normal class, and all flows of HTTP DoS attack are used in the evaluation.

The algorithm's performance in detecting HTTP DoS flows is provided in Table 6.4. The features are extracted from flows of 3,5, and 10 packets. It is observed that using 5 packets improves performance compared to using 3 packets. However, 10 packets decrease the accuracy and recall ratio while significantly improving the false-positive rate. Therefore, there is a decrease in the algorithm's overall performance when 10-packet is used. The primary reason for this issue is that some attack flows have fewer than 10 packets, which results in performance degradation.

The training dataset has a total of 4632 flows for each class, whereas the test dataset contains 1986 flows. In Figure 6.3, a detailed analysis of the 5-packet case is presented as a complexity matrix. It is seen that number of the wrong prediction is

very small. There are no missed HTTP DoS attack flows, but there are seven false positives for legitimate flows. Although the number of training samples is relatively limited for this attack situation, features can be used to properly describe network flows to distinguish legitimate flows from malicious ones.

Table 6.4. HTTP DoS - IDS 2012 - Sunday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9955	0.9932	0.9939	0.9969	0.0060	0.9955
5 Packet	0.9960	0.9940	0.9940	0.9980	0.0060	0.9960
10 Packet	0.9950	0.9970	0.9970	0.9930	0.0030	0.9950

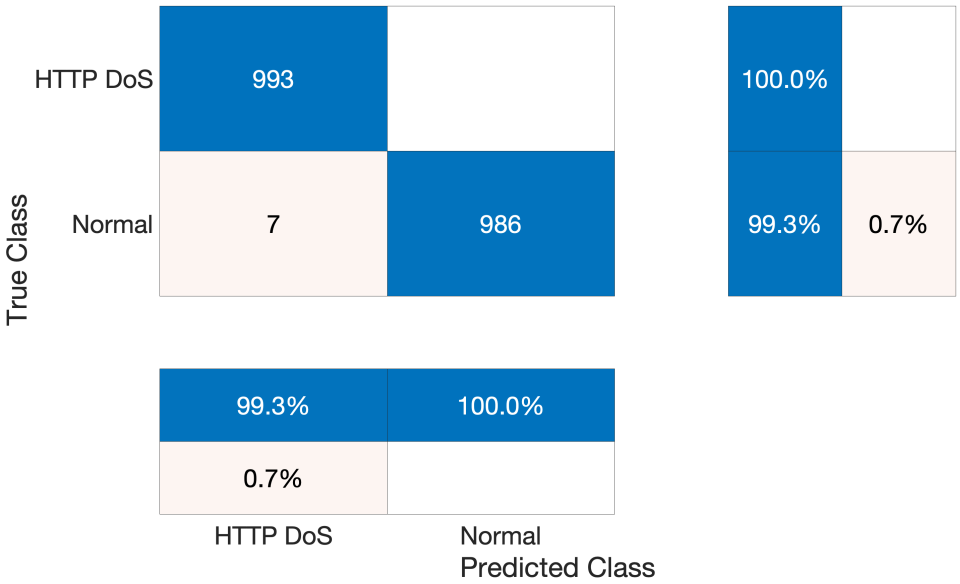

Figure 6.3. Http DoS - IDS 2012 - Sunday - Confusion Matrix.

6.2.2.3. DDoS via IRC - IDS 2012 - Tuesday. The number of flows belonging to this attack is 10995, and the number of legitimate flows on this day equals 232888. Randomly selected 10995 flows from the normal class, and all flows from the DDOS attack are used in the evaluation.

DDoS attack flows simulated via the Internet Relay Chat (IRC) protocol are detected by the proposed payload-based intrusion detection system. Performance evaluations are given for 3,5 and 10 packet cases in Table 6.5. There is a small performance increase for all performance metrics when the number of packets increases. DDoS at-

tack flows are detected with a low false-positive rate. Therefore, these features can effectively characterize DDoS attacks performed using the IRC protocol.

Table 6.5. DDoS IRC - IDS 2012 - Tuesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9938	0.9933	0.9933	0.9942	0.0067	0.9938
5 Packet	0.9941	0.9939	0.9939	0.9942	0.0061	0.9941
10 Packet	0.9948	0.9948	0.9948	0.9948	0.0052	0.9948

The training dataset contains a total of 15392 flows for each class, and the test dataset has 6598 flows. A detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.4. There are a total of 38 wrong predictions out of 6598 network flows in the DDoS IRC attack case. It corresponds to 0.6% of the tested network flows. It can be seen that extracted features performed well to distinguish DDoS attacks from legitimate ones.

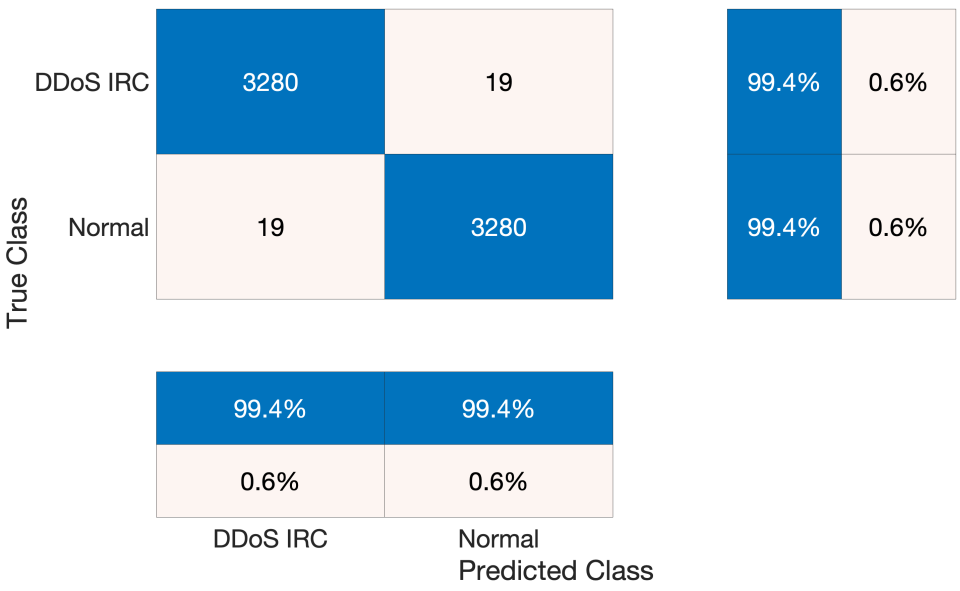

Figure 6.4. DDoS IRC- IDS 2012 - Tuesday - Confusion Matrix.

6.2.2.4. Brute Force SSH - IDS 2012 - Thursday. The total number of flows associated with this attempt is 4753, while the total number of legitimate activities on this day is 152233. The evaluation uses randomly selected 4753 flows from the normal class, and all flows from the Brute Force SSH attack.

The presented payload-based intrusion detection system identifies Brute Force-SSH attack flows. Performance evaluations for 3, 5, and 10 packet cases are provided in Table 6.6. The table represents how the overall performance of the attack detection algorithm declines as the number of packets used in feature extraction grows. It is due to the nature of Brute Force attacks. It contains a small number of packets in a flow, whereas valid flows contain a greater number of packets. It results in a decline in performance. It indicates the importance of selecting the appropriate number of packets to extract features for different attack cases. With a Brute Force attack on the SSH protocol, a high detection rate can be obtained with early attack detection by processing only 3 packets.

Table 6.6. Brute Force SSH - IDS 2012 - Thursday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9968	0.9944	0.9944	0.9993	0.0056	0.9969
5 Packet	0.9961	0.9930	0.9930	0.9993	0.0070	0.9962
10 Packet	0.9919	0.9979	0.9979	0.9860	0.0021	0.9919

The training dataset contains a total of 6654 flows for each class, and the test dataset has 2852 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.5. There are only 2 missed attacks out of 1426 attack flows, which corresponds to 0.1% of attack flows. There were 14 false alarms out of 1426 legitimate flows. In brute force attack cases, the total error rate is 0.5% percent.

6.2.2.5. Brute Force - Patator - FTP- IDS 2017 - Tuesday. The number of flows belonging to this attack is 3942, and the number of legitimate flows on this day equals 149964. Randomly selected 3942 flows from the normal class, and all flows of Brute Force - Patator - FTP attack are used in the evaluation.

Brute Force - Patator - FTP attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.7. Brute force attacks can be easily detected by using proposed payload-based features. Precision for the detection of attack samples is very high.

At the same time, the ratio of false alarms is very small. These attack flows can be detected even by using 3 packet features. It means that damage of the Brute Force attack can be prevented with early detection.

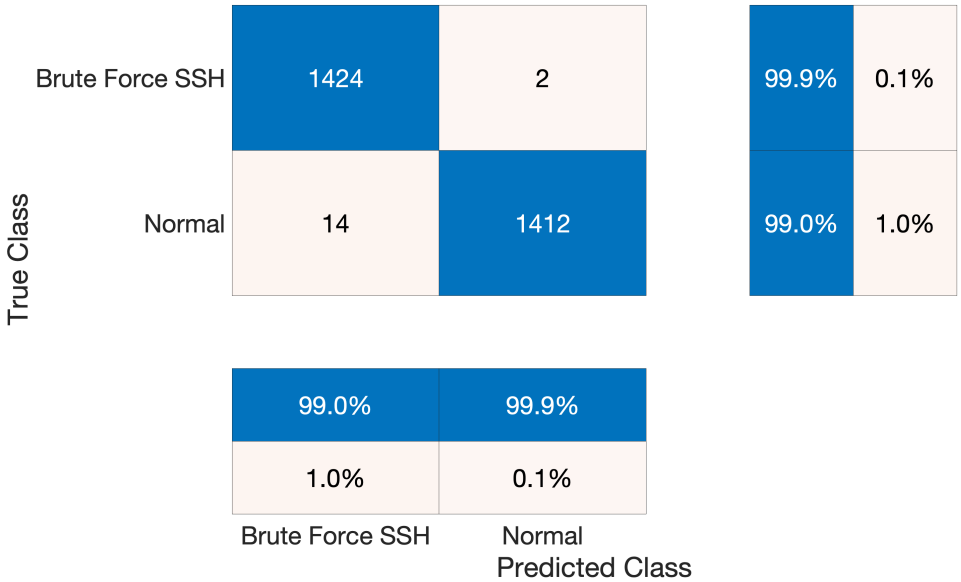

Figure 6.5. Brute Force SSH- IDS 2012 - Thursday - Confusion Matrix.

Table 6.7. BruteForce Patator - FTP - IDS 2017 - Tuesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9987	0.9975	0.9975	0.9999	0.0025	0.9987
5 Packet	0.9996	0.9999	0.9999	0.9992	0.0001	0.9996
10 Packet	0.9987	0.9992	0.9992	0.9983	0.0008	0.9987

The training dataset contains a total 5518 number of flows for each class, and the test dataset has 2366 flows. Detailed analysis of 5 packet case is given as a complexity matrix in Figure 6.6. It can be seen that there is no misclassified attack flow in the complexity matrix for the FTP-Patator type Brute Force attack. Attack flows are effectively characterized by the proposed payload-based features. Only 2 normal flow is labeled as attack flows, and it generates false alarms, but the ratio of false alarms is quite small and 0.2%.

6.2.2.6. Brute Force - Patator - SSH- IDS 2017 - Tuesday. The number of flows belonging to this attack is 2955, and the number of legitimate flows on this day equals

149964. Randomly selected 2955 flows from the normal class, and all flows of Brute Force - Patator - SSH attack are used in the evaluation.

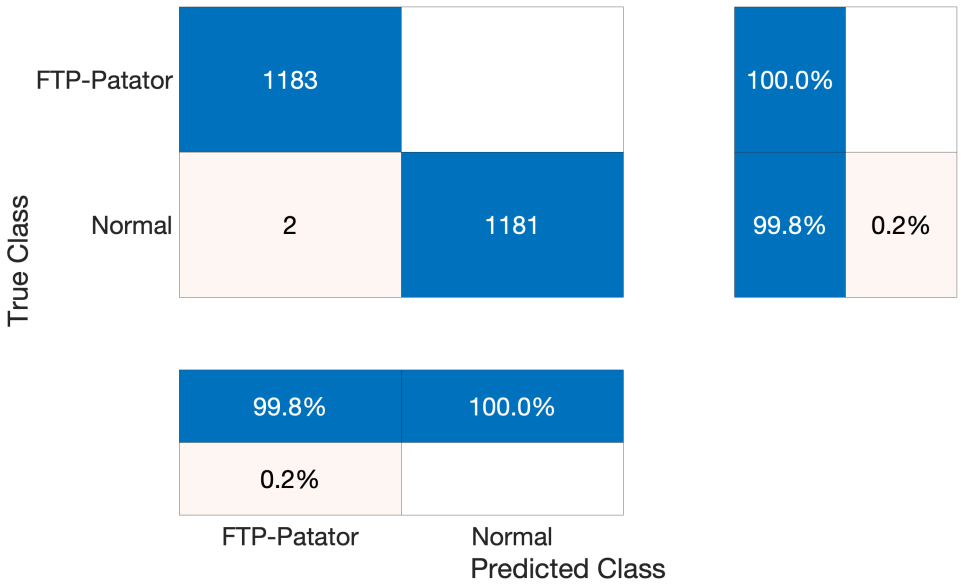

Figure 6.6. BruteForce Patator - FTP - IDS 2017 - Tuesday - Confusion Matrix.

Brute Force - Patator - SSH attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.8. Similar to the FTP-Patator attack, the Brute Force attack, which exploits SSH protocol, is also easily detectable with payload-based features. The performance metrics indicate that attack flow detection is executed for all 3, 5, and 10 packet features with very low false-positive rates.

The training dataset contains a total 4136 number of flows for each class, and the test dataset has 1774 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.7. Similar to the FTP-Brute Force attack, there are only 2 missed samples of the SSH-Brute Force attack in the test phase. The missed sample ratio is 0.2% of the total attack samples. There is no false alarm in this classification. Payload-based features are pretty valuable for detecting Brute Force attacks.

Table 6.8. BruteForce Patator - SSH - IDS 2017 - Tuesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9977	0.9989	0.9989	0.9966	0.0011	0.9977
5 Packet	0.9983	0.9989	0.9989	0.9977	0.0011	0.9983
10 Packet	0.9977	0.9955	0.9955	0.9999	0.0045	0.9978

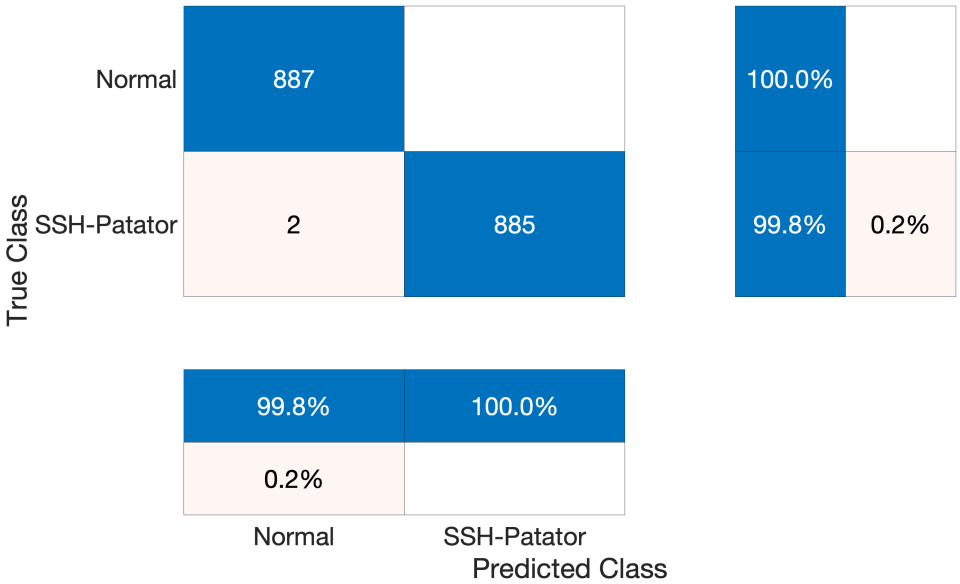

Figure 6.7. BruteForce Patator - SSH - IDS 2017 - Tuesday - Confusion Matrix.

6.2.2.7. DoS GoldenEye- IDS 2017 - Wednesday. The number of flows belonging to this attack is 7367, and the number of legitimate flows on this day equals 152638. Randomly selected 7367 flows from the normal class, and all flows of the DoS GoldenEye attack are used in the evaluation. DoS GoldenEye attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.13.

The training dataset contains a total 10312 number of flows for each class, and the test dataset has 4422 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.8.

6.2.2.8. DoS Slowhttptest- IDS 2017 - Wednesday. The number of flows belonging to this attack is 810, and the number of legitimate flows on this day equals 152638. Ran-

domly selected 810 flows from the normal class, and all flows of the DoS Slowhttptest attack are used in the evaluation.

Table 6.9. DoS GoldenEye - IDS 2017 - Wednesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9907	0.9815	0.9818	1.000	0.0185	0.9910
5 Packet	0.9948	0.9905	0.9906	0.9991	0.0095	0.9948
10 Packet	0.9966	0.9941	0.9941	0.9991	0.0059	0.9966

Figure 6.8. DoS GoldenEye - IDS 2017 - Wednesday - Confusion Matrix.

DoS Slowhttptest attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.10. DoS Slowhttptest attack is effectively detected with proposed payload features. Even if the number of samples is small for this attack scenario, the SVM classifier is able to learn the boundaries between attack and legitimate samples. For all 3, 5, and 10-packet cases, proposed features can model the attribute of the attack flows. High F1-Score ratios are obtained for these three cases.

The training dataset contains a total 1134 number of flows for each class, and the test dataset has 486 flows. Detailed analysis of the 5-packet case is given as a

complexity matrix in Figure 6.9. The misclassification rates for DoS Slowhttptest and normal class are meager and 0.8% and 0.4%, respectively. DoS Slowhttptest can be detected efficiently with payload analysis. The false-positive rate is 0.2% for this case.

Table 6.10. DoS Slowhttptest - IDS 2017 - Wednesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9938	0.9992	0.9993	0.9877	0.0011	0.9938
5 Packet	0.9938	0.9959	0.9979	0.9918	0.0021	0.9938
10 Packet	0.9959	0.9994	0.9992	0.9918	0.0011	0.9959

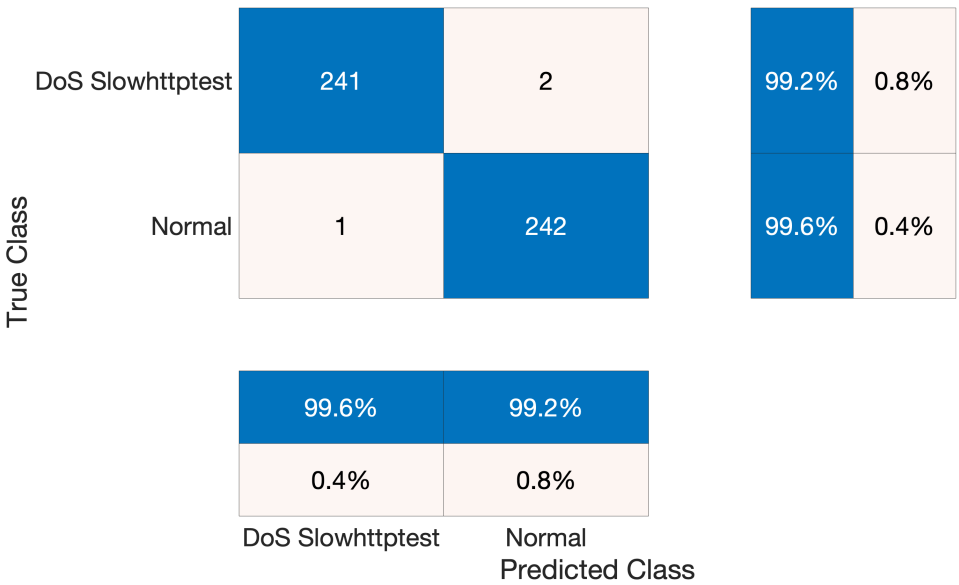

Figure 6.9. DoS Slowhttptest - IDS 2017 - Wednesday - Confusion Matrix.

6.2.2.9. DoS Hulk- IDS 2017 - Wednesday. The number of flows belonging to this attack is 152318, and the number of legitimate flows on this day equals 152638. Randomly selected 152318 flows from the normal class, and all flows of the DoS Hulk attack are used in the evaluation.

DoS Hulk attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.10. Even if 3 packet features are capable of representing the attack attributes, they can be used on early attack detection with a high precision rate. The performance is slightly improved, and false-positive rates are decreased when the number of the

processed packets increases. F1-Score obtained for the 5-packet case is 0.9921, implying that DoS Hulk attack samples can be detected effectively.

Table 6.11. DoS Hulk - IDS 2017 - Wednesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9920	0.9865	0.9866	0.9975	0.0135	0.9920
5 Packet	0.9921	0.9865	0.9867	0.9976	0.0135	0.9921
10 Packet	0.9927	0.9877	0.9879	0.9976	0.0123	0.9927

The training dataset contains a total 213244 number of flows for each class, and the test dataset has 91392 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.10. DDoS Hulk attacks create floods on the network to exhaust network resources. The number of samples in the evaluation dataset is relatively high. Only 111 flow belonging to the attack is missed out of 45596 flows. It corresponds to 0.2% of the total attack flows. There are 615 false alarms generated on the system, but it is 0.65% of the whole network flows evaluated in the test phase.

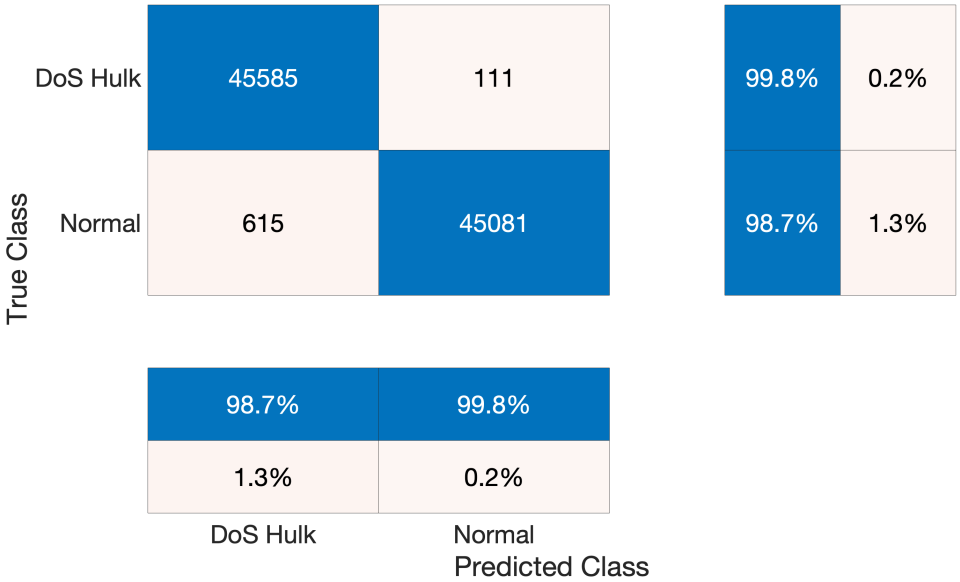

Figure 6.10. DoS Hulk - IDS 2017 - Wednesday - Confusion Matrix.

6.2.2.10. DoS Slowloris- IDS 2017 - Wednesday. The number of flows belonging to this attack is 1989, and the number of legitimate flows on this day equals 152638. Randomly selected 810 flows from the normal class and all flows of the DoS Slowrois

attack are used in the evaluation.

DoS Slowloris attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.12. Slowloris attack can be detected efficiently by using 3, 5, and 10 packet features. 3-packet features provide early detection of attack samples with very high precision. The false-positive rate is very low for 5 and 10 packet cases. There is no big difference between 3, 5, and 10 packet cases, but the best performance is taken for the 5-packet case according to F1-Score.

Table 6.12. DoS Slowloris - IDS 2017 - Wednesday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9950	0.9933	0.9933	0.9966	0.0067	0.9950
5 Packet	0.9958	0.9983	0.9983	0.9933	0.0017	0.9958
10 Packet	0.9933	0.9933	0.9933	0.9933	0.0067	0.9933

The training dataset contains a total of 2784 flows for each class, and the test dataset has 1194 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.11. The misclassification rate for both classes is 0.3% for the DoS Slowloris attack. These attack flows are differentiated by the proposed payload-based features efficiently, and the SVM classifier is suitable for classifying these flows.

6.2.2.11. Bot Ares- IDS 2017 - Friday. The number of flows belonging to this attack is 1153, and the number of legitimate flows on this day equals 82782. In the evaluation, 1153 flows from the normal class, and all flows from the Bot-Ares attack are chosen at random.

Attack flows simulated with the python-based ARES tool are detected by the proposed payload-based intrusion detection system. Performance evaluations are given for 3, 5, and 10 packet cases in Table 6.13. Detection performance for 3,5 and 10 packet cases does not differ too much. The detection performance is similar in each case, but the false positive rate decreases, and the detection performance slightly improves as

the number of used packets increases. The F1-Score metric can be used to assess the classifier's overall performance, which depends on both precision and recall rates. It is increased from 0.9730 to 0.9787 when the number of analyzed packets increases. 10-packet can be used for more accurate detection, but 3 packet cases also provide early detection of attack flows with high accuracy.

Figure 6.11. DoS Slowloris - IDS 2017 - Wednesday - Confusion Matrix.

The training dataset contains a total of 1614 flows for each class, and the test dataset has 692 flows. A detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.12. In the complexity matrix, it can be seen that the number of missed attack flows is really low and corresponds to 0.3% of total attack flows. 4.3% of the normal flows are predicted as Bot Ares attacks. It has a relatively high misclassification rate. It can be improved by increasing the number of samples in the training dataset.

Table 6.13. BotNet Ares - IDS 2017 - Friday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9725	0.9566	0.9580	0.9884	0.0434	0.9730
5 Packet	0.9769	0.9566	0.9583	0.9971	0.0434	0.9773
10 Packet	0.9783	0.9595	0.9610	0.9971	0.0405	0.9787

6.2.2.12. DDoS LOIC- IDS 2017 - Friday. The number of flows belonging to this attack is 47158, and the number of legitimate flows on this day equals 82782. Randomly selected 47158 flows from the normal class, and all flows of a DDoS LOIC attack were used in the evaluation.

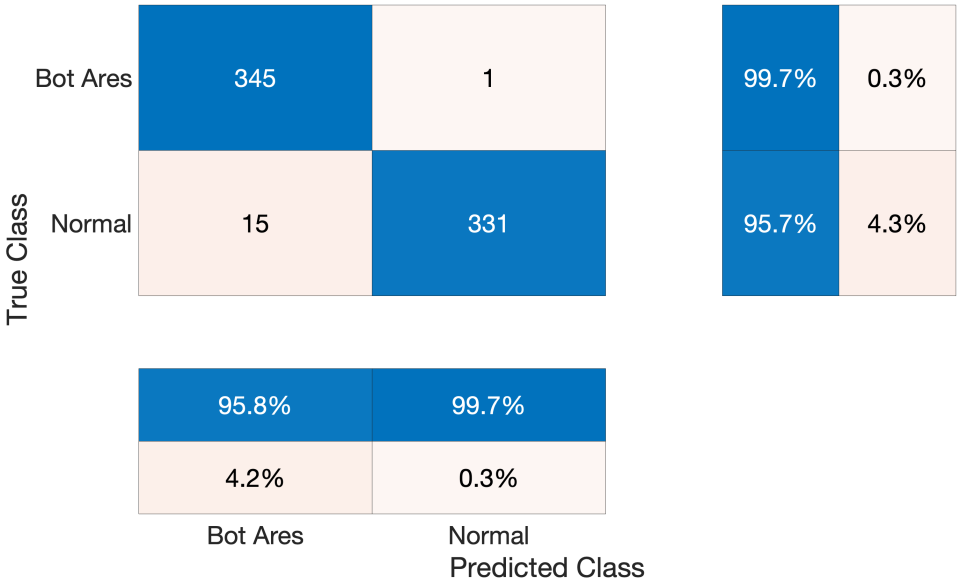

Figure 6.12. BotNet ARES- IDS 2017 - Friday - Confusion Matrix.

The DDoS LOIC attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluations are given for 3, 5, and 10 packet cases in Table 6.14. There is a big improvement in the detection performance of the 5 and 10 packet cases compared with the 3-packet case. The features extracted from the 3 packets in a flow are not sufficient to explain the attributes of DDoS attack flows. The F1-Score for the 5 packet case is 2 percentage points higher than the 3 packet case. However, there is no significant improvement in performance for the 10 packet case. The performance metrics are slightly higher than in the 5 packet case. Five packet features are preferable when the importance of early detection with high accuracy is accounted for.

The training dataset contains a total of 33011 flows, and the test dataset has 14148 flows for each class. A detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.13. It can be seen that DDoS attacks are detected with a very high recall rate. However, the number of false alarms and the misclassification of the normal

flows is 5.7% of the total legitimate samples. It means the false positive rate for DDoS attacks performed with a tool called LOIC is higher than the other attack cases.

Table 6.14. DDoS LOIC - IDS 2017 - Friday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9496	0.9242	0.9278	0.9751	0.0758	0.9509
5 Packet	0.9709	0.9418	0.9450	0.9999	0.0582	0.9717
10 Packet	0.9715	0.9430	0.9461	1.000	0.0570	0.9723

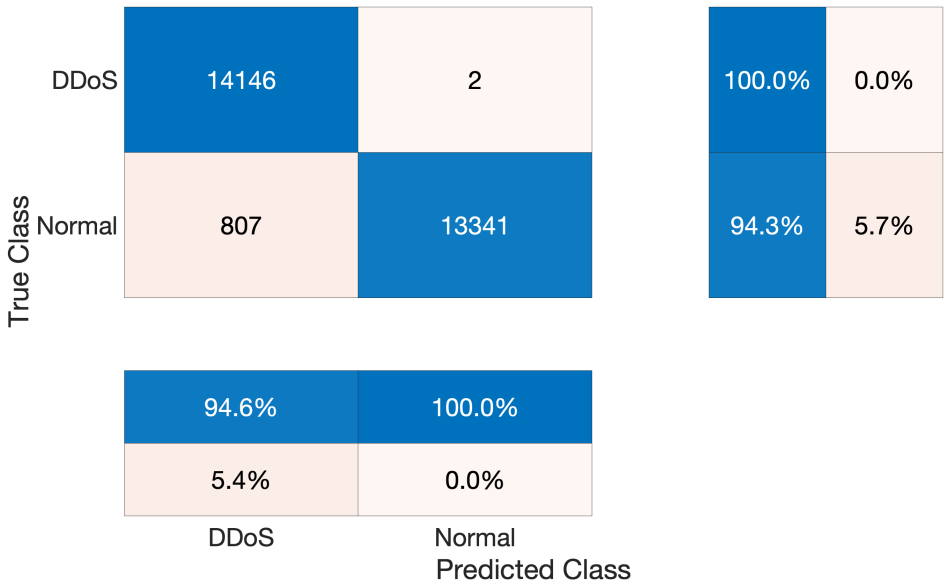

Figure 6.13. DDoS LOIC- IDS 2017 - Friday - Confusion Matrix.

6.2.2.13. PortScan - IDS 2017 - Friday. The number of flows belonging to this attack is 86738, and the number of legitimate flows on this day equals 82782. Randomly selected 82782 flows from the PortScan attack class, and all flows of legitimate flows on Friday are used in the evaluation.

PortScan attack flows are detected by the proposed payload-based intrusion detection system. Performance evaluation given for 3,5 and 10 packet cases in Table 6.15. The portScan attack can be detected with a very low false-positive rate using even 3 packet features. Increasing the number of packets improves the overall performance according to F1-Score with an increasing recall rate. However, false-positive rates, which indicate the false alarm ratio, are also increasing for the PortScan attack.

The training dataset contains a total 115894 number of flows for each class, and the test dataset has 49670 flows. Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.14. The total number of misclassified samples is 718 in the PortScan attack case. It corresponds to 1.4% of the total flows in test data. The ratio of false-positive rate is 2.6%, and the number of false alarms is 644. The number of missed attack flows is 74, and it is 0.3% of attack flows in the test data. Although there are some false alarms, the detection of the PortScan attack is performed with a very high detection rate.

Table 6.15. PortScan - IDS 2017 - Friday - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive - Rate**	**F1-Score**
3 Packet	0.9845	0.9964	0.9963	0.9725	0.0036	0.9843
5 Packet	0.9851	0.9729	0.9736	0.9972	0.0271	0.9852
10 Packet	0.9858	0.9743	0.9749	0.9974	0.0257	0.9860

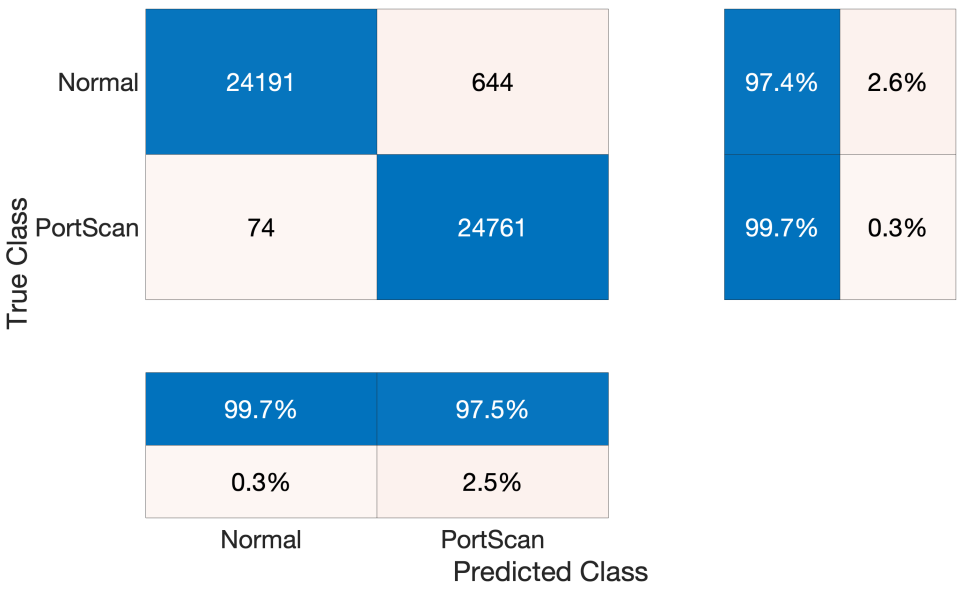

Figure 6.14. PortScan - IDS 2017 - Friday - Confusion Matrix.

6.2.3. Case B: Dataset Performance Analysis

In this case, one-class SVM is trained for all attacks and legitimate flows in IDS 2012 and IDS 2017 datasets separately. These datasets are unbalanced dataset with varying sample numbers from 810 to 232888 for each class from Table 6.1 and 6.2.

In order to reduce to effects of unbalanced distribution, the maximum sample size is determined as 20 000 flows for attack classes. If it is lower than 20 000, all network flows of a class are used in the evaluation. 20 000 flow is randomly selected for the classes with a higher sample size. Legitimate network flows are randomly selected according to the total number of flows in the attack class. Then, 70% of each class is selected randomly to train a one-class SVM model, and a test dataset is generated using the remaining samples from each class. In order to handle non-linear distinctions between the classes, a one-class SVM algorithm is trained using a quadratic kernel.

In this case, multi-class SVM is also trained to evaluate the ability of the proposed features to distinguish different attack types. Multi-class SVM is trained with both attack and legitimate flows in the datasets.

6.2.3.1. IDS 2012 Dataset. The IDS 2012 dataset contains four different attack types, each with its particular objectives. While HTTP DoS attacks and DDoS attacks try to exhaust network resources, Brute Force attacks and Infiltrating attacks try to gain unauthorized access in different ways. These attacks are assigned to only one attack class, and randomly selected same number of legitimate flows are used in the evaluation.

Performance evaluation metrics for trained one-class SVM are given in Table 6.16 for 3, 5, and 10 packet cases. It can be seen that the best performance is taken for the 5 packet cases based on the F1-Score. There is a clear performance increase in the 5 packet case compared to the 3 packet case. In the 10 packet case, there is quite a decrease in F1-Score, where there is an improvement in the false positive rate. There is no big improvement because the number of payload packets, specifically flows belonging to attack, is generally less than 10. Using more packets in the network disrupts the characterization of legitimate and attack flows. There is no big difference between 5 and 10 packet cases. The 5 packet case will provide early detection with almost the same accuracy as the 10 packet case. Also, using the 5 packet case will reduce the computational load of the anomaly detection algorithm.

Table 6.16. IDS 2012 - Performance Evaluation.

	Accuracy	Specificity	Precision	Recall	FalsePositive Rate	F1-Score
3 Packet	0.9853	0.9913	0.9940	0.9812	0.0087	0.9874
5 Packet	0.9887	0.9915	0.9941	0.9868	0.0085	0.9904
10 Packet	0.9859	0.9918	0.9943	0.9819	0.0082	0.9880

Confusion matrix for the 5 packet case is given in Figure 6.15. There is an equal number of flows in the evaluation dataset for both legitimate and attack flows. 141 attack flow is missed out of 8608 attack flows, and it corresponds to 1.6% of the total attack flows. Misclassified normal flows are 0.3% of total normal flows, and only 26 normal flows are labeled as attack flows. Proposed features can reveal the attributes of the legitimate and attack flows, and quadratic SVM can effectively classify these network flows.

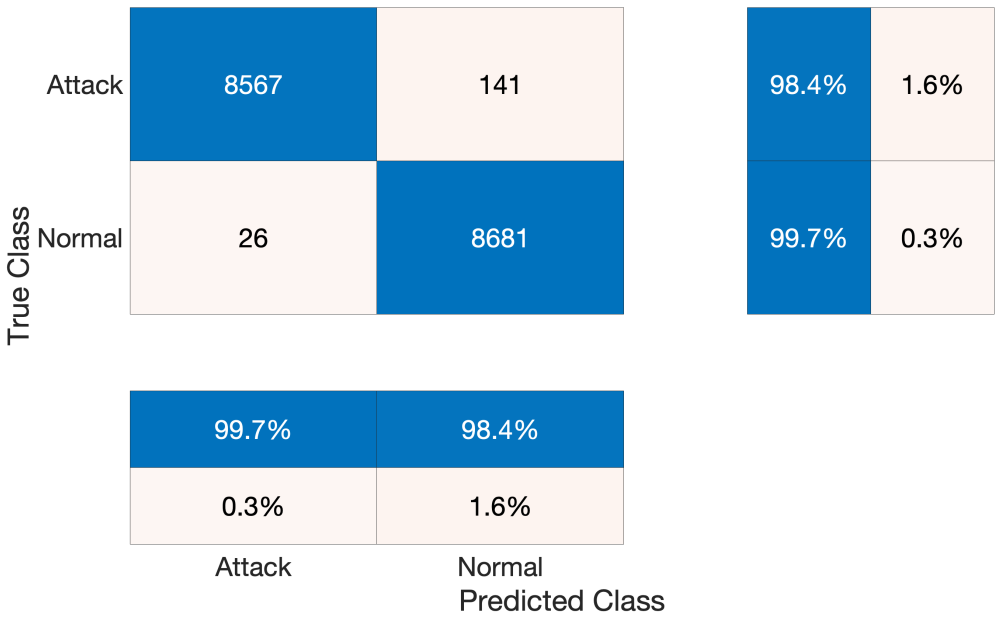

Figure 6.15. IDS 2012 - Confusion Matrix.

The performance of the proposed feature to characterize various attack types and the legitimate flows is evaluated with the trained multi-class SVM. There are Brute Force, DDoS, DoS, and Intrafiltrating attacks in this dataset. Therefore, a 5-class SVM classifier was trained. In the Multi-class SVM, the most misclassified classes are HTTP DoS and Infiltrating Attacks. These features do not show discriminating attributes for these classes. Specifically, network flows belonging to an HTTP DoS attack are labeled as infiltrating attacks. Considering missed HTTP DoS attack flows, 47% of HTTP DoS attack flows are misclassified. However, the detection rate of the attack flows is still high for the multi-class SVM case. Best performance is taken for the DDoS attack flows, and only 0.5% of the DDoS attack flows are mislabeled. The general separation capability of the proposed features is relatively high, except for one class.

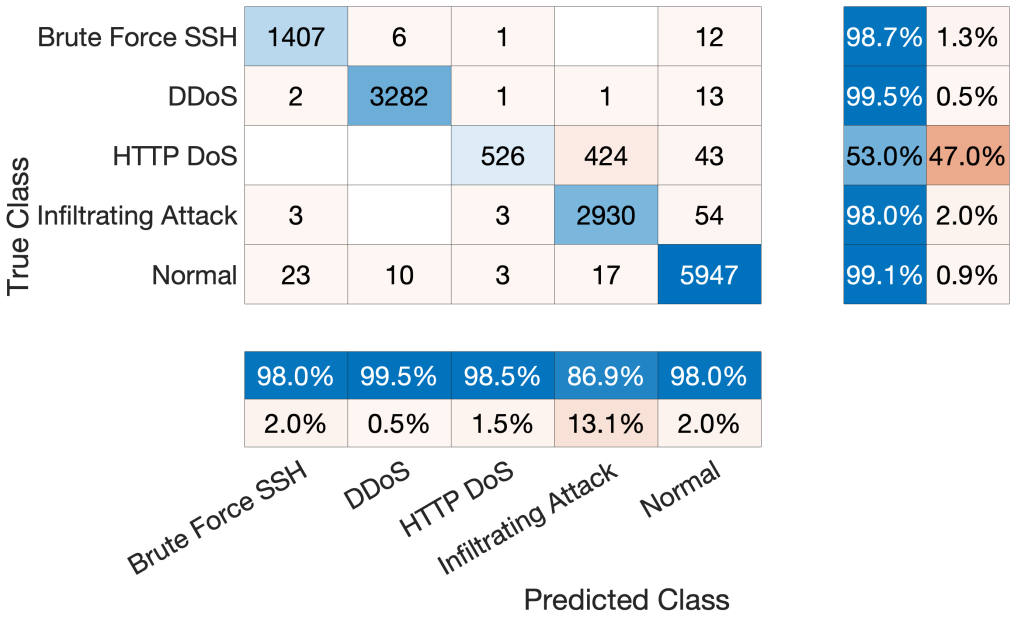

Figure 6.16. IDS 2012 - Multi Class - Confusion Matrix.

6.2.3.2. IDS 2017 Dataset. The IDS 2017 dataset contains 9 different attack types, which are given in Table 6.2. There are two different types of brute force attacks, which are done via SSH and FTP protocols. Network activity captured on Wednesday contains different types of DoS attacks, such as DoS Hulk and DoS Slowhttptest, having completely different attributes from each other. The data captured on Friday contains three different attack types: DDoS, Bot, and PortScan. Therefore, this dataset contains a variety of attacks that are performed with particular objectives.

Although this dataset contains various attacks, extracted features are able to diversify attack flows from the usual user activities. A one-class SVM is trained by assigning all attacks to one general "Attack" class. It helps to evaluate the real intrusion detection performance of the features. Besides that, multi-class SVM is trained to identify attack types.

All attack flows in the IDS 2017 dataset are detected by the proposed payload-based intrusion detection system. Performance evaluations are given for 3, 5, and 10 packet cases in Table 6.17. It is seen that the lowest performance is obtained in the 3 packet case, particularly for some attack cases, such as DDOS LOIC. Detection performance for extracted features using 3 packet payload information is weak, as shown in Case A. Similarly, using 10 packets to extract features can reduce the detection rate for some attacks such as DoS Slowloris and GoldenEye. According to the attributes of attacks, the optimal number of packets can be diversified attack by attack. The best performance was obtained in the 5-packet case overall. Although it costs more to delay detection than a 3-packet case, it provides better precision and recall rates. It also provides early detection and better detection rates compared with the 10-packet case.

Table 6.17. IDS 2017 - Performance Evaluation.

	Accuracy	**Specificity**	**Precision**	**Recall**	**FalsePositive Rate**	**F1-Score**
3 Packet	0.9761	0.9598	0.9610	0.9923	0.0402	0.9764
5 Packet	0.9803	0.9634	0.9646	0.9972	0.0366	0.9806
10 Packet	0.9789	0.9607	0.9621	0.9971	0.0393	0.9793

Detailed analysis of the 5-packet case is given as a complexity matrix in Figure 6.17. The training dataset contains a total of 23467 samples for each class. The number of missed attack flows is 68, while its ratio corresponds to 0.3% of total flows belonging to intrusion activity. 96.3% of legitimate flows are correctly labeled, while the 3.7% of them cause false alarms. The total error rate in this scenario is equal to 2%.

All attacks in the IDS 2017 dataset are grouped into one "Attack" class, and the performance of the one-class SVM is analyzed. The detection of the attack type

is critical to taking countermeasures against the attacker. It enables us to reduce the damage of the attack and preserve the security of the information. In order to evaluate the performance in distinguishing different attacks, a multi-class SVM classifier is trained by using both attack and normal network flows. There are 9 different types of network attacks and a legitimate class, then 10-class SVM classifier is trained. The complexity matrix for the multi-class SVM trained with the features extracted using 5 packets in a flow is given in Figure 6.18. In the complexity matrix, it can be seen that 231 network flows belonging to the Bot attack are labeled as a PortScan attack. The misclassification ratio of the Bot attack flows is very high at 66.8%. Proposed features can not differentiate Bot Ares attack flows and PortScan flows. Second, most misclassification is observed between DDoS and Normal network flows. Both classes have a large number of training and test flows, and this ratio is 1.4% of total DDoS flows. 0.6% of the overall attack flows is labeled as legitimate flows, and they are missed. Although the trained multi-class SVM contains 10 different attack classes, the total number of missed attack flows is 34. The proposed features efficiently characterize FTP-Patator attack flows, and there are no missed flows. At the same time, there are no flows that belong to the other classes and are predicted as FTP-Patator. It can be considered the best-performed class.

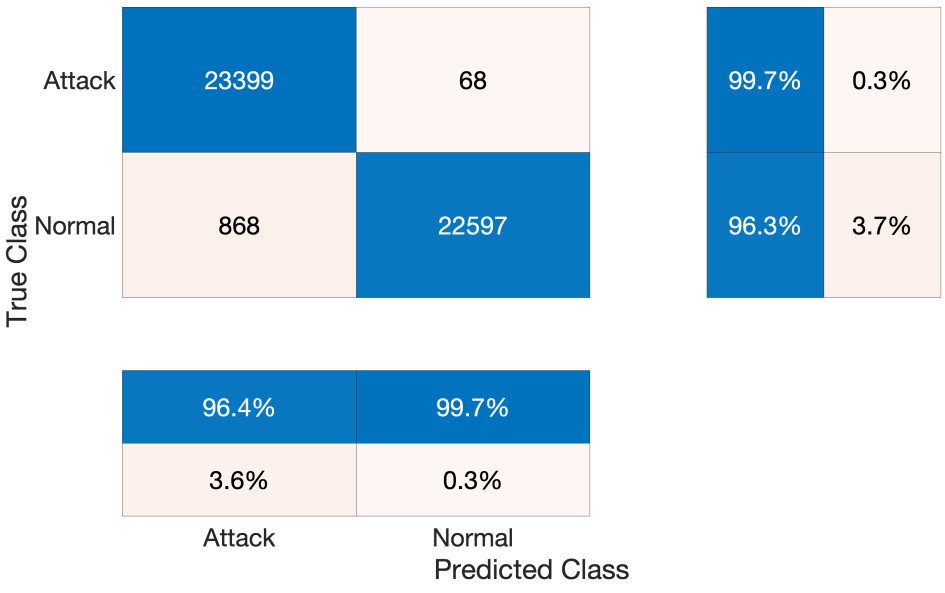

Figure 6.17. IDS 2017 - Confusion Matrix.

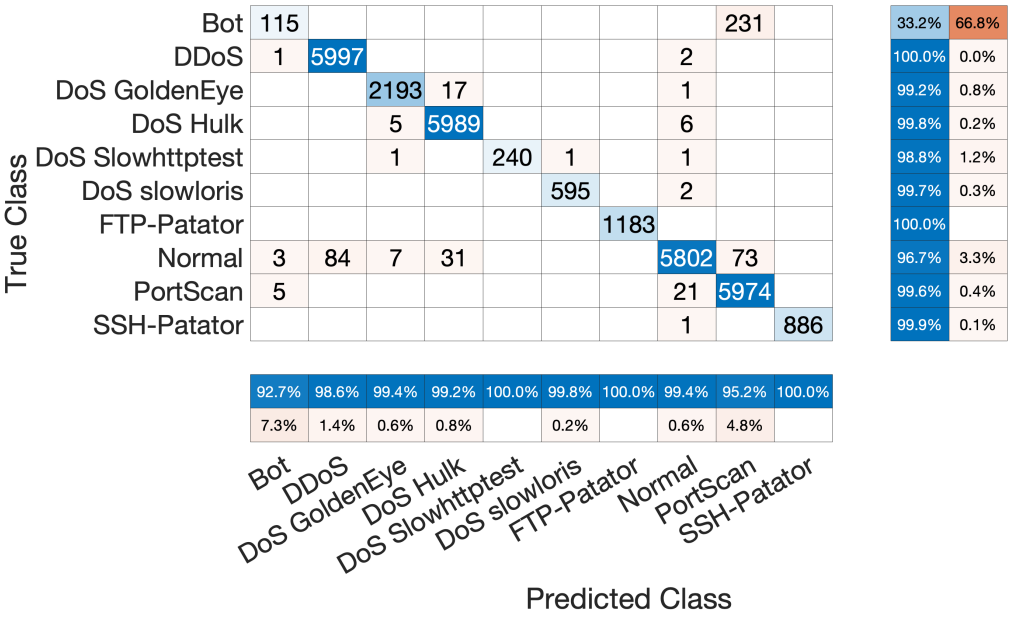

Figure 6.18. IDS 2017 - Confusion Matrix.

7. CONCLUSION

Early detection of network attacks is crucial to provide security of network systems against intrusions affecting users in various ways. Early detection of malicious connections enables us to prevent the potential loss caused by these activities. Detecting whether there are any attacks on the network is an essential operation to take the necessary precautions. Besides that, distinguishing the legitimate connections from the attackers is significant work for the system to continue to serve. Analyzing network packet payload is an approach to identify whether there is an attack on the network and determine which connections belong to the attack. Its primary reason is that the construction way of packet payloads of legitimate flows and intrusions have distinct patterns in their payloads. Moreover, imitating the characteristics of the legitimate packet payloads on the network is quite a challenging task for the attacker side. Therefore, packet payload analysis is significant in characterizing and classifying network flows.

This thesis evaluates the performance of features conventionally used in application classification in detecting attacks on the network. Proposed features reveal both patterns in the byte distribution within the packet and patterns between packets. Greedy algorithm-based metrics allow comparing defined probability distributions over different sample spaces at various lengths to analyze the attributes of these distributions and characterize the network. Besides that, new features are proposed to capture more complicated patterns in payload vectors. In this respect, spectral-domain analysis of the payload sequences allows us to extract features to classify normal and attack flows more accurately. Discrete Cosine Transform coefficients of payload sequences, commonly used in image processing to classify and extract summarizing information, are utilized to characterize network flows. These features are extracted from the packet payload sequence obtained with N-gram analysis for different N values. Also, these features are calculated for different segments of the network flow to characterize each flow more effectively.

This thesis analyzes the efficiency of these payload-based features to separate different types of attacks. These features are extracted for each flow on the network using different packet numbers. In our analysis, flows with 3, 5, and 10 packets are separately used for the feature extraction. There is a large number of features that can be used to model the characteristics of both legitimate and attack flows. All of these features are used in the classification process to train and test datasets. A one-class and multi-class support vector machine (SVM) algorithm are used in the classification stage. The classification performance of the SVM model trained with these features was evaluated on the IDS 2012 and IDS 2017 datasets, which are widely used in the literature. Performance analysis of the SVM algorithm is given in detail for both datasets, which contain various types of network attacks.

The performance of the SVM algorithm trained with proposed payload-based features that exploit attributes of flows is tested for mainly six different types of attack. Some of these attacks are executed in the simulation environment via different tools and methods. These attacks are also considered different attack types, and performance evaluation is executed for 13 different attack types. High accuracy rates are achieved against these attack types even when the features extracted from 3-packet flows are used. These 3-packet features provide early detection of an attack with high accuracy rates. However, 5 and 10 packet cases generally improve the performance of detection. When the trade-off between early detection and detection accuracy is considered, 5-packet cases are generally preferable ones for all attack types. The 5-packet features provide high precision and recall rates alongside early detection of attack flows.

In this thesis, all features are used in the training stage of the classification model. Instead of calculating all the features for the real-time application, only some of the discriminating features can be used according to the requirements of the network system. In this thesis, it is shown that network payloads can be used as an identifier of network attacks. Features indicating the similarities between payload byte distributions are valuable for modeling network flows. Frequency domain analysis and discrete cosine transform, previously not applied in this domain, provide a crucial contribution to detecting network intrusions.

REFERENCES

1. Cisco, U., "Cisco Annual Internet Report (2018–2023) White Paper", *Cisco: San Jose, CA, USA*, 2020.

2. Specht, S. and R. Lee, "Taxonomies of Distributed Denial of Service Networks, Attacks, Tools and Countermeasures", *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

3. Roy, S., C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya and Q. Wu, "A Survey of Game Theory as Applied to Network Security", *2010 43rd Hawaii International Conference on System Sciences*, pp. 1–10, Institute of Electrical and Electronics Engineers, 2010.

4. Vidyasagar, M., "A Metric Between Probability Distributions on Finite Sets of Different Cardinalities and Applications to Order Reduction", *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, Vol. 57, No. 10, pp. 2464–2477, 2012.

5. Mirkovic, J. and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms", *Association for Computing Machinery Special Interest Group on Data, Computer Communication Review*, Vol. 34, No. 2, pp. 39–53, 2004.

6. Sharafaldin, I., A. H. Lashkari, S. Hakak and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", *2019 International Carnahan Conference on Security Technology*, pp. 1–8, Institute of Electrical and Electronics Engineers, 2019.

7. Gupta, B. B. and O. P. Badve, "Taxonomy of DoS and DDoS Attacks and Desirable Defense Mechanism in a Cloud Computing Environment", *Neural Computing and Applications*, Vol. 28, No. 12, pp. 3655–3682, 2017.

8. Dantas Silva, F. S., E. Silva, E. P. Neto, M. Lemos, A. J. Venancio Neto and F. Esposito, "A Taxonomy of DDoS Attack Mitigation Approaches Featured by SDN Technologies in IoT Scenarios", *Sensors*, Vol. 20, No. 11, pp. 3078–3084, 2020.

9. Warburton, D., *DDoS Attack Trends for 2020*, 2020, `https://www.f5.com/labs/articles/threat-intelligence`, accessed in June 2021.

10. Kaspersky, *Distributed Denial of Service: Anatomy and Impact of DDoS Attacks*, 2021, `https://usa.kaspersky.com/resource-center/preemptive-safety`, accessed in June 2021.

11. Postel, J., "Transmission Control Protocol", *Defense Advanced Research Projects Agency Internet Program Protocol Specification*, 2007.

12. Adi, E., *Denial-of-service attack modelling and detection for HTTP/2 services*, Ph.D. Thesis, Edith Cowan University, Research Online, Perth, Western Australia, 2017.

13. Criscuolo, P. J., *Distributed Denial of Service: Trin00, Tribe Flood nNetwork, Tribe Flood Network 2000, and Stacheldraht ciac-2319*, Tech. rep., California Univ Livermore Radiation Lab, 2000.

14. Singh, J. and V. Grewal, "A Survey of Different Strategies to Pacify ARP Poisoning Attacks in Wireless Networks", *International Journal of Computer Applications*, Vol. 116, No. 11, 2015.

15. Marchette, D. J. and D. Marchette, *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*, Springer, 2001.

16. Bujlow, T., V. Carela-Español and P. Barlet-Ros, *Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification*, 2014,

https://vbn.aau.dk/en/publications/extended-independent-comparison, accessed in September 2021.

17. Tavallaee, M., E. Bagheri, W. Lu and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Dataset", *2009 Institute of Electrical and Electronics Engineers Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, Institute of Electrical and Electronics Engineers, 2009.

18. Lau, F., S. H. Rubin, M. H. Smith and L. Trajkovic, "Distributed Denial of Service Attacks", *Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions*, Vol. 3, pp. 2275–2280, Institute of Electrical and Electronics Engineers, 2000.

19. Cambiaso, E., G. Papaleo, G. Chiola and M. Aiello, "Slow DoS Attacks: Definition and Categorisation", *International Journal of Trust Management in Computing and Communications*, Vol. 1, No. 3-4, pp. 300–319, 2013.

20. Sharafaldin, I., A. H. Lashkari and A. A. Ghorbani, "Toward Generating A New Intrusion Detection Dataset and Intrusion Traffic Characterization", *International Conference on Information Systems Security and Privacy*, Vol. 1, pp. 108–116, 2018.

21. Mathur, N. O., *Application of Autoencoder Ensembles in Anomaly and Intrusion Detection using Time-based Analysis*, Ph.D. Thesis, University of Cincinnati, 2020.

22. Vinayakumar, R., M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System", *Institute of Electrical and Electronics Engineers Access*, Vol. 7, pp. 41525–41550, 2019.

23. Hofstede, R., M. Jonker, A. Sperotto and A. Pras, "Flow-based Web Application Brute-Force Attack and Compromise Detection", *Journal of Network and Systems Management*, Vol. 25, No. 4, pp. 735–758, 2017.

24. Gadge, J. and A. A. Patil, "Port Scan Detection", *2008 16th Institute of Electrical and Electronics Engineers International Conference on Networks*, pp. 1–6, Institute of Electrical and Electronics Engineers, 2008.

25. Kamboj, P., M. C. Trivedi, V. K. Yadav and V. K. Singh, "Detection Techniques of DDoS Attacks: A Survey", *2017 4th Institute of Electrical and Electronics Engineers Uttar Pradesh Section International Conference on Electrical, Computer and Electronics*, pp. 675–679, Institute of Electrical and Electronics Engineers, 2017.

26. Mahjabin, T., Y. Xiao, G. Sun and W. Jiang, "A Survey of Distributed Denial-of-Service Attack, Prevention, and Mitigation Techniques", *International Journal of Distributed Sensor Networks*, Vol. 13, No. 12, pp. 1–33, 2017.

27. Lima Filho, F. S. d., F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar and L. F. Silveira, "Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning", *Security and Communication Networks*, Vol. 2019, 2019.

28. Jin, S. and D. S. Yeung, "A Covariance Analysis Model for DDoS Attack Detection", *2004 Institute of Electrical and Electronics Engineers International Conference on Communications*, Vol. 4, pp. 1882–1886, Institute of Electrical and Electronics Engineers, 2004.

29. Koay, A., A. Chen, I. Welch and W. K. Seah, "A New Multi Classifier System Using Entropy-based Features in DDoS Attack Detection", *2018 International Conference on Information Networking*, pp. 162–167, Institute of Electrical and Electronics Engineers, 2018.

30. Idhammad, M., K. Afdel and M. Belouch, "Detection System of HTTP DDoS attacks in a Cloud Environment based on Information Theoretic Entropy and Random Forest", *Security and Communication Networks*, Vol. 2018, 2018.

31. Fouladi, R. F., C. E. Kayatas and E. Anarim, "Frequency based DDoS Attack

Detection Approach Using Naive Bayes Classification", *2016 39th International Conference on Telecommunications and Signal Processing*, pp. 104–107, Institute of Electrical and Electronics Engineers, 2016.

32. Erhan, D. and E. Anarım, "Boğaziçi University Distributed Denial of Service Dataset", *Data in Brief*, Vol. 32, pp. 106187–106188, 2020.

33. Santanna, J. J., R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville and A. Pras, "Booters—An Analysis of DDoS-as-a-Service Attacks", *2015 Institute of Electrical and Electronics Engineers International Symposium on Integrated Network Management*, pp. 243–251, Institute of Electrical and Electronics Engineers, 2015.

34. Ramadas, M., S. Ostermann and B. Tjaden, "Detecting Anomalous Network Traffic with Self-Organizing Maps", *International Workshop on Recent Advances in Intrusion Detection*, pp. 36–54, Springer, 2003.

35. Zhang, Z., J. Li, C. Manikopoulos, J. Jorgenson and J. Ucles, "HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification", *Institute of Electrical and Electronics Engineers Workshop on Information Assurance and Security*, Vol. 85, pp. 90–94, 2001.

36. Aygun, R. C. and A. G. Yavuz, "Network Anomaly Detection with Stochastically Improved Autoencoder based Models", *2017 Institute of Electrical and Electronics Engineers 4th International Conference on Cyber Security and Cloud Computing*, pp. 193–198, Institute of Electrical and Electronics Engineers, 2017.

37. Chen, Z., C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based Network Anomaly Detection", *2018 Wireless Telecommunications Symposium*, pp. 1–5, Institute of Electrical and Electronics Engineers, 2018.

38. Yuan, X., C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning", *2017 Institute of Electrical and Electronics Engineers International*

Conference on Smart Computing, pp. 1–8, Institute of Electrical and Electronics Engineers, 2017.

39. Wu, K., Z. Chen and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks", *Institute of Electrical and Electronics Engineers Access*, Vol. 6, pp. 50850–50859, 2018.

40. Vinayakumar, R., K. Soman and P. Poornachandran, "Applying Convolutional Neural Network for Network Intrusion Detection", *2017 International Conference on Advances in Computing, Communications and Informatics*, pp. 1222–1228, Institute of Electrical and Electronics Engineers, 2017.

41. McLaughlin, N., J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé *et al.*, "Deep Android Malware Detection", *Proceedings of the Seventh Association for Computing Machinery on Conference on Data and Application Security and Privacy*, pp. 301–308, 2017.

42. Wang, W., M. Zhu, X. Zeng, X. Ye and Y. Sheng, "Malware Traffic Classification Using Convolutional Neural Network for Representation Learning", *2017 International Conference on Information Networking*, pp. 712–717, Institute of Electrical and Electronics Engineers, 2017.

43. Kim, T., B. Kang, M. Rho, S. Sezer and E. G. Im, "A Multimodal Deep Learning Method for Android Malware Detection Using Various Features", *Institute of Electrical and Electronics Engineers Transactions on Information Forensics and Security*, Vol. 14, No. 3, pp. 773–788, 2018.

44. Doriguzzi-Corin, R., S. Millar, S. Scott-Hayward, J. Martinez-del Rincon and D. Siracusa, "LUCID: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection", *Institute of Electrical and Electronics Engineers Transactions on Network and Service Management*, Vol. 17, No. 2, pp. 876–889, 2020.

45. Hamamoto, A. H., L. F. Carvalho, L. D. H. Sampaio, T. Abrão and M. L.

Proença Jr, "Network Anomaly Detection System Using Genetic Algorithm and Fuzzy Logic", *Expert Systems with Applications*, Vol. 92, pp. 390–402, 2018.

46. Matias, R., A. M. Carvalho, L. B. Araujo and P. R. Maciel, "Comparison Analysis of Statistical Control Charts for Quality Monitoring of Network Traffic Forecasts", *2011 Institute of Electrical and Electronics Engineers International Conference on Systems, Man, and Cybernetics*, pp. 404–409, Institute of Electrical and Electronics Engineers, 2011.

47. Wang, K. and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection", *International Workshop on Recent Advances in Intrusion Detection*, pp. 203–222, Springer, 2004.

48. Wang, K., J. J. Parekh and S. J. Stolfo, "Anagram: A Content Anomaly Detector Resistant to Mimicry Attack", *International Workshop on Recent Advances in Intrusion Detection*, pp. 226–248, Springer, 2006.

49. Bolzoni, D., S. Etalle and P. Hartel, "Poseidon: A 2-Tier Anomaly-based Network Intrusion Detection System", *Fourth Institute of Electrical and Electronics Engineers International Workshop on Information Assurance*, pp. 10–20, Institute of Electrical and Electronics Engineers, 2006.

50. Hubballi, N., S. Biswas and S. Nandi, "Layered Higher Order N-Grams for Hardening Payload based Anomaly Intrusion Detection", *2010 International Conference on Availability, Reliability and Security*, pp. 321–326, Institute of Electrical and Electronics Engineers, 2010.

51. Swarnkar, M. and N. Hubballi, "OCPAD: One Class Naive Bayes Classifier for Payload based Anomaly Detection", *Expert Systems with Applications*, Vol. 64, pp. 330–339, 2016.

52. Olivain, J. and J. Goubault-Larrecq, *Detecting Subverted Cryptographic Protocols by Entropy Checking*, Ph.D. Thesis, LSV, ENS Cachan, 2006.

53. Bolthausen, E. and M. V. Wüthrich, "Bernoulli's Law of Large Numbers", *ASTIN Bulletin: The Journal of the International Academic Association*, Vol. 43, No. 2, pp. 73–79, 2013.

54. Cover, T. M., *Elements of Information Theory*, John Wiley & Sons, 1999.

55. Hammersley, J., *Monte Carlo Methods*, Methuen, London, 1964.

56. Damashek, M., "Gauging Similarity with N-Grams: Language-Independent Categorization of Text", *Science*, Vol. 267, No. 5199, pp. 843–848, 1995.

57. Kohonen, T., "Self-Organizing Maps, Ser", *Information Sciences Berlin: Springer*, Vol. 30, 2001.

58. Mamun, M. S. I., A. A. Ghorbani and N. Stakhanova, "An Entropy-based Encrypted Traffic Classifier", *International Conference on Information and Communications Security*, pp. 282–294, Springer, 2015.

59. Dorfinger, P., G. Panholzer and W. John, "Entropy Estimation for Real-Time Encrypted Traffic Identification", *International Workshop on Traffic Monitoring and Analysis*, pp. 164–171, Springer, 2011.

60. Casino, F., K.-K. R. Choo and C. Patsakis, "HEDGE: Efficient Traffic Classification of Encrypted and Compressed Packets", *Institute of Electrical and Electronics Engineers Transactions on Information Forensics and Security*, Vol. 14, No. 11, pp. 2916–2926, 2019.

61. Wang, Y., Z. Zhang, L. Guo and S. Li, "Using Entropy to Classify Traffic More Deeply", *2011 Institute of Electrical and Electronics Engineers Sixth International Conference on Networking, Architecture, and Storage*, pp. 45–52, Institute of Electrical and Electronics Engineers, 2011.

62. Velan, P., M. Čermák, P. Čeleda and M. Drašar, "A Survey of Methods for

Encrypted Traffic Classification and Analysis", *International Journal of Network Management*, Vol. 25, No. 5, pp. 355–374, 2015.

63. Dorfinger, P., G. Panholzer, B. Trammell and T. Pepe, "Entropy-based Traffic Filtering to Support Real-time Skype Detection", *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pp. 747–751, 2010.

64. Zhang, H., C. Papadopoulos and D. Massey, "Detecting Encrypted Botnet Traffic", *2013 Proceedings Institute of Electrical and Electronics Engineers International Conference on Computer Communications*, pp. 3453–1358, Institute of Electrical and Electronics Engineers, 2013.

65. Białczak, P. and W. Mazurczyk, "Characterizing Anomalies in Malware-generated HTTP Traffic", *Security and Communication Networks*, Vol. 2020, 2020.

66. Thorat, S. A., A. K. Khandelwal, B. Bruhadeshwar and K. Kishore, "Payload Content-based Network Anomaly Detection", *2008 First International Conference on the Applications of Digital Information and Web Technologies*, pp. 127–132, Institute of Electrical and Electronics Engineers, 2008.

67. Ariu, D. and G. Giacinto, "HMMPayl: an Application of HMM to the Analysis of the HTTP Payload", *Proceedings of the First Workshop on Applications of Pattern Analysis*, pp. 81–87, Proceedings of Machine Learning Research, 2010.

68. Yang, C., F. Wang and B. Huang, "Internet Traffic Classification Using dbscan", *2009 WASE International Conference on Information Engineering*, Vol. 2, pp. 163–166, Institute of Electrical and Electronics Engineers, 2009.

69. Moore, A., D. Zuev and M. Crogan, *Discriminators for Use in Flow-based Classification*, Tech. rep., 2013.

70. Heckbert, P., "Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm", *Computer Graphics*, Vol. 2, pp. 15–463, 1995.

71. Smith, J. O., *Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications*, Julius Smith, 2007.

72. Frigo, M. and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT", *Proceedings of the 1998 Institute of Electrical and Electronics Engineers International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 1381–1384, Institute of Electrical and Electronics Engineers, 1998.

73. Frigo, M. and S. G. Johnson, "The Design and Implementation of FFTW3", *Proceedings of the Institute of Electrical and Electronics Engineers*, Vol. 93, No. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

74. Frigo, M. and S. G. Johnson, *The Fastest Fourier Transform in the West*, Tech. Rep. MIT-LCS-TR-728, Massachusetts Institute of Technology, 1997.

75. Frigo, M. and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT", *1998 Institute of Electrical and Electronics Engineers International Conference Acoustics Speech and Signal Processing*, Vol. 3, pp. 1381–1384, Institute of Electrical and Electronics Engineers, 1998.

76. Frigo, M., "A Fast Fourier Transform Compiler", *Proceedings 1999 of Association for Computing Machinery's Special Interest Group Conference on Programming Language Design and Implementation*, Vol. 34, pp. 169–180, Association for Computing Machinery, 1999.

77. Phinyomark, A., S. Thongpanja, H. Hu, P. Phukpattaranont and C. Limsakul, "The Usefulness of Mean and Median Frequencies in Electromyography Analysis", *Computational Intelligence in Electromyography Analysis-A Perspective on Current Applications and future Challenges*, pp. 195–220, 2012.

78. Oskoei, M. A. and H. Hu, "Support Vector Machine-based Classification Scheme for Myoelectric Control Applied to Upper Limb", *Institute of Electrical and Electronics*

Engineers Transactions on Biomedical Engineering, Vol. 55, No. 8, pp. 1956–1965, 2008.

79. Thongpanja, S., A. Phinyomark, P. Phukpattaranont and C. Limsakul, "A Feasibility Study of Fatigue and Muscle Contraction Indices based on EMG Time-Dependent Spectral Analysis", *Procedia Engineering*, Vol. 32, pp. 239–245, 2012.

80. Liu, L., B. Liu, H. Huang and A. C. Bovik, "No-reference Image Quality Assessment based on Spatial and Spectral Entropies", *Signal processing: Image communication*, Vol. 29, No. 8, pp. 856–863, 2014.

81. Asgharzadeh-Bonab, A., M. C. Amirani and A. Mehri, "Spectral Entropy and Deep Convolutional Neural Network for ECG Beat Classification", *Biocybernetics and Biomedical Engineering*, Vol. 40, No. 2, pp. 691–700, 2020.

82. Ahmed, N., T. Natarajan and K. R. Rao, "Discrete Cosine Transform", *Institute of Electrical and Electronics Engineers transactions on Computers*, Vol. 100, No. 1, pp. 90–93, 1974.

83. Benhassine, N. E., A. Boukaache and D. Boudjehem, "Classification of Mammogram Images Using the Energy Probability in Frequency Domain and Most Discriminative Power Coefficients", *International Journal of Imaging Systems and Technology*, Vol. 30, No. 1, pp. 45–56, 2020.

84. Shiravi, A., H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection", *Computers & Security*, Vol. 31, No. 3, pp. 357–374, 2012.

85. McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks", *Association for Computing Machinery's Special Interest Group on Data Communications, Computer Communication Review*, Vol. 38, No. 2, pp. 69–74, 2008.

86. Shramos, *Pcap Splitter*, 2020, `https://github.com/shramos/pcap-splitter`, accessed in September 2021.

87. Vural, V. and J. G. Dy, "A Hierarchical Method for Multi-Class Support Vector Machines", *Proceedings of the Twenty-first International Conference on Machine Learning*, pp. 105–112, 2004.

88. Mountrakis, G., J. Im and C. Ogole, "Support Vector Machines in Remote Sensing: A Review", *International Society for Photogrammetry and Remote Sensing Journal of Photogrammetry and Remote Sensing*, Vol. 66, No. 3, pp. 247–259, 2011.

89. James, G., D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning*, Vol. 112, Springer, 2013.

90. Lin, H.-T. and C.-J. Lin, "A Study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels by SMO-Type Methods", *Neural Computation*, Vol. 3, No. 1-32, pp. 16–24, 2003.

APPENDIX A: FLOW BASED PAYLOAD FEATURE LIST

Table A.1. Flow Based Payload Feature Name List.

No	**Feature Names**
1	N=1 - General Ratio of Printable Characters - Mean
2	N=1 - General Ratio of Printable Characters - Min
3	N=1 - General Ratio of Printable Characters - Max
4	N=1 - General Ratio of Printable Characters - Std
5	N=1 - General - Ratio of Unique Bytes - Mean
6	N=1 - General - Ratio of Unique Bytes - Min
7	N=1 - General - Ratio of Unique Bytes - Max
8	N=1 - General - Ratio of Unique Bytes - Std
9	N=1 - General - Entropy - Mean
10	N=1 - General - Entropy - Min
11	N=1 - General - Entropy - Max
12	N=1 - General - Entropy - Std
13	N=1 - General - Actual Entropy - Mean
14	N=1 - General - Actual Entropy - Min
15	N=1 - General - Actual Entropy - Max
16	N=1 - General - Actual Entropy - Std
17	N=1 - General - Greedy Distance of Packets - Mean
18	N=1 - General - Greedy Distance of Packets - Min
19	N=1 - General - Greedy Distance of Packets - Max
20	N=1 - General - Greedy Distance of Packets - Std
21	N=1 - General - Mean Frequency - Mean
22	N=1 - General - Mean Frequency - Min
23	N=1 - General - Mean Frequency - Max
24	N=1 - General - Mean Frequency - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

Header 1	**Header 2**
25	N=1 - General - Peak Frequency - Mean
26	N=1 - General - Peak Frequency - Min
27	N=1 - General - Peak Frequency - Max
28	N=1 - General - Peak Frequency - Std
29	N=1 - General - Spectral Entropy - Mean
30	N=1 - General - Spectral Entropy - Min
31	N=1 - General - Spectral Entropy - Max
32	N=1 - General - Spectral Entropy - Std
33	N=1 - General - Greedy Distance of PSDs - Mean
34	N=1 - General - Greedy Distance of PSDs - Min
35	N=1 - General - Greedy Distance of PSDs - Max
36	N=1 - General - Greedy Distance of PSDs - Std
37	N=1 - General - DCT Coefficient 1 - Mean
38	N=1 - General - DCT Coefficient 1 - Min
39	N=1 - General - DCT Coefficient 1 - Max
40	N=1 - General - DCT Coefficient 1 - Std
41	N=1 - General - DCT Coefficient 2 - Mean
42	N=1 - General -DCT Coefficient 2 - Min
43	N=1 - General - DCT Coefficient 2 - Max
44	N=1 - General - DCT Coefficient 2 - Std
45	N=1 - General - DCT Coefficient 3 - Mean
46	N=1 - General -DCT Coefficient 3 - Min
47	N=1 - General - DCT Coefficient 3 - Max
48	N=1 - General - DCT Coefficient 3 - Std
49	N=1 - Forward - Ratio of Printable Characters - Mean
50	N=1 - Forward - Ratio of Printable Characters - Min
51	N=1 - Forward - Ratio of Printable Characters - Max
52	N=1 - Forward - Ratio of Printable Characters - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
53	N=1 - Forward - Ratio of Unique Bytes - Mean
54	N=1 - Forward - Ratio of Unique Bytes - Min
55	N=1 - Forward - Ratio of Unique Bytes - Max
56	N=1 - Forward - Ratio of Unique Bytes - Std
57	N=1 - Forward - Entropy - Mean
58	N=1 - Forward - Entropy - Min
59	N=1 - Forward - Entropy - Max
60	N=1 - Forward - Entropy - Std
61	N=1 - Forward - Actual Entropy - Mean
62	N=1 - Forward - Actual Entropy - Min
63	N=1 - Forward - Actual Entropy - Max
64	N=1 - Forward - Actual Entropy - Std
65	N=1 - Forward - Greedy Distance of Packets - Mean
66	N=1 - Forward - Greedy Distance of Packets - Min
67	N=1 - Forward - Greedy Distance of Packets - Max
68	N=1 - Forward - Greedy Distance of Packets - Std
69	N=1 - Forward - Mean Frequency - Mean
70	N=1 - Forward - Mean Frequency - Min
71	N=1 - Forward - Mean Frequency - Max
72	N=1 - Forward - Mean Frequency - Std
73	N=1 - Forward - Peak Frequency - Mean
74	N=1 - Forward - Peak Frequency - Min
75	N=1 - Forward - Peak Frequency - Max
76	N=1 - Forward - Peak Frequency - Std
77	N=1 - Forward - Spectral Entropy - Mean
78	N=1 - Forward - Spectral Entropy - Min
79	N=1 - Forward - Spectral Entropy - Max
80	N=1 - Forward - Spectral Entropy - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
81	N=1 - Forward - Greedy Distance of PSDs - Mean
82	N=1 - Forward - Greedy Distance of PSDs - Min
83	N=1 - Forward - Greedy Distance of PSDs - Max
84	N=1 - Forward - Greedy Distance of PSDs - Std
85	N=1 - Forward - DCT Coefficient 1 - Mean
86	N=1 - Forward - DCT Coefficient 1 - Min
87	N=1 - Forward - DCT Coefficient 1 - Max
88	N=1 - Forward - DCT Coefficient 1 - Std
89	N=1 - Forward - DCT Coefficient 2 - Mean
90	N=1 - Forward -DCT Coefficient 2 - Min
91	N=1 - Forward - DCT Coefficient 2 - Max
92	N=1 - Forward - DCT Coefficient 2 - Std
93	N=1 - Forward - DCT Coefficient 3 - Mean
94	N=1 - Forward -DCT Coefficient 3 - Min
95	N=1 - Forward - DCT Coefficient 3 - Max
96	N=1 - Forward - DCT Coefficient 3 - Std
97	N=1 - Backward - Ratio of Printable Characters - Mean
98	N=1 - Backward - Ratio of Printable Characters - Min
99	N=1 - Backward - Ratio of Printable Characters - Max
100	N=1 - Backward - Ratio of Printable Characters - Std
101	N=1 - Backward - Ratio of Unique Bytes - Mean
102	N=1 - Backward - Ratio of Unique Bytes - Min
103	N=1 - Backward - Ratio of Unique Bytes - Max
104	N=1 - Backward - Ratio of Unique Bytes - Std
105	N=1 - Backward - Entropy - Mean
106	N=1 - Backward - Entropy - Min
107	N=1 - Backward - Entropy - Max
108	N=1 - Backward - Entropy - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
109	N=1 - Backward - Actual Entropy - Mean
110	N=1 - Backward - Actual Entropy - Min
111	N=1 - Backward - Actual Entropy - Max
112	N=1 - Backward - Actual Entropy - Std
113	N=1 - Backward - Greedy Distance of Packets - Mean
114	N=1 - Backward - Greedy Distance of Packets - Min
115	N=1 - Backward - Greedy Distance of Packets - Max
116	N=1 - Backward - Greedy Distance of Packets - Std
117	N=1 - Backward - Mean Frequency - Mean
118	N=1 - Backward - Mean Frequency - Min
119	N=1 - Backward - Mean Frequency - Max
120	N=1 - Backward - Mean Frequency - Std
121	N=1 - Backward - Peak Frequency - Mean
122	N=1 - Backward - Peak Frequency - Min
123	N=1 - Backward - Peak Frequency - Max
124	N=1 - Backward - Peak Frequency - Std
125	N=1 - Backward - Spectral Entropy - Mean
126	N=1 - Backward - Spectral Entropy - Min
127	N=1 - Backward - Spectral Entropy - Max
128	N=1 - Backward - Spectral Entropy - Std
129	N=1 - Backward - Greedy Distance of PSDs - Mean
130	N=1 - Backward - Greedy Distance of PSDs - Min
131	N=1 - Backward - Greedy Distance of PSDs - Max
132	N=1 - Backward - Greedy Distance of PSDs - Std
133	N=1 - Backward - DCT Coefficient 1 - Mean
134	N=1 - Backward - DCT Coefficient 1 - Min
135	N=1 - Backward - DCT Coefficient 1 - Max
136	N=1 - Backward - DCT Coefficient 1 - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
137	N=1 - Backward - DCT Coefficient 2 - Mean
138	N=1 - Backward -DCT Coefficient 2 - Min
139	N=1 - Backward - DCT Coefficient 2 - Max
140	N=1 - Backward - DCT Coefficient 2 - Std
141	N=1 - Backward - DCT Coefficient 3 - Mean
142	N=1 - Backward -DCT Coefficient 3 - Min
143	N=1 - Backward - DCT Coefficient 3 - Max
144	N=1 - Backward - DCT Coefficient 3 - Std
145	N=2 - General Ratio of Printable Characters - Mean
146	N=2 - General Ratio of Printable Characters - Min
147	N=2 - General Ratio of Printable Characters - Max
148	N=2 - General Ratio of Printable Characters - Std
149	N=2 - General - Ratio of Unique Bytes - Mean
150	N=2 - General - Ratio of Unique Bytes - Min
151	N=2 - General - Ratio of Unique Bytes - Max
152	N=2 - General - Ratio of Unique Bytes - Std
153	N=2 - General - Entropy - Mean
154	N=2 - General - Entropy - Min
155	N=2 - General - Entropy - Max
156	N=2 - General - Entropy - Std
157	N=2 - General - Actual Entropy - Mean
158	N=2 - General - Actual Entropy - Min
159	N=2 - General - Actual Entropy - Max
160	N=2 - General - Actual Entropy - Std
161	N=2 - General - Greedy Distance of Packets - Mean
162	N=2 - General - Greedy Distance of Packets - Min
163	N=2 - General - Greedy Distance of Packets - Max
164	N=2 - General - Greedy Distance of Packets - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
165	N=2 - General - Mean Frequency - Mean
166	N=2 - General - Mean Frequency - Min
167	N=2 - General - Mean Frequency - Max
168	N=2 - General - Mean Frequency - Std
169	N=2 - General - Peak Frequency - Mean
170	N=2 - General - Peak Frequency - Min
171	N=2 - General - Peak Frequency - Max
172	N=2 - General - Peak Frequency - Std
173	N=2 - General - Spectral Entropy - Mean
174	N=2 - General - Spectral Entropy - Min
175	N=2 - General - Spectral Entropy - Max
176	N=2 - General - Spectral Entropy - Std
177	N=2 - General - Greedy Distance of PSDs - Mean
178	N=2 - General - Greedy Distance of PSDs - Min
179	N=2 - General - Greedy Distance of PSDs - Max
180	N=2 - General - Greedy Distance of PSDs - Std
181	N=2 - General - DCT Coefficient 1 - Mean
182	N=2 - General - DCT Coefficient 1 - Min
183	N=2 - General - DCT Coefficient 1 - Max
184	N=2 - General - DCT Coefficient 1 - Std
185	N=2 - General - DCT Coefficient 2 - Mean
186	N=2 - General -DCT Coefficient 2 - Min
187	N=2 - General - DCT Coefficient 2 - Max
188	N=2 - General - DCT Coefficient 2 - Std
189	N=2 - General - DCT Coefficient 3 - Mean
190	N=2 - General -DCT Coefficient 3 - Min
191	N=2 - General - DCT Coefficient 3 - Max
192	N=2 - General - DCT Coefficient 3 - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
193	N=2 - Forward - Ratio of Printable Characters - Mean
194	N=2 - Forward - Ratio of Printable Characters - Min
195	N=2 - Forward - Ratio of Printable Characters - Max
196	N=2 - Forward - Ratio of Printable Characters - Std
197	N=2 - Forward - Ratio of Unique Bytes - Mean
198	N=2 - Forward - Ratio of Unique Bytes - Min
199	N=2 - Forward - Ratio of Unique Bytes - Max
200	N=2 - Forward - Ratio of Unique Bytes - Std
201	N=2 - Forward - Entropy - Mean
202	N=2 - Forward - Entropy - Min
203	N=2 - Forward - Entropy - Max
204	N=2 - Forward - Entropy - Std
205	N=2 - Forward - Actual Entropy - Mean
206	N=2 - Forward - Actual Entropy - Min
207	N=2 - Forward - Actual Entropy - Max
208	N=2 - Forward - Actual Entropy - Std
209	N=2 - Forward - Greedy Distance of Packets - Mean
210	N=2 - Forward - Greedy Distance of Packets - Min
211	N=2 - Forward - Greedy Distance of Packets - Max
212	N=2 - Forward - Greedy Distance of Packets - Std
213	N=2 - Forward - Mean Frequency - Mean
214	N=2 - Forward - Mean Frequency - Min
215	N=2 - Forward - Mean Frequency - Max
216	N=2 - Forward - Mean Frequency - Std
217	N=2 - Forward - Peak Frequency - Mean
218	N=2 - Forward - Peak Frequency - Min
219	N=2 - Forward - Peak Frequency - Max
220	N=2 - Forward - Peak Frequency - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
221	N=2 - Forward - Spectral Entropy - Mean
222	N=2 - Forward - Spectral Entropy - Min
223	N=2 - Forward - Spectral Entropy - Max
224	N=2 - Forward - Spectral Entropy - Std
225	N=2 - Forward - Greedy Distance of PSDs - Mean
226	N=2 - Forward - Greedy Distance of PSDs - Min
227	N=2 - Forward - Greedy Distance of PSDs - Max
228	N=2 - Forward - Greedy Distance of PSDs - Std
229	N=2 - Forward - DCT Coefficient 1 - Mean
230	N=2 - Forward - DCT Coefficient 1 - Min
231	N=2 - Forward - DCT Coefficient 1 - Max
232	N=2 - Forward - DCT Coefficient 1 - Std
233	N=2 - Forward - DCT Coefficient 2 - Mean
234	N=2 - Forward -DCT Coefficient 2 - Min
235	N=2 - Forward - DCT Coefficient 2 - Max
236	N=2 - Forward - DCT Coefficient 2 - Std
237	N=2 - Forward - DCT Coefficient 3 - Mean
238	N=2 - Forward -DCT Coefficient 3 - Min
239	N=2 - Forward - DCT Coefficient 3 - Max
240	N=2 - Forward - DCT Coefficient 3 - Std
241	N=2 - Backward - Ratio of Printable Characters - Mean
242	N=2 - Backward - Ratio of Printable Characters - Min
243	N=2 - Backward - Ratio of Printable Characters - Max
244	N=2 - Backward - Ratio of Printable Characters - Std
245	N=2 - Backward - Ratio of Unique Bytes - Mean
246	N=2 - Backward - Ratio of Unique Bytes - Min
247	N=2 - Backward - Ratio of Unique Bytes - Max
248	N=2 - Backward - Ratio of Unique Bytes - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
249	N=2 - Backward - Entropy - Mean
250	N=2 - Backward - Entropy - Min
251	N=2 - Backward - Entropy - Max
252	N=2 - Backward - Entropy - Std
253	N=2 - Backward - Actual Entropy - Mean
254	N=2 - Backward - Actual Entropy - Min
255	N=2 - Backward - Actual Entropy - Max
256	N=2 - Backward - Actual Entropy - Std
257	N=2 - Backward - Greedy Distance of Packets - Mean
258	N=2 - Backward - Greedy Distance of Packets - Min
259	N=2 - Backward - Greedy Distance of Packets - Max
260	N=2 - Backward - Greedy Distance of Packets - Std
261	N=2 - Backward - Mean Frequency - Mean
262	N=2 - Backward - Mean Frequency - Min
263	N=2 - Backward - Mean Frequency - Max
264	N=2 - Backward - Mean Frequency - Std
265	N=2 - Backward - Peak Frequency - Mean
266	N=2 - Backward - Peak Frequency - Min
267	N=2 - Backward - Peak Frequency - Max
268	N=2 - Backward - Peak Frequency - Std
269	N=2 - Backward - Spectral Entropy - Mean
270	N=2 - Backward - Spectral Entropy - Min
271	N=2 - Backward - Spectral Entropy - Max
272	N=2 - Backward - Spectral Entropy - Std
273	N=2 - Backward - Greedy Distance of PSDs - Mean
274	N=2 - Backward - Greedy Distance of PSDs - Min
275	N=2 - Backward - Greedy Distance of PSDs - Max
276	N=2 - Backward - Greedy Distance of PSDs - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
277	N=2 - Backward - DCT Coefficient 1 - Mean
278	N=2 - Backward - DCT Coefficient 1 - Min
279	N=2 - Backward - DCT Coefficient 1 - Max
280	N=2 - Backward - DCT Coefficient 1 - Std
281	N=2 - Backward - DCT Coefficient 2 - Mean
282	N=2 - Backward -DCT Coefficient 2 - Min
283	N=2 - Backward - DCT Coefficient 2 - Max
284	N=2 - Backward - DCT Coefficient 2 - Std
285	N=2 - Backward - DCT Coefficient 3 - Mean
286	N=2 - Backward -DCT Coefficient 3 - Min
287	N=2 - Backward - DCT Coefficient 3 - Max
288	N=2 - Backward - DCT Coefficient 3 - Std
289	N=3 - General Ratio of Printable Characters - Mean
290	N=3 - General Ratio of Printable Characters - Min
291	N=3 - General Ratio of Printable Characters - Max
292	N=3 - General Ratio of Printable Characters - Std
293	N=3 - General - Ratio of Unique Bytes - Mean
294	N=3 - General - Ratio of Unique Bytes - Min
295	N=3 - General - Ratio of Unique Bytes - Max
296	N=3 - General - Ratio of Unique Bytes - Std
297	N=3 - General - Entropy - Mean
298	N=3 - General - Entropy - Min
299	N=3 - General - Entropy - Max
300	N=3 - General - Entropy - Std
301	N=3 - General - Actual Entropy - Mean
302	N=3 - General - Actual Entropy - Min
303	N=3 - General - Actual Entropy - Max
304	N=3 - General - Actual Entropy - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
305	N=3 - General - Greedy Distance of Packets - Mean
306	N=3 - General - Greedy Distance of Packets - Min
307	N=3 - General - Greedy Distance of Packets - Max
308	N=3 - General - Greedy Distance of Packets - Std
309	N=3 - General - Mean Frequency - Mean
310	N=3 - General - Mean Frequency - Min
311	N=3 - General - Mean Frequency - Max
312	N=3 - General - Mean Frequency - Std
313	N=3 - General - Peak Frequency - Mean
314	N=3 - General - Peak Frequency - Min
315	N=3 - General - Peak Frequency - Max
316	N=3 - General - Peak Frequency - Std
317	N=3 - General - Spectral Entropy - Mean
318	N=3 - General - Spectral Entropy - Min
319	N=3 - General - Spectral Entropy - Max
320	N=3 - General - Spectral Entropy - Std
321	N=3 - General - Greedy Distance of PSDs - Mean
322	N=3 - General - Greedy Distance of PSDs - Min
323	N=3 - General - Greedy Distance of PSDs - Max
324	N=3 - General - Greedy Distance of PSDs - Std
325	N=3 - General - DCT Coefficient 1 - Mean
326	N=3 - General - DCT Coefficient 1 - Min
327	N=3 - General - DCT Coefficient 1 - Max
328	N=3 - General - DCT Coefficient 1 - Std
329	N=3 - General - DCT Coefficient 2 - Mean
330	N=3 - General -DCT Coefficient 2 - Min
331	N=3 - General - DCT Coefficient 2 - Max
332	N=3 - General - DCT Coefficient 2 - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
333	N=3 - General - DCT Coefficient 3 - Mean
334	N=3 - General -DCT Coefficient 3 - Min
335	N=3 - General - DCT Coefficient 3 - Max
336	N=3 - General - DCT Coefficient 3 - Std
337	N=3 - Forward - Ratio of Printable Characters - Mean
338	N=3 - Forward - Ratio of Printable Characters - Min
339	N=3 - Forward - Ratio of Printable Characters - Max
340	N=3 - Forward - Ratio of Printable Characters - Std
341	N=3 - Forward - Ratio of Unique Bytes - Mean
342	N=3 - Forward - Ratio of Unique Bytes - Min
343	N=3 - Forward - Ratio of Unique Bytes - Max
344	N=3 - Forward - Ratio of Unique Bytes - Std
345	N=3 - Forward - Entropy - Mean
346	N=3 - Forward - Entropy - Min
347	N=3 - Forward - Entropy - Max
348	N=3 - Forward - Entropy - Std
349	N=3 - Forward - Actual Entropy - Mean
350	N=3 - Forward - Actual Entropy - Min
351	N=3 - Forward - Actual Entropy - Max
352	N=3 - Forward - Actual Entropy - Std
353	N=3 - Forward - Greedy Distance of Packets - Mean
354	N=3 - Forward - Greedy Distance of Packets - Min
355	N=3 - Forward - Greedy Distance of Packets - Max
356	N=3 - Forward - Greedy Distance of Packets - Std
357	N=3 - Forward - Mean Frequency - Mean
358	N=3 - Forward - Mean Frequency - Min
359	N=3 - Forward - Mean Frequency - Max
360	N=3 - Forward - Mean Frequency - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
361	N=3 - Forward - Peak Frequency - Mean
362	N=3 - Forward - Peak Frequency - Min
363	N=3 - Forward - Peak Frequency - Max
364	N=3 - Forward - Peak Frequency - Std
365	N=3 - Forward - Spectral Entropy - Mean
366	N=3 - Forward - Spectral Entropy - Min
367	N=3 - Forward - Spectral Entropy - Max
368	N=3 - Forward - Spectral Entropy - Std
369	N=3 - Forward - Greedy Distance of PSDs - Mean
370	N=3 - Forward - Greedy Distance of PSDs - Min
371	N=3 - Forward - Greedy Distance of PSDs - Max
372	N=3 - Forward - Greedy Distance of PSDs - Std
373	N=3 - Forward - DCT Coefficient 1 - Mean
374	N=3 - Forward - DCT Coefficient 1 - Min
375	N=3 - Forward - DCT Coefficient 1 - Max
376	N=3 - Forward - DCT Coefficient 1 - Std
377	N=3 - Forward - DCT Coefficient 2 - Mean
378	N=3 - Forward -DCT Coefficient 2 - Min
379	N=3 - Forward - DCT Coefficient 2 - Max
380	N=3 - Forward - DCT Coefficient 2 - Std
381	N=3 - Forward - DCT Coefficient 3 - Mean
382	N=3 - Forward -DCT Coefficient 3 - Min
383	N=3 - Forward - DCT Coefficient 3 - Max
384	N=3 - Forward - DCT Coefficient 3 - Std
385	N=3 - Backward - Ratio of Printable Characters - Mean
386	N=3 - Backward - Ratio of Printable Characters - Min
387	N=3 - Backward - Ratio of Printable Characters - Max
388	N=3 - Backward - Ratio of Printable Characters - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
389	N=3 - Backward - Ratio of Unique Bytes - Mean
390	N=3 - Backward - Ratio of Unique Bytes - Min
391	N=3 - Backward - Ratio of Unique Bytes - Max
392	N=3 - Backward - Ratio of Unique Bytes - Std
393	N=3 - Backward - Entropy - Mean
394	N=3 - Backward - Entropy - Min
395	N=3 - Backward - Entropy - Max
396	N=3 - Backward - Entropy - Std
397	N=3 - Backward - Actual Entropy - Mean
398	N=3 - Backward - Actual Entropy - Min
399	N=3 - Backward - Actual Entropy - Max
400	N=3 - Backward - Actual Entropy - Std
401	N=3 - Backward - Greedy Distance of Packets - Mean
402	N=3 - Backward - Greedy Distance of Packets - Min
403	N=3 - Backward - Greedy Distance of Packets - Max
404	N=3 - Backward - Greedy Distance of Packets - Std
405	N=3 - Backward - Mean Frequency - Mean
406	N=3 - Backward - Mean Frequency - Min
407	N=3 - Backward - Mean Frequency - Max
408	N=3 - Backward - Mean Frequency - Std
409	N=3 - Backward - Peak Frequency - Mean
410	N=3 - Backward - Peak Frequency - Min
411	N=3 - Backward - Peak Frequency - Max
412	N=3 - Backward - Peak Frequency - Std
413	N=3 - Backward - Spectral Entropy - Mean
414	N=3 - Backward - Spectral Entropy - Min
415	N=3 - Backward - Spectral Entropy - Max
416	N=3 - Backward - Spectral Entropy - Std

Table A.1. Flow Based Payload Feature Name List. (cont.)

No	**Feature Names**
417	N=3 - Backward - Greedy Distance of PSDs - Mean
418	N=3 - Backward - Greedy Distance of PSDs - Min
419	N=3 - Backward - Greedy Distance of PSDs - Max
420	N=3 - Backward - Greedy Distance of PSDs - Std
421	N=3 - Backward - DCT Coefficient 1 - Mean
422	N=3 - Backward - DCT Coefficient 1 - Min
423	N=3 - Backward - DCT Coefficient 1 - Max
424	N=3 - Backward - DCT Coefficient 1 - Std
425	N=3 - Backward - DCT Coefficient 2 - Mean
426	N=3 - Backward -DCT Coefficient 2 - Min
427	N=3 - Backward - DCT Coefficient 2 - Max
428	N=3 - Backward - DCT Coefficient 2 - Std
429	N=3 - Backward - DCT Coefficient 3 - Mean
430	N=3 - Backward -DCT Coefficient 3 - Min
431	N=3 - Backward - DCT Coefficient 3 - Max
432	N=3 - Backward - DCT Coefficient 3 - Std