DETECTION AND MITIGATION OF NETWORK ATTACKS USING FEATURE GROUPING IN CROSS LAYER NETWORKS

Thesis Submitted for the Award of the Degree of

DOCTOR OF PHILOSOPHY

in

Computer Science and Engineering

By

Ravishanker

Registration Number: 41500075

Supervised By

Dr. Prateek Agrawal (13714)

(Professor and Associate Dean)

Computer Science and Engineering

Lovely Professional University

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2024

<u>DECLARATION</u>

I, hereby declared that the presented work in the thesis entitled "DETECTION AND MITIGATION OF NETWORK ATTACKS USING FEATURE GROUPING IN CROSS LAYER NETWORKS" in fulfilment of degree of **Doctor of Philosophy (Ph. D.)** is outcome of research work carried out by me under the supervision Dr. Prateek Agrawal, working as Professor and Associate Dean, in the School of Computer Science and Engineering of Lovely Professional University, Punjab, India. In keeping with general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of other investigator. This work has not been submitted in part or full to any other University or Institute for the award of any degree.

(Signature of Scholar)

Name of the scholar:	Ravishanker
Registration No.:	41500075
Department/school:	School of Computer Science and Engineering

Lovely Professional University,

Punjab, India

CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled “DETECTION AND MITIGATION OF NETWORK ATTACKS USING FEATURE GROUPING IN CROSS LAYER NETWORKS” submitted in fulfillment of the requirement for the reward of degree of **Doctor of Philosophy (Ph.D.)** in the School of Computer Science and Engineering, is a research work carried out by Ravishanker, 41500075, is bonafide record of his/her original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma or equivalent course.

(Signature of Supervisor)

Name of supervisor: Dr. Prateek Agrawal

Designation: Professor and Associate Dean

Department/school: School of Computer Science and Engineering

University: Lovely Professional University

Abstract: Information is generated every second because of the expansion of online platforms, and there is an immediate need to collect the massive volume of data and then safeguard it. Every day, the level of complexity rises, and with it, the need for extreme computational power, backups and other resources also rising. As a result of technological improvements and the emergence of globalization, the necessity to share information is becoming increasingly vital. Intrusion Detection Systems (IDS) have a lot of features, but it can be hard to figure out which ones are the best and requires regular updating of its signature. Unnecessary features in a dataset will make it harder to use and slow down the rate at which information is transferred.

To monitor the attack at different layer of TCP/IP model, a layered approach is followed starting from physical layer to application layer. IDS tools are available to capture packets in the form of datasets, and various well-known datasets are now accessible for benchmarking network attacks for IDS, which has attracted researchers to analyze current and potential attacks. To recognize cross-layer attacks, these datasets contain a variety of attributes. Cross-layer refers in context to the data link layer, network layer, transport layer, and application layer. These databases can be used to identify new attacks. This work is more focused on the feature grouping, recognition of attack types and classification of network attacks. Physical layer attack related to interception or jamming of signal in case of wireless network or capturing of packet in monitor mode can be possible. Same applies for wired network by tapping of wire. In the research more effort is made on analysis of frame and packets at data link layer and network layer respectively because a packet or frame is the only PDU which comes in or out from a network to end system or end system to the network and this makes an attacker easier to identify the target and attack by modifying, intercepting, replaying, or sending attack packet. Transport layer provide end to end communication and many application layer tools are available for detection of transport layer-based attack by monitoring TCP or UDP ports and related services.

In the beginning, investigation started with the data link layer (DLL) for checking the feasibility of employing Snort as tool for identifying DLL attacks. The conclusion was the IDS cannot identify all data link layer attacks since Snort typically operates at network layer and above. Snort can be used to analyze attacks at the data connection layer using a variety of solutions that have already been developed. This analysis helped in comprehending the possibilities of identifying all network attacks by combining multiple attack at the data link layer using Snort as a tool.

In next section, analysis on different machine learning model is used to select best approach for effective network attack detection by training and testing using CICIDS2017 and KDD'99 dataset. The system first gets the feature subset of each classifier, which depends on the normal weight and at later stage the strategy is applied for combining other subsets. After conducting research, it was reduced to only14 features for finding DDoS attacks in the CICIDS2017 data set. Overall performance using this method was effective in term of identification time based on selection of feature weightage. During the analysis phase it was found that some of the features in the dataset was duplicates or it had almost nothing to relate with the identification of an attack. This was an additional task as it was just adding more computational process and harder to compute. In our research, all such consideration was taken care of that was not used and added only added relevant features that were being counted when building the machine learning model. Considering this has shown a great computationally effective performance.

The accuracy has been computed as a three-step process: (1) collecting and pre-processing data, (2) training a ML model, and (3) Analysis of performance of the model. Nine ML approach is used in this work and accuracy is compared. This work is further extended by using IG-Feature Selection Subset, CR-Feature Selection Subset, and ReF-Feature Selection Subset, when using 48, 28, and 14 feature selection subsets, the respective results were 99.9981%, 99.9873%, and 99.9974%. The results show that the selection of key information for features is crucial for planning IDS that is simple, effective, and feasible for intrusion detection systems. This concludes that, choosing key attributes for features is important for planning IDS. Network attacks have a different set of parameters compared to datalink, transport or physical layer attacks, so an individual approach needs to be applied for the detection of particular attacks so cross layer detection using a single tool is not feasible but with the help of careful planning, accurate and quick analysis can be done.

ACKNOWLEDGEMENT

First of all, I would like to express my thanks to the Lord Almighty for his grace and blessings, without which this study would not have been possible.
I want to convey my gratitude and admiration to Prof. (Dr.) Prateek Agrawal, Associate Dean in the computer science and engineering department, whose advice, assistance, and encouragement have been really helpful for me during my research. For the technical assistance and supporting time to time, I like to thank Dr. Vishu and Dr. Usha Mittal.

Additionally, I would like to thank Lovely Professional University for continuing to support my Ph.D's thesis on "DETECTION AND MITIGATION OF NETWORK ATTACKS USING FEATURE GROUPING IN CROSS LAYER NETWORKS".

Last but not the least, I must express my gratitude to my family and friends for their unwavering support throughout this extremely demanding academic year.

List of Abbreviations

ANN	Artificial neural network
ART	Adaptive Resonance Theory
BN	Bayesian Network
BNN	BAT-Neural Network
CICIDS	Canadian Institute for Cybersecurity-Intrusion Detection Systems
CPS	Cyber-physical systems
CR	Correlation
CSRF	Cross-site request forgery
CVM	Cluster Variation Method
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed denial-of-service
DMIFS	Dynamic mutual information feature selection
DPI	Deep Packet Inspection
DT	Decision tree
EFS	Ensemble Feature Selection
FGLCC	Feature grouping combined with linear correlation coefficient
FGMMI	Feature grouping based on multivariate mutual information
FNN	Feedforward Neural Networks
FPR	False Positive Rate
FRNN	Feedback recurrent neural network
FSS-PART	Feature Selection Subset - Projective Adaptive Resonance Theory
HCA	Hierarchical cluster analysis
IANA	Internet Assigned Numbers Authority
ID3	Iterative Dichotomiser 3
IG	Information Gain
IPFIX	IP Flow Information Export
KISS	Chi-Square Signatures
LGP	Linear genetic programs
MARS	Multivariate adaptive regression splines
MaxEnt	Maximum Entropy

MLP	Multilayer Perceptron
NB	Naive Bayes
NSL-KDD	Network Security Laboratory-Knowledge Discovery and Data Mining
PCA	Principal component analysis
QDA	Quadratic Discriminant Analysis
ReF	ReliefF
REP	Reduced error pruning
SMLC	Semi-supervised Multi-Layer Clustering
SMOTE	Synthetic Minority Oversampling Technique
SVR	Support Vector Regression
TPR	True Positive Rate

Contents

List of Tables

List of Figures

Chapter-1

1. Introduction

1.1 Motivation

The internet has experienced exponential growth since its beginning, evolving from a limited network to a global infrastructure connecting billions of users worldwide. It unites people from across the globe, demonstrating the vast potential achievable through collective effort towards a common objective. It has revolutionized communication, commerce, and access to information, enabling instant connectivity and collaboration across geographical boundaries. Each day, billions of individuals utilize the internet to explore emerging trends and technologies, with millions of young people viewing it as their playground - a space for learning and generating novel ideas to foster personal growth and potentially benefit society. The proliferation of internet-enabled devices, including smartphones and IoT devices, has led to ubiquitous internet access and integration into various aspects of daily life. Internet technologies have driven digital transformation across industries, fostering innovation, economic growth, and new opportunities for businesses and individuals. In short, the Internet is the start of a new era as a transformative power just like how fire helped people learn in the

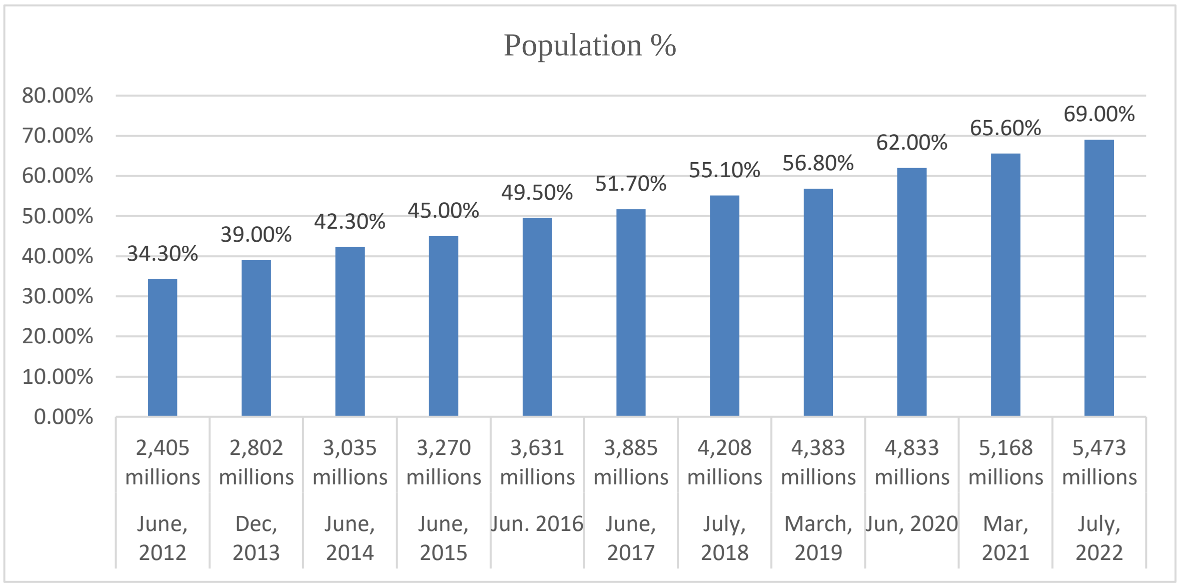


Figure 1. Internet Users (World) in 10 years from 2012-2022 [1]

ancient time. Internet is a wild, raging fire that lights the way for future innovations and will help a lot of people work together to make their dreams come true. The volume of internet users in world has surpassed 5.4 billion where as in India this count to 1.06 billon, and this number is quickly growing each day as millions of individuals use it [1].

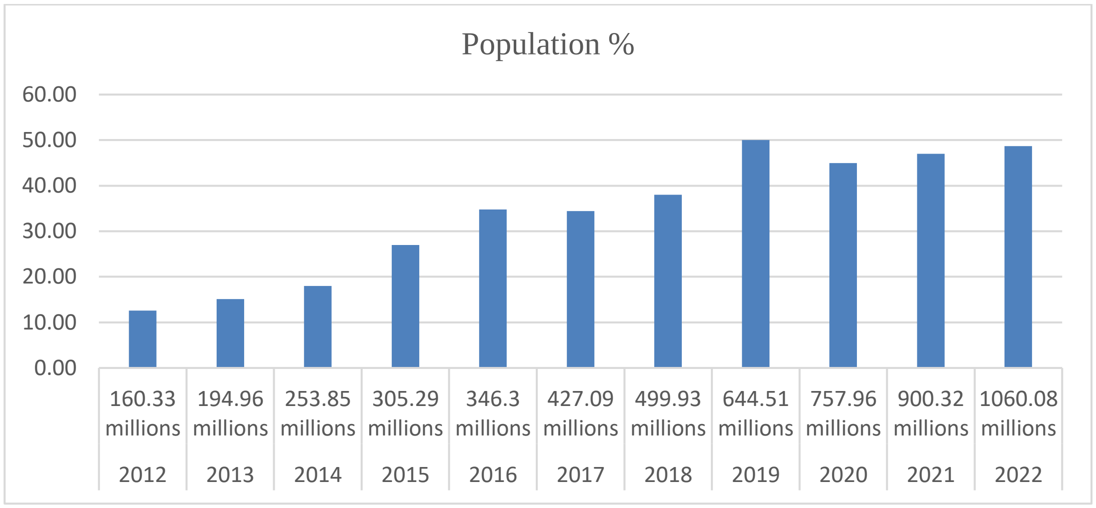


Figure 2. Internet Users (India) in 10 years from 2012-2022 [1]

Growth of Internet also brings certain disadvantages such as cybersecurity threats, privacy concerns, digital divide, misinformation, cyberbullying, online harassment, internet dependency, disruption of traditional industries etc. The internet's growth has led to an increase in cyber threats such as hacking, malware, phishing, and data breaches, posing risks to individuals, businesses, and governments. With the vast amount of personal data being collected and shared online, privacy breaches and unauthorized surveillance have become major concerns, compromising individuals' privacy rights. Despite increased internet access, disparities in connectivity, affordability, and digital literacy persist, exacerbating inequalities between urban and rural areas, as well as among different socio-economic groups. The internet facilitates the rapid spread of misinformation and fake news, leading to social polarization, erosion of trust, and manipulation of public opinion. Social media platforms and online forums have become breeding grounds for cyberbullying, harassment, and hate speech, affecting individuals' mental health and well-being. Excessive internet usage, particularly among youth, can lead to addiction, social isolation, and negative impacts on physical and mental health. The rise of the internet economy has disrupted traditional industries such as retail, publishing, and media, leading to job displacement and economic disturbance in some sectors. As a result of these developments, there are more attacks on the Internet. The internet offers immense benefits, addressing these disadvantages requires concerted efforts from governments, businesses, and civil society to mitigate risks and maximize its positive impact on society.

With respect to the above challenges, India is facing more and more cyberattacks, which are a major threat to national security, businesses, people, and critical infrastructure. India has seen a rise in cybercrime, from data breaches and ransomware attacks to espionage and state-sponsored cyberwarfare, as more and more areas become digital. Some of the most known cyber-attack India has faced in recent years is Mirai Botnet Attack, WannaCry Ransomware, SolarWinds Supply Chain Attack, COVID-19 Vaccine Data Breach etc.

In October 2016, a massive Distributed Denial of Service (DDoS) attack targeted Dyn, a major Domain Name System (DNS) provider, disrupting internet services for millions of users worldwide. This attack utilized the Mirai botnet, which infected IoT devices such as cameras and routers. It proved the global impact of cyber threats, raising concerns about the vulnerability of India's rapidly expanding IoT ecosystem.

The WannaCry ransomware attack in May 2017 affected over 150 countries, including India. It exploited a vulnerability in Microsoft Windows systems, encrypting data and demanding ransom payments in Bitcoin. Indian organizations across various sectors, including healthcare and banking, were disrupted, highlighting the susceptibility of outdated software and the need for robust cybersecurity measures.

In December 2020, the SolarWinds cyberattack came to light, affecting numerous organizations worldwide, including those in India. Hackers compromised SolarWinds' Orion software updates, allowing them to infiltrate networks and conduct espionage activities. While the full extent of the breach in India remains unclear, it underscored the risks associated with supply chain vulnerabilities and the need for enhanced cybersecurity frameworks.

To combat the COVID-19 pandemic, reports arisen of a cyber espionage operation targeting organizations involved in vaccine research and distribution. The incident raised concerns about the security of sensitive medical data and intellectual property. India has crucial role in vaccine production and distribution, such attacks could have severe implications for public health and national security.

These examples highlight the diverse range of cyber threats facing India as well as world, including malware, ransomware, espionage, and supply chain attacks. To mitigate these risks, India needs to prioritize cybersecurity by investing in technology, infrastructure, workforce training, and international collaboration.

1.2 Issues and Challenges

The current research work is focused more on cross layer attack for which there are two primary ways for detecting it: signature and anomaly-based detection. Signature based totally dependent on signature of known attack database and cannot detect novel attack unless the database is up to date. In certain case these updated databases are also inefficient like zero-day attacks and to prevent these attacks, the database needs information about them. There is also an ever-increasing percentage of internet traffic that is encrypted with SSL/TLS protocols. In such case of encrypted internet traffic, signature-based approaches are inoperable because it is impossible to see what is in the stream. In such cases data is analyzed using general attributes like packet/frame size, connection time, and packet count and measured under anomaly-based approach.

Recently, hackers and network programmers have used a range of methods to flood the packets and take over devices to attack an organization's services. Example of such attack at the transport layer is flooding attack, which can produce a lot of malicious packets. Attacks at the application layer have entirely different features from those at the datalink, network, and transport layers. Determining attack is crucial so IDS offers defense against different attacks. Each packet contains a several features, and combinations of these features allow each packet to uniquely identify the protocols and data packets it is carrying, some of which may be attack data packets and others just regular data packets. Internet traffic carrying packets comprises attributes such as source IP address, destination IP address, protocols, flag bits, etc. These attributes assist IDS in identifying attacks. However, the performance of IDS in terms of accuracy and timeliness may be compromised by improperly configured or unwanted attributes, leading to the consumption of additional computational resources. To address this issue, various characteristics or feature grouping approaches can be employed through machine learning approach to identify relevant or irrelevant attributes within the packet or stored dataset. This approach reduces the time required to detect attacks and improves the likelihood of detecting an attack accurately.

1.3 Thesis Organization

To find features relevant to attack detection, various cross layer attack analysis is done. There are several attacks, but this research is limited to identifying such attack for future detection. The creation of sustainable infrastructure is one of the ways to prevent from the attacks. An approach for DLL based attack uses different approach compared to network, transport and application layer attack. DLL related work is discussed literature survey part only as implementation of these attack is specific to hardware and its environment. Other layers of attack are commonly covered by considering a real-life datasets CICIDS 2017 and its detection using machine learning approach. Also, a synthetic dataset KDD'99 is considered for creating a framework for detection of attack specific to transport layer protocol. Most of the attacks happened from ingress or egress on the network layer so more focus is given on the other layer attack instead of DLL.

For network and above layer attack packet need to be captured and then requires a three-step method that entails (I) data collection and pre-processing, (II) machine learning model selection and training, and (III) performance model evaluation has been taken into consideration. For the simulation for the same a real-life dataset CICIDS 2017 is considered. The selection of CICIDS 2017 datasets serves as the initial step in the process. The ranker algorithm is used to consider appropriate feature selection processes. Here is a list of the tasks performed in the later section:

I. Observing attacks using feature grouping.

II. Employ function to speed up comparison and testing with CICIDS2017 and KDD'99 dataset.

III. Examine the performance of suggested model.

A new model is also proposed FSS-PART (Feature Selection Subset - Projective Adaptive Resonance Theory) to justify the performance and accuracy for justifying the benefits of feature grouping. In last a framework is designed that may help an administrator of the network to analyze the network traffic.

Chapter-2

2 Literature Review

2.1 Attack analysis at physical layer

Attack analysis at physical layer: These attacks are hardware-centric and, despite being easy to conduct, require some form of hardware to be effective. Attacks on other layers can be seen as simple alterations of packets and vulnerability of applications of already-in-use technology. These attacks can be easily conducted without comprehensive understanding of the technology [2], [3]. Such threats at this layer include eavesdropping, interference, and jamming [4], [5] which include intercepting communications without permission i.e., eavesdropping. In MANET, mobile hosts use a wireless medium to communicate and access the naturally broadcast RF band. With receivers tuned to the correct frequency, it is simple to intercept signals transmitted over the wireless medium. Thus, transmitted messages can be intercepted and forged messages can be introduced into the network. Jamming or interfering with radio transmissions can result in message corruption or loss. If the attacker has a strong transmitter, he or she can send out a signal strong enough to block communications and block up the intended signals. Jamming attacks can even be launched from places far from the target networks. Frequency hopping can be considered for the defensive approach against jamming attacks. The WEP protocol utilized by IEEE 802.11 is WEP uses a relatively weak encryption algorithm as well as WEP requires manual configuration of encryption keys on both the access point and client devices. This process is cumbersome and prone to errors. Additionally, the use of static keys makes it difficult to revoke or change encryption keys, especially in large-scale deployments. This lack of efficient key management increases the risk of unauthorized access. WEP does not offer forward secrecy, meaning that if an attacker captures encrypted network traffic and later obtains the encryption key, they can decrypt the entire captured data. This lack of forward secrecy compromises the confidentiality of past communications if the encryption key is compromised in the future. It has been widely replaced by more secure alternatives such as WPA (Wi-Fi Protected Access) and WPA2. [6]. In VANET, the attacker stays near the other moving vehicle and intercept or interrupt communication[7]. These attacks are mor feasible on static topologies such as sensor networks and rather than in dynamic topologies such as VANET to chase a moving vehicle.

2.2 Attack analysis at data link layer

Data link layer is an important component of network communication, as it oversees ensuring the reliable transport of data between linked devices. It is, nevertheless, vulnerable to a variety of security attacks that jeopardize integrity, confidentiality, and availability of network resources. Attack analysis at the data link layer entails identifying, comprehending, and mitigating these threats to provide a safe and strong network infrastructure. Attack analysis in this context refers to the examination and evaluation of various attack vectors and tactics that target the data connection layer. It entails investigating the flaws and vulnerabilities inherent in data link layer protocols, devices, and setups, as well as the potential implications of successful attacks.

Purpose of data link layer attack analysis is to improve network security by evaluating the data connection layer, security experts may discover and comprehend the numerous attack vectors that bad actors can use. This includes investigating protocols like Ethernet and Wi-Fi, as well as the accompanying methods for data encapsulation, error detection, and flow management.

The current study looks at the attack at the data link layer using different tools to look into the protocol and procedure. These approaches are not new, still they will assist with further study and identifying the issues using available tools. They can also be employed to automate tasks within the fields of machine learning and deep learning, respectively.

SNORT [8] is not capable of identifying every possible form of network attack at the cross layer due to configuration and signature updating related specific limitations.

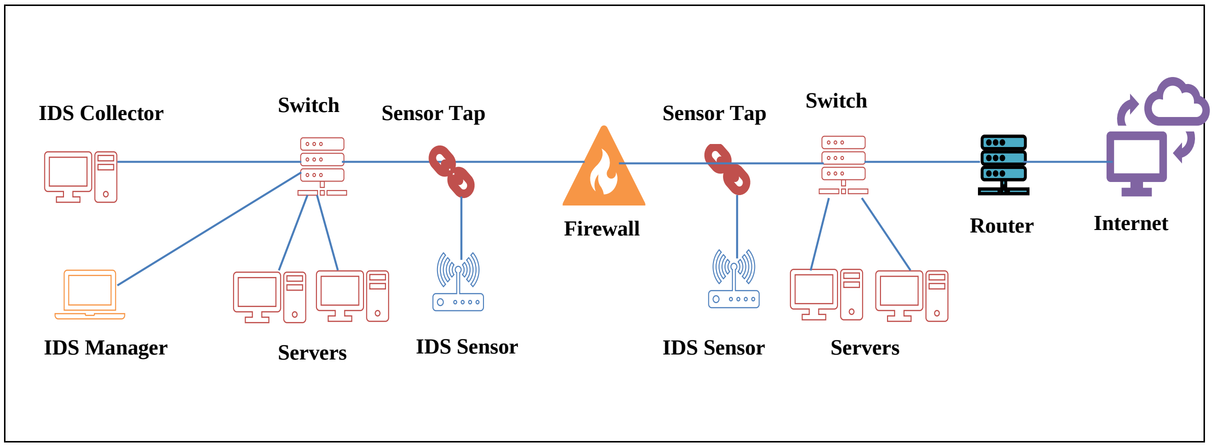


Figure 3. IDS placement and monitoring

In Figure 3 IDS placement is shown where the IDS collector is gathering all the logs and IDS manager is for configuring and monitoring purpose. IDS sensors are placed in between the firewall to detect malicious activity for incoming and outgoing traffic connected to the outside server and from internet. These all activities can be monitored at application layer and an administrator can configure security rules at this layer. Data link layer devices serve a crucial role in network communication by filtering unwanted frames based on source and destination addresses, controlling frame access, and authenticating devices. However, their scope is limited to self and neighbor connected devices, making it challenging to prevent attacks depends on IP addresses, protocols, or end-to-end connectivity. The primary security parameter for the DLL to work is the verifying neighboring MAC addresses, establishing trust between already linked devices [9].

Research attention towards data link layer security has been relatively limited in contrast to other cross-layer security aspects and due to this result, new attack types have not been extensively identified recently. Some common attacks in wired networks include CAM table overflow, ARP spoofing, DHCP starvation, and VLAN jumping. In wireless networks, attacks such as Wi-Fi de-authentication, hidden-terminal, rogue access point, malicious behavior of nodes, and selfish behavior of nodes are prevalent.

While attacks do occur at the network layer due to its accessibility and availability of open-source tools, most attacks target the network layer. These attacks can be categorized as either active or passive. Active attacks disrupt communication, modify packets, and provide the attacker full access to network resources. Passive attacks involve the observation of traffic patterns without making changes. Network layer attacks include IP spoofing, network hijacking, link withholding, smurf attacks, tear drop attacks, ICMP floods, ping floods, and replay attacks.

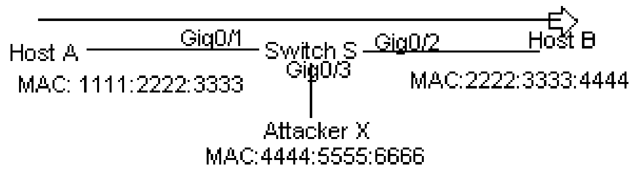
Transport layer attacks primarily occur during connection setup or ongoing communication. Examples include TCP sequence prediction/session hijacking, SYN floods, UDP flooding, and TCP flooding. Application layer threats encompass GET floods, Slowloris attacks, invite floods, SQL injections, SMTP attacks, malware attacks, Slow read attacks, and FTP bounce attacks. These attacks exploit vulnerabilities in system or application configurations, and they can often be mitigated through patches and recommended security measures.

Table 1. List of attacks on different layers

OSI Layer	**Attack Type**
Data Link Layer	CAM table overflow attack, ARP spoofing, DHCP starvation attack, Virtual LAN hopping attack, Wi-Fi de-authentication attack, hidden-terminal attack, rouge access point attack, malicious behavior of nodes and selfish behavior of nodes
Network Layer	IP spoofing, network hijacking, link withholding, smurf attack, tear drop, ICMP flood, ping flood, replay
Transport Layer	TCP sequence prediction/session hijacking, SYN floods, UDP flooding, TCP flooding
Application Layer	GET flood, Slowloris, invite flood, SQL injection, SMTP attack, Malware attacks, Slow read and FTP bounce

Several research has already been done on DLL attack and authors has suggested their counter measures also. In this section all such methods are discussed along with the possible solutions suggested by them. Trabelsi, Z. [10] has effectively illustrated the practical execution of a

CAM table overflow attack. Through his efforts, he has presented the process by which a switch constructs its MAC table, and how an attacker can exploit this by transmitting numerous messages, each carrying distinct MAC addresses. Consequently, this manipulation causes an influx of entries into the switch's CAM table. As the table reaches its capacity, an overflow situation occurs. One of the prominent tools recognized for executing such an attack is macof.



CAM Table on Switch S	MAC Addresses
Gig0/1	1111:2222:3333
Gig0/2	2222:3333:4444
Gig0/3	4444:5555:6666

Figure 4. An attacker overflowing CAM table.

In this case, the attacker will try to send a frame with more than one MAC address. The switch will pick up on this and start flooding. This will make it impossible for PC1 and PC2 to talk to each other. Since SNORT only records packets at the network layer, it can't find this kind of event at the switch and stop it from causing it to overflow. Switchport port-security can be configured to protect against this kind of attack. Ghazi A. et al. [11] and X. Hou [12] has shown the procedure of ARP spoofing in his research paper. ARP spoofing is a malicious technique where the attacker sends forged messages containing their forged MAC address along with genuine IP address of sender. By associating forged MAC address with legitimate IP, an attacker can capture all messages intended for actual host. Once the attacker gains access, they can eavesdrop, modify, or even block the entire communication. This type of attack is typically executed within a local LAN, but it can be further extended to enable denial-of-service (DoS) attacks, session hijacking, and Man-in-the-Middle (MITM) attacks by utilizing network tools such as Arpspoof, Cain & Abel, Arpoison, and Ettercap.

Mukhtar et. al. [13] in his work shown how DHCP starvation attack can be carried out and procedure for its mitigation. The attacker attempts to exhaust all IP addresses that a DHCP server reserves for allocation to legitimate hosts. The attackers attempt to submit requests using multiple fabricated identities.

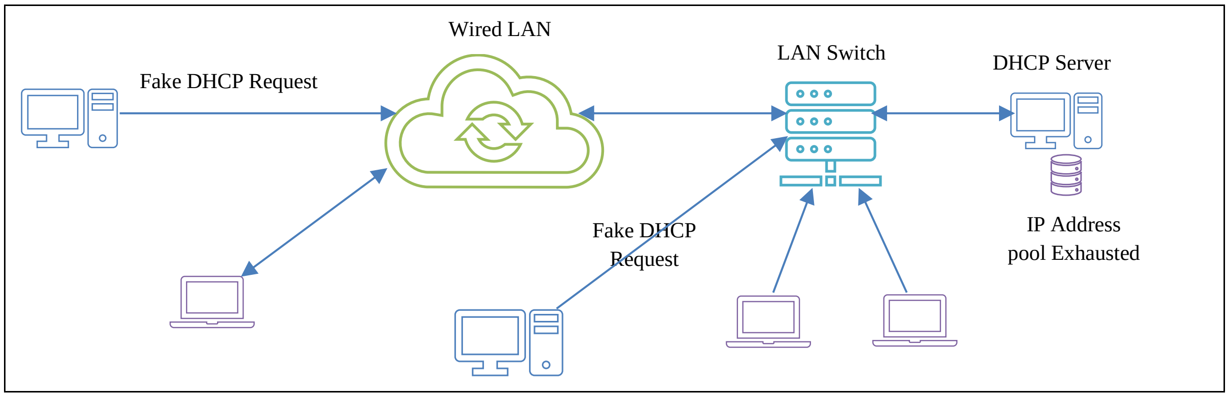


Figure 5. An attacker sends fake DHCP request to DHCP server.

Another form of attack related to DHCP involves the setup of a forged DHCP server. In this scenario, when a host requests an IP address from a DHCP server, the deceptive DHCP server responds more rapidly than the legitimate one. Consequently, the host becomes connected to the fraudulent DHCP server, which then becomes the exclusive point of communication. This imposter DHCP server can exploit this situation to intercept and retrieve all transmitted data. In a study by Rouiller [14] a technique was presented illustrating how an unauthorized individual can breach into a separate VLAN. This approach involves the intruder initially gaining access to the switch and subsequently leveraging a trunking protocol like IEEE802.1Q or DTP to facilitate communication between two distinct VLANs. Ordinarily, a host is restricted to conversing solely with other hosts within its designated VLAN and cannot establish connections with hosts in different VLANs. However, employing tactics such as switch spoofing or double tagging, the attacker endeavors to foster communication across other VLANs. This involves the manipulation of trunking protocols and frame tagging, which are conventionally employed to uphold VLAN maintenance.

Agarwal et al. [15] in their research showed the method that execute Wi-Fi de-authentication attack between the host access point. Wi-Fi de-authentication has similarity with denial-of-service attack. Attackers often intercept the data transmitted between a server and an access point, employing various techniques to compromise the security of the wireless network. One common approach involves the manipulation of the host's MAC address and the transmission of a "de-authentication frame," which exploits a feature in the IEEE 802.11 protocol. This type of attack can lead to the exploitation of two specific vulnerabilities: “evil twin access point” and “password” attack. In an evil twin, an attacker creates a fake access point that impersonate the legitimate one. By enticing the host to connect to the fake access point instead of the genuine one, the attacker gains unauthorized access to the host's communications and sensitive information. A password attack, on the other hand, targets the WPA or WPA2 handshake process used for authentication. The attacker leverages brute force or dictionary attacks to obtain the password. Numerous methods exist to execute such attacks, and popular tools like Aircrack-ng, Scapy, and Zulu are commonly utilized by attackers for these purposes.

It is important to note that these attacks can have serious security implications. Organizations and individuals should employ robust security measures, such as strong encryption, regular password updates, and network monitoring, to mitigate the risks associated with these attacks. Milliken et al. [16] in their work, used machine learning to stop "de-authentication attacks," which worked 96% of the time and suggested to use SHA-512 and UUID to mitigate such attack. Chang S and Hu C [17] identified hidden terminal attack mitigation technique by RTS timer and CTS timer validation. A duration field is included in the MAC frame of the early network allocation vector in IEEE 802.11 standard. It contains NAV with counter whose value drops when the channel is idle until it reaches zero, which only serves to validate the communication by holding back all frame transmission time. Studies conducted by other researchers demonstrate work of a similar nature based on RTS, CTS timers, and NAV duration counters. An illustration of such study may be seen in Jamal et al. [18]. Shetty et al. [19] In their work, had shown how to find and stop attacks from rogue access point with MAC filtering. There are two parts to the work. The first one is based on the difference between how traffic moves on a wired LAN and a wireless LAN. The second one is based on straight access attempt and cross access attempt threshold factors. Other researchers, who worked on MAC filtering with different approach can be found in [20].

Attack mitigation analysis: Several data link layer attacks have already been addressed above, and numerous researchers have come up with several ways to prevent them. This part will discuss about how to stop a data link layer attack, with the CAM table [21] overflow attack, ARP spoofing, DHCP starvation attack and Virtual LAN hopping attack. CAM table overflow attack: To defend such attack, switchport security can be configured on the switch. Multiple manufacturers of switches have implemented security by binding ports with restricted MAC address learning. This protects the switch from CAM table overload by allowing only a desired number of ports to learn. Example to configure on Cisco switch for such security is by putting static access port:

```
Switch(config-if)# switchport mode access

Switch(config-if)# switchport port-security

Switch(config-if)# switchport port-security maximum 2          //It allow only 2 MAC address

Switch(config-if)# switchport port-security mac-address sticky

                                  Or

Switch(config-if)# switchport port-security mac-address 1111.2222.3333      // Using static host MAC entry
```

In the configuration displayed above, port-security is set to a maximum of 2, allowing only the first two learned MAC addresses to associate with the switch. Manually binding a static host MAC entry is another way to do the configuration.

ARP spoofing: With packet filtering, a trust relationship, arp spoofing detection tools, and cryptographic protocols such as SSL/TLS, SSH, and HTTPS, this type of attack can be countered. Since ARP spoofing [22] impact the higher layer protocols so it can be easily configured for detection in SNORT as described by X Hou [12]. Since the attacker tries to set itself up as the default gateway, frame from the attacker only goes to the real gateway switch, so a host does not notice that anything has changed. For example, to stop this kind of attack on a Cisco-based switch, we need to set the port-security limit to 1, so that it will only let in the first MAC address it learns.

```
Switch(config-if)# switchport port-security maximum 1   //Set as 1 to learn only initial learnt MAC address
```

DHCP spoofing attack / DHCP starvation attack: Researchers have come up with many ways to solve this problem, such as making a static allocation where an administrator can bind an IP address to a MAC address and store the record on a DHCP server. This method won't work for a large group of network users. The other way is to look for a DHCP rate request [23], [24].

Their work was mostly centered on dynamic IP allocation under the need of fair allocation. Cisco has also demonstrated their strategy for defending from starvation attacks as elaborated in given example:

switch(config)# ip dhcp snooping

switch(config)# ip dhcp snooping vlan number 10 **// Give the VLAN number**

switch(config)# ip verify source vlan dhcp-snooping port-security

switch(config)# switchport port-security limit rate invalid-source-mac 15

switch(config)# ip source binding 10.1.0.2 aaaa.bbbb.cccc vlan 5 interface eth0/0

switch (config-if)# ip dhcp snooping trust

VLAN hopping attack: Switch Spoofing/Double Tagging: To mitigate the VLAN hopping [25], [26] authors have suggested proper switch configuration. Snort can only keep track on VLAN devices, so VLAN hopping, which uses frame encapsulation at the data link layer, can't be tracked. Switch configuration commands have been developed by Cisco as a defense against switch spoofing attacks is shown below:

A. Disabling DTP by changing trunk port to nonegotiate:

Switch (config-if)# switchport nonegotiate

B. Unused port must be set as access ports especially unallocated trunk port by:

Switch (config-if)# switchport mode access

Solution to prevent double tagging attack by switch configuration command as shown:

A. Changing default VLAN i.e. VLAN 1 to access VLAN.

Switch (config-if)# switchport access vlan 2

B. Changing native VLAN on all trunk ports to any unimplemented VLAN ID.

Switch (config-if)# switchport trunk native vlan 800

C. All trunk ports must be tagged explicitly with native VLAN.

```
Switch(config)# vlan dot1q tag native
```

Wi-Fi de-authentication attack: Wi-Fi de-authentication can be done in many ways, but this paper focuses on the data link layer attack and use of Snort as an IDS. Wi-Fi de-authentication attacks are all about the data link layer, so snort needs to use the latest patch snort-wireless 2.3.3-sgracia [27]. It can set up alerts on Snort to notice this kind of attack as shown below:

```
alert wifi any -> any (msg:"de-authentication"; stype:STYPE_DEAUTH;)
```

Other rules in Snort-wireless facilitate the detection of rogue access points on the wireless medium, programs that search for access points, and de-authentication or authentication floods that lead to the access point and wireless hosts.

Only a passive technique to hijack a session while using the RTS-CTS handshake has been studied for the hidden terminal problem, it becomes apparent after glancing at a few sources. There is no method for Snort IDS to detect a hidden terminal attack. Future research will be done on Snort-wireless to determine its applicability. How to locate a rogue access point has not been covered in much detail up to this point. However, MAC screening, which will be the focus of my upcoming work in this sector, can find it to some extent. An effort can be made to learn more about this subject using Snort-wireless.

Detection capabilities using snort: The prospect of employing Snort for attack detection at the data link layer was investigated by studying various literature surveys. It was discovered that Snort typically performs well at the network, transport, and application layers, concluded that, all types of attack detection at the data link layer cannot be detected by IDS. After consulting a number of articles, it is discovered that numerous solutions have already been investigated. This section will focus on a summary of all known attack types and the potential application of Snort at the data link layer.

Table 2. Strategies to address and minimize attack at DLL using Snort

Attack at Data link layer	Strategies to address and minimize using Snort
CAM table overflow attack	Due to Snort's limitation in capturing packets solely at the network layers, it remains unable to identify and prevent overflow incidents occurring at the switch level. Although the examination of Snort's configuration for countering

	such attacks has not been validated during the analysis, it is conceivable to enhance protection by incorporating a static MAC address list within its configuration file. This measure could potentially generate alerts and enhance the system's ability to address such scenarios.
ARP spoofing	ARP spoofing directly affects higher-layer protocols, making it feasible to configure SNORT for its detection as described by X. Hou [11].
DHCP starvation attack	The attack's detection relies on the proper placement of an IDS. However, no conclusive research has yet been identified in this regard.
Virtual LAN hopping attack	Snort's monitoring capabilities are limited to VLAN devices, thus restricting its ability to effectively monitor VLAN hopping due to its reliance on frame encapsulation at the data link layer.
Wi-Fi de-authentication attack	By utilizing the Snort-wireless patch extension, it becomes feasible to identify this type of attack. Snort-wireless not only offers the potential to detect such attacks but also includes additional rules that strengthen its capabilities as an IDS. These extended functionalities encompass the detection of rogue access points within the wireless medium, the identification of programs scanning for access points, and the recognition of de-authentication or authentication floods targeting both the access point and its wireless hosts.
Hidden-terminal attack	The current Snort IDS lacks the capability to detect hidden terminal attacks, prompting future exploration in the domain of Snort-wireless for addressing this concern.
Rouge access point attack	Properly placed IDS configurations can potentially detect the attack; however, there is a dearth of relevant literature on this particular attack, indicating a potential avenue for future investigation using Snort-wireless.

Several authors suggested the defenses against DLL attacks and their mitigating strategy was comparatively unique. When compared to a wireless network attack, wired approach needs a different strategy. The characterization of attacks was influenced by the specific devices in use, including the host computer, switch, and DHCP server. These reasons led to the conclusion that it is challenging to create a single point of attack detection or gathering all attack detection in one location. The focus of this investigation was to pinpoint the attack at DLL and determine whether it was possible to use Snort to consolidate all attacks in one place. Typically, Snort operates at or above the network layer. IDS cannot always detect data connection layer attacks. In the current effort, Snort will be used to investigate attacks at the data connection layer using a variety of previously developed solutions. Cyber-attacks are becoming more advanced every day, and the major difficulty is to implement algorithms that can distinguish unknown attacks [28], [29], [30], i.e., attacks for which a preset set of signature patterns is not specified for new attacks. Apart from this, data is generated from number of network sources, and it is not necessary that the captured would be useful in identifying an attack. Algorithms for intrusion detection, such as signature and anomaly detection, have been developed by several researchers [31], [32], [33].

2.3 Attack analysis at Network, Transport and Application layer

Attacks are detected by comparing the present activities to the desired activities of an attacker. An anomaly detection mechanism creates an activity profile. Misuse detection necessitates the labeling of training observations as either normal or malicious. During the training phase of anomaly detection, only valid behavior is considered. In the test data, anomalous behavior is detected if it deviates significantly from the modeled behavior. For the following reasons [34], labeled data are not easy to train. It is possible that different data sources generated the records needed to identify an attack. There are no publicly available datasets that include actual user traffic. Researchers often lack access to comprehensive data regarding the frequencies and impact of cyberattacks. Nonetheless, it remains crucial to evaluate the scalability of machine learning-based solutions in handling large volumes of data, numerous equipment, and diverse applications for addressing various network-related issues. Currently, only a limited number of machine learning techniques for fault and security management can be effectively applied to multitenant and multi-layer networks. To effectively manage faults and security in future networks, it will be necessary to integrate the concept of multi-tenancy in multi-layer networks into existing machine learning approaches. This adaptation will enable more robust fault and security management capabilities. [35], [36]. In general, machine learning combined with large datasets provides effective solutions for predicting and addressing various problems. ML techniques can uncover hidden patterns within data, allowing for clustering, classification, regression, and rule extraction. This can be particularly useful in the field of networking, where classification challenges such as denial-of-service, user-to-root, root-to-local, and probing attacks can be grouped together to predict the type of attack. To anticipate future failures, regression problems can be developed using machine learning algorithms. Although machine learning has a broad scope in addressing various problems, a consistent approach is often followed in constructing ML-based solutions. This involves data collection, which encompasses the gathering, generation, or definition of a relevant dataset and associated classes. Feature grouping is a technique that can reduce the dimensionality of datasets while identifying distinguishing characteristics. This not only saves computation time but also enhances the accuracy of predictions. By utilizing machine learning and appropriate data preprocessing techniques, organizations can leverage the power of big datasets to extract valuable insights, make informed decisions, and improve various aspects of their operations. [37], [38], [39]. As a result, once the ML model has been developed, it may be used to predict results from new data. To avoid "concept drifting," which is the gradual deterioration of a concept over time, results are checked on a regular basis. When this happens, the machine learning model can be re-trained, and its incremental learning can be monitored.

Khalil El-Khatib [40] has this a novel approach to picking the optimal attributes for 802.11-based intrusion detection. To pick relevant features, the strategy employs a hybrid approach that incorporates both filter and wrapper models. The total number of characteristics was decreased from 38 to 8 because of this strategy. They have also investigated how different classifiers powered by neural networks perform when features are selected. The classifiers' learning time is slashed by 33% while detection accuracy is increased by 15% because of the smaller range of possibilities. Their upcoming investigation is on comparing performance of classifier based ANNs with models supported by SVMs, MARSs, and LGPs to see how the decreased feature set affects those models.

Song J et. al. [41] has this a new strategy for grouping features based on mutual knowledge. Fuzzy C-means algorithmic rules are used to form groups or teams in the proposed algorithmic rules. When selecting an attribute from a group, the shared knowledge between a feature and its class labels is utilized. Experiment findings on the KDD’99 dataset show that the strategy this by them beats the DMIFS algorithm in most cases. An additional benefit of using separate classification methods to compare 10 and 41 features is that the performance indicators are now more accurate. The work provides a starting point for additional inquiry. It is possible to create a novel algorithm based on features and class labels and a variety of categorization methods are used to form group.

An IDPS based on Machine Learning (ML) requires an enormous volume of labeled and training data to identify intrusions and generalize to fresh attacks, as indicated by Omar Y. Al-Jarrah et al. [42]. Labeling information, on the other hand, is expensive and cannot be done with big amounts of data, like those created by IOT applications. For the present effect, it is important to build an ML model that can learn from data that is not labeled or is only partially labeled. In this study, Semi-supervised Multi-Layer Clustering (SMLC) is used to find and stop network intrusions. When learning from partially labeled data, SMLC can be compared to supervised ML-based IDPS in terms of how well it can find things. The NSL and Kyoto 2006+ datasets are widely recognized benchmarks for evaluating the performance of SMLC (Semi-Supervised Learning Classifier) models. These datasets serve as standards to compare the effectiveness of SMLC against other established semi-supervised models, such as tri-training, as well as ensemble machine learning models like Random Forest and Bagging. By utilizing these datasets, researchers can objectively assess and compare the performance of different models to determine their effectiveness in handling semi-supervised learning tasks. The result concluded that, SMLC performed better in terms of accuracy of detection while using 20% less labeled training sample. Their results show that this method is accurate at identifying problems as guided ensemble models.

It has been attempted by Chandrashekar and Sahin [43] that an introduction to feature selection methods be provided. Machine learning and pattern recognition are only two examples of feature selection approaches that have been studied extensively. One dataset is the only way to compare feature selection techniques since each algorithm can react differently when applied to multiple datasets. In machine learning, feature selection algorithms indicate that a lot of information is not necessarily helpful. They tested several methods on the data at hand before settling on a final feature selection algorithm based on classification performance metrics. They used feature selection to enhance accuracy of predictors and to analyze fault prediction based on information about fault modes, which has worked well.

To reduce the modeling complexity, [44] stated that, in data mining or pattern recognition, one of the most important preprocessing steps is choosing groups of attributes based on rough sets. To deal with new technology line "big data," new ways need to be identify. In this paper, the authors look at new work on using decision theory to choose subsets of attributes in rough set models. In this study, research on selecting feature sets and supporting rough sets was reviewed. Author proposed MapReduce version of parallel genetic algorithm to find least feature reduction. Parallel GA model focuses on three key areas: the construction of distinction tables, the evaluation of the GA population, and the administration of the GA. For constructing the distinction table, they frequently employ several mappers. It was during this phase that all the genetic operations occurred. Using a set of criteria, the driver code selects the simplest and best candidate. Using intrusion detection datasets as a test bed, it was found that the MapReduce technique will speed up execution without sacrificing solution quality. The number of attributes and instances has also been examined, as have the successive and MapReduce implementations. Due to our propensity to work in increasingly complex datasets, the studies revealed that the MapReduce implementation consistently produced better results than the other approaches. The planned approach will be tested in additional areas in the future.

The problem of high dimensionality in bio-medical data classification was tackled in a novel way by Yonghong Peng et al. [45]. Research in the disciplines of pattern recognition and machine learning has been extensive. Filter, wrapper, and hybrid feature selection strategies have been discussed in their work. When evaluating feature subsets, filter strategies use an independent test, whereas wrapper strategies require a specified learning procedure. They all have their own set of drawbacks, yet they all complement each other in some way. Their work includes filter and wrapper algorithms as part of a sequential search process to improve how well the hand-picked options can be grouped. The main parts of the planned method are pre-selecting feature subsets with better classification performance and misusing ROC curves to describe each option and subgroup within the classification.

To keep the classifiers' performance results intact while reducing the number of alternatives, Bolón-Canedo et al. [46] present a solution to enable a mix of wrapper, filter, and classification algorithms. The winner of the KDD'99 Competition and other writers have used it to reduce the number of options in two different techniques by a contrasting difference. The best ways to classify things are into two categories or into more than two classes. Using 17% of all features, the binary approach did better than the KDD winning result. More than that, the approach was used on several big binary datasets with the same results, which means that machine learning algorithms have fewer options. This method seems to be good for big datasets like the KDD'99 dataset because it saves time and memory. KDD'99 has also been looked at as a setback that fits into more than one group, like traditional connections and four other kinds of attacks. People often suggest using a multiple class method or multiple binary classifiers to deal with many category problems. This work changes the KDD winner result, even though it only uses 17% of the traits. This method works with large databases because it uses naive Bayes or C4.5 classifiers instead of SVMs, multilayer perceptrons, or functional networks. These machine learning algorithms are faster and use less computing power than other classifiers used in the literature. When compared to other methods, it looks like ours did better than those of the other authors, especially in the areas that were hard to tell apart. Because they did not have enough data, applied math could not use the work of other writers. During a comparison, it was found that, based on the category, some combinations were much more effective than others. In the future, they suggested to put together completely different combinations to make sure that the smaller categories are taken care of and enhancing the performance at larger scale.

In one of the research, IDS feature selection problem has been highlighted by Sara Mohammadi et al. [47] due to additional and unused features containing in a dataset, as a result these systems are prone to inaccurate categorization and a poor rate of detection. They discussed FGMMI, FGLCC, and try-wise MI as possible solutions to the problem. There was a lower rate of false-positive detection and accuracy with the projected feature choice strategies compared to try-wise linear parametric statistics and thus the try-wise MI used in many previous algorithms, as demonstrated by experimenting NSL and KDD'99 datasets. The FGLCC and FGPMI algorithms have introduced a novel approach, enhancing the FGMI algorithm through strategic structural adjustments. In KDD'99 and NSL, the FGMMI algorithm produced the best results with AR and F values of 95.65 and 96.12 percent, respectively and have high detection and accuracy rates.

According to Mowei Wang et al. [48], A reliable ML model for networking problems necessitates representative and unbiased data. The process of data collection is crucial, as datasets vary across different problems and time periods. Data can be gathered through two methods: in-person and online. Offline data collection enables the accumulation of extensive legacy data, which could be utilized for enhancing model training. During the live phase, real-time network data can be used to retrain the model and incorporated into a feedback loop.

Monitoring network activity can be done actively or passively. Active monitoring involves sending probe packets into the network to gather useful information, while passive monitoring observes network behavior without injecting additional traffic. Active monitoring incurs additional costs due to bandwidth usage and the need for extra equipment, whereas passive monitoring avoids these issues. After gathering the data, it is partitioned into sets designated for training, validation, and testing purposes. The training dataset plays a pivotal role in refining model parameters, such as the neural network's connection weights, to achieve optimization. The validation dataset aids in the choice of an optimal model configuration, like determining the ideal number of hidden layers in a neural network from a range of potential models. The test dataset serves the purpose of impartially assessing the model's performance.

The split percentages for training, validation and testing can vary with ratio of 60/20/20 or 70/30, depending on the need for validation. Rule-of-thumb ratios work well for smaller collections, while extreme ratios like 98/1 or 99/0.4/0.1 are useful for large datasets. It is crucial to ensure that training datasets are not skewed towards specific groups of interest, to prevent overfitting or poor generalization of the model. Validation and test datasets should have the same distribution as the training set and remain separate from it. To assess the model's ability to handle changes in time and location, training and validation datasets from different periods or networks can be used. Models that accurately predict results from datasets collected over a year or from different networks demonstrate stability across both time and space.

Using machine learning, Moore and Zuev [49] proposed solution to network traffic classification challenge. Before utilizing collected data for learning purposes, it is essential to address any inaccuracies or missing information through a data cleaning process. Feature extraction is a crucial step that precedes model learning or training, as it helps simplify the learning process and enables drawing meaningful conclusions. When dealing with networking tasks, there are various options available, which can be categorized into coarse, medium, and fine levels of detail.

At the packet-level, statistics such as mean, root mean square (RMS), and variance can be computed from the collected packets. Time series data, such as hurst, can also be derived from the packets. One benefit of using packet-level analysis is that they are not affected by packet sampling, which are sometimes employed to collect data but may alter the behavior of features. Flow-level features, such as average flow length, average packet flow count, and average bytes flow count, can be calculated using simple statistical methods. Additionally, characteristics at the transport layer, like throughput and window size mentioned in TCP connection headers, can provide further insights into connections. However, these features may require more setup time and are inclined to sampling and route asymmetries.

The selection and extraction of features play a vital role in machine learning. The primary objectives of this process are to reduce the dimensionality of large datasets and recognize distinctive features that minimize computational overhead while enhancing the correctness of ML models. "Feature selection" involves removing unnecessary or redundant features, as utilizing irrelevant characteristics that have little or no impact on accuracy can lead to overfitting. Feature extraction is a complex computational process that leverages methods for instance entropy, Fourier transform, and principal component analysis (PCA) to obtain additional or transformed features from existing ones. Tools like NetMate and WEKA can be employed to assist in the selection and extraction of features. By carefully selecting and extracting relevant features, the dimensionality of datasets can be effectively reduced, computational efficiency can be improved, and ML models can achieve higher accuracy in networking tasks.

According to Jun Zhang et al. [50], data can be labeled using the features in many ways. Manual labeling by field experts with DPI assistance is primarily required. To help with classification when there is a lack of training data, this research offers a new traffic classification scheme. Uses discretized statistical features to describe traffic flows, and bag-of-flow correlation information to model flow correlation in this approach (BoF). They use a classifier combination framework to solve the BoF-based traffic classification and then conduct an analysis of the function. The NB predictions combined using a novel BoF-based traffic classification algorithm, they examine the aggregation strategies' sensitivity to prediction error. This strategy was put to the test two real traffic datasets. According to the experimental results, this technique outperformed current traffic classification algorithms in terms of accuracy. This new traffic classification technique increased classification performance in absence of insufficient training data. Using the new BoF-NB approach, NB predictions were efficiently aggregated. The success of the suggested strategy was proved through testing on two real network traffic datasets. In this study, traffic classification was achieved with less time for labeling of training samples.

The performance of the model was shown to be impacted by the amount of the training data for different classes, as demonstrated by Soysal Schmidt [51]. Traffic classification accuracy is critical for network administration and monitoring as well as network provisioning. Consideration must also be given to scalability, processing costs for classification and user privacy when selecting a classification method. Based on accuracy and execution cost like classification and system buildup time, the study suggests that BN, DT, and MLP is best for flow-based traffic classification. Several categories of data were taken into account, including P2P, HTTP, Akamai content delivery, FTP, bulk uploads, DNS, and SMTP email. A piece of software called WEKA is used to run the tests. There hasn't been much research on ML algorithms for traffic classification because earlier studies used flow traces that were labeled with likely mistakes because of payload inspection or port-based methods. Also, the data sets don't include internet flows that can't be found with these methods, which limits how useful the results are. They used more than a million recent flow traces that were correctly labeled to train and test the ML systems. Peer-to-peer (P2P) traffic is an example of this kind of traffic. P2P traffic can move ports or pretend to be something else. Flow-based classification is being used for the first time to classify Akamai content delivery data.

For BNs, DTs, and MLPs, their work was to look at how training data was put together in an organized way. Experiments show a link between volume of training data related to each traffic type and how it affects accuracy of classification. Volume of training data for a specified traffic categories can influence how well it can classify traffic. Because of how we do things, there are some training sets that give the most accurate results. Traffic which associates to well-known default ports for a port-based classifiers work better. ML approach, on the other hand, are useful for classifying traffic, especially for types of traffic like P2P that do not use system ports.

The National Academic Network of Turkey was used to get a big set of correctly labeled data that was used to test BNs, DTs, and MLPs. According to the test results, DTs are more accurate and have a better categorization rate than BNs. DTs, on the other hand, are longer to create and are more vulnerable to inaccurate or insufficient training data. Researchers have conducted a thorough investigation on traffic classification using MLPs trained using back propagation to detect any shortcomings. They discovered that the sorting of web traffic, bulk traffic, and email traffic is highly dependent on one another and significant to amount of training data. When using the MLP method, we cannot get good recall numbers for these types of traffic at the same time. For instance, they demonstrate how P2P programs that masquerade (that is, utilizing ports from other applications, like port 80) and FTP programs that run in passive mode (that is, use random port assignments in a certain range) may impair the performance of other programs. While FTP applications that are running in the background may adversely affect the accuracy of the BN algorithm, same is not applicable with the DT approach.

Support Vector Regression (SVR) can be used to estimate link load forecasts in traffic prediction, according to Bermolen and Rossi [52]. With a hands-on approach, the researchers tinker with the SVR performance and compare it to that of the MA and AR models. For short-term link load predictions, the improvement that SVR has over basic prediction methods like MA or AR is not enough to support its use, even though the actual data matches the predictions made by SVR well. It's important to note that SVR models have some advantages over other forecasting methods. For example, they aren't too sensitive to change in the parameters, their computational complexity isn't too high, and the forecasting horizon can be further extended with only a modest loss in accuracy using a cascaded SVR model. This paper serves as a springboard for subsequent research, the directions of which are briefly discussed below. Different network traces from various scenarios should be used to validate the outcomes of the research in order to get more robust data. Preliminary results suggest that changing the time series (such as by changing its differentiation or statistical features) could improve forecast accuracy by a large amount. This would require a comparison to more advanced methods for forecasting time series. Also, it remains to be seen if the accuracy of SVR can be improved by using different kernels (such as multi-linear or others can account for characteristics of the time series) to reduce need for expensive time-series manipulation. To wrap up, this study could look at other possible paths, such as evaluating different forecast targets (like peak load) and analyzing longer timescales. It could be worthwhile to investigate whether the SVR can better anticipate load changes by including information about the duration in a day or a week.

It has been claimed that traffic prediction is a non-TSF problem and that an alternative model for inter-DC network traffic prediction. Inter-DC traffic, in contrast to typical network traffic, is driven by a few numbers of large applications that produce huge traffic flow. The key to traffic forecasting is to break down the traffic pattern into its component parts. The primary objective is to predict the volume of traffic traversing an inter-data center link, with a particular emphasis on handling substantial traffic flows. To achieve this, models employ Feedforward Neural Networks (FNNs) trained using Backpropagation (BP) with basic gradient descent. Additionally, wavelet transforms are employed to capture temporal and frequency characteristics inherent in time series form for traffic data. By incorporating these techniques, the models can effectively analyze and interpret the complex patterns present in the traffic data, enabling accurate traffic volume prediction for inter-data center links. The estimate considers more factors, such as traffic flows. But it costs more to count the number of byte-volumes than to gather all traffic flows at high rates. So, to reduce the cost of gathering traffic flow data, it is done less often and then the missing numbers are filled in by interpolation. For six weeks, SNMP counters were used to check Baidu's DC routers as well as inter-DC connection every 30 seconds to record a consolidated amount of egress and ingress traffic. Most of data on egress and ingress traffic are gathered every 5 minutes, and lost data count are estimated every 30 seconds using interpolation. A level 10 wavelet transform are used to consolidate time series, resulting in 120 features per timestamp [53].

Bakhshi and Ghita [54] addressed traffic classification in their article for task performed by the network administrator on various operation activities. Numerous important aspects need to be considered in networking, which include infrastructure allocation, quality of service (QoS) and service categorization, ensuring security, detecting malicious intrusions, tracking logs, and evaluating performance. These factors collectively contribute to the efficient functioning and management of networks, ensuring optimal resource allocation, reliable service delivery, robust security measures, and effective monitoring of network performance. Pre-defined classes of interest must be appropriately linked to network traffic to achieve traffic classification. HTTP, FTP, WWW, DNS and P2P are all examples of types of applications. Other examples include services such as Skype, YouTube, and Netflix. When it comes to QoS (Quality of Service), for example, a class of service covers everything that falls under the same QoS specifications. This means that applications with seemingly disparate functionality may have a common set of service characteristics despite their outward differences. It is possible to categorize all types of network traffic by port number, payload, host behavior, or flow characteristics in four major categories. Use of flow measurement to classify traffic allows network operators to perform crucial network management. Classification approaches like NetFlow, which require extra packet-level information and host behavior analysis as well as specific hardware requirements are regarded as insufficient for flow accounting. Dealing with such issues, two-stage machine learning classification technique is proposed which has input data from NetFlow. C5.0 decision tree classifiers are trained using k-means clustering on flow classes derived per application. Unsupervised flow data from 15 major Internet sources were used in the initial round of validation to define distinct flow classes created by each application individually using k-means clustering. After that, a supervised C5.0 decision tree was trained and tested using the generated flow classes. Adaptive boosting increased accuracy from 92.37 percent to 96.67 percent on about 3.4 million test cases with the resulting classifier. Between 98.37 percent and 99.57 percent of the time, the specification agent of classifier, that differentiate content-specific from additional flows, was found as a result, they suggested that this methodology is applicable to a vast array of instances, including traffic classification, because of its computational efficiency and accuracy.

Dainotti et al. [55], stated that the assignment of port numbers to programs is a responsibility handled by IANA. However, it has been observed that port numbers alone do not hold much significance owing to widespread usage of dynamic negotiation, tunneling and the allocation of ports to commonly used programs for the purpose of masking traffic and bypassing firewalls. To enhance the efficiency of classifiers, it becomes crucial to combine port numbers with additional methods. In the forthcoming sections, various ML-based traffic classification methods are discussed. Employing a multiclassifier system or integrating multiple techniques has the potential to yield higher accuracy compared to a single classifier, as different techniques excel in different traffic categories. With recent advancements in machine learning, multi-classifier systems have emerged, which harness the expertise of multiple classifiers trained on the same flow objects. As a result, these systems are more accurate than a single classifier and can better withstand variations in the sample population, such as shifts in the types and combinations of applications. However, traffic categorization tools have only dabbled in simplified techniques, such as relying on hosts or machine learning if payload inspection fails. This strategy has been successfully utilized by network anomaly and intrusion detection systems.

Haffner et al. [56], suggested to use first some bytes to reduce computational work for unencrypted traffic containing TCP packets which are unidirectional as binary feature vectors. This is another approach which is in addition to port-specific this type of traffic classification, which uses the contents of the payload. Due to its high computational and storage requirements, it is not the most cost-effective method. Sources of data and their dynamics are constantly changing, maintaining, and adapting signatures manually is time-consuming. Additionally, because to growing security and privacy issues, encryption is required to the payload and privacy rule is applied to access it. As a result, extracting signature from a payload is not an easy task. TCP encryption settings are negotiated by extracting information from the unencrypted handshake in encrypted communication such as SSH and HTTPS. For traffic classification, they make use of NB, AdaBoost, and MaxEnt. Overall, AdaBoost surpasses NB (99 percent accuracy) and MaxEnt (less than 0.5 percent error rate). Author used ML techniques in this research to recognize signatures for a variety of applications automatically. According to the results, this method is highly accurate and may be used on high-speed networks to identify online applications. Content signatures continue to function even when encrypted, according to a recent discovery. In these cases, they talk about the encryption settings for a specific link to get content signatures for handshakes that aren't encrypted.

Their ML models are scalable and reliable because they can handle partial packages, one-way flows, and a variety of usage patterns. Asymmetric routing problems are avoided by flows that only go in one way. Residential network data is more varied in terms of age, social group, and interests, and there are less connections between usage trends in space and time. For the AdaBoost traffic analyzer to work well with noisy data, it needs to know ahead of time about the protocols in the application classes.

There is no need for prior knowledge of the application classes to achieve payload-based traffic classification using unsupervised clustering, according to Ma et al. [57]. They train their classifiers to identify structure in packet by giving them label of one case of a protocol and the partial correlations between several protocols. For every pair of sessions, there are source to destination one-way flows and its reverse. Their work contained, a hierarchical cluster analysis (HCA) was used to order the protocols that were seen and to find the classes of interest. The distance measure for PD and MP is weighted relative entropy, but for CSG it is approximate graph similarity. The PD-based protocol models were the most accurate when it came to the total number of wrong classifications. Because of this, binary and text systems like DNS and XML have a high degree of invariance at fixed offsets. Using HTTP, which is the Internet Protocol. Even though the CSG had a higher rate of wrong classification (about 7%), it was still the best choice for SSH-encrypted communication. Encryption, on the other hand, makes the data less predictable, which slows it down.

Finamore et al. [58] uses another approach where signature extraction part is not necessary for the encrypted traffic. With the rise of streaming apps, they are focusing on packet payload analysis to get application fingerprints from UDP data that has been sent for a long time. Using Pearson's chi-square test, author first took some bytes from packet and compared with randomness of other uniformly distributed experimental data with several grouping some consecutive bits in a set of some set of packets. With the signatures, an SVM classifier is trained to make a 99.6 percent accurate distinction between the classes of interest. FP, on the other hand, is more affected by the length of considered bytes and is only reduced by 5% for windows larger than 80. Despite the drawbacks, payload-based classifiers are frequently used to establish ground truth despite their high accuracy. Chi-Square Signatures are derived from the test findings and reflect application fingerprints compactly. As fingerprints derived from packet inspection, Chi-square Signatures have several advantages over traditional DPI signatures. No arduous and time-consuming reverse-engineering is necessary to come up with them. They can be used to sort data based on both where it goes and where it ends up. Online categorization is possible because of their small computational and memory needs. Although KISS and DPI classifiers both need to look at application layer signals, they are quite different tools. The disadvantage of both approaches is that they are rendered useless when dealing with encrypted payloads. The behavior patterns of the network hosts are used to identify these classes. Focusing on how different application interact end to end with much concern on encrypted traffic and port-based flow. As an example, a P2P host may use a separate port number to communicate with each of its peers [59].

Several viewers could connect to a web server through the same port. Timing and protocol connections are used to find HTTPS webmail traffic. They use the following in their favor: Most SMTP, IMAP, and POP servers, which can be found by their port numbers, are in the same domain or subnet as webmail services. (ii) Users of IMAP, POP, and webmail all use their accounts in different ways on a daily and weekly basis. For example, POP and IMAP users are more likely to leave the web client open to check for new messages, while IMAP users are more likely to stop the web client to check for new messages. Application timers cause webmail traffic to have regular trends (iv), like when AJAX-based clients check for new messages at different times. Using an SVM classifier, the authors show that these characteristics can be used to tell the difference between webmail traffic and other types of traffic. Author has used 5-fold cross validation with 93.2% accuracy. Value of FN came out was large because the classifier did not know how to tell the difference between VPN and webmail sites. It is important to remember that info from P2P apps is very selective. A P2P application can be designed to download video with large size chunk from a fellow peer and similarly a short flow can grab fixed-size chunks of video at the same time.

P2P application signatures can be derived from the packets transmitted between peers in small time frames proposed by Bermolan et al. [60]. The authors look at the sensitivity of the parameters so they can improve their settings, which means a higher True Positive Rate (TPR) and a lower False Positive Rate (FPR) for getting the best result. On applying their proposed algorithm, the TPR is about 95% and the FPR is less than 0.1%. Signatures are also shown to be portable across time and place with only a small drop in performance. Still, observed communication patterns can be changed by routing differences in the network core, so the position of the monitoring system has a lot to do with how accurate host behavior-based traffic classification is. Support Vector Machines are used to accurately identify P2P-TV traffic and traffic from other types of apps in this classification framework. This means that almost no wrong categorizations happen. In this work, the reliability of the rejection criterion is calculated by identifying rate of false alarms in worst-case scenario. The conclusive part was signatures used by two peers gets the result (62% in CAMPUS and 82% in ISP) for real traffic, which is used to measure how many false positives there are: There is a match between 72% and 84% of the UDP traffic. The categorization framework is more robust because this traffic cannot be misclassified.

Jun Zhang et al. [61] suggested RTC which is based on supervised full flow feature-based traffic classification. They use both supervised and unsupervised ML techniques that classify previously unidentified zero-day traffic to improve accuracy. The reason is that zero-day activity is in unlabeled data. This RTC structure is made up of three parts: figuring out what is unknown, putting traffic into categories based on the BoF, and keeping the system up to date. The BoF module makes sure that zero-day samples are clean and puts flows that are related together. The system update module adds to what is already known by learning new classes from zero-day data that has already been found. RTC is special because it can use correlated flows to show real-world scenarios and find zero-day applications. In case of considering a least size of labeled trained datasets also, RTC can be up to 15% more accurate with flow and up to 10% more accurate with bytes than the second-best method also using the discriminative MLP-NN classifier, increases the accuracy of traffic classification with 99%. A study of travel data from the real world has shown that this plan works. For zero-day attack their work performed better. In another approach by Auld et al. [62], classifier's temporal accuracy was increased to 95% by MLP-NN with BNN approach.

Este et al. [63] sorts traffic using both single-class and multi-class SVM for categorizing traffic in IP networks is Support Vector Machines (SVM). There are still certain issues to be resolved before SVMs are routinely employed as traffic classifiers. It is still being investigated how to apply them to problems with more than two classes because they were primarily designed for situations with two classes. Additionally, how well their functioning parameters are tuned greatly affects how well they perform. In this study, we discuss the traffic classification capabilities of SVM. The author classified statistical traffic using an approach for solving multi-class issues by considering SVM classifier before describing a straightforward optimization strategy that enables the classifier to function properly with as few as a few hundred examples of training data. Even though the results are still very recent, they demonstrate that even with a limited training set, SVM-based classifiers were effective for traffic from various sources. However, for encrypted communication, the performance is not as good because it was relying on the port-based labeling, which degrades its throughput but with the increase in computation costs.

Jing et al. [64] suggests an SVM approach for context-based traffic. At each round during SVM design, the candidate classes are paired at random. A binary SVM classifier then selects one class from each combination, thereby halving the number of potential classes which reduces count of support vectors that rely on two-pair classes. Using multi-class classification in SVM significantly reduces the cost of classification compared to complete training dataset. Traditional and multi-class SVM are not as effective as they could be at managing large datasets. It is essential to remember that the best classification could be removed, resulting in more incorrect classifications with increase computational cost. The authors contrast the existing SVM with the basic and FT-SVM schemes they have developed with 96% accuracy while reducing classification error by 2.35 and computation by 7.65 times. It is uncommon to have information about all the programs operating on a network, so supervised learning cannot always be applied to network traffic.

Liu et al. [65], to achieve up to 90% accuracy with k-Means. Unsupervised training was part of the experimental analysis that utilized full traffic flows and log transformations of characteristics to fetch features to get as close as feasible to achieve stable distribution, removing anomalies from the data. However, it would be illogical to use rigid clustering to determine membership because flow characteristics from sources like HTTP and FTP may match.

Yang et al. [66] built and showed first convolutional neural network that automatically extract features from short strings for cyber security problems. Using embeddings and convolutions as the top layers of our neural network and training it under supervision lets us automatically get a set of features that are directly optimized for classification. Even though similar approaches have been this for natural language processing (NLP), eXpose is the first approach that shows how top-to-bottom deep-learning methods resolve several important cyber security problems in an adversarial environment where strings are purposely messed up to stop obvious feature extraction. During our experiments, one of the biggest problems was that training on longer strings took a lot of computing power, so we could not try out more complex architectures. With recent improvements in hardware and the addition of distributed training modules to modern frameworks, the results of some of the more expensive architectures may be able to get even better. For end-to-end learning, these ideas are built into eXpose by using labeled datasets that already exist. As hardware and data sets get better, the difference between automatically extracted features and traditional ways of extracting features will only get bigger.

Huda [67] proposed new malware detection system for CPS which is semi-supervised. The new thing about this system is that it can protect CPS against new types of malwares without the manual work needed to create signatures and keep the repository of antimalware tools up to date. This technique can compensate for the inability of CPS to be utilized with older control software or operating systems (OS) due to impossibility of updating organization (due to time constraints, personnel costs, and anticipated asset loss). In this system, the hidden data structure's geometric information that was recovered also contains current details on any new variations. Unsupervised learning is used to accomplish this, and the supervised detection engine is subsequently updated with the new data. So, this method protects against new malware without requiring the database to be manually updated or labeled.

Kamarudin et al. [68] Anomalies and invasions have been studied extensively. Still, recognizing invisible threats and reducing false alarms is difficult. The Logitboost-based classifier detects known and unknown online attack activity in this work. Their approach used NSL-KDD and UNSW-NB15, two publicly accessible labeled intrusion detection testing datasets, to create diverse integration testing scenarios. Preprocessing eliminated redundant or unimportant elements. Data mining with the Logitboost classification algorithm yielded high detection rates and low false alarms. This study found that ensemble technique may detect unknown attack and their results used to signed and stored in a database. Since new traffic may be matched with signatures of good or bad traffic from prior detection, finding something is much faster. Finally, this ensemble approach can be tested online utilizing previously intercepted and current encrypted traffic.

Ziyang Guo et al. [69], proposed residue-based detection technique that detect sensor communication pattern by the malicious attacker to alter sensor data. They recommend a linear deception attack technique and display the associated feasibility constraint. Due to its rapid growth and safety-critical applications, CPS security has drawn more attention recently. These systems are susceptible to cyber threats because CPS frequently sends measurement and control data across unencrypted communication networks. Any successful CPS attack might result in a few negative outcomes, including the disclosure of consumer information, harm to the national economy, destruction of infrastructure, and even the risk to people's lives.

Offline data fetched from numerous archives can be utilized for research purpose as long as its validity exist in networking. Some most common are the UNSW-NB15 [70], the UCI KDD Archive [71], MAWI [72], and IMPACT Archive [73]. Using monitoring and measurement tools is a good way to get data from both offline and online sources. These tools give more control over different parts of data collection, like the rate of data sampling, the length of monitoring, and the position of network core and edges. Monitoring networks is usually done with SNMP [74], Cisco Net-Flow [75], and IPFIX [76].

Apart from the above data several other researchers also worked on the known datasets and achieved certain level of accuracy. Ambusaidi et al. [77] It gives general data that can be changed for IDS feature description. Using SVM with 19 selected features, KDD'99 record got the highest accuracy of 99.79 percent. The model applies code correlation analysis to find the required attributes and the ideal attributes (depending on the value below the upper limit of the CCA upper limit in the NSL-KDD dataset) with a 98.9% accuracy [78]. How well different groups of features work in KDD datasets, Khammassi and Krichen [79] came up with a way to find outages and fix them using repeated calculations, genetic calculations, and strict research methods increased their accuracy of DTC by 99.9% and 0.105% for the KDD'99 dataset with 18 features. Akashdeep et al. [80] suggested correlation and data retrieval will reduce if wrong artificial neural network classifiers for detection is used. The system is the most accurate in the KDD'99 dataset, with a 99.9% accuracy rate. It is used for DoS attacks on ANNs with 25 features.

Divyasree and Sherly [81] proposed a screening technique for network IDS, that uses chi-square tests to suggest many simplified KDD'99 functions. This method reduces 10 features that could be used in DoS attacks. Using CVM, the detection rate is 99.12%. For the definition of the logical dependence of IDS, it is best to use rough and well-thought-out calculations. In the research [82], 22 attributes were used, and fuzzy rules were added to the k-nearest neighbor classifier (FRNN). It gives 99.87 percent of the detection area in a dataset with only one direction and half the types of data. The framework [83] has a 99.79% recognition rate because it uses 26 functions from a one-time dataset. Yulianto et al. [84] used an intrusion detection system based on Adaboost to come up with a plan to cut down on the number of parts in an IDS. With 25 functions and the PCA feature combination along with Ensemble Feature Selection (EFS), it can be accurate 81.83 percent of the time. Using a selection-based Bayesian and roughset approach, Prasad et al. [85] came up with a way to cut down on noise. With a random population structure calculated with negative definitions for the NSL-KDD dataset [86], it can improve accuracy by up to 99 percent. Also, REP Tree [87] uses 10 features from the global identifier and the small dataset, which can give the highest accuracy of 99.73%. Only small differences can be seen between the signatures of different attacks and is the best approach to identify who is breaking into the system [88].

Numerous authors have worked on different datasets with different machine learning approach and found variation in result. Different methodologies such as FGLCC-CFA, C4.5 DMIFS, PKID + Cons + C4.5, SVM, GA-LR wrapper, ANN, CVM, Ensemble of C5 DTs, NN and C4.5 DT, BN and CART, NB, and others have been applied. Performance metrics vary across methodologies and features, with detection rates ranging from 80% to 100% for normal instances and varying rates for different types of attacks such as DoS, Probe, U2R, R2L, and Satan while using KDD Cup datasets.

Using CICIDS 2017 dataset, multiple methodologies such as Bayesian Rough set, Ensemble MPML, TCM K-NN, Random Forest, Information Gain + Random Forest, and AdaBoost have been employed. Detection rates for different types of attacks like DoS are consistently high, reaching up to 99.96%. The number of features used in each methodology according to selection of features varies and affecting the performance metrics.

Table 3. Review of various work done on different datasets and ML techniques

Paper Title [Ref]	**Dataset**	**Methodology**	**Features**	**Performance**	**Limitation**
Cyber intrusion detection by combined feature selection algorithm [39]	KDD Cup	FGLCC-CFA FGLCC 10Fold - FGLCC-CFA	10 Features 15 Features 15 Features	95.03 92.59 99.84	Low detection Low detection High detection

Feature Grouping for Intrusion Detection Based on Mutual Information [41]	KDD Cup	C4.5 DMIFS FGMI	13 features	Normal: 99.3 Anomaly: 91.4	Low detection
Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset [46]	KDD Cup	PKID + Cons + C4.5	13 Features	Normal: 99.3 U2R: 25 DoS: 96.08 R2L:8.12 Probe:73.62	Low detection then KDD Winner 99.45
Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm [77]	KDD Cup	SVM	19 Features	99.79	High Accuracy
A GA-LR wrapper approach for feature selection in network intrusion detection [79]	KDD Cup	GA-LR wrapper	18 Features	99.9	High Accuracy with delay
A feature reduced intrusion detection system using ANN classifier [80]	KDD Cup	ANN	25 Features	99.9	High Accuracy with 25 features
A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach [81]	KDD Cup	CVM	10 Features	DoS:99.12	High Accuracy for DoS attack only
An efficient feature selection based Bayesian and Rough set approach for intrusion detection [85]	CICIDS 2017	Bayesian Rough set	40 Features	96.38	Low accuracy as 40 features used.
Artificial neural networks for misuse detection [89]	RealSecure	Supervised NN	Payload, header of TCP, IP, and ICMP	Detection Ration: 89-91	Low Detection
Winning the KDD99 classification cup [90]	KDD Cup	Supervised Ensemble of C5 DTs	41 features	Normal:99.5 Probe:83.3 DoS:97.1 U2R:13.2 R2L:8.4	Normal:High Probe:Medium DoS:High U2R:Low R2L:Low Using all features
Hybrid neural network and C4.5 for misuse detection [91]	KDD Cup	Supervised NN and C4.5 DT	41 features	Normal:99.5 DoS:97.3 Satan:95.3 Portsweep: 94.9 U2R:72.7	High Detection for R2L attack

				R2L: 100	
A neural network-based system for intrusion detection and classification of attacks [92]	KDD Cup	Supervised NN	35 features	MLP:80 ESVM:90 ESVM DR:87	Low Detection
Feature deduction and ensemble design of intrusion detection systems [93]	KDD Cup	Supervised BN and CART	41 features	Normal:100 Probe:100 DoS:100 U2R:84 R2L:99.47	High detection for Normal, DoS and R2L
Naive Bayes vs decision trees in intrusion detection systems [94]	KDD Cup	Supervised NB	41 features	Normal: 97.68 DoS: 96.65 R2L: 8.66 U2R: 11.84 Probing: 88.33	Low Performance
Decision tree classifier for network intrusion detection with GA-based feature selection [95]	KDD Cup	Supervised C4.5 DT	GA-based Feature Selection	DoS:97.88 Probe: 98.33 R2L: 80.01 U2R: 99.99	High performance for U2R
Feature deduction and ensemble design of intrusion detection systems et al. [96]	KDD Cup	Supervised Ensemble of SVM, DT, and SVM-DT	all 41 features	Normal: 99.7 Probe:100 DoS: 99.92 U2R: 68 R2L: 97.16	High detection for Probe and DoS
Practical real-time intrusion detection using machine learning approaches [97]	RLD09	Supervised C4.5 DT	Header of TCP, UPD, and ICMP	Normal: 99.43 DoS: 99.17 Probe: 98.73	High detection
Multi-Perspective Machine Learning a Classifier Ensemble Method for Intrusion Detection [98]	NSL-KDD	Supervised Ensemble MPML	all 41 features	84.137	Low accuracy
An active learning based TCM-KNN algorithm for	KDD Cup	Supervised TCM K-NN Chi-square	41 features 8 features	99.7 99.6	High accuracy

supervised network intrusion detection [99]					
Kshirsagar et al. [100]	CICIDS 2017	IGR-SCS1 CR-SCS2 ReF-SCS3	48 Features 24 Features 12 Features	DoS:99.9586 DoS:99.9593 DoS:98.8698	High accuracy
CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection [101]	CICIDS 2017	Random Forest	15 Features	99.81	High accuracy with only 15 features used
An efficient feature reduction method for the detection of DoS attack [102]	CICIDS 2017	Information Gain + Random Forest	22 Features 28 Features	99.83 99.79	High accuracy but more features used
Development of a Method for Detecting Network Attack on Machine Learning Algorithms [103]	CICIDS 2017	AdaBoost	10 Features	99	High accuracy with less features

These research outcomes by various authors demonstrate the effectiveness of various methodologies for intrusion detection across different datasets. Performance metrics vary based on the methodology, features used, and the specific dataset characteristics. High detection rates for normal instances and different types of attacks indicate the potential of these methodologies in effectively identifying and mitigating these cross-layer attacks.

2.4 Anomaly and misuse detection techniques

Anomaly detection technique reflects any deviations in the normal pattern. Mismatches can be found by looking for both static and dynamic deviations [104]. The static deviation exception works because it is known that the part of the network that needs to be checked won't change. Most static detectors look at the software part in networked system because it is thought that there is no need to check the hardware. The part of the frame that doesn't change is the frame structure and the consistent information segment. This information is important for the situation to work right. For example, the system information and PC startup information will never change. This information is stored in system for long duration of time unless a new hardware is added and software has changed. However, the system can deviate from its unique structure at any time if an error happens or if an attacker changes the static parts of the system. The framework that finds the log does not keep track of every case. All they do is keep track of interesting cases.

Misuse detection: Information about framework weaknesses and actual attack designs can help find misuse. Misuse detection tries to find deviation a hacker who are trying to break into a framework by misusing a weakness that is already known. If there are no outside protections, a misuse detection framework tries to find out about every known weakness and fix it. If it does not, it could lead to a system attack [105]. An intrusion detection framework constantly checks a running process for known intrusions. This is done to make sure that at least one attacker is not trying to take advantage of known weaknesses, so every intrusion should be found. Misuse detection systems uses rules to detect events for unauthorized activity that a security module checks in system. It can be hard to keep track of and explain complicated rules. Because there are many ways to find intrusions, it is unlikely that the rules will not be caught. In this way, changes to the standard rate can be complicated because the rules that are affected are spread out over the whole standard rate. The new stratification strategy in the guide links model-based principles to government progress to deal with these problems. As part of figuring out when someone is misusing something, these principles are used to find situations that might be good for doing something bad. It can be seen in real time by looking at the packets coming in or later by looking at the log.

Anomaly detection and misuse detection techniques have their advantages and disadvantages. Often, new, or unknown attacks, as well as variations of normal attacks, go unnoticed by traditional abuse detection strategies. Responding to such attacks with conventional methods may not effectively mitigate the harm they cause. Irregular identification methods can identify novel or undisclosed attacks in the programming environment, detect variations of known attacks, and identify deviations from the typical usage patterns of a project, whether they originate from authorized internal users or unauthorized external users. However, attacks that are observable may not appear distinct, especially if they align with established client profiles. Understanding the motives behind launching attacks for scientific purposes can also be challenging to explain. Additionally, certain unique identification methods are prone to manipulation by malicious clients who gradually modify their profiles over time, altering the breach detection framework to reveal the attacker's personal profile. This tactic is commonly employed as a means of seeking retribution. The false alarm rate in intrusion detection can be influenced by well-trained methods for estimating targets or determining uniqueness.

Table 4. Comparison of Anomaly and Misuse detection technique

	Anomaly Detection	Misuse Detection
Motivation	Identifying deviations from normal behavior or patterns within a system	Focuses malicious activity or specific attack signatures
Methodology	Baseline of normal behavior using historical data or statistical models and flags any data points or activities that significantly deviate from this baseline as anomalies	Relies on predefined signatures, rules, or patterns of known attacks to identify malicious behavior
Resource Intensity	Regular updates to its signature database	Require more computational resources to keep baseline of normal behavior
Use case	Detect novel attacks or previously unseen threats, such as zero-day exploits or insider threats	Effective at catching known threats, such as viruses, worms, or known types of cyberattacks
False Positives	Generate more false positives because it flags any deviation from normal behavior	May have fewer false positives but may miss new, unknown threats
Example	Unusual login times or locations, unexpected spikes in network traffic, unusual file access patterns	Matching network traffic against a database of known malware signatures, identifying specific command

		sequences indicative of a known attack

In terms of intrusion detection systems, there are two main categories. Host-based intrusion detection systems count on data sourced from individual hosts, typically for monitoring purposes. On the other hand, network-based intrusion detection systems analyze network traffic within the organization to which the host is connected, providing a broader view of potential intrusions.

A. A Host-based Intrusion Detection System (HIDS) is a cybersecurity tool designed to monitor and analyze the activities occurring on a single host or endpoint within a network. HIDS focuses on detecting unauthorized or malicious actions that occur directly on a specific host. HIDS operates by continuously monitoring various aspects of a host's activities, such as file system changes, log entries, process executions, and system calls. It compares observed behavior against known patterns of malicious activity or predefined rules to identify potential security breaches or anomalies. By providing real-time monitoring and detection capabilities at the host level, HIDS can help organizations identify and respond to security incidents promptly, protecting against threats such as unauthorized access, malware infections, insider attacks, and system vulnerabilities [106].

B. Network-based IDS: A Network Intrusion Detection System (NIDS) is a cybersecurity tool designed to monitor and analyze network traffic for signs of suspicious or malicious activity. NIDS operates at the network level, inspecting traffic as it flows across the network. As use of Internet increases with number of hosts but if there is misconfiguration in the organization then chances of intrusion may happened [107]. By continuously monitoring network traffic, NIDS can identify various types of threats, including malware infections, denial-of-service (DoS) attacks, port scans, intrusion attempts, and unauthorized access attempts. Upon detecting suspicious activity, NIDS can generate alerts or trigger automated responses to mitigate the threat and protect the network infrastructure [108].

Data mining method for detecting intruders: Data mining techniques are increasingly being applied to intrusion detection systems (IDS) to improve their ability to detect and respond to cyber threats. leverage large datasets to identify patterns, anomalies, and indicators of malicious activity within network traffic or system logs [100]. The information retrieval method is a new strategy in a data mining method for detecting intrusions. It changes how rare information is, how rules and designs are made, and how much information there is. The goal of intelligent analysis of information is to get rid of information that cannot be seen well with the usual methods [109]. Mining calculations come in many forms, such as sequence, repetition, grouping, membership rule retrieval, checking for differences, and array exploration. Different mining strategies are used to avoid being found because they have important benefits. The large summary information is used to index information about how the client or program acts [110]. By default, known attacks and errors are written into NIDS. Because of this, it is not sensitive to certain actions and can only tell between a limited number of types of intrusions. A SVM can map vector training in a space with many elements and figure out which group each vector belongs to. Support vector machines have been a good way to find intrusions because they are fast and easy to change.

2.5 Data filtering and feature selection

The aim of data filtering corresponds to cut down undesired information that the IDS must deal with directly. Some information may not help the IDS, so it can be discarded before it is handled. This has the benefit of reducing the amount of space needed for storage, cutting down on preparation time, and improving the rate of identification [111].

Feature selection plays a crucial role in the detection of network attacks as it helps identify the most relevant attributes for differentiating between normal and malicious network traffic. By selecting a subset of features, the complexity of the data is reduced, leading to improved accuracy and efficiency in the classification models employed for attack detection. This process is essential for enhancing the overall effectiveness of network security measures.

For the feature selection process following steps were followed:

a. Define the problem: To define the problem and identify specific type of network attacks that need to be detected. This will help determine the relevant features that need to be selected for the analysis.

b. Data collection: The second step is to collect data related to network traffic, which includes various features like source and destination IP addresses, packet size, protocol, and timestamps, among others. This data can be obtained from network sensors, firewalls, or intrusion detection systems.

c. Data pre-processing: The collected data is often noisy, incomplete, and inconsistent, which can affect identification of correct feature selection process. Therefore, a data needs pre-processing to remove irrelevant or redundant features, fill missing values, and standardize the data.

d. Feature ranking: After the pre-processing of data, it becomes necessary to prioritize the features according to their significance in solving the problem at hand. Several techniques can be employed for feature ranking, such as correlation, information gain, and chi-square feature selection, among others. These methods help identify the most influential features that contribute to the desired outcomes, enabling more effective decision-making and problem-solving in the given context.

e. Feature subset selection: After the features have been ranked, the subsequent stage entails choosing the most pertinent subset of features. This selection process involves employing a feature selection algorithm, such as greedy forward selection, greedy backward elimination, or genetic algorithms. These algorithms aim to identify the finest subset of features which can maximizes performance of classification model, thereby enhancing the accuracy and efficiency of the analysis. By carefully selecting the subset of features, the subsequent classification or prediction tasks can be significantly improved.

f. Classification model: After the relevant feature subset has been selected, the next step involves training a classification model on the data to effectively differentiate between normal and malicious traffic. Various models can be employed for this purpose, such as decision trees, neural networks, support vector machines (SVM), or any other suitable model that aligns with the specific problem at hand. These models are trained using the selected features to learn patterns and create a reliable framework for classifying incoming network traffic accurately.

g. Evaluation: Finally performance of feature selection process and classification model needs to be evaluated using a range of metrics for instance accuracy, precision, recall, and F1 score. Such evaluation measures offer insights into effectiveness and reliability of chosen features as well as the classification model in accurately identifying and classifying network traffic. By assessing these metrics, researchers and practitioners can gauge the overall performance and effectiveness of their approach, allowing for further refinement and optimization if necessary.

Feature selection process involves selecting the most relevant feature subsets from available data for enhancing the accuracy and efficiency of the classification models used for attack detection. The process involves several steps, including defining the problem, data collection, data pre-processing, feature ranking, feature subset selection, classification model, and evaluation. By following this process, network administrators can detect and prevent network attacks in real-time, ensuring the security and integrity of their network infrastructure. Some features can be used more than once because the data they add is already in other features [112]. This makes IDS less accurate. Defined functions improve characterization by finding subsets of features that can better group information such as IP address destination packet, protocol, timestamp etc.

In conclusion, many IDS relies on a database of known attack signatures to identify malicious activity. As a result, they can only detect attacks for which signatures are available. Zero-day attacks or variants of known attacks that deviate from existing signatures may go undetected. Also, making new signatures requires expensive and time-consuming manual review by human experts. This means that there is often a gap between finding a new attack and making a signature for it by selecting appropriate features for reduced computation.

2.6 Datasets for Intrusion Detection System

IDS were developed to find and stop hackers before they could cause any damage to a computer network. There are several datasets available that can be used for IDS research and evaluation.

2.6.1 DARPA 1998

In connection with a DARPA IDS Evaluation program, this dataset was compiled and includes network traffic data from various sources, including LAN and WAN. The dataset contains processes like FTP, browsing the internet, sending/receiving e-mail, IRC data etc. Aside from normal network traffic, there are 38 attacks classified as DoS, U2R, Probe, and R2L [113]. DARPA dataset is out-of-date with non-real-world network traffic. It contains some flows that also interprets as false positive because it interprets some of the benign as attack [114]. It is longer used but importance of DARPA98 dataset still exist because it was used to create datasets like KDD'99 and NSL-KDD.

2.6.2 KDD Cup 1999 (KDD'99)

For research, its most used IDS dataset that contains appropriate number of network traffic instances, both normal and anomalous to train and test IDS algorithms, derived from DARPA98 dataset[115]. Training and a testing portion are provided on the website where training section contains 4898431 data streams, while the testing section contains 311029 data streams. A total of 38 different attack types can be found within the KDD'99 dataset out of which 14 were newly added unknown attacks. As a result, it is possible to regulate for identifying novel attack using test section. Many studies have used the KDD'99 dataset instead of DARPA99 because of it contain selected 41 features with training and test parts, which are more suitable for machine learning methods [116].

2.6.3 NSL-KDD

An extension of the KDD'99 dataset that addressed some of its limitations. It contains a more balanced distribution of normal and anomalous instances, as well as more realistic attack scenarios. It was found that the number of repetitions in KDD'99 adversely affects the results of studies and desired output of algorithms for applied ML approach. Apart from this, researchers have attempted to use only a portion of the KDD'99 dataset due to its large size. When a portion of dataset is considered, it does not cover all the dataset's features, so selecting dataset need proper attention.

For addressing these issues, Siddique et al. [117] came up with NSL-KDD dataset in 2009. Errors and repetitions in KDD'99 were eliminated in this version. This proposed NSL-KDD dataset was partitioned into two main categories: KDDTrain+ and KDDTest+ with KDDTrain+ 20Percent for training.

2.6.4 UNSW-NB15

This dataset contains network traffic instances generated in a realistic environment that contains attacks such as DoS, probing, and exploitation.

2.6.5 ISCX 2012

This dataset includes network traffic data captured in a realistic environment that contains several attack categories such as DoS, DDoS and botnet. On the Canadian Institute for Cybersecurity testbed, a seven-day Internet stream was used to create ISCX 2012 IDS dataset [118]. This dataset was built in real scenario, which contains both malicious and normal flow like “FTP”, “HTTP”, “IMAP”, “POP3”, “SMTP” and “SSH” protocols (Infiltration, DoS, DDoS and SSH Brute Force). According to ISCX 2012, more than half of all Internet traffic is made up of SSL/TLS traffic, not included in the ISCX 2012 dataset [119]. These datasets can be used for a variety of purposes, including training and testing IDS algorithms, evaluating performance of different IDS techniques along with comparing effective IDS systems.

When using machine learning to identify network anomalies, it must contain a substantial amount of malicious and normal traffic for training and testing. Privacy issue is the major concern due to this an actual network traffic cannot be used. Several datasets have been and continue to be developed in order to accommodate this demand. For analyzing attack in other than datalink and network layer of TCP/IP model, KDD'99 and CICIDS dataset is selected for final implementation.

2.6.6 CICIDS2017

This dataset contains network traffic data captured in a realistic environment that includes new attacks and malware. This dataset contains five days of data. The Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) is a standard dataset for network security intrusion detection systems (IDS) that covers large set of actual and variety of network traffic scenarios. The dataset was created by capturing real network traffic and conducting series of attacks on test network. The traffic was then processed to remove any sensitive information and anonymized to protect the privacy of individuals and organizations involved. The resulting dataset contains approximately 2.5 million network flows, including both benign and malicious traffic, and a total of 79 features or attributes for each flow.

The CICIDS2017 dataset is composed of two parts:

Training set: This set contains approximately 560,000 network flows, including both benign traffic and covering several attacks at different layers of TCP/IP model, such as DoS (Denial of Service), DDoS (Distributed Denial of Service), and Brute Force attacks, among others by conducting attacks using a variety of tools, including Metasploit, Nmap, and Hping3, among others. Its testing set contains approximately 1.9 million network flows.

Features in CICIDS2017 dataset are clustered into six categories:

Basic flow features: These features describe basic information about the flow, like source and destination IP addresses, the protocol used, and packet flow count.

Content features: Describe the content such as the payload size and the occurrence of definite keywords or patterns.

Traffic features: These features describe the traffic patterns, such as the inter-arrival time between packets flow intervals.

Time-based features: Describe the timing characteristics of flow, such as start and end times and flow interval.

Connection-based features: These features describe the characteristics of the connection, like count of packets and bytes exchanged, count of packets and bytes per second, and number of connections between the same source and destination IP addresses.

Network behavior features: This describes overall behavior of the network, such as number of flows and connections per second and the overall network throughput.

The CICIDS2017 dataset provides a comprehensive and realistic benchmark for evaluating the performance of IDS in detecting several network attacks. Diverse features and attack ranges in the dataset makes it asset for investigators associated in area of network security. It contains more protocols than any other data set. Apart from protocol such as FTP, HTTP, SSH, and e-mail, it supports HTTPS (Hypertext Transfer Protocol Secure) but in contrast to KDD'99 and NSL-KDD, this dataset does contain separate files for training and testing.

Table 5. Dataset features and attack types

Datasets	No. of Features	Attack types	Remarks
DARPA 1998	41	DoS, R2L, U2R, Probe	Combined Non real traffic, irregular attack instance
KDD Cup 99	41	DoS, R2L, U2R, Probe	Contains many duplicate data samples
NSL-KDD	41	DoS, R2L, U2R, Probe	non-redundant instances from KDD Cup, Limited count of attack
UNSW-NB15	49	DoS, R2L, U2R, Probe, Reconnaissance	Data collected from realistic virtualized environment
ISCX 2012	42	DoS, PortScan, and DDoS	Network traffic from medium sized organization
CICIDS2017	79	DoS, DDoS, Brute force, Portscan, Botnet, Web, Infiltration	Data collected from real-world network traffic

2.7 Network Attacks and its categories

In this section, the various attack types found in the data set are discussed in depth.

DoS HULK: Denial-of-Service HULK [120] attack falls under category of distributed DoS attack which attempts to interrupt web server with huge volume of HTTP requests with the goal of consuming server resources which interrupt the service to legitimate users. DoS HULK attack uses botnet that are controlled by command-and-control center hosted by an attacker with a huge number of compromised computers. The botnet is used to flood target server by high volume of HTTP GET and POST requests, targeting specific pages or resources on the server. The requests are sent in rapid succession, overwhelming the server and causing it to become unresponsive or crash. The requests are often designed to be malformed or include invalid parameters, which can consume additional server resources and make it more difficult for the server to process legitimate requests. The HULK attack is particularly effective against web servers that have limited resources and are not designed to handle high levels of traffic. It can be difficult to mitigate the attack once it has started, as the botnet can be distributed across many different computers and IP addresses, making it difficult to block all the traffic.

To mitigate a HULK attack, web server administrators can implement various techniques, including rate limiting, filtering out suspicious traffic, and using load balancers to distribute the traffic across multiple servers. Additionally, network administrators can use intrusion prevention systems (IPS) and firewalls to detect and block suspicious traffic coming from known malicious IP addresses.

GET Flooding via the Internet: A GET Flooding via the Internet attack is category of DoS attack which attempts to interrupt web server with huge volume of HTTP GET requests. The botnet is used to flood target server by high volume of GET requests, targeting specific pages or resources on the server. The requests are sent in rapid succession, consuming the server's resources and causing it to become unresponsive or crash. The requests are typically targeted towards a specific page or resource on the server, such as a login page, and are designed to mimic legitimate user requests. However, since the requests are coming from a botnet and are being sent at a much higher rate than legitimate users, the server is unable to keep up with the request volume and is overwhelmed. One of the challenges of defending against a GET Flooding via the Internet attack is that the requests are typically legitimate HTTP requests, making it difficult to filter them out without also blocking legitimate user traffic. Additionally, the botnet can be distributed across many different computers and IP addresses, making it difficult to block all the traffic.
To mitigate a GET Flooding via the Internet attack, web server administrators can implement various techniques, including rate limiting, filtering out suspicious traffic, and using load balancers to distribute the traffic across multiple servers. Additionally, network administrators can use intrusion prevention systems (IPS) and firewalls to detect and block suspicious traffic coming from known malicious IP addresses [121].

PortScan: Nmap [122] is responsible for the PortScan attack on the CICIDS2017 dataset. A PortScan attack does network reconnaissance by scanning target host or network for open ports and services. The purpose of a PortScan attack is to collect information specific to target network that for identification of potential vulnerabilities or to plan further attacks. The attacker typically uses a tool such as Nmap to send a customized set of network packets to the target network, with the goal of identifying running services and open ports. The attacker may use several PortScan techniques, like TCP SYN scans, UDP scans, or FIN scans, for such running services or open ports. Attackers further plans for attacks once these running service or open port vulnerability is identified. For example, if attacker identifies an open SSH port, they may attempt to brute force the login credentials to get unauthorized access to target system. Alternatively, if an attacker identifies an open web server port, they may attempt to exploit known vulnerabilities in the web server software to gain access to sensitive data or to launch further attacks.

To defend against PortScan attacks, network administrators can implement various techniques, such as network segmentation, firewall rules, and IDS, to identify and block suspicious network traffic. Additionally, system administrators can reduce the attack surface of their systems by closing unnecessary ports and disabling unused services. It is important to monitor network traffic for unusual patterns and keep update with security patches to prevent exploitation of known vulnerabilities [123]. Categorization port scanning methods into the following categories:

Port scanning methods can be categorized by type of network packets that are sent to the target network to identify open ports and services. Some common categories of port scanning methods include:

a. TCP SYN Scans: This is the most common category of PortScan attack where in an attacker sends SYN packet to selected network's ports. When the port is open in response to the SYN packet it replies with SYN-ACK. RST (reset) packet is sent by an attacker after this to close this connection. This technique is stealthy because it only initiates a partial connection, and the target network may not log the event.

b. UDP Scans: In this method, the attacker sends a UDP (user datagram protocol) packet to selected network's ports. When a port found to be open, the target network may reply with a UDP packet and when a closed port, the target network may respond with an ICMP (Internet control message protocol) packet. UDP scanning is typically slower than TCP SYN scans, as UDP packets are not guaranteed to be delivered and may require multiple attempts.

c. FIN Scans: In this method, the attacker pushes FIN (finish) packet to desired network's ports. When a port found to be open, the target network may reply with an RST (reset) packet. and when a closed port, the target network may not respond at all. This technique can be effective against older systems, but many modern operating systems have implemented defenses against FIN scans.

d. XMAS Scans: In this method, the attacker pushes packet with the FIN, URG (urgent), and PUSH flags set to desired network's ports. When a port found to be open, the target network may reply with an RST (reset) packet. and when a closed port, the target network may not respond at all. XMAS scans can be effective against some systems that are vulnerable to this type of attack.

e. NULL Scans: In this method, the attacker pushes packet with no flags set to desired network's ports. When a port found to be open, the target network may reply with an RST (reset) packet. and when a closed port, the target network may not respond at all. NULL scans can be effective against some systems that are vulnerable to this type of attack.

Overall, there are various types of PortScan techniques, each with their own strengths and weaknesses, and network administrators need to be aware of these techniques to detect and prevent PortScan attacks.

DOS: The program LOIC [124] is responsible for the DDoS attack on the CICIDS2017 dataset, which sent HTTP, TCP, and UDP requests. DoS attack intended to deny normal operations of a target system, network, or service by sending huge volume of traffic. The requests are sent in rapid succession, overwhelming the server and causing it to become unresponsive or crash. The requests are often designed to be malformed or include invalid parameters, which can consume additional server resources and make it challenging for server to process legitimate requests. DoS attacks are executed by a variety of techniques, like flooding the target system with traffic, exploiting vulnerabilities in the system to crash it, or overwhelming the system's resources by consuming its memory or CPU. Some common types of DoS attacks include:

a. DDoS: It involves compromised sources or "bots" to flood the destined system or network with traffic. DDoS attacks can be difficult to defend against, as the traffic can originate from many different sources.

b. Application-layer DoS: This type of DoS attack targets specific applications or services by exploiting their vulnerabilities. For example, an attacker may flood many requests to a web server, overwhelming its capacity for reacting legitimate requests. These attacks are referred as a "HTTP flood" or "Slowloris" attack.

c. Network-layer DoS: This type of DoS attack targets the network infrastructure, such as routers or switches, by pushing a huge traffic to consume their resources. Such attack is also termed as "ping flood" or "smurf" attack.

Some common DoS attack types are:

a. TCP SYN Flood: It exploits the three-way handshake mechanism of TCP that flood server with connection requests that are never completed, causing the server to become unavailable to legitimate users.
b. UDP Flood: This attack involves overwhelming a server with a flood of UDP packets that crashes or inactivate targeted server.
c. Smurf Attack: The broadcast address on the network receives an ICMP echo request and in response to this all host replies back which creates flooding of request on targeted server.
d. Ping of Death: Attackers sends a large size packet to a server or network, which can cause the system that crashes or inactivate targeted server.
e. HTTP Flood: On target server an attacker sends huge volume of HTTP requests with the aim of consuming its resources and denying services to users.
f. Slowloris: Attackers sends an incomplete HTTP request to targeted server, results in holding connections open that denying services to legitimate requests.

g. DNS Amplification: Attackers send DNS request to a server including spoofed source IP address, causing server to respond with a larger packet size than the original request and overwhelming the targeted system.

h. NTP Amplification: This attack involves sending a request to a vulnerable NTP server including spoofed source IP address, causing server to respond with a larger packet size than the original request and overwhelming the targeted system.

DoS attacks can have serious consequences, such as disrupting critical services or causing financial losses for businesses. Organizations can put in place several defenses against DoS attacks, including firewalls, IDPS, and CDNs that guard against traffic overload. Additionally, organizations can use traffic filtering and monitoring tools to detect and block suspicious traffic patterns, and regularly update their software and systems to address known vulnerabilities.

DoS Goldeneye: GoldenEye is a DoS tool that was developed in Python and is often used by hackers to launch DoS attacks against web servers. It is designed to flood a target server with HTTP GET and POST requests, which can overload the server and cause it to crash or become unresponsive. The GoldenEye tool uses several techniques to make it more effective at causing damage to a target server. For example, it can randomly generate User-Agents and referrer headers to mimic the behavior of legitimate users, making it harder for the target server to distinguish between genuine requests and malicious traffic. A single TCP connection can transmit a larger file size by using the Keep-Alive method. Because of this, it disables HTTP Cache Control with the 'No Cache' option. System resources are depleted quickly when either of these two features is activated.

To protect against GoldenEye and other types of DoS attacks, website administrators can take steps such as applying firewalls, load balancers, and various security approaches which can identify and prevent suspicious traffic. Additionally, using a Content Delivery Network (CDN) can help to distribute the load of incoming requests, reducing the risk of a single server being overwhelmed [125].

FTP-Patator: Python-based FTP-Patator is a type of cyber-attack that targets File Transfer Protocol (FTP) servers using automated brute-force techniques. The attack is carried out using a tool called FTP-Patator, which is designed to test thousands of usernames and passwords in very less time to gain unauthorized access to the FTP server. The FTP-Patator tool works by sending login attempts to the target FTP server using database of pre-defined usernames and passwords. The tool can also generate custom password with different set of character, numbers, and alphabet combination along with common password patterns or word combinations. If the tool is successful in finding a valid username and password combination form the database of password or self-created password list, attacker can access FTP server and potentially compromise the system. To protect against FTP-Patator, it is recommended to use a strong password along with two-factor authentication, IP-based access controls, and intrusion detection systems. It is also recommended to monitor FTP server logs for suspicious activity and to regularly update and patch the server software to address any known vulnerabilities [126].

SSH-Patator: The Patator, a multithreaded Python program and SSH-Patator is a type of brute-force attack that targets systems using the SSH protocol. The attack involves an automated tool called "Patator," which is used to launch a massive number of logins attempts with combinations of username and password to take advantage of unauthorized access to the target system. SSH protocol is a widely used method for remotely accessing and managing systems, especially in the context of servers and cloud infrastructure. SSH relies on secure cryptographic keys and encrypted communication to prevent unauthorized access. However, if an attacker can guess the correct username and password combination, they can gain access to targer system. The SSH-Patator attack is designed to automate this guessing process, typically using a large list of common usernames and passwords, or even using dictionary attacks to guess potential passwords based on words found in the system or user data. The tool uses target's IP address along with list of username and password combinations to test against the SSH server.

Aim of SSH-Patator attack is to gain access to SSH enabled system, which can then be used to execute further attacks, install malware, or steal sensitive data. To prevent such attacks, it is important to use strong passwords, disable password authentication for SSH, limit SSH access to authorized users, and monitor SSH logs for suspicious activity.

Slowloris DoS: An attacker overwhelms the target server by keeping many connections open simultaneously. A huge number of partial HTTP requests are sent by attacker to destination server and keep on hold long as possible, without completing the request. This causes the server's resources to be tied up, and it cannot accept any new connections, eventually causing the server to become unavailable. Slowloris DoS is called "slow" because the attack is carried out slowly and steadily, rather than in a sudden and quick burst. The attacker sends requests slowly, and each request is incomplete, so the server cannot close the connection. By doing this, the attacker can keep many connections open simultaneously with little bandwidth, thereby overwhelming the server's capacity to handle incoming connections. Slowloris can be difficult to detect because it uses legitimate HTTP requests and appears to be normal traffic. It is effective against servers with limited resources or configurations that allow many connections to be open simultaneously. Slowloris attacks are prevented by restricting number of connections request per IP address, and by configuring the server to time out incomplete requests after a certain period.

DoS SlowHTTPTest: It is an application layer DoS attack aiming to keep connection open as long as possible by slowly and exhausting server resources by sending HTTP requests [127]. The following are the steps involved in a Slow HTTP Test attack:

a. Target selection: The attacker identifies the target server that they want to attack.

b. Initial connection: HTTP request is sent to creates connection with targeted server. It's request typically contains large headers size in attempt to take more server processing time.

c. Slow data transfer: Once connected, the attacker starts sending the HTTP request data very slowly. Attacker sends small chunks of data at a time, with long delays between each chunk. This makes it difficult for the server to process the request efficiently and can cause the server to keep the connection open for a long time.

d. Connection keep-alive: The attacker sends a "keep-alive" header with the request, which instructs the server to remain open its connection for long time. This allows the attacker to keep the connection open and continue sending data slowly.

e. Exhaustion: The attacker continues to send slow HTTP requests with keep-alive headers for exhausting server's resources so that it cannot handle new connection request. This results in the server becoming unavailable to legitimate users.

Slow HTTP Test attacks can be prevented or mitigated by implementing security measures such as rate limiting, load balancing, and using tools such as firewalls and IDS/IPS.

Botnet: A network of internet-connected devices injected with malicious software, called "bots" or "zombies." These infected devices, which can include computers, servers, smartphones, and Internet of Things (IoT) devices, managed by single server. The operators of a botnet use the compromised devices to perform several malicious activities, like "DDoS" attacks, "spamming", "phishing", "click fraud", and "data theft". They can also be used to spread malware, such as ransomware, across the internet. Botnets are typically created through malware infections, which can occur through a variety of methods such as phishing emails, software vulnerabilities, and social engineering. Once infected, the device becomes a part of the botnet and can be controlled remotely by the botnet operator. Botnets can be difficult to detect and take down due to their distributed nature and the use of encryption and other techniques to evade detection.

Ares [128], a Python-based attack tool, is used to conduct a botnet attack. "Ares" is the name of a specific botnet that was active between 2008 and 2016. It was a peer-to-peer botnet that primarily spread through file sharing networks such as Ares Galaxy, hence its name. The Ares botnet was primarily used for unethical purpose like distributed denial-of-service (DDoS) attacks, spamming, and stealing confidential information such as usernames and passwords. It was also capable of downloading and executing additional malware on infected machines. The Ares botnet was eventually taken down in a joint operation between law enforcement agencies and cybersecurity researchers. However, botnets continue to be a significant threat to cybersecurity and can cause significant harm to individuals, organizations, and even entire countries.

Web-based Attack: A web-based attack [129] targets a website or web application, exploiting vulnerabilities in the software and infrastructure that underpins the site. Web-based attacks can be carried out using a range of techniques, including:

Cross-site scripting (XSS): Attackers inject malicious code inside web page accessed by users, potentially stealing sensitive information or executing arbitrary commands on the user's computer.

a. SQL injection: Inserting malicious code inside web form, which can then be used to manipulate a website's database and gain unauthorized access to sensitive data.

b. Cross-site request forgery (CSRF): Attackers tricks user to perform any activity on webpage, like clicking any link or submitting a form, without knowledge of user.

c. Distributed denial-of-service (DDoS): It involve overwhelming a website using data packets, causing it to become unavailable to users.

d. Man-in-the-middle (MITM) attacks: This involves eavesdropping user communications, potentially stealing login credentials or other sensitive information.

Web-based attacks can have serious consequences, including data theft, financial losses, and reputational damage. To protect against web-based attacks, website owners and developers should ensure that their software is up to date, implement strong authentication mechanisms, and use encryption to protect sensitive data. Users should also be vigilant when browsing the web, avoiding suspicious links and keeping their software up to date.

Infiltration: An infiltration [130] attack has goal to identify security flaws that could be misused by a real attacker and getting recommendations for further improving security. In an infiltration attack, a team of ethical hackers or security professionals attempts to take control of system or network using a variety of techniques, including social engineering, malware, password cracking, and network scanning. The attackers then attempt to exploit any vulnerabilities they find to access entry to crucial data or resources. Infiltration attacks are conducted by numerous methodologies, such as "white box", "gray box", and "black box" testing. Infiltration is an important part of a comprehensive security testing program, allowing organizations to identify and remediate vulnerabilities before they can be exploited by attackers. However, it is important to conduct infiltration attacks only with proper authorization and with appropriate safeguards in place to avoid causing unintended harm to the system or network being tested.

Table 6. Comparative analysis of various attack methodology/detection techniques

Attack Type	**Description**	**Attack methodology/detection techniques**	**Target**
Bot	Malicious software designed to perform automated tasks	Infiltrates systems, often part of a botnet, used for various purposes such as DDoS attacks, data theft, or spamming	Various

DDoS	Distributed Denial of Service	Overwhelms a target system or network with a flood of traffic from multiple sources, causing a service outage	Websites, Networks
DoS GoldenEye	Denial of Service attack using the GoldenEye tool	Employs HTTP and HTTPS flooding to overwhelm the target, disrupting services	Websites, Servers
DoS Hulk	Denial of Service attack using the Hulk tool	Utilizes a large number of HTTP GET/POST requests to exhaust server resources, leading to service unavailability	Web Servers
DoS Slowhttptest	Denial of Service attack testing tool for slow HTTP	Exploits slow POST requests to consume server resources gradually, leading to a potential service outage	Web Servers
DoS Slowloris	Denial of Service attack using the Slowloris tool	Keeps multiple connections to the target web server open, preventing them from serving other legitimate requests	Web Servers
FTP-Patator	Brute-force attack targeting FTP servers	Attempts to gain unauthorized access by systematically trying different username and password combinations	FTP Servers
Heartbleed	OpenSSL vulnerability allowing unauthorized access	Exploits a vulnerability in the OpenSSL cryptographic software library, potentially leaking sensitive data from servers	Servers
Infiltration	Unauthorized access or penetration into a system	Involves gaining entry into a secure system or network with the intent of extracting or manipulating data	Networks, Systems
PortScan	Systematic probing of a network for open ports	Scans for open ports on a target system to identify potential vulnerabilities or entry points	Networks
SSH-Patator	Brute-force attack targeting SSH servers	Tries multiple username and password combinations to gain unauthorized access to secure shell (SSH) servers	SSH Servers
Web Attack	Various attacks targeting web applications	Includes SQL injection, cross-site scripting (XSS), and other methods to exploit vulnerabilities in web applications	Websites, Servers
Back	Unauthorized access to a system using a Trojan	Involves compromising a system and creating a backdoor for future access	Systems
Land	Spoofed packet causing a system to reply to itself	Exploits a vulnerability where the target system responds to a maliciously crafted packet	Systems
Neptune	Denial of Service attack using the Neptune tool	Floods a target network with TCP packets, causing service unavailability	Networks
Pod	Denial of Service attack using the Pod tool	Overwhelms a target system with a high volume of traffic, leading to service disruption	Systems
Smurf	ICMP echo request flood with a spoofed source	Amplifies a single ICMP echo request into a flood, causing network congestion and service disruption	Networks
Teardrop	Fragmented packet attack causing system crashes	Sends overlapping, fragmented packets to the target, causing the system to crash	Systems
Satan	Network vulnerability scanner and exploit tool	Scans networks for vulnerabilities and exploits them to gain unauthorized access	Networks, Systems
Ipsweep	Network scanning for active IP addresses	Probes a network to identify active IP addresses, mapping the network structure	Networks

Nmap	Network scanning and discovery tool	Scans a target network to discover hosts, open ports, and services, aiding in penetration testing	Networks
Portsweep	Scanning for open ports on multiple systems	Identifies open ports on multiple systems within a network, potentially indicating vulnerabilities	Networks
Guess_passwd	Brute-force attack targeting password guessing	Repeatedly attempts different password combinations to gain unauthorized access	User Accounts
ftp_write	Unauthorized write access to FTP server	Exploits vulnerabilities to gain unauthorized write access to an FTP server	FTP Servers
Imap	Unauthorized access to an IMAP server	Gains unauthorized access to an Internet Message Access Protocol (IMAP) server	Email Systems
Phf	Exploitation of the "phf" CGI vulnerability	Exploits a vulnerability in the "phf" CGI script to gain unauthorized access	Web Servers
Multihop	Unauthorized use of multiple hosts for attacks	Uses multiple hosts to launch attacks, making it challenging to trace the origin	Networks
Warezmaster	Distribution of pirated software and files	Involves the distribution of copyrighted or unauthorized software and files	File Sharing
Warezclient	Downloading pirated software and files	Involves downloading copyrighted or unauthorized software and files	Individuals, Systems
Spy	Unauthorized monitoring or espionage	Involves spying on user activities, capturing sensitive information	Individuals, Systems
Buffer_overflow	Exploiting buffer overflow vulnerabilities	Overflows buffer memory to execute malicious code, potentially leading to system compromise	Systems
Load_module	Unauthorized loading of kernel modules	Attempts to load unauthorized kernel modules, potentially leading to system compromise	Systems
Perl	Exploitation of Perl interpreter vulnerabilities	Exploits vulnerabilities in the Perl interpreter, allowing execution of malicious code	Systems
Rootkit	Stealthy software for unauthorized access	Installs malicious software to gain unauthorized access and maintain control over a system	Systems

2.8 Machine Learning

Machine learning is a powerful tool for detecting and mitigating network attacks. Here are some ways in which machine learning is helpful in the research of network attacks:

a. Automated detection: This ML approach automate identification of network attacks by analyzing network traffic in real-time. The algorithms can train large datasets of both benign and malicious traffic to learn the patterns of different attack categories. Once trained, the algorithms can identify and flag any suspicious network activity that deviates from these patterns, enabling quick detection and response to potential threats.

b. Improved accuracy: Machine learning algorithms can analyze vast amounts of network traffic data and detect subtle patterns that may be difficult for humans to detect. This can lead to more accurate detection of network attacks and fewer false positives, which can save time and resources.

c. Speed and scalability: ML algorithms can synthesize large volumes of network traffic data effectively and quickly, that make possible to evaluate vast real-time data. This scalability allows network administrators to monitor and respond to network attacks quickly, reducing the potential impact of these attacks.

d. Continuous learning: It can adapt to new network threats and adjust their detection patterns as new threats emerge. This continuous learning capability makes machine learning-based detection systems more robust and adaptable to evolving threat landscapes.

e. Predictive analysis: It can predict future network attacks associated with historical data and patterns. This can help network administrators prepare for potential threats and implement proactive measures to prevent attacks from occurring.

Machine learning approaches has its own strengths and weaknesses. Some common ML approach are listed below:

Supervised learning: This type of approach uses labeled dataset for training a model, where the desired output is already known. The model learns to relate input variables to validated output variable, allowing to make predictions on new and previously uncovered data. This technique is applied for classification and regression problems. Each dataset flow must be accurately identified and labeled in training data with a description of its characteristics (such as normal or malicious). At this point, the algorithm's performance is evaluated using these tags, which are then compared to what it predicted. The method's performance is excellent. When using an external service (such as manual tagging) for labeling, the cost of supervised learning is high. Decision Trees, K-Nearest Neighbors, and Random Forests are a few examples of this type of algorithm.

Examples are:

a. Classification: Predicting whether an email is spam or not spam

b. Regression: Predicting the price of a house based on its features

Unsupervised learning: This method trains a model on an unlabeled dataset with unknown output. The model detects data patterns and outliers. Clustering and anomaly detection are major area for conducting unsupervised learning. Methods that do not use labels are known as unsupervised learning. Algorithms use various properties to categorize the data into groups and look for correlations between them. Accuracy detection and relationship learning are just two of the many areas where it has been used extensively. Labeling, on the other hand, requires specialized expertise that is expensive to outsource. Examples are:

a. Clustering: Grouping customers into segments based on their purchasing habits
b. Anomaly Detection: Identifying fraudulent credit card transactions based on patterns in the data

Semi-supervised learning: It includes both labeled and unlabeled data. It learns from labeled data and apply this knowledge for predicting on the unlabeled data and used when labeled data is scarce or expensive to obtain. Unsupervised and supervised learning methods are combined to create semi-supervised learning: method. In most cases, only small percentage of data are labeled. This approach combines advantages of supervised and unsupervised learning, resulting in a high level of performance at a lower cost.

a. Text classification: Labeling news articles like politics or sports containing only few labeled and large unlabeled dataset
b. Medical diagnosis: Identifying diseases in medical images containing only few labeled and large unlabeled dataset

Reinforcement learning: It involves decision making consideration by feedback from its surroundings. Reinforcement learning is commonly used in robotics and gaming applications.

a. Robotics: Teaching a robot to navigate a maze by rewarding it for taking the correct path and penalizing it for taking the wrong path
b. Gaming: Training a computer to play chess by rewarding it for making good moves and penalizing it for making bad moves

Deep learning: It requires training the model on neural networks, which are complex mathematical models. Deep learning has proficiency of learning complex patterns and associations in data, making it well-suited for image and speech recognition applications.

a. Image recognition: Classifying images of animals, objects, or people based on their features

b. Speech recognition: Transcribing spoken words into written text

Chapter-3

3. Hypothesis, objectives, and methods

3.1 Hypothesis

To analyze, how a better and efficient system can be formed that secures the network from attacks, below are the hypotheses that lead to the firm statement of how current research work proceeded:

Several authors have proposed various methods to detect attack at DLL, network, transport and application layer along with its prevention techniques. Attack analysis at the physical layer involves attacks like eavesdropping and jamming. While conducting cross layer analysis it is found that the physical layer does not have any much to do with software part as intruders generally capture packet by tapping or intercepting signals or do the jamming for interrupting service as discussed in section 2.1.

Attack analysis at the data link layer is crucial for network security, as it helps ensure reliable data transport between devices. However, it is vulnerable to various security attacks that compromise network integrity, confidentiality, and resource availability. Research on data link layer security is limited compared to other cross-layer aspects. However, there is no method for tools like Snort to detect hidden terminal attacks, and further research is needed to determine its applicability. Tools like Snort typically performs well at network, transport, and application layers, but cannot detect all types of attack as discussed in section 2.2.

Most of the attacks happened inside or outside of the network at network, transport, and application layer only and can be identified based on signature or anomalies. Attack at these three layers can be easily detected if the signature database is properly configured but it will be inefficient in case of zero-day attack or novel attack. Machine learning (ML) can be used to detect anomalies, clustering, classification, regression, and rule extraction. However, labelled data is challenging to train due to various factors, such as lack of publicly available datasets and access to comprehensive data on cyberattack frequency and impact. It is assumed that, integrating cross layer network attack dataset together in into existing machine learning approaches will enable more robust fault and security management capabilities. Also, it is needed to improve the utilization of feature grouping by applying existing clustering algorithms for efficient detection of known/unknown attacks along with the need for unbiased data to train in ML. These issues are discussed in section 2.3.

3.2 Objectives of this work

a. To improve the utilization of feature grouping and applying existing clustering algorithms for detecting known/unknown attacks.

In our increasingly connected society, network attacks are a major challenge. Recent studies have used traditional machine learning to find network attacks. They did this by looking at the patterns of how networks interact and training a classification model. Most of the time, these models use large, labeled datasets. However, due to the speed and unpredictability of cyber-attacks, this labeling is unlikely to happen in real time. Many researchers have suggested using the "feature grouping" method to find new and unknown attacks by transferring information about known attacks. The feature grouping approach can find the different kinds of attacks and learn an optimized representation that is not affected by changes in how the attacks behave.

By feature grouping, we can automatically find the link between the new attack and the known attack. This grouping can be used to simulate situations in which different attack styles or subtypes from the training set are included in the testing dataset. Traditional classification models like decision trees, random forests, KNN, and other approaches to grouping features can be used for this. In the present work, nine ML approach is used and applied on 77 features and reducing up to 14 to increase performance and accuracy in detecting attack.

b. To design the fault and security management framework for future networks with existing machine learning approaches.

To achieve the objective, a selective ML algorithm are employed along with feature grouping, performance analysis, training and testing, and identification of known or unknown attacks in the future. Existing frameworks for intrusion detection systems (IDS) typically consider all information features to differentiate between interruption and misuse patterns. However, some of these features may be redundant or contribute minimally to network attack identification. Hence, the aim of this research was to determine the crucial feature groups for developing an effective and easily programmable IDS.

To address this, we propose the FSS-PART model, which utilizes Adaptive Resonance Theory (ART) to identify similarities and patterns in network traffic. By leveraging machine learning algorithms, predictive models capable of detecting network attacks can be automatically generated. In this research, an attempt is done to implement an application of the K-means and KNN methods within a web-based framework to identify network attacks based on protocol type TCP, UDP and ICMP.

To facilitate the identification of new network attacks in a specific domain, we included source and target parameters that accurately reflect distinct or similar network environments. Additionally, we ensure that different attacks are recorded at various times and in separate instances. The approach for detecting novel network threats involves the following steps:

i. Extraction of features from collected data
ii. Classification of the extracted data
iii. Representation of the data in graph for attack analysis

By following this framework, it aimed to streamline the identification of new network threats as well as improving the overall effectiveness of intrusion detection systems.

c. Comparative analysis of this method with existing methods using standards datasets and parameters.

For network model or intrusion detection systems (IDS) to figure out the best functions, selecting or removing functions is a complex task. Calculations based on filters, especially for information gain (IG), correlation (CR), and relief F (ReF). The system first gets the feature subset of each classifier. Depending full features and the strategy for combining other subsets, and after many tests, it was found that reducing the number of features improves performance, while adding more features outperformed but makes it easier to identify attacks. InfoGain, Coorelation and ReliefF were analyzed, and the result is included in Table 25.

3.3 Approach for achievement of the objectives

Table 7. Methodology/ Tools/ Instruments used

Objective	Analysis	Processes/ software used
To improve the utilization of feature grouping and applying existing clustering algorithms for detecting known/unknown attacks.	Dataset analysis and its features along with grouping of features for effective detection of attack	KDD'99 datasets, python, Weka
To design the fault and security management framework for future networks with existing machine learning approaches.	Feature grouping, analysing performance, training, and testing using selective machine learning algorithm	KDD'99, CICIDS 2017 datasets, python, Weka
Comparative analysis of this method with existing methods using standards datasets and parameters.	Analysing the impact of feature grouping based on number of features taken by earlier approach and impact on the performance.	KDD'99, CICIDS 2017 datasets, python, Weka

In recent years, cyberattacks have become a significant threat to government, military, and industrial networks. These attacks are increasingly complex and diverse, including zero-day attacks and Denial of Service (DoS). Traditional signature-based detection methods often lack to keep up with the evolving nature of cyberattacks. It is crucial to explore new approaches that can identify anomalies, enable network learning and adaptation and detect threats in various network settings.

Machine learning and data mining have been employed to enhance the detection rate of network attacks in networked environments. Supervised data-driven models have shown higher accuracy compared to unsupervised methods, but they require a large number of labeled malicious examples. As attack patterns change, the distribution of functions can also shift, rendering the training models less effective in identifying new attacks.

Obtaining enough identified data for continuously emerging attacks is challenging. Additionally, when new attacks are discovered, incorporating new functions from different network levels becomes necessary. Since these functions have varying variables, it becomes possible to retrain the models. To address these challenges, we propose a method known as "feature grouping" to facilitate the identification of new threats. The "feature grouping approach" is a novel machine learning technique that adapts features in a "target domain" where labeled data is limited by leveraging knowledge attained from an associated "source domain." Present work inspired by the fact that most network attacks are variations of well-known attack families, sharing similar features that align with the feature grouping approach. The data from the source and destination networks are of different time periods within the same network setting. The assumption is that attacks on the source network are previously known and recorded, while attacks on the target network are new and distinct from those in the source network. The feature grouping method is examined in this thesis through parameter adjustments and various sizes of training sets. Furthermore, this method can be applied to other machine learning techniques.

Existing network Intrusion Detection Systems (IDS) primarily rely on signature-based systems, which utilize collections of attack signatures created manually by experts. This process is slow and computing intensive. Machine learning models have been developed to predict and detect network attacks, reducing the reliance on manual analysis by experts. While IDS technology is beneficial, there is a need for improved detection of hidden or complex attack patterns. Building an effective intrusion detection system relies on a robust IDS dataset, which is used for model construction and testing to detect intrusions. Machine learning approaches leverage this data to build predictive models, where incorporating diverse and interesting data enhances the model's capabilities. It is significant to thoroughly evaluate both normal and malicious packets to identify and address vulnerabilities. Testing the system's ability to detect and minimize false alarms is critical. Ensuring low false alarm rates provides crucial information to programs, protocols, and lower-level network entities, enabling them to detect and respond to attacks. Creating and simulating realistic profiles aids in obtaining accurate information, as profiles can be described and executed by individuals, autonomous agents, or random distributions. Current intrusion detection tools often fail to accurately represent network traffic as it occurs in the real world. Collecting intrusion detection data as part of the process for developing and reviewing detection methods for computer network attacks is essential. Such datasets should encompass real-world network situations to ensure their effectiveness. Such a dataset is needed to show how this method for finding network threats works in real life. Study various forms of attacks, and various methods of machine learning, including classification, identification of anomalies and selection mechanisms for building host and network level detection models, the following are the outcome:

- Use of ML approach for understanding attacks at different layers.
- For each attack to be found, there should be a thorough review and consideration for ML method.
- How data is collected, missing data are taken out, and the data is analyzed.
- For each attack, get representative data from a live university network ex. CICIDS 2017, such as network traffic or server logs. Data collection for each attack is well thought out so that the gaps in the available databases can be fixed.
- Considering all attack in the dataset, research done on these collected datasets are correct.
- This work also justifies how ML approach is helpful in building host and network-based detection model.
- Feature selection approach is used to reduce attack detection time under different ML approach.

3.4 Tools selection

3.4.1 Python

A high-level programming language which is used for diverse tasks, such as building websites, analyzing data, creating AI, doing scientific computing, and more. Guido van Rossum made it public for the first time in 1991. Since then, it has become most used computer languages in the world. It is a great language for learning because its code is simple and easy to understand. It has an active community of developers involved in making third-party libraries and tools that make it even more powerful and flexible. Some of the key features of Python include dynamic typing, automatic memory management, object-oriented programming, and support for multiple programming paradigms. Python code can run on several platform such as Windows, macOS, Linux, and more.

3.4.2 Sklearn

Sklearn, better known as scikit-learn, is a renowned open-source machine learning library created on top of "NumPy," "SciPy" and "matplotlib." It offers a convenient and efficient platform for applying a diverse array of machine learning algorithms, together with classification, regression, clustering, and dimensionality reduction. Sklearn offers a wealth of tools for data preprocessing, model selection, and evaluation. Additionally, it offers access to several widely used datasets that serve as benchmarks for testing and comparing different machine learning methods. Within Sklearn, popular methods for instance "linear regression," "logistic regression," "k-nearest neighbors," "decision trees," "random forests," "SVM" and "neural networks" are frequently employed to address various machine learning tasks.

3.4.3 Pandas

An open-source tool for Python which analyzes data and supports other tools for working with organized data, such as tools for loading, cleaning, transforming, and displaying data. Pandas is built on top of NumPy, which is another popular scientific computing tool for Python. It has an interface that is made to make data analysis easier and more intuitive. Pandas adds Series and DataFrame, two new data formats that make it easier to work with tabular data. The Series object is basically a one-dimensional array with a labeled index, while the DataFrame object is a two-dimensional table with named rows and columns. Pandas makes it easy to index, filter, and manipulate these data structures. It also lets us do more complicated things with data, like grouping, pivoting, and merging. Pandas also has a lot of tools for working with missing data, data in a time series, and data that can be put into different groups. It also works with other popular Python tools, like Matplotlib for displaying data and Scikit-learn for teaching computers how to do things.

3.4.4 Matplotlib

The Python tool Matplotlib makes it easier to see how data looks. Matplotlib is a useful Python library for plotting data. It has many tools for making plots, charts, and graphs of high quality, like line plots, scatter plots, bar plots, histograms, and more. NumPy is another popular scientific computing library for Python, and Matplotlib is built on top of it. It works well with other tools for data analysis and machine learning, like Pandas and Scikit-learn. It has an easy-to-use interface for making custom plots and visualizations, and it has a lot of customization choices for fine-tuning how plots look. Matplotlib are used for data visualization jobs, such as exploratory data analysis, making figures that are good enough for publication, and telling stakeholders about the results. It also supports a wide range of output files, such as PNG, PDF, SVG, and more.

NumPy: Scientific computing, math and logic operations were made faster and easier by using this library. It can do a lot of things with arrays, such as indexing, slicing, sorting, and changing multidimensional arrays. NumPy is built on top of optimized C and Fortran tools and is meant to be fast and efficient. It has a lot of math features, such as linear algebra, Fourier transforms, and making random numbers. NumPy also works well with other scientific computer libraries such as Pandas, Scikit-learn, and Matplotlib. One of NumPy's most important parts is its multidimensional array object, which makes it easy to work with big data sets. We can make these arrays from Python lists or tuples, and they can have any number of dimensions. NumPy offers a variety of methods for doing math operations on these arrays, including vectorized operations that can be done on the whole array at once.

3.4.5 Weka

Weka is an open-source software that helps with data mining and machine learning along with giving us a full set of tools for preprocessing data, classifying, regressing, clustering, and finding associations between rules. Weka is written in Java and has both a graphical user interface and a command-line interface for running batch tasks. It also offers a extensive array of machine learning methods, such as "decision trees", "support vector machines", "random forests", "k-nearest neighbors", and "neural networks", as well as tools for feature selection, ensemble learning, and model evaluation. Weka's ease of use is one of its best features. This makes it a good choice for both beginners and pros. It also has a lot of information about how to use it and a big, active community of developers and users who work together to improve the software and help new users.

3.5 Hardware platform selection

The time it takes for a machine learning algorithm to run is one way to judge its performance. It is, however, possible that the execution time will be affected by the computer's performance. In this research work following are the hardware configuration considered.

Operating System: Windows 11

Processor: Intel Core i7-1065G7

Clock Rate: 1.30GHz (8 CPUs), ~1.5GHz

Memory (RAM): 16384MB

Card name: NVIDIA GeForce MX350 with Display Memory: 10064 MB

Card name: Intel Iris Plus Graphics with Display Memory: 8212 MB

Solid State Drive: 1 TB (3400 MB/s reading and 3100 MB/s writing)

* Machine learning Training and Testing part is done on Google Colab with 12 GB RAM and allocated GPU.

3.6 Data collection process

KDD'99 and CICIDS 2017 datasets were analyzed for any improvement by selecting and reducing different features. As these datasets were large and features in these datasets were 41 for KDD'99 and 79 for CICIDS 2017 respectively so selection of important features and detection of attack with accuracy and putting the provision for improving the efficiency of detecting the attack were a challenging task.

The first task in the process was to select the datasets on which feature grouping need to be done. Following is the description regarding the datasets:

Synthetic datasets: To meet condition or specific needs synthetic datasets are generated that satisfy real data. When we design any system for theoretical analysis, we use this dataset, and this dataset can be refined accordingly. It is used for creating different types of test scenarios. Designers utilize datasets to analyze and create realistic profiles for evaluating the effectiveness of methods and techniques. However, it is often challenging to determine how well algorithms perform in real-world scenarios using these datasets.

Benchmark datasets: These datasets are systematic and well-defined. They rectify the strengths and weaknesses of various algorithm. These given below the available benchmark datasets produced using simulated environment or by different attack scenarios:

NSL-KDD dataset: It is most used benchmarked dataset derived from DARPA98.

Several other datasets include KDD'99, DARPA 2000 dataset, DEFCON dataset, CAIDA dataset, LBNL dataset etc.

Benchmark datasets are not effective for real world traffic as they do not represent dataset for simulating real time network. As DARPA dataset does not represent real network traffic as it is produced synthetically.

Real life datasets:

Example of some real-life datasets includes CICIDS 2017, UNIBS dataset, ISCX-UNB dataset, TUIDS dataset etc. In this work KDD'99 and CICIDS 2017 datasets is taken. Different datasets are available for research purpose, and this can be implemented for training and testing purpose for future attack. Most famous among these datasets are KDD'99 & CICIDS2017 dataset. KDD'99 is most well-known benchmark and researched datasets in IDS development. It is a statistically pre-processed datasets available by DARPA since 1999. KDD'99 is mainly used for offline intrusion detection, but can it be used for online datasets. Presently, by comparing the structure of KDD'99 Dataset to that of any simulated attack, it’s clear that KDD'99 model is showing its age. It is not the best choice to train any machine learning algorithm using KDD'99 for any real-life datasets but it can be taken for reference purpose for benchmarking, and this is the reason CICIDS 2017 dataset is considered for further analysis of work.

Chapter-4

4 Implementation

Utilizing machine learning approaches for attack detection has proven to be effective in the domain of network security. However, the success of these models heavily depends on the quality of the data used for training and testing. To ensure the reliability and accuracy of the results, data cleansing plays a vital role in the initial phase.

During the data cleansing process, various techniques are employed to eliminate errors, inconsistencies, and incomplete records from the dataset. This step is crucial as it helps to remove noise and ensure that the subsequent analysis is conducted on reliable and high-quality data. By fixing problems with data integrity early on, the performance and usefulness of the machine learning models can be greatly improved. Following data cleansing, the dataset divides in two distinct categories: training and testing set. The training set train the machine learning models, enabling them to learn and identify patterns within the data. On the other hand, the testing set is used to evaluate the performance of the trained models by assessing their ability to accurately detect and classify attacks.

Feature selection is a crucial aspect of the machine learning process. It involves identifying the most relevant properties from dataset applied by ML algorithms for attack detection. Purpose of feature selection is to optimize performance and efficiency of models by selecting most informative and discriminative features. By carefully choosing these features, the models can focus on the most relevant aspects of the data, improving their ability to accurately identify and differentiate between normal network behavior and potential attacks. Once the feature selection process is complete, the machine learning algorithms are ready to implement with dataset. These algorithms utilize the chosen features and leverage their underlying mathematical and statistical properties to detect and classify attacks. They analyze the patterns, relationships, and characteristics present in the data to identify any anomalies or suspicious activities that may indicate a potential attack.

By following this approach, which involves data cleansing, feature selection and applying machine learning algorithms, attack detection accuracy and effectiveness can be significantly enhanced. The combination of these steps allows for the creation of robust and efficient models that can successfully differentiate between normal network behavior and malicious activities. The goal is to provide network administrators and security analysts with reliable tools to detect and mitigate potential threats in real-time, safeguarding the integrity and security of network systems. The entire procedure is shown in greater detail in Figure 6. Two different datasets are used. In the first approach CICIDS 2017 datasets are used and another using KDD'99.

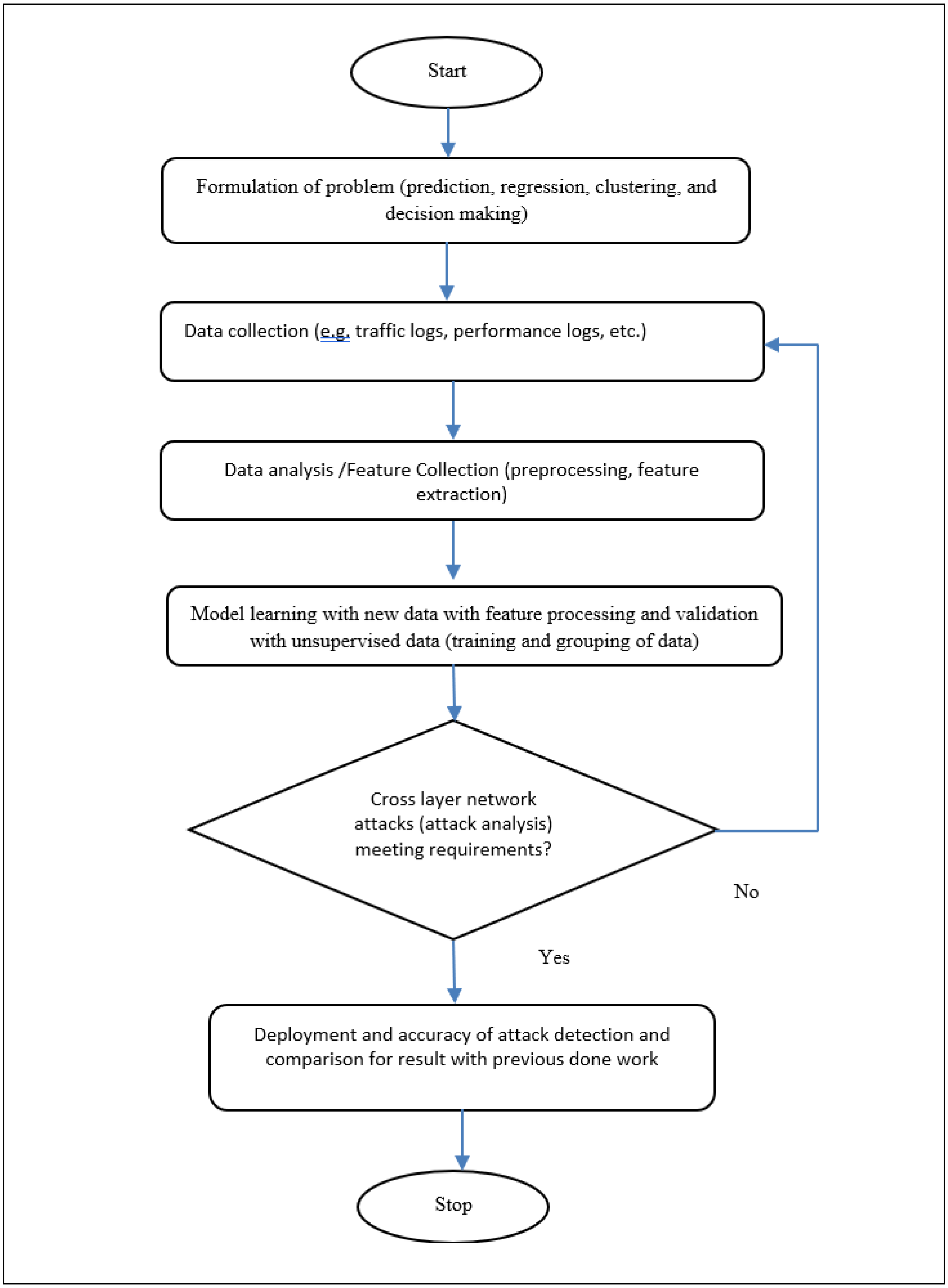

Start
Formulation of problem (prediction, regression, clustering, and decision making)
Data collection (e.g. traffic logs, performance logs, etc.)
Data analysis /Feature Collection (preprocessing, feature extraction)
Model learning with new data with feature processing and validation with unsupervised data (training and grouping of data)
Cross layer network attacks (attack analysis) meeting requirements?
No
Yes
Deployment and accuracy of attack detection and comparison for result with previous done work
Stop

Figure 6. Flow diagram for identification of attack

4.1 Data Cleansing

Preprocessing eliminated redundant or unimportant elements.

Following data preprocessing steps were done:

i. Imports necessary libraries and modules, including pandas, os, sklearn.preprocessing, and time.

ii. Define list of CSV file names and a list of main labels representing column headers.

iii. Write the main labels as the header in each CSV file, based on the given file names.

iv. For each CSV file, read the file line by line, checks if the line starts with a number, replaces any instances of " – " with " - ", and write the line to a temporary CSV file.

v. Read the temporary CSV file using pandas, fills any missing values with 0, and performs the following steps for specific string features:

    Replace 'Infinity' with -1.

    Replace 'NaN' with 0.

   Convert string representations of numbers to their corresponding numerical values.

vi. Identify string features by checking the data type of each column and stores them in the variable string_features.

vii. Initialize a label encoder from sklearn.preprocessing.

viii. Encode the string features using the label encoder, replacing the original values with encoded values.

ix. Drop unnecessary column from the DataFrame.

x. Write the preprocessed DataFrame to the file final CVS file and remove temporary cvs file.

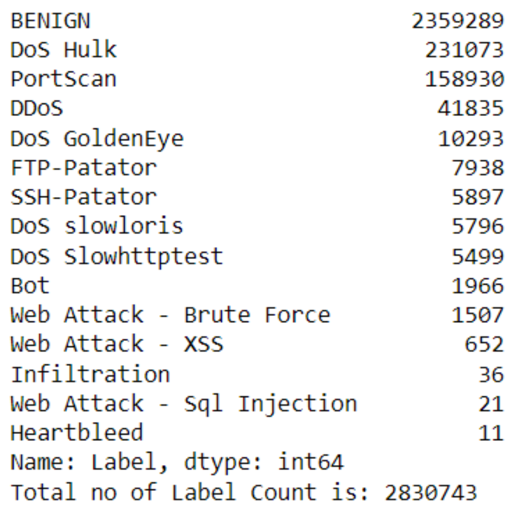
Table 8. Overview of the CICIDS2017 dataset.

Flow Recording Day	**pcap File size**	**Duration**	**CSV File Size**	**Attack Name**	**Flow Count**

Monday	10 GB	All Day	257 MB	No Attack	529918
Tuesday	10 GB	All Day	166 MB	SSH-Patator, FTP-Patator	445909
Wednesday	12 GB	All Day	272 MB	DoS Slowhttptest, DoS Hulk, DoS slowloris, Heartbleed, DoS GoldenEye	692703
Thursday	7.7GB	Morning	87.7 MB	Web Attacks (XSS, Brute Force, Sql Injection)	170366
		Afternoon	103 MB	Infiltration	288602
Friday	8.2GB	Morning	71.8 MB	Bot	192033
		Afternoon	92.7 MB	DDoS	225745
		Afternoon	97.1 MB	PortScan	286467

Before putting the dataset to use, it's possible that some alterations will be required to make it more effective. Considering CICIDS2017 dataset all the incomplete records and errors have been corrected.

Table 9. CICIDS 2017 dataset with number of attacks after correction



```
BENIGN                          2359289
DoS Hulk                         231073
PortScan                         158930
DDoS                              41835
DoS GoldenEye                     10293
FTP-Patator                        7938
SSH-Patator                        5897
DoS slowloris                      5796
DoS Slowhttptest                   5499
Bot                                1966
Web Attack - Brute Force           1507
Web Attack - XSS                    652
Infiltration                         36
Web Attack - Sql Injection           21
Heartbleed                           11
Name: Label, dtype: int64
Total no of Label Count is: 2830743
```

There are 3119345 records in the dataset. Table 9 shows numerical distribution of these records. A closer look at these records reveals that the 288602 record is incorrect or incomplete. After Removing these unnecessary records as the first step in pre-processing total count is 2830743. The provided analysis includes the count of different types of network activities or attacks observed in a dataset. Here is a breakdown of the occurrences:

The majority of the network activity in the dataset is classified as "BENIGN," representing normal network traffic with 2,359,289 occurrences. However, it is worth noting that there are several different types of attacks present in the dataset as well. Among the attacks, the most frequent ones are "DoS Hulk" with 231,073 occurrences, followed by "PortScan" with 158,930 occurrences. These attacks involve overwhelming the target network or scanning it for vulnerabilities, respectively. The "DDoS" (Distributed Denial of Service) attack is also significant, with 41,835 occurrences. This attack involves multiple sources overwhelming a target with traffic to disrupt its normal functioning.

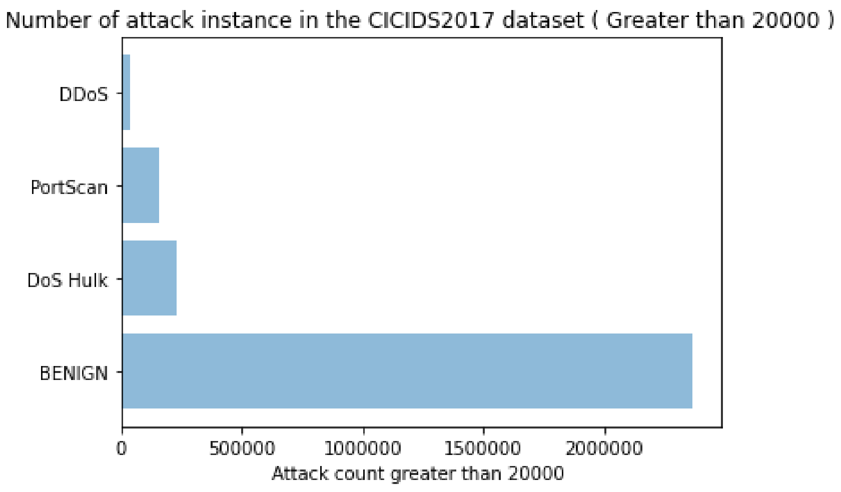


Figure 7. Attack instance in the CICIDS2017 dataset (Greater than 20000)

Less frequent attacks include "Infiltration," "Web Attack - SQL Injection," and "Heartbleed." These attacks involve unauthorized access to networks, exploitation of SQL vulnerabilities in web applications, and exploiting a vulnerability in the OpenSSL encryption library, respectively.

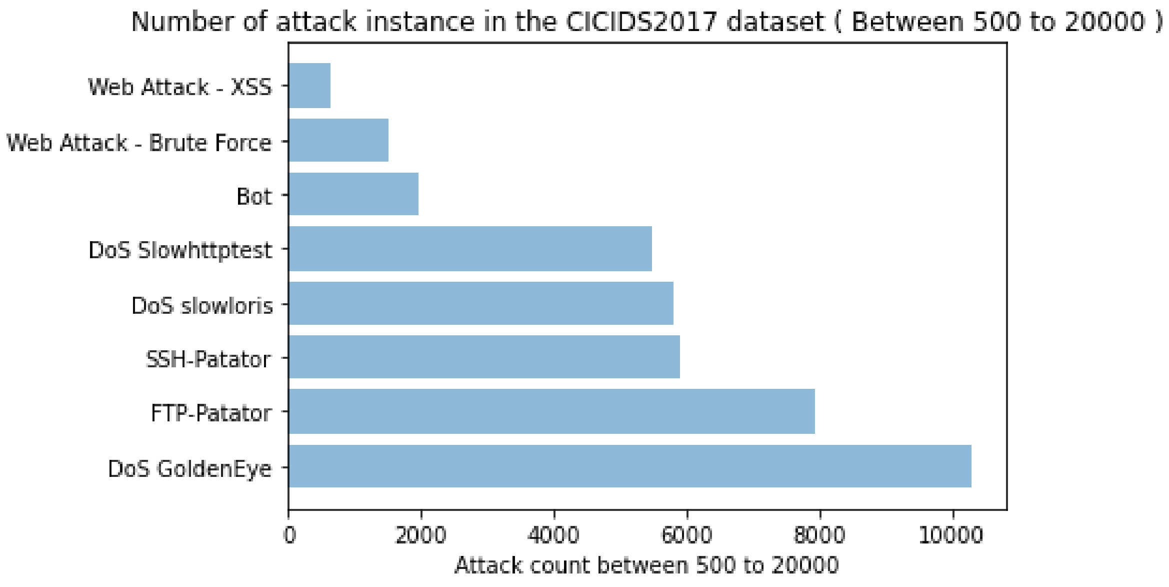


Figure 8. Attack instance in the CICIDS2017 dataset (Between 500 to 20000)

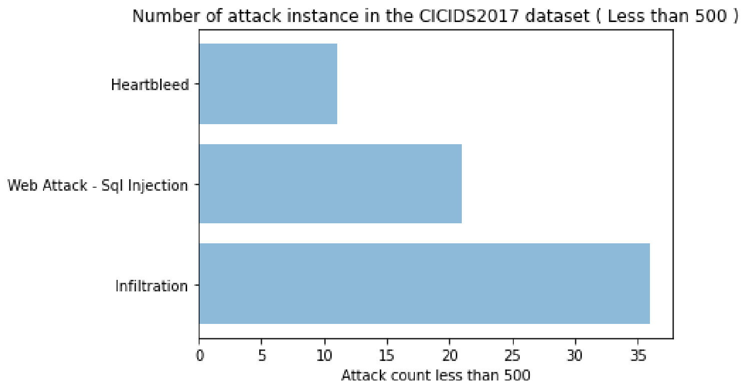


Figure 9. Attack instance in the CICIDS2017 dataset (Less than 500)

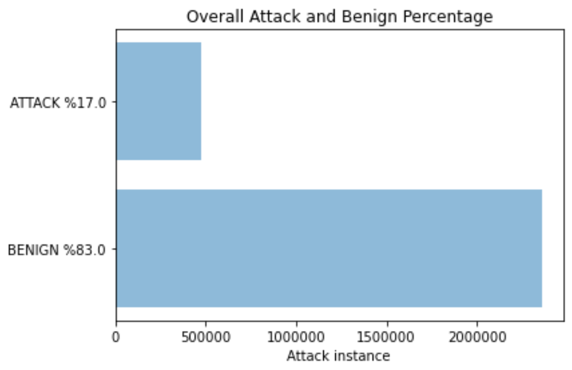


Figure 10. Distribution of attack vs. benign percentage

Understanding the distribution and frequency of these attacks within the dataset can help in developing and enhancing network security measures. It provides insights into the types of threats that networks may face and the areas that require attention to strengthen the overall security posture.

The dataset consists of 86 columns, with a total of 85 columns dedicated to features. However, there is an error in the feature columns where the "Fwd Header Length" attribute is duplicated in both the 41st and 62nd columns. To rectify this error, the repeated column (column 62) should be deleted. To utilize the dataset for machine learning algorithms, certain properties need to be converted into numerical data. These properties include Flow ID, Source IP, Destination IP, Timestamp and External IP. To achieve this, the "LabelEncoder()" class from the Sklearn library can be utilized. By applying the LabelEncoder(), string values within these columns can be transformed into corresponding integer values. The integers will range between 0 and j-1, where j represents total count of unique attributes within the column.

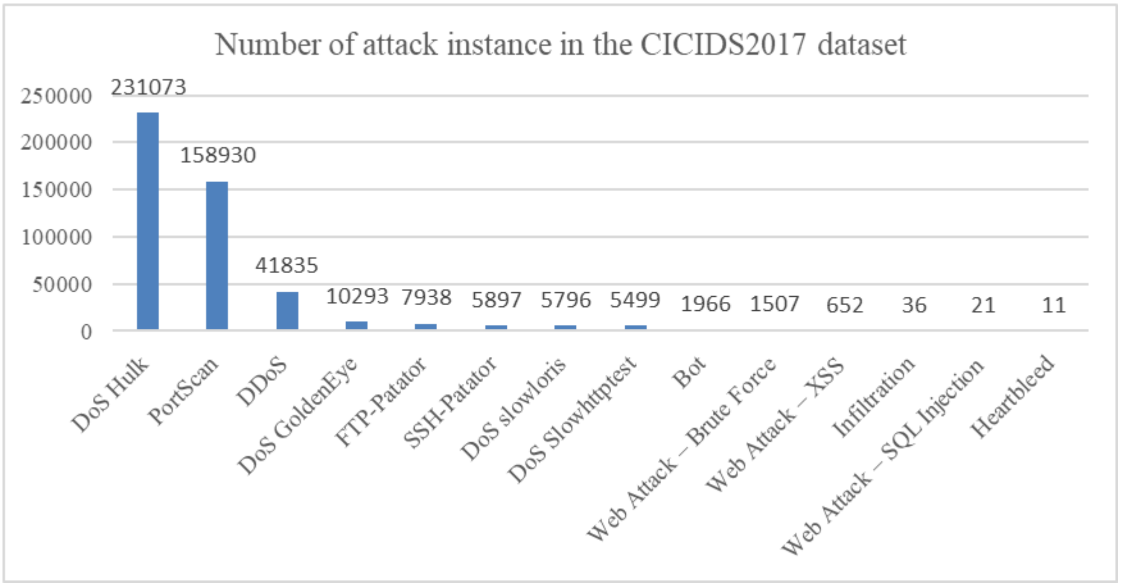


Figure 11. Overall Attack instance in the CICIDS2017 datasets

Even though the "Label" field is a categorical one, it should not be changed. This is because the original categories within the "Label" column are required during the processing stage. Different attack types can take on various forms and approaches, making it necessary to retain the original categories for accurate analysis and classification. By addressing the error in the duplicated column and converting relevant properties into numerical data using the LabelEncoder() class, the dataset will be prepared for machine learning operations. The numerical representation of the data will allow machine learning algorithms to effectively analyze and classify the network activities, enabling the detection and identification of different attack types based on their characteristic patterns and behaviors. Attack instance in the dataset clearly shows only three attacks are dominant compared to remaining 11 attack categories. Further research will require on the same for balance between the attack count and benign data. Individual attack based data extraction from whole dataset for training and testing is another aspect when a model is designed instead of training model with all attack together in the initial phase of machine learning process.

4.2 Creation of Training and Test Data

A significant amount of data is required to train and evaluate the performance of algorithms in algorithm learning process. Testing the algorithm's performance goes beyond the training data and requires additional data specifically for evaluation purposes. The algorithm learns from the training data and then put on its learned knowledge to the test data, allowing for an assessment of its performance based on unseen examples.

In the case of the CICIDS2017 dataset, there is no predefined separation of dedicated training and testing datasets. Instead, it is provided as a single, unpartitioned dataset. To address this, it is necessary to partition the dataset into distinct training and testing sections. This can be achieved using a command from the Sklearn library called "train_test_split." This command allows users to specify the desired sizes or proportions of the training and testing data.

The separation of data into training and testing sets typically follows a general rule of thumb. For instance, common proportions include 20:80, 30:70, 35:65, 40:60 and 50:50 respective percentage ratio for testing and training. These proportions ensure a balance between having enough data for training the algorithm and a sufficient amount for evaluating its performance.

When creating these data groups using the train_test_split command, the selection process is randomized to avoid any biases. This procedure is identified as cross-validation, which helps in obtaining reliable and unbiased performance results. To ensure the adequacy of results gained during running the model, training and test data creation is repeated multiple times. In this case, the process is repeated 10 times consecutively. By averaging the results of these repeated operations, a more accurate and robust assessment of the algorithm's performance can be obtained. The application of cross-validation and the averaging of results across multiple iterations provide a more comprehensive evaluation of the machine learning algorithm's capabilities. It helps mitigate the impact of randomness and variability in the dataset, ensuring that the results are reliable and representative of the algorithm's performance across different data splits.

4.3 Feature Selection

This process encompasses computing the importance of features in the training dataset, categorized based on different "Benign" and "Attack" ratios. Three ratios are considered:

50:50, 80:20, and All. Additionally, this process is performed for the complete dataset. In this research work, attack types considered include "Bot", "DDoS", "DoS GoldenEye", "DoS Hulk", "DoS Slowhttptest", "DoS slowloris", "FTP-Patator", "Heartbleed", "Infiltration", "PortScan", "SSH-Patator" and "Web Attack". For the Web Attack category, Brute Force, XSS, and SQL Injection are merged into a single file due to their minimal occurrence compared to other attacks.

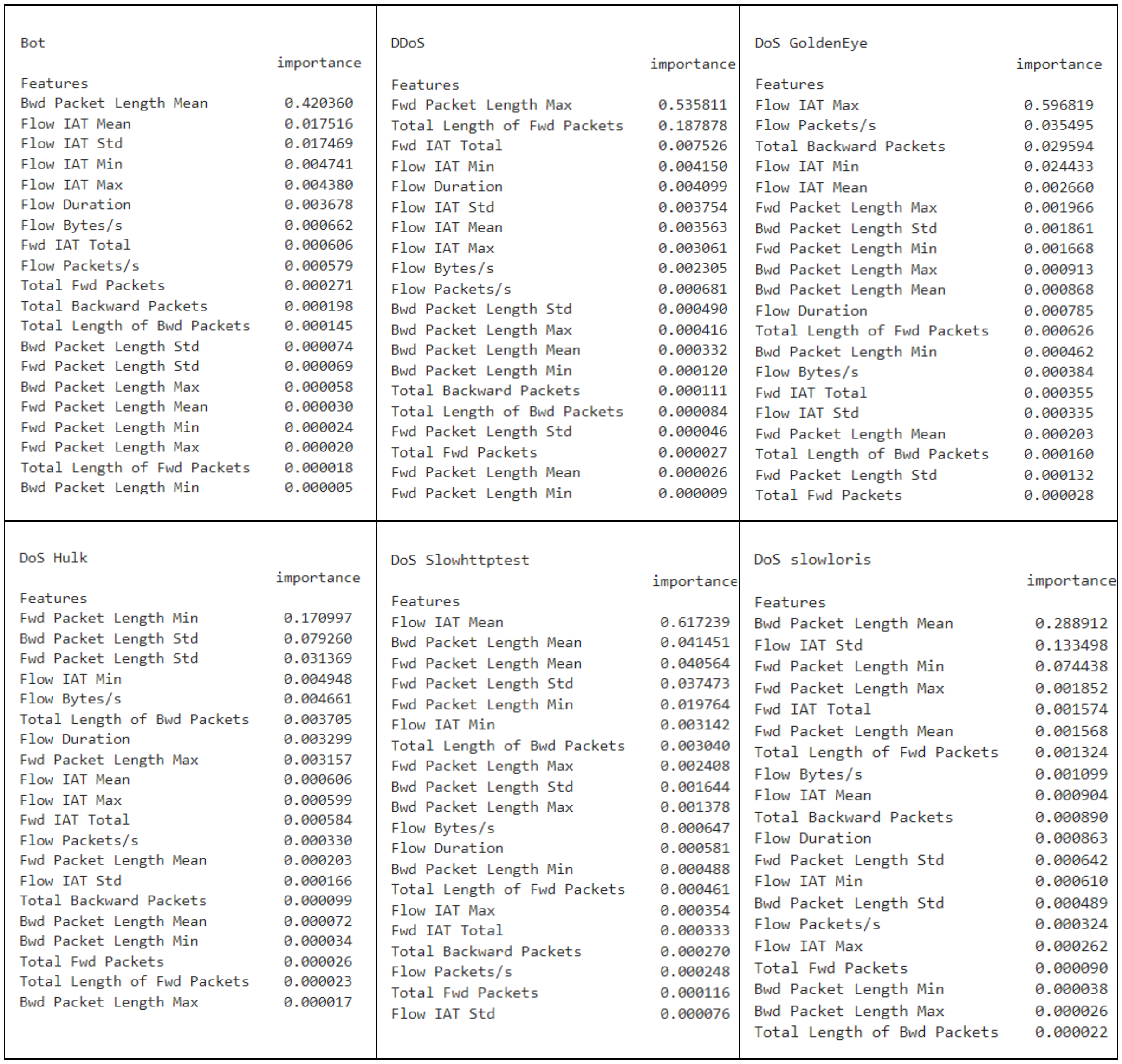
The feature selection process utilizes the Random Forest Regressor algorithm and set the value 250 for the n_estimators parameter, which specifies the number of trees to be built. By combining the predictions of multiple trees, the algorithm determines the maximum voting or averages of predictions, with higher values leading to better performance. To perform this calculation, it is necessary to separate each type of attack from the rest of the attacks, ensuring that all data streams categorized as "Benign" and "Attack" are included.

To calculate the importance weights for features, the Sklearn library's Random Forest Regressor class is used. The weight assigned to a feature for decision tree is determined through sum of weights of all parameters. By evaluating the score of a specific feature against the total tree score, it becomes possible to determine the feature's significance within the decision tree. Although there are 85 properties available, only 8 of them are considered in calculating the importance: "Flow ID", "source IP", "source port", "destination IP", "destination port", "protocol, timestamp", and "external IP". Despite all prevalence of these features in classical approaches, attackers may attempt to bypass operating system restrictions or evade detection by using unfamiliar ports or generating/faking IP addresses. Furthermore, dynamic port usage is common, and multiple applications can be sent over the same port simultaneously. Thus, relying solely on port numbers can be misleading.

In such scenarios, effectiveness will be enhanced by eliminating ambiguous features like IP address, port number and timestamp. As a substitute, the focus is shifted towards utilizing more common and constant attributes that describe the attack. The profile and characteristics of data provide valuable insights into whether it represents an attack or not, enabling better detection and classification.

I. Feature importance when file stream is kept in ratio of 50:50 for "Benign" and "Attack", following results were achieved:

Table 10. Weightage based on the 50:50 ratio of "Benign" vs. "Attack"



```
Bot
                                  importance
Features
Bwd Packet Length Mean              0.420360
Flow IAT Mean                       0.017516
Flow IAT Std                        0.017469
Flow IAT Min                        0.004741
Flow IAT Max                        0.004380
Flow Duration                       0.003678
Flow Bytes/s                        0.000662
Fwd IAT Total                       0.000606
Flow Packets/s                      0.000579
Total Fwd Packets                   0.000271
Total Backward Packets              0.000198
Total Length of Bwd Packets         0.000145
Bwd Packet Length Std               0.000074
Fwd Packet Length Std               0.000069
Bwd Packet Length Max               0.000058
Fwd Packet Length Mean              0.000030
Fwd Packet Length Min               0.000024
Fwd Packet Length Max               0.000020
Total Length of Fwd Packets         0.000018
Bwd Packet Length Min               0.000005
```

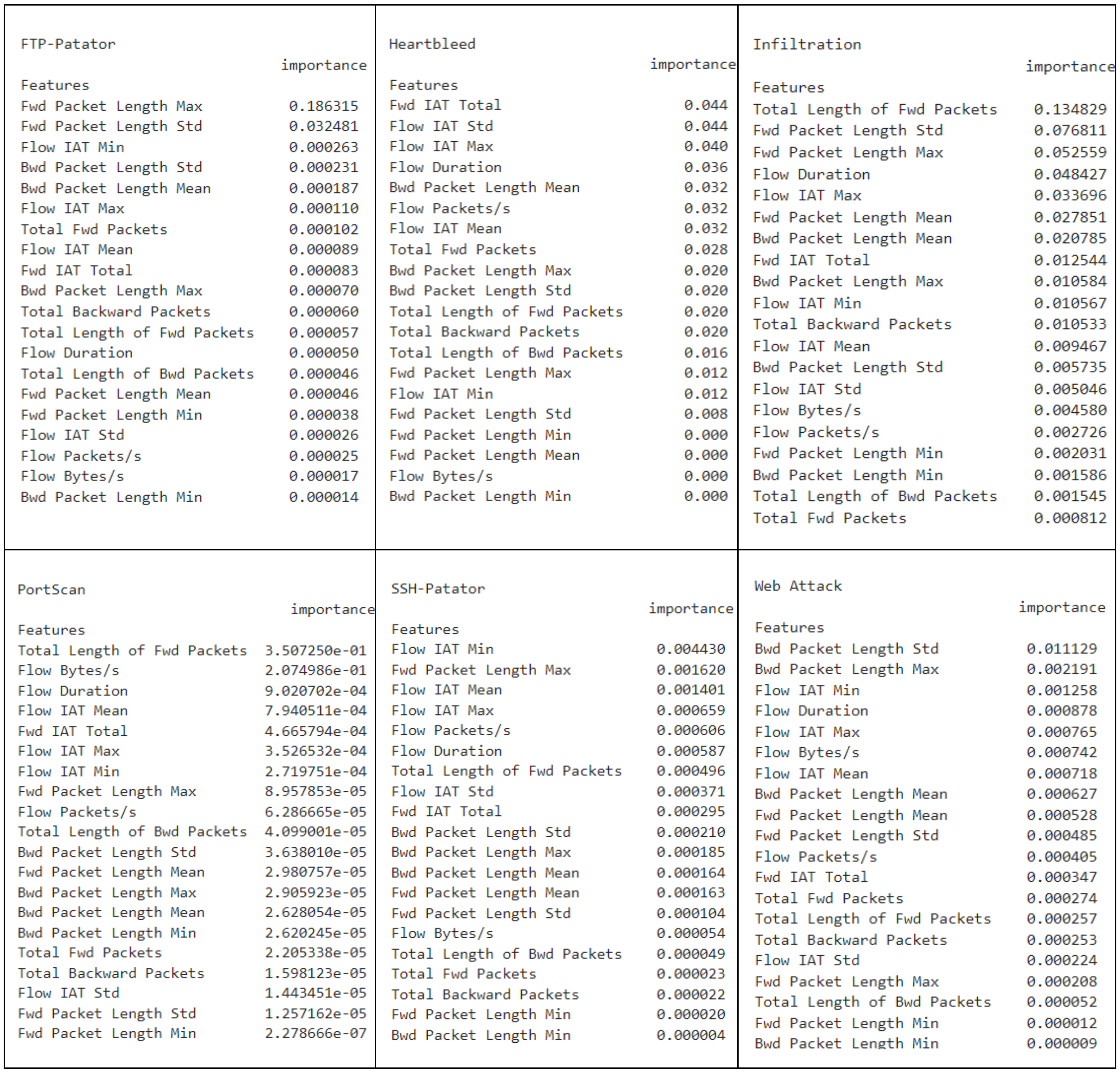
```
DDoS
                                  importance
Features
Fwd Packet Length Max               0.535811
Total Length of Fwd Packets         0.187878
Fwd IAT Total                       0.007526
Flow IAT Min                        0.004150
Flow Duration                       0.004099
Flow IAT Std                        0.003754
Flow IAT Mean                       0.003563
Flow IAT Max                        0.003061
Flow Bytes/s                        0.002305
Flow Packets/s                      0.000681
Bwd Packet Length Std               0.000490
Bwd Packet Length Max               0.000416
Bwd Packet Length Mean              0.000332
Bwd Packet Length Min               0.000120
Total Backward Packets              0.000111
Total Length of Bwd Packets         0.000084
Fwd Packet Length Std               0.000046
Total Fwd Packets                   0.000027
Fwd Packet Length Mean              0.000026
Fwd Packet Length Min               0.000009
```

```
DoS GoldenEye
                                  importance
Features
Flow IAT Max                        0.596819
Flow Packets/s                      0.035495
Total Backward Packets              0.029594
Flow IAT Min                        0.024433
Flow IAT Mean                       0.002660
Fwd Packet Length Max               0.001966
Bwd Packet Length Std               0.001861
Fwd Packet Length Min               0.001668
Bwd Packet Length Max               0.000913
Bwd Packet Length Mean              0.000868
Flow Duration                       0.000785
Total Length of Fwd Packets         0.000626
Bwd Packet Length Min               0.000462
Flow Bytes/s                        0.000384
Fwd IAT Total                       0.000355
Flow IAT Std                        0.000335
Fwd Packet Length Mean              0.000203
Total Length of Bwd Packets         0.000160
Fwd Packet Length Std               0.000132
Total Fwd Packets                   0.000028
```

```
DoS Hulk
                                  importance
Features
Fwd Packet Length Min               0.170997
Bwd Packet Length Std               0.079260
Fwd Packet Length Std               0.031369
Flow IAT Min                        0.004948
Flow Bytes/s                        0.004661
Total Length of Bwd Packets         0.003705
Flow Duration                       0.003299
Fwd Packet Length Max               0.003157
Flow IAT Mean                       0.000606
Flow IAT Max                        0.000599
Fwd IAT Total                       0.000584
Flow Packets/s                      0.000330
Fwd Packet Length Mean              0.000203
Flow IAT Std                        0.000166
Total Backward Packets              0.000099
Bwd Packet Length Mean              0.000072
Bwd Packet Length Min               0.000034
Total Fwd Packets                   0.000026
Total Length of Fwd Packets         0.000023
Bwd Packet Length Max               0.000017
```

```
DoS Slowhttptest
                                  importance
Features
Flow IAT Mean                       0.617239
Bwd Packet Length Mean              0.041451
Fwd Packet Length Mean              0.040564
Fwd Packet Length Std               0.037473
Fwd Packet Length Min               0.019764
Flow IAT Min                        0.003142
Total Length of Bwd Packets         0.003040
Fwd Packet Length Max               0.002408
Bwd Packet Length Std               0.001644
Bwd Packet Length Max               0.001378
Flow Bytes/s                        0.000647
Flow Duration                       0.000581
Bwd Packet Length Min               0.000488
Total Length of Fwd Packets         0.000461
Flow IAT Max                        0.000354
Fwd IAT Total                       0.000333
Total Backward Packets              0.000270
Flow Packets/s                      0.000248
Total Fwd Packets                   0.000116
Flow IAT Std                        0.000076
```

```
DoS slowloris
                                  importance
Features
Bwd Packet Length Mean              0.288912
Flow IAT Std                        0.133498
Fwd Packet Length Min               0.074438
Fwd Packet Length Max               0.001852
Fwd IAT Total                       0.001574
Fwd Packet Length Mean              0.001568
Total Length of Fwd Packets         0.001324
Flow Bytes/s                        0.001099
Flow IAT Mean                       0.000904
Total Backward Packets              0.000890
Flow Duration                       0.000863
Fwd Packet Length Std               0.000642
Flow IAT Min                        0.000610
Bwd Packet Length Std               0.000489
Flow Packets/s                      0.000324
Flow IAT Max                        0.000262
Total Fwd Packets                   0.000090
Bwd Packet Length Min               0.000038
Bwd Packet Length Max               0.000026
Total Length of Bwd Packets         0.000022
```



```
FTP-Patator
                              importance
Features
Fwd Packet Length Max           0.186315
Fwd Packet Length Std           0.032481
Flow IAT Min                    0.000263
Bwd Packet Length Std           0.000231
Bwd Packet Length Mean          0.000187
Flow IAT Max                    0.000110
Total Fwd Packets               0.000102
Flow IAT Mean                   0.000089
Fwd IAT Total                   0.000083
Bwd Packet Length Max           0.000070
Total Backward Packets          0.000060
Total Length of Fwd Packets     0.000057
Flow Duration                   0.000050
Total Length of Bwd Packets     0.000046
Fwd Packet Length Mean          0.000046
Fwd Packet Length Min           0.000038
Flow IAT Std                    0.000026
Flow Packets/s                  0.000025
Flow Bytes/s                    0.000017
Bwd Packet Length Min           0.000014
```

```
Heartbleed
                              importance
Features
Fwd IAT Total                      0.044
Flow IAT Std                       0.044
Flow IAT Max                       0.040
Flow Duration                      0.036
Bwd Packet Length Mean             0.032
Flow Packets/s                     0.032
Flow IAT Mean                      0.032
Total Fwd Packets                  0.028
Bwd Packet Length Max              0.020
Bwd Packet Length Std              0.020
Total Length of Fwd Packets        0.020
Total Backward Packets             0.020
Total Length of Bwd Packets        0.016
Fwd Packet Length Max              0.012
Flow IAT Min                       0.012
Fwd Packet Length Std              0.008
Fwd Packet Length Min              0.000
Fwd Packet Length Mean             0.000
Flow Bytes/s                       0.000
Bwd Packet Length Min              0.000
```

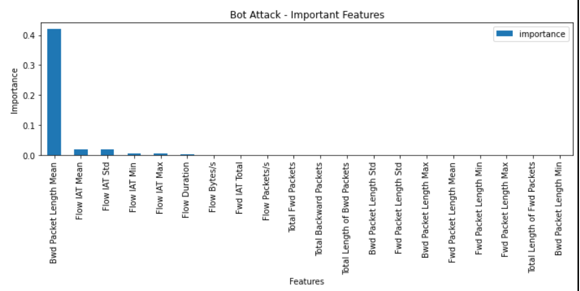
```
Infiltration
                              importance
Features
Total Length of Fwd Packets     0.134829
Fwd Packet Length Std           0.076811
Fwd Packet Length Max           0.052559
Flow Duration                   0.048427
Flow IAT Max                    0.033696
Fwd Packet Length Mean          0.027851
Bwd Packet Length Mean          0.020785
Fwd IAT Total                   0.012544
Bwd Packet Length Max           0.010584
Flow IAT Min                    0.010567
Total Backward Packets          0.010533
Flow IAT Mean                   0.009467
Bwd Packet Length Std           0.005735
Flow IAT Std                    0.005046
Flow Bytes/s                    0.004580
Flow Packets/s                  0.002726
Fwd Packet Length Min           0.002031
Bwd Packet Length Min           0.001586
Total Length of Bwd Packets     0.001545
Total Fwd Packets               0.000812
```

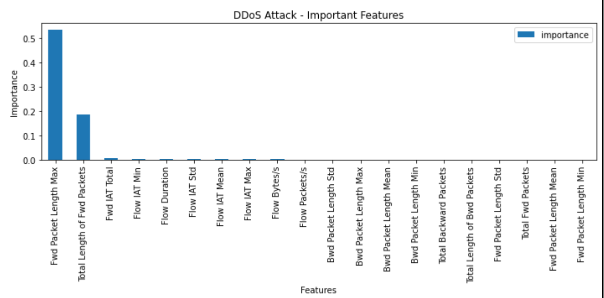
```
PortScan
                              importance
Features
Total Length of Fwd Packets   3.507250e-01
Flow Bytes/s                  2.074986e-01
Flow Duration                 9.020702e-04
Flow IAT Mean                 7.940511e-04
Fwd IAT Total                 4.665794e-04
Flow IAT Max                  3.526532e-04
Flow IAT Min                  2.719751e-04
Fwd Packet Length Max         8.957853e-05
Flow Packets/s                6.286665e-05
Total Length of Bwd Packets   4.099001e-05
Bwd Packet Length Std         3.638010e-05
Fwd Packet Length Mean        2.980757e-05
Bwd Packet Length Max         2.905923e-05
Bwd Packet Length Mean        2.628054e-05
Bwd Packet Length Min         2.620245e-05
Total Fwd Packets             2.205338e-05
Total Backward Packets        1.598123e-05
Flow IAT Std                  1.443451e-05
Fwd Packet Length Std         1.257162e-05
Fwd Packet Length Min         2.278666e-07
```

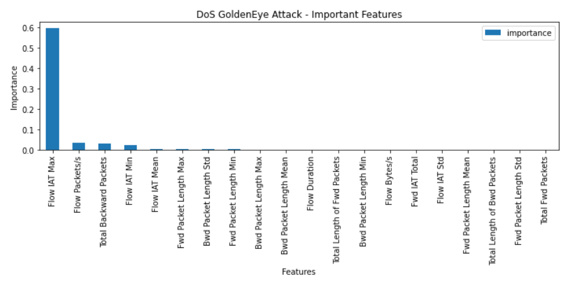
```
SSH-Patator
                              importance
Features
Flow IAT Min                    0.004430
Fwd Packet Length Max           0.001620
Flow IAT Mean                   0.001401
Flow IAT Max                    0.000659
Flow Packets/s                  0.000606
Flow Duration                   0.000587
Total Length of Fwd Packets     0.000496
Flow IAT Std                    0.000371
Fwd IAT Total                   0.000295
Bwd Packet Length Std           0.000210
Bwd Packet Length Max           0.000185
Bwd Packet Length Mean          0.000164
Fwd Packet Length Mean          0.000163
Fwd Packet Length Std           0.000104
Flow Bytes/s                    0.000054
Total Length of Bwd Packets     0.000049
Total Fwd Packets               0.000023
Total Backward Packets          0.000022
Fwd Packet Length Min           0.000020
Bwd Packet Length Min           0.000004
```

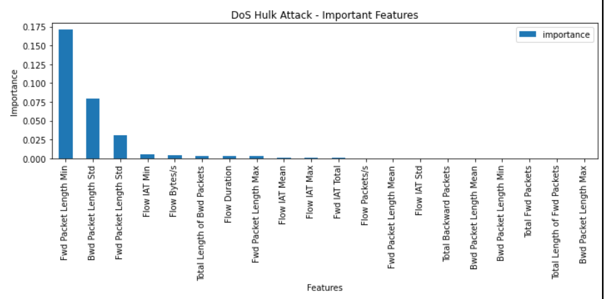
```
Web Attack
                              importance
Features
Bwd Packet Length Std           0.011129
Bwd Packet Length Max           0.002191
Flow IAT Min                    0.001258
Flow Duration                   0.000878
Flow IAT Max                    0.000765
Flow Bytes/s                    0.000742
Flow IAT Mean                   0.000718
Bwd Packet Length Mean          0.000627
Fwd Packet Length Mean          0.000528
Fwd Packet Length Std           0.000485
Flow Packets/s                  0.000405
Fwd IAT Total                   0.000347
Total Fwd Packets               0.000274
Total Length of Fwd Packets     0.000257
Total Backward Packets          0.000253
Flow IAT Std                    0.000224
Fwd Packet Length Max           0.000208
Total Length of Bwd Packets     0.000052
Fwd Packet Length Min           0.000012
Bwd Packet Length Min           0.000009
```

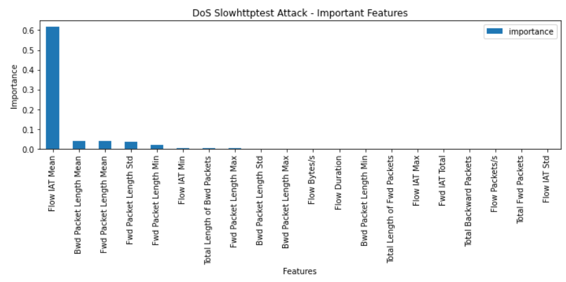
In Table 10. Weightage based on the 50:50 ratio of "Benign" vs. “Attack" shows the feature weightage. Weightage importance for each type of attack is calculated with the Random Forest Regressor classifier. Each type of attacks has different feature importance which can be seen in table for respective type of attack. The result shown in table is showing only top 20 most important features out of 79 features.

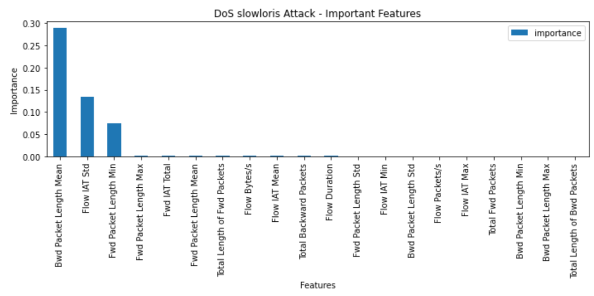

Bot Attack - Important Features
importance
Importance
Features
Bwd Packet Length Mean
Flow IAT Mean
Flow IAT Std
Flow IAT Min
Flow IAT Max
Flow Duration
Flow Bytes/s
Fwd IAT Total
Flow Packets/s
Total Fwd Packets
Total Backward Packets
Total Length of Bwd Packets
Bwd Packet Length Std
Fwd Packet Length Std
Bwd Packet Length Max
Fwd Packet Length Mean
Fwd Packet Length Min
Fwd Packet Length Max
Total Length of Fwd Packets
Bwd Packet Length Min

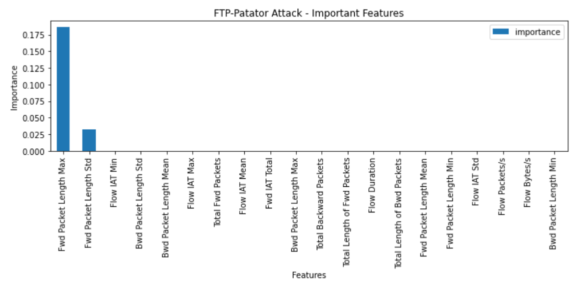

DDoS Attack - Important Features
importance
Importance
Features
Fwd Packet Length Max
Total Length of Fwd Packets
Fwd IAT Total
Flow IAT Min
Flow Duration
Flow IAT Std
Flow IAT Mean
Flow IAT Max
Flow Bytes/s
Flow Packets/s
Bwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Mean
Bwd Packet Length Min
Total Backward Packets
Total Length of Bwd Packets
Fwd Packet Length Std
Total Fwd Packets
Fwd Packet Length Mean
Fwd Packet Length Min

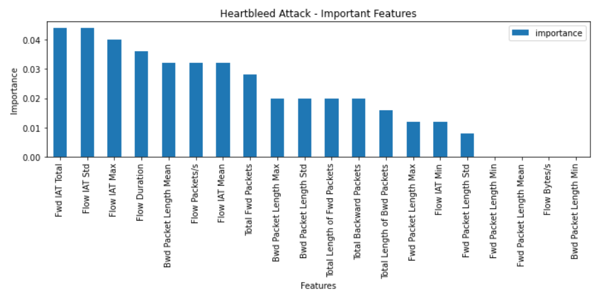

DoS GoldenEye Attack - Important Features
importance
Importance
Features
Flow IAT Max
Flow Packets/s
Total Backward Packets
Flow IAT Min
Flow IAT Mean
Fwd Packet Length Max
Bwd Packet Length Std
Fwd Packet Length Min
Bwd Packet Length Max
Bwd Packet Length Mean
Flow Duration
Total Length of Fwd Packets
Bwd Packet Length Min
Flow Bytes/s
Fwd IAT Total
Flow IAT Std
Fwd Packet Length Mean
Total Length of Bwd Packets
Fwd Packet Length Std
Total Fwd Packets

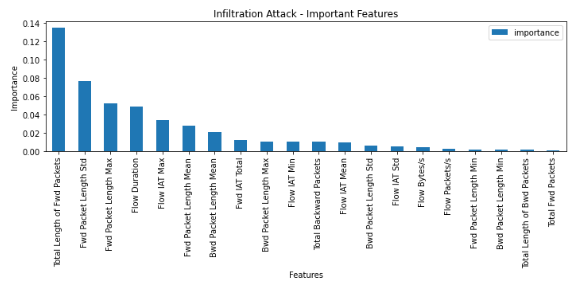

DoS Hulk Attack - Important Features
importance
Importance
Features
Fwd Packet Length Min
Bwd Packet Length Std
Fwd Packet Length Std
Flow IAT Min
Flow Bytes/s
Total Length of Bwd Packets
Flow Duration
Fwd Packet Length Max
Flow IAT Mean
Flow IAT Max
Fwd IAT Total
Flow Packets/s
Fwd Packet Length Mean
Flow IAT Std
Total Backward Packets
Bwd Packet Length Mean
Bwd Packet Length Min
Total Fwd Packets
Total Length of Fwd Packets
Bwd Packet Length Max

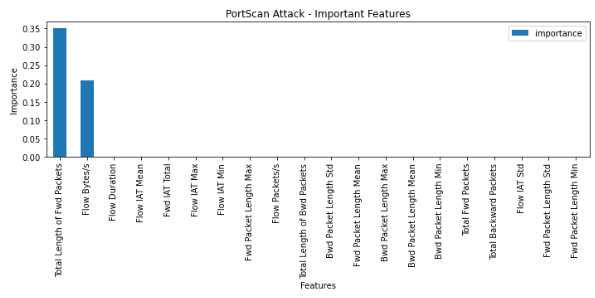

DoS Slowhttptest Attack - Important Features
importance
Importance
Features
Flow IAT Mean
Bwd Packet Length Mean
Fwd Packet Length Mean
Fwd Packet Length Std
Fwd Packet Length Min
Flow IAT Min
Total Length of Bwd Packets
Fwd Packet Length Max
Bwd Packet Length Std
Bwd Packet Length Max
Flow Bytes/s
Flow Duration
Bwd Packet Length Min
Total Length of Fwd Packets
Flow IAT Max
Fwd IAT Total
Total Backward Packets
Flow Packets/s
Total Fwd Packets
Flow IAT Std


DoS slowloris Attack - Important Features
importance
Importance
Features
Bwd Packet Length Mean
Flow IAT Std
Fwd Packet Length Min
Fwd Packet Length Max
Fwd IAT Total
Fwd Packet Length Mean
Total Length of Fwd Packets
Flow Bytes/s
Flow IAT Mean
Total Backward Packets
Flow Duration
Fwd Packet Length Std
Flow IAT Min
Bwd Packet Length Std
Flow Packets/s
Flow IAT Max
Total Fwd Packets
Bwd Packet Length Min
Bwd Packet Length Max
Total Length of Bwd Packets


FTP-Patator Attack - Important Features
importance
Importance
Features
Fwd Packet Length Max
Fwd Packet Length Std
Flow IAT Min
Bwd Packet Length Std
Bwd Packet Length Mean
Flow IAT Max
Total Fwd Packets
Flow IAT Mean
Fwd IAT Total
Bwd Packet Length Max
Total Backward Packets
Total Length of Fwd Packets
Flow Duration
Total Length of Bwd Packets
Fwd Packet Length Mean
Fwd Packet Length Min
Flow IAT Std
Flow Packets/s
Flow Bytes/s
Bwd Packet Length Min


Heartbleed Attack - Important Features
importance
Importance
Features
Fwd IAT Total
Flow IAT Std
Flow IAT Max
Flow Duration
Bwd Packet Length Mean
Flow Packets/s
Flow IAT Mean
Total Fwd Packets
Bwd Packet Length Max
Bwd Packet Length Std
Total Length of Fwd Packets
Total Backward Packets
Total Length of Bwd Packets
Fwd Packet Length Max
Flow IAT Min
Fwd Packet Length Std
Fwd Packet Length Min
Fwd Packet Length Mean
Flow Bytes/s
Bwd Packet Length Min


Infiltration Attack - Important Features
importance
Importance
Features
Total Length of Fwd Packets
Fwd Packet Length Std
Fwd Packet Length Max
Flow Duration
Flow IAT Max
Fwd Packet Length Mean
Bwd Packet Length Mean
Fwd IAT Total
Bwd Packet Length Max
Flow IAT Min
Total Backward Packets
Flow IAT Mean
Bwd Packet Length Std
Flow IAT Std
Flow Bytes/s
Flow Packets/s
Fwd Packet Length Min
Bwd Packet Length Min
Total Length of Bwd Packets
Total Fwd Packets


PortScan Attack - Important Features
importance
Importance
Features
Total Length of Fwd Packets
Flow Bytes/s
Flow Duration
Flow IAT Mean
Fwd IAT Total
Flow IAT Max
Flow IAT Min
Fwd Packet Length Max
Flow Packets/s
Total Length of Bwd Packets
Bwd Packet Length Std
Fwd Packet Length Mean
Bwd Packet Length Max
Bwd Packet Length Mean
Bwd Packet Length Min
Total Fwd Packets
Total Backward Packets
Flow IAT Std
Fwd Packet Length Std
Fwd Packet Length Min

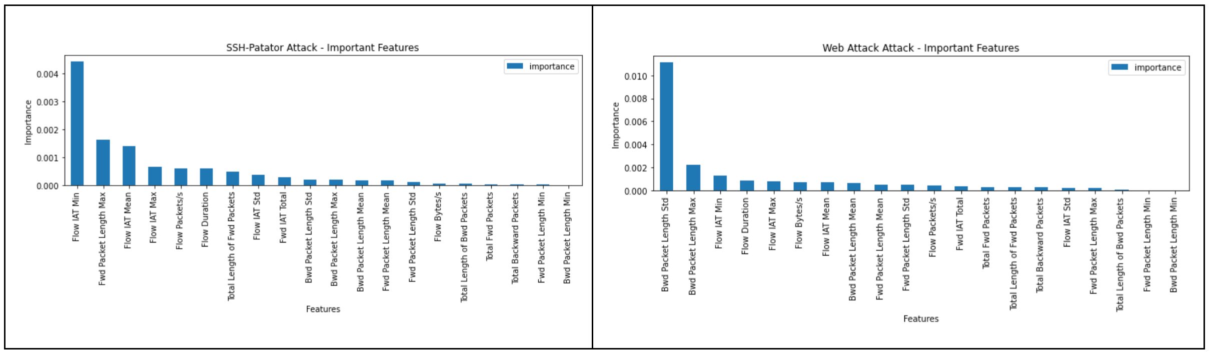
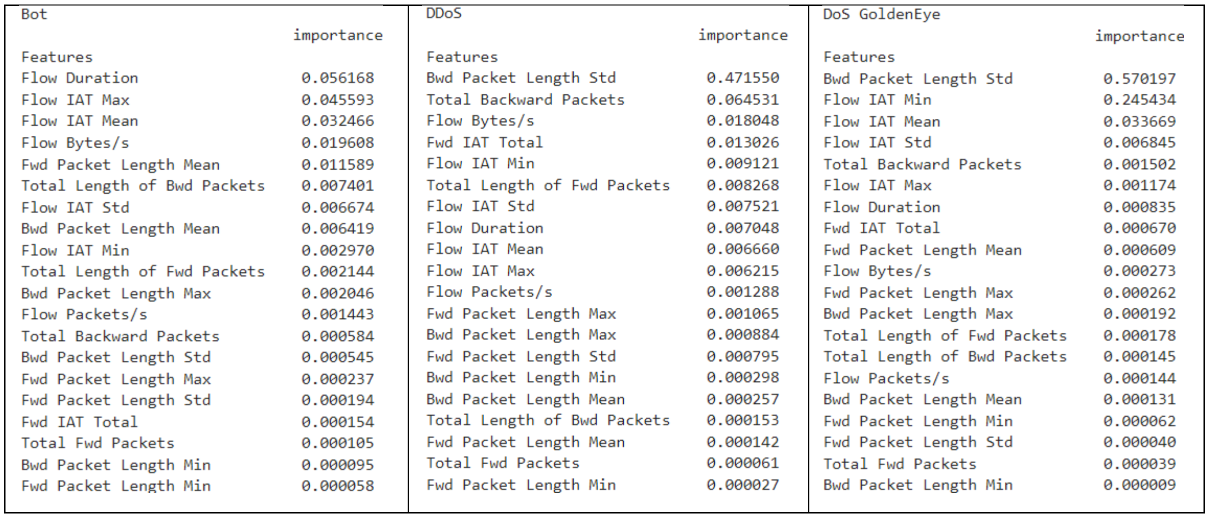


Figure 12. Feature weightage for 50:50 ratio of "Benign" vs. “Attack"

Figure 12 shows its equivalent graph. Top 20 features are displayed in the output. In the observation it is found that most of the attacks are using mostly three to six dominant features which is highly correlated for the attack detection. The Heartbleed attack, on the other hand, have quite different properties and their more than ten attributes are of similar dominance. Top 5 features from all 12 attack categories are taken which counts to total of 60 features. Many features were redundant in each attack categories, so after removing duplicate features only 17 most used features were extracted.

II. Feature importance when file stream is kept in ratio of 80:20 for "Benign" and "Attack", following results were achieved:

Table 11. Weightage based on the 80:20 ratio of "Benign" vs. “Attack"



```
Bot
                              importance
Features
Flow Duration                   0.056168
Flow IAT Max                    0.045593
Flow IAT Mean                   0.032466
Flow Bytes/s                    0.019608
Fwd Packet Length Mean          0.011589
Total Length of Bwd Packets     0.007401
Flow IAT Std                    0.006674
Bwd Packet Length Mean          0.006419
Flow IAT Min                    0.002970
Total Length of Fwd Packets     0.002144
Bwd Packet Length Max           0.002046
Flow Packets/s                  0.001443
Total Backward Packets          0.000584
Bwd Packet Length Std           0.000545
Fwd Packet Length Max           0.000237
Fwd Packet Length Std           0.000194
Fwd IAT Total                   0.000154
Total Fwd Packets               0.000105
Bwd Packet Length Min           0.000095
Fwd Packet Length Min           0.000058
```

```
DDoS
                              importance
Features
Bwd Packet Length Std           0.471550
Total Backward Packets          0.064531
Flow Bytes/s                    0.018048
Fwd IAT Total                   0.013026
Flow IAT Min                    0.009121
Total Length of Fwd Packets     0.008268
Flow IAT Std                    0.007521
Flow Duration                   0.007048
Flow IAT Mean                   0.006660
Flow IAT Max                    0.006215
Flow Packets/s                  0.001288
Fwd Packet Length Max           0.001065
Bwd Packet Length Max           0.000884
Fwd Packet Length Std           0.000795
Bwd Packet Length Min           0.000298
Bwd Packet Length Mean          0.000257
Total Length of Bwd Packets     0.000153
Fwd Packet Length Mean          0.000142
Total Fwd Packets               0.000061
Fwd Packet Length Min           0.000027
```

```
DoS GoldenEye
                              importance
Features
Bwd Packet Length Std           0.570197
Flow IAT Min                    0.245434
Flow IAT Mean                   0.033669
Flow IAT Std                    0.006845
Total Backward Packets          0.001502
Flow IAT Max                    0.001174
Flow Duration                   0.000835
Fwd IAT Total                   0.000670
Fwd Packet Length Mean          0.000609
Flow Bytes/s                    0.000273
Fwd Packet Length Max           0.000262
Bwd Packet Length Max           0.000192
Total Length of Fwd Packets     0.000178
Total Length of Bwd Packets     0.000145
Flow Packets/s                  0.000144
Bwd Packet Length Mean          0.000131
Fwd Packet Length Min           0.000062
Fwd Packet Length Std           0.000040
Total Fwd Packets               0.000039
Bwd Packet Length Min           0.000009
```

```
DoS Hulk
                                importance
Features
Fwd Packet Length Std         5.148903e-02
Bwd Packet Length Std         3.267070e-02
Bwd Packet Length Mean        1.864779e-02
Flow IAT Min                  3.155069e-03
Fwd Packet Length Max         2.241508e-03
Fwd Packet Length Mean        1.886793e-03
Flow Duration                 1.622871e-03
Total Length of Fwd Packets   4.629747e-04
Fwd IAT Total                 4.492172e-04
Flow IAT Std                  3.146338e-04
Total Length of Bwd Packets   2.843317e-04
Total Backward Packets        2.815263e-04
Flow IAT Mean                 1.789805e-04
Flow Bytes/s                  1.465322e-04
Flow Packets/s                9.395857e-05
Flow IAT Max                  9.247921e-05
Total Fwd Packets             4.598578e-05
Bwd Packet Length Max         3.771230e-05
Bwd Packet Length Min         6.580825e-06
Fwd Packet Length Min         5.271332e-08
```

```
DoS Slowhttptest
                              importance
Features
Flow IAT Mean                   0.643451
Fwd Packet Length Min           0.093049
Bwd Packet Length Mean          0.016968
Bwd Packet Length Max           0.015690
Total Length of Bwd Packets     0.014859
Bwd Packet Length Std           0.002886
Fwd Packet Length Std           0.002300
Fwd Packet Length Mean          0.002134
Flow IAT Min                    0.001537
Total Fwd Packets               0.000740
Total Length of Fwd Packets     0.000706
Fwd Packet Length Max           0.000694
Flow Duration                   0.000532
Flow Bytes/s                    0.000516
Bwd Packet Length Min           0.000462
Fwd IAT Total                   0.000213
Total Backward Packets          0.000206
Flow IAT Max                    0.000188
Flow IAT Std                    0.000176
Flow Packets/s                  0.000072
```

```
DoS slowloris
                              importance
Features
Flow IAT Mean                   0.407084
Bwd Packet Length Mean          0.127372
Total Fwd Packets               0.015042
Fwd IAT Total                   0.011822
Total Length of Bwd Packets     0.002669
Flow IAT Min                    0.002007
Flow Bytes/s                    0.001102
Flow IAT Std                    0.000862
Fwd Packet Length Std           0.000816
Flow IAT Max                    0.000772
Total Backward Packets          0.000760
Flow Packets/s                  0.000745
Flow Duration                   0.000597
Fwd Packet Length Max           0.000484
Fwd Packet Length Mean          0.000451
Total Length of Fwd Packets     0.000331
Bwd Packet Length Std           0.000273
Bwd Packet Length Max           0.000243
Fwd Packet Length Min           0.000123
Bwd Packet Length Min           0.000010
```

```
FTP-Patator
                              importance
Features
Fwd Packet Length Std           0.030042
Bwd Packet Length Std           0.000827
Bwd Packet Length Max           0.000605
Bwd Packet Length Mean          0.000542
Total Length of Bwd Packets     0.000507
Fwd Packet Length Max           0.000441
Fwd Packet Length Mean          0.000292
Flow IAT Min                    0.000254
Total Length of Fwd Packets     0.000137
Fwd IAT Total                   0.000092
Total Backward Packets          0.000065
Flow Duration                   0.000058
Flow IAT Mean                   0.000057
Flow Packets/s                  0.000046
Total Fwd Packets               0.000040
Fwd Packet Length Min           0.000033
Flow IAT Max                    0.000033
Flow IAT Std                    0.000030
Flow Bytes/s                    0.000018
Bwd Packet Length Min           0.000002
```

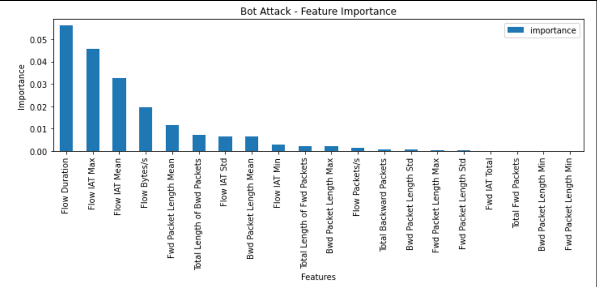
```
Heartbleed
                              importance
Features
Total Backward Packets             0.052
Bwd Packet Length Max              0.048
Bwd Packet Length Mean             0.048
Total Length of Bwd Packets        0.044
Bwd Packet Length Std              0.036
Total Fwd Packets                  0.036
Total Length of Fwd Packets        0.032
Flow Duration                      0.020
Flow IAT Min                       0.016
Fwd IAT Total                      0.012
Fwd Packet Length Max              0.004
Fwd Packet Length Std              0.000
Fwd Packet Length Mean             0.000
Fwd Packet Length Min              0.000
Flow Bytes/s                       0.000
Flow Packets/s                     0.000
Flow IAT Mean                      0.000
Flow IAT Std                       0.000
Flow IAT Max                       0.000
Bwd Packet Length Min              0.000
```

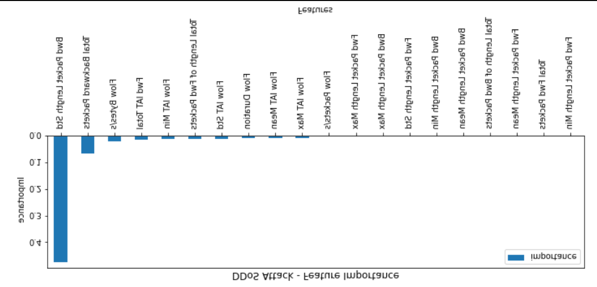
```
Infiltration
                              importance
Features
Fwd Packet Length Mean          0.250581
Fwd Packet Length Max           0.072484
Total Length of Fwd Packets     0.062854
Flow IAT Mean                   0.014998
Bwd Packet Length Std           0.013175
Bwd Packet Length Max           0.012933
Bwd Packet Length Mean          0.012682
Flow IAT Min                    0.009434
Fwd IAT Total                   0.005731
Flow IAT Std                    0.005230
Flow Bytes/s                    0.004643
Fwd Packet Length Std           0.004506
Flow IAT Max                    0.004505
Flow Duration                   0.003462
Total Backward Packets          0.003185
Total Fwd Packets               0.001600
Total Length of Bwd Packets     0.001091
Flow Packets/s                  0.000833
Fwd Packet Length Min           0.000434
Bwd Packet Length Min           0.000177
```

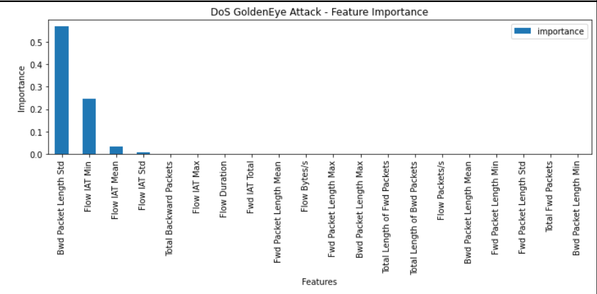
```
PortScan
                              importance
Features
Flow Bytes/s                    0.437283
Total Length of Fwd Packets     0.246983
Flow IAT Max                    0.000220
Fwd Packet Length Max           0.000200
Flow IAT Mean                   0.000066
Total Fwd Packets               0.000065
Bwd Packet Length Min           0.000061
Fwd Packet Length Mean          0.000060
Fwd Packet Length Min           0.000046
Flow Duration                   0.000039
Total Length of Bwd Packets     0.000031
Fwd IAT Total                   0.000030
Bwd Packet Length Mean          0.000025
Flow IAT Std                    0.000022
Flow Packets/s                  0.000019
Flow IAT Min                    0.000017
Total Backward Packets          0.000015
Bwd Packet Length Std           0.000014
Bwd Packet Length Max           0.000013
Fwd Packet Length Std           0.000012
```

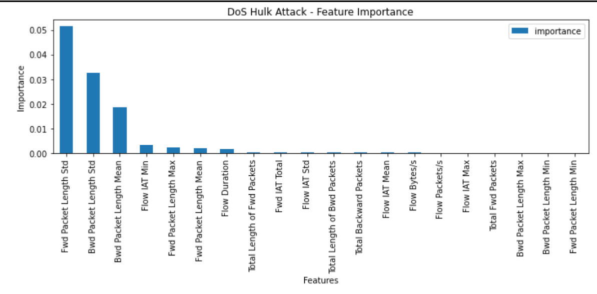
```
SSH-Patator
                              importance
Features
Flow Duration                   0.001400
Fwd Packet Length Max           0.000627
Flow IAT Mean                   0.000597
Flow Bytes/s                    0.000357
Flow IAT Max                    0.000350
Fwd IAT Total                   0.000333
Flow Packets/s                  0.000332
Flow IAT Std                    0.000291
Total Length of Fwd Packets     0.000263
Fwd Packet Length Std           0.000187
Fwd Packet Length Mean          0.000182
Flow IAT Min                    0.000172
Total Fwd Packets               0.000110
Total Backward Packets          0.000069
Bwd Packet Length Max           0.000055
Bwd Packet Length Mean          0.000048
Total Length of Bwd Packets     0.000032
Bwd Packet Length Std           0.000023
Bwd Packet Length Min           0.000009
Fwd Packet Length Min           0.000008
```

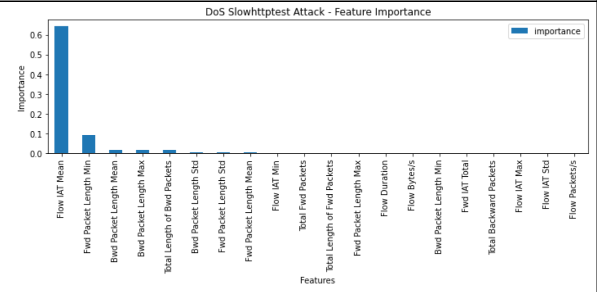
```
Web Attack
                              importance
Features
Total Length of Fwd Packets     0.006104
Fwd Packet Length Std           0.002943
Fwd Packet Length Mean          0.002221
Flow IAT Min                    0.002077
Bwd Packet Length Max           0.001997
Flow IAT Max                    0.001854
Flow Bytes/s                    0.001411
Bwd Packet Length Std           0.001266
Total Length of Bwd Packets     0.000829
Fwd Packet Length Max           0.000806
Flow Packets/s                  0.000500
Flow Duration                   0.000494
Flow IAT Mean                   0.000415
Total Backward Packets          0.000332
Fwd IAT Total                   0.000329
Total Fwd Packets               0.000224
Bwd Packet Length Mean          0.000200
Flow IAT Std                    0.000170
Bwd Packet Length Min           0.000007
Fwd Packet Length Min           0.000003
```

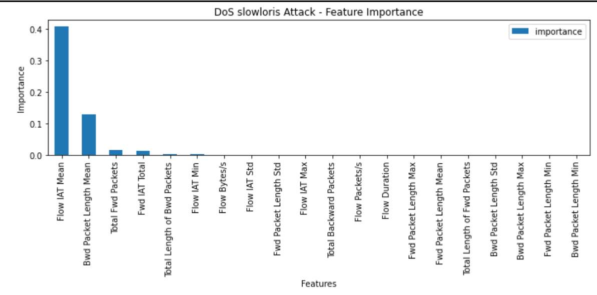

Bot Attack - Feature Importance
importance
Importance
Features
Flow Duration
Flow IAT Max
Flow IAT Mean
Flow Bytes/s
Fwd Packet Length Mean
Total Length of Bwd Packets
Flow IAT Std
Bwd Packet Length Mean
Flow IAT Min
Total Length of Fwd Packets
Bwd Packet Length Max
Flow Packets/s
Total Backward Packets
Bwd Packet Length Std
Fwd Packet Length Max
Fwd Packet Length Std
Fwd IAT Total
Total Fwd Packets
Bwd Packet Length Min
Fwd Packet Length Min

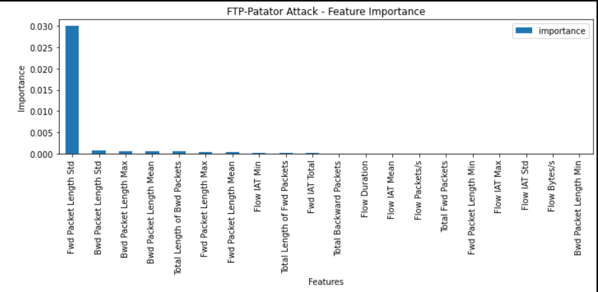

DDoS Attack - Feature Importance
importance
Importance
Features
Bwd Packet Length Std
Total Backward Packets
Flow Bytes/s
Fwd IAT Total
Flow IAT Min
Total Length of Fwd Packets
Flow IAT Std
Flow Duration
Flow IAT Mean
Flow IAT Max
Flow Packets/s
Fwd Packet Length Max
Bwd Packet Length Max
Fwd Packet Length Std
Bwd Packet Length Min
Bwd Packet Length Mean
Total Length of Bwd Packets
Fwd Packet Length Mean
Total Fwd Packets
Fwd Packet Length Min

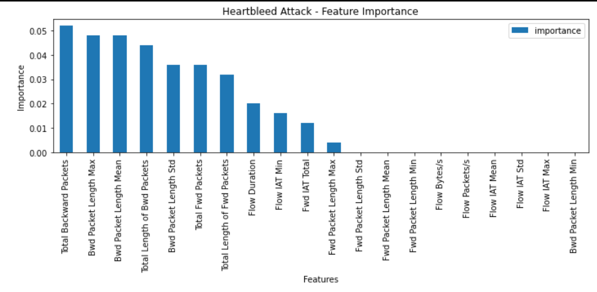

DoS GoldenEye Attack - Feature Importance
importance
Importance
Features
Bwd Packet Length Std
Flow IAT Min
Flow IAT Mean
Flow IAT Std
Total Backward Packets
Flow IAT Max
Flow Duration
Fwd IAT Total
Fwd Packet Length Mean
Flow Bytes/s
Fwd Packet Length Max
Bwd Packet Length Max
Total Length of Fwd Packets
Total Length of Bwd Packets
Flow Packets/s
Bwd Packet Length Mean
Fwd Packet Length Min
Fwd Packet Length Std
Total Fwd Packets
Bwd Packet Length Min

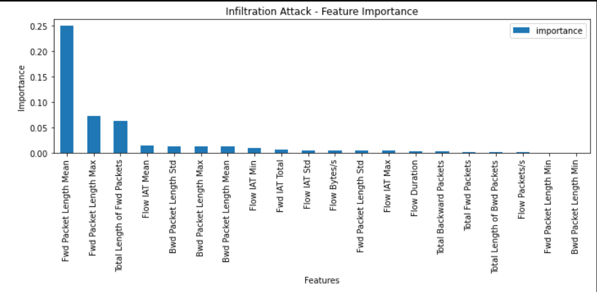

DoS Hulk Attack - Feature Importance
importance
Importance
Features
Fwd Packet Length Std
Bwd Packet Length Std
Bwd Packet Length Mean
Flow IAT Min
Fwd Packet Length Max
Fwd Packet Length Mean
Flow Duration
Total Length of Fwd Packets
Fwd IAT Total
Flow IAT Std
Total Length of Bwd Packets
Total Backward Packets
Flow IAT Mean
Flow Bytes/s
Flow Packets/s
Flow IAT Max
Total Fwd Packets
Bwd Packet Length Max
Bwd Packet Length Min
Fwd Packet Length Min

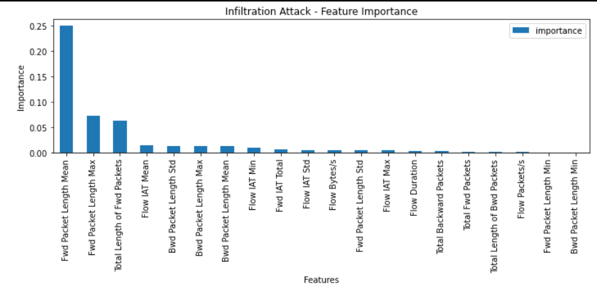

DoS Slowhttptest Attack - Feature Importance
importance
Importance
Features
Flow IAT Mean
Fwd Packet Length Min
Bwd Packet Length Mean
Bwd Packet Length Max
Total Length of Bwd Packets
Bwd Packet Length Std
Fwd Packet Length Std
Fwd Packet Length Mean
Flow IAT Min
Total Fwd Packets
Total Length of Fwd Packets
Fwd Packet Length Max
Flow Duration
Flow Bytes/s
Bwd Packet Length Min
Fwd IAT Total
Total Backward Packets
Flow IAT Max
Flow IAT Std
Flow Packets/s


DoS slowloris Attack - Feature Importance
importance
Importance
Features
Flow IAT Mean
Bwd Packet Length Mean
Total Fwd Packets
Fwd IAT Total
Total Length of Bwd Packets
Flow IAT Min
Flow Bytes/s
Flow IAT Std
Fwd Packet Length Std
Flow IAT Max
Total Backward Packets
Flow Packets/s
Flow Duration
Fwd Packet Length Max
Fwd Packet Length Mean
Total Length of Fwd Packets
Bwd Packet Length Std
Bwd Packet Length Max
Fwd Packet Length Min
Bwd Packet Length Min


FTP-Patator Attack - Feature Importance
importance
Importance
Features
Fwd Packet Length Std
Bwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Mean
Total Length of Bwd Packets
Fwd Packet Length Max
Fwd Packet Length Mean
Flow IAT Min
Total Length of Fwd Packets
Fwd IAT Total
Total Backward Packets
Flow Duration
Flow IAT Mean
Flow Packets/s
Total Fwd Packets
Fwd Packet Length Min
Flow IAT Max
Flow IAT Std
Flow Bytes/s
Bwd Packet Length Min


Heartbleed Attack - Feature Importance
importance
Importance
Features
Total Backward Packets
Bwd Packet Length Max
Bwd Packet Length Mean
Total Length of Bwd Packets
Bwd Packet Length Std
Total Fwd Packets
Total Length of Fwd Packets
Flow Duration
Flow IAT Min
Fwd IAT Total
Fwd Packet Length Max
Fwd Packet Length Std
Fwd Packet Length Mean
Fwd Packet Length Min
Flow Bytes/s
Flow Packets/s
Flow IAT Mean
Flow IAT Std
Flow IAT Max
Bwd Packet Length Min


Infiltration Attack - Feature Importance
importance
Importance
Features
Fwd Packet Length Mean
Fwd Packet Length Max
Total Length of Fwd Packets
Flow IAT Mean
Bwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Mean
Flow IAT Min
Fwd IAT Total
Flow IAT Std
Flow Bytes/s
Fwd Packet Length Std
Flow IAT Max
Flow Duration
Total Backward Packets
Total Fwd Packets
Total Length of Bwd Packets
Flow Packets/s
Fwd Packet Length Min
Bwd Packet Length Min


Infiltration Attack - Feature Importance
importance
Importance
Features
Fwd Packet Length Mean
Fwd Packet Length Max
Total Length of Fwd Packets
Flow IAT Mean
Bwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Mean
Flow IAT Min
Fwd IAT Total
Flow IAT Std
Flow Bytes/s
Fwd Packet Length Std
Flow IAT Max
Flow Duration
Total Backward Packets
Total Fwd Packets
Total Length of Bwd Packets
Flow Packets/s
Fwd Packet Length Min
Bwd Packet Length Min

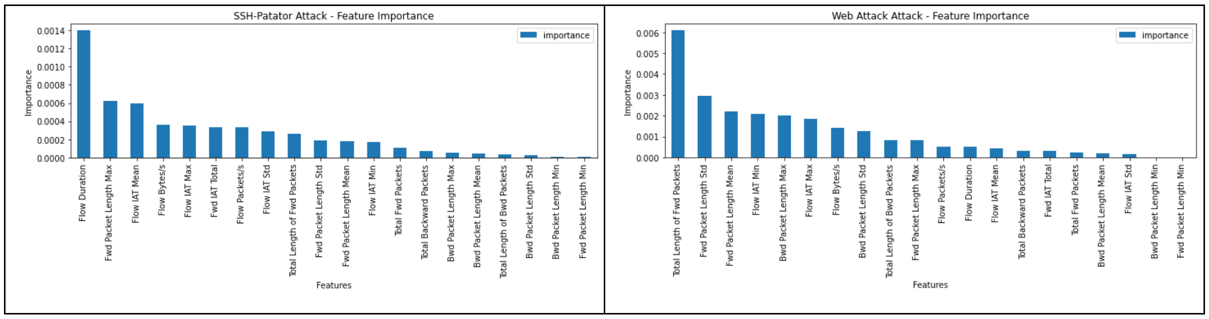
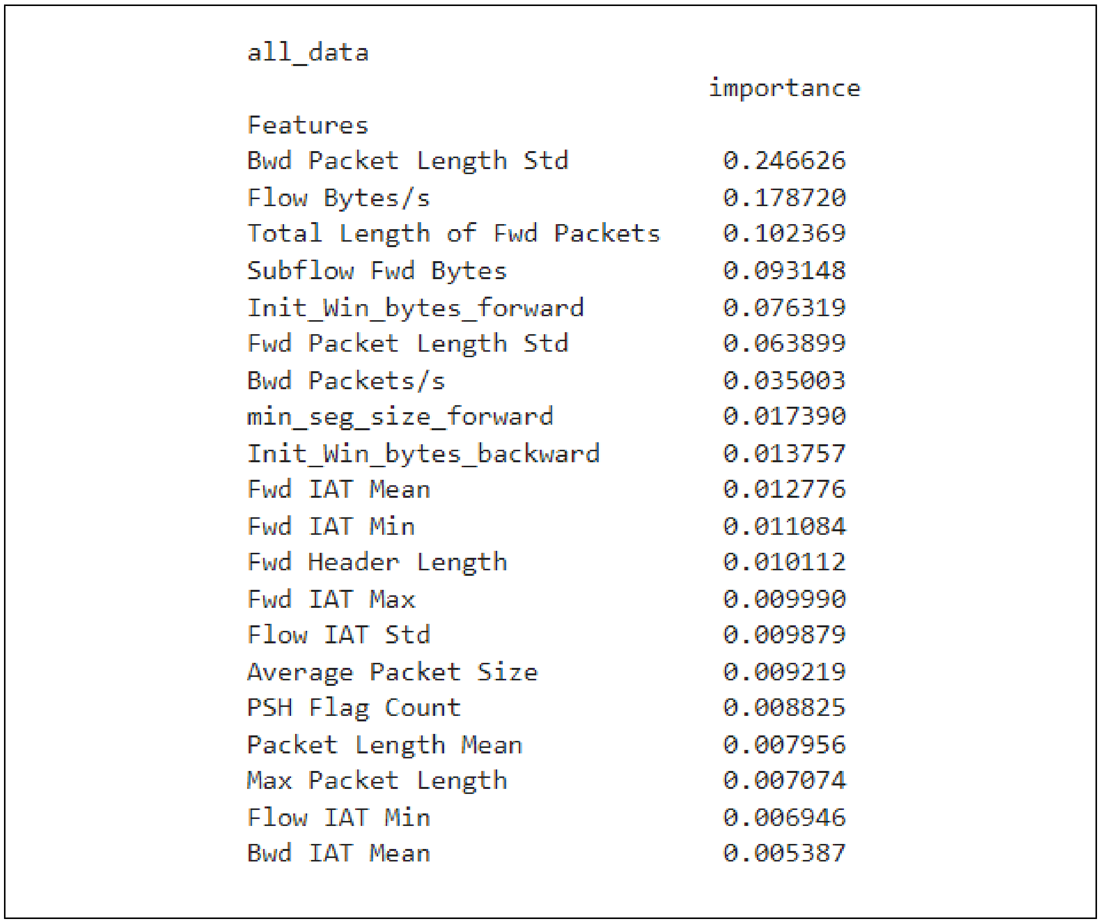


Figure 13. Feature weightage for 80:20 ratio of "Benign" vs. “Attack"

When the file stream was kept in the ratio of 80:20 for the "Benign" vs. “Attack", it is found that the similar features were identified as shown in Figure 12 and Figure 13. This experiment given the similar output for the top 17 features after removing duplicates from the 60 identified features from all the 12 attack categories.

Table 12. Top 20 feature weightage of all data of CICIDS 2017



```
all_data
                              importance
Features
Bwd Packet Length Std           0.246626
Flow Bytes/s                    0.178720
Total Length of Fwd Packets     0.102369
Subflow Fwd Bytes               0.093148
Init_Win_bytes_forward          0.076319
Fwd Packet Length Std           0.063899
Bwd Packets/s                   0.035003
min_seg_size_forward            0.017390
Init_Win_bytes_backward         0.013757
Fwd IAT Mean                    0.012776
Fwd IAT Min                     0.011084
Fwd Header Length               0.010112
Fwd IAT Max                     0.009990
Flow IAT Std                    0.009879
Average Packet Size             0.009219
PSH Flag Count                  0.008825
Packet Length Mean              0.007956
Max Packet Length               0.007074
Flow IAT Min                    0.006946
Bwd IAT Mean                    0.005387
```

In Table 12 representing top 20 features that covers almost all the attack also, it can be seen that the "Bwd Packet Length Std, Flow Bytes/s", "Total Length of Fwd Packets", "Subflow Fwd Bytes", "Init_Win_bytes_forward" and "Fwd Packet Length Std" are the most used feature that almost covers 85% of the attacks.

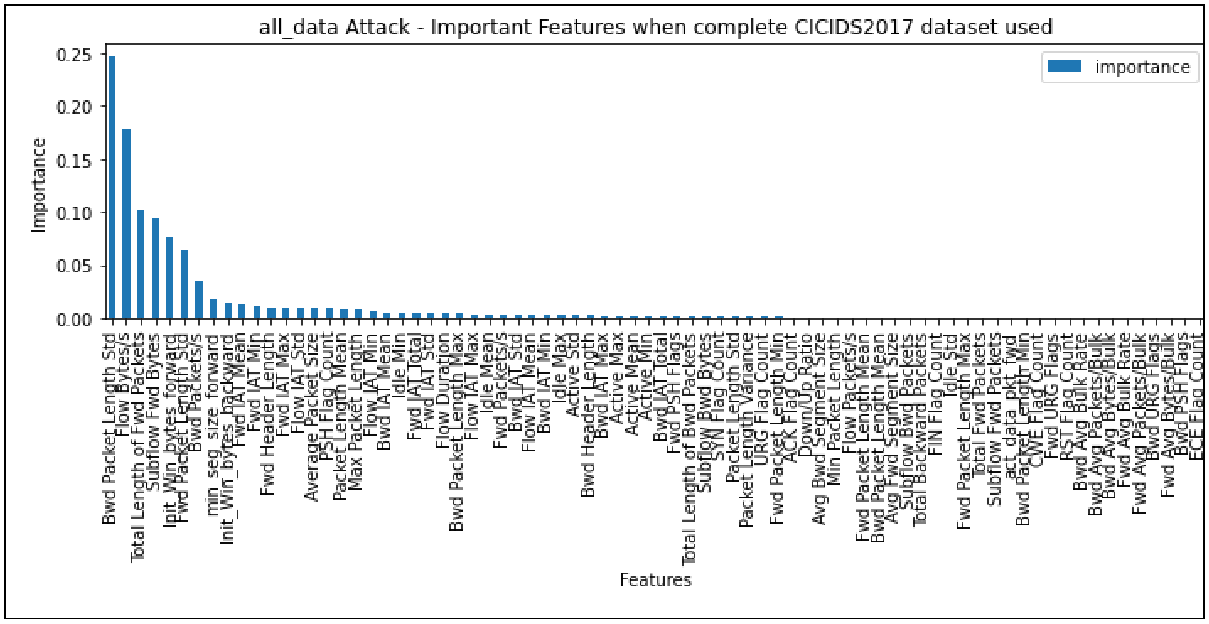


Figure 14. Feature weightage of all data of CICIDS 2017

In Figure 14 representing all the 77 features used for the feature selection and its distribution in the data stream. This graph clearly shows that there are only few features which are most dominant and capable to represent maximum of the attack categories detection. Data stream representing very a smaller number of attacks with these few dominant features may have less chances to identify and need to carefully train such attack types. All the top 20 features from the 77 features taken from the complete dataset and then comparing with all 12 individual attack categories of 50:50 and 80:20 for the "Benign" vs. “Attack" data stream given only 6 most dominant features. These features are:

i. Bwd Packet Length Std
ii. Flow Bytes/s
iii. Total Length of Fwd Packets
iv. Fwd Packet Length Std
v. Flow IAT Std
vi. Flow IAT Min

The feature selection process plays a crucial role in machine learning, especially when considering the distinction between attacks and benign instances. It involves identifying the features that carry the most significant weight and have high importance scores within the dataset. In this study, a weight threshold of 80 percent is employed to determine the relevance of features. By setting the weight threshold at 80 percent, the focus is placed on selecting features that possess substantial importance in distinguishing between attacks and benign instances. This threshold acts as a criterion for filtering out features that may have lower predictive power or contribute less to the classification task at hand. The feature selection methodology aims to prioritize those features that demonstrate a strong influence on determining whether an instance belongs to an attack or benign category. These features, with their high weighting and importance scores, are deemed more influential in accurately classifying instances.

By selecting features based on their importance scores and applying a weight threshold of 80 percent, the feature selection process in this study ensures that the most informative and influential features are retained. This approach enhances the performance and efficiency of the machine learning model by focusing on such features that contribute the maximum towards the classification task, ultimately leading to improved accuracy and robustness in distinguishing between attacks and benign instances.

Table 13. Feature Importance with 80% dataset for "Attack or Benign"

Top 20 Features (80% Dataset considered) Feature Importance			
Bwd Packet Length Std	0.246626	Flow IAT Mean	0.003275
Flow Bytes/s	0.17872	Total Length of Bwd Packets	0.001325
Total Length of Fwd Packets	0.102369	Fwd Packet Length Min	0.000686
Fwd Packet Length Std	0.063899	Flow Packets/s	0.000541
Flow IAT Std	0.009879	Fwd Packet Length Mean	0.000537
Flow IAT Min	0.006946	Bwd Packet Length Mean	0.000526
Fwd IAT Total	0.005136	Total Backward Packets	0.000177
Flow Duration	0.00415	Fwd Packet Length Max	0.000138
Bwd Packet Length Max	0.004014	Total Fwd Packets	0.000127
Flow IAT Max	0.003534	Bwd Packet Length Min	0.000076
Total Length of Fwd Packets	0.102369	Total Length of Fwd Packets	0.102369

The provided analysis results show the importance scores for various features in the dataset. The higher the importance score, the more influential the feature is in distinguishing between different classes or making predictions. Here is a brief analysis of the given results:

- Bwd Packet Length Std: This feature has a relatively high importance score of 0.246626, indicating that it carries significant information for classification or prediction tasks.
- Flow IAT Mean: Although this feature has a lower importance score of 0.003275, it still contributes to the overall classification process.
- Flow Bytes/s: With an importance score of 0.17872, this feature demonstrates its significance in differentiating between classes or making predictions.
- Total Length of Bwd Packets: This feature has a relatively low importance score of 0.001325, suggesting that it may have a lesser impact on the classification process compared to other features.
- Total Length of Fwd Packets: This feature appears twice in the results, indicating its importance. With an importance score of 0.102369, it plays a significant role in classification tasks.

The analysis highlights that feature like Bwd Packet Length Std, Flow Bytes/s, and Total Length of Fwd Packets are among the top influential features for the classification or prediction task at hand. Meanwhile, features with lower importance scores, such as Flow IAT Mean and Total Length of Bwd Packets, may have a relatively smaller impact on the overall classification performance. It is essential to consider these importance scores when selecting the most relevant features for the ML model to enhance accuracy and efficiency in distinguishing between different classes or making predictions.

4.4 Implementation of Machine Learning Algorithms

To examine the implementation of different ML algorithms on the dataset, the data was divided into testing and training sets using ratios of 50:50 and 80:20. Nine machine learning algorithms were employed for this purpose, namely "Bagging," "Naive Bayes," "QDA" (Quadratic Discriminant Analysis), "Random Forest," "ID3," "AdaBoost," "GradientBoost," "MLP" (Multi-Layer Perceptron), and "Nearest Neighbors."

Each algorithm was trained and tested using specific parameters to optimize their performance. The parameters were carefully selected to ensure efficient learning and accurate predictions. By tuning these parameters, the algorithms can adapt to the characteristics of the dataset and improve their predictive capabilities. Use of diverse algorithms allows for a comprehensive evaluation of their performance on the specified dataset. Each algorithm has its own strengths and weaknesses, and by evaluating them, we can identify the most suitable algorithm for the task at hand.

By conducting experiments with various algorithms and parameter settings, valuable insights can be gained regarding their effectiveness in handling the dataset. The performance metrics obtained from these experiments will provide a basis for selecting the most appropriate algorithm for future applications or further optimization. Overall, this approach enables a systematic and rigorous analysis of the dataset using a diverse set of machine learning algorithms, leading to informed decision-making and potentially improving the accuracy and reliability of predictions in various domains.

i. "Bagging": n estimators = 5, K Neighbors Classifier = 5, max samples = 1.0, max features = 1.0
ii. "Naive Bayes": priors = none, var smoothing = .20
iii. "QDA": priors = None, reg param=0.0
iv. "Random Forest": max depth = 5, n estimators = 100, max features=5,
v. "ID3": Decision Tree – max depth = 5, criterion = "entropy"
vi. "AdaBoost": estimator = None, n estimators = 50, learning rate = 1.0, algorithm = 'SAMME.R', random state = None, base estimator = 'deprecated'
vii. "GradientBoost": n estimators = 100, learning rate=0.1, max depth = 3, random_state = 0.
viii. "MLP": hidden layer sizes=100, max iter = 200
ix. "Nearest Neighbors": n neighbors = 5, weights = 'uniform', algorithm = 'auto', leaf_size=30

Parameters for these were also changed while conducting the experiment but the result was quite similar with slight change in the fraction part. Machine learning implementation is done based on considering the top 5 features weightage for each attack categories.

Table 14. Attack types with top 5 important features

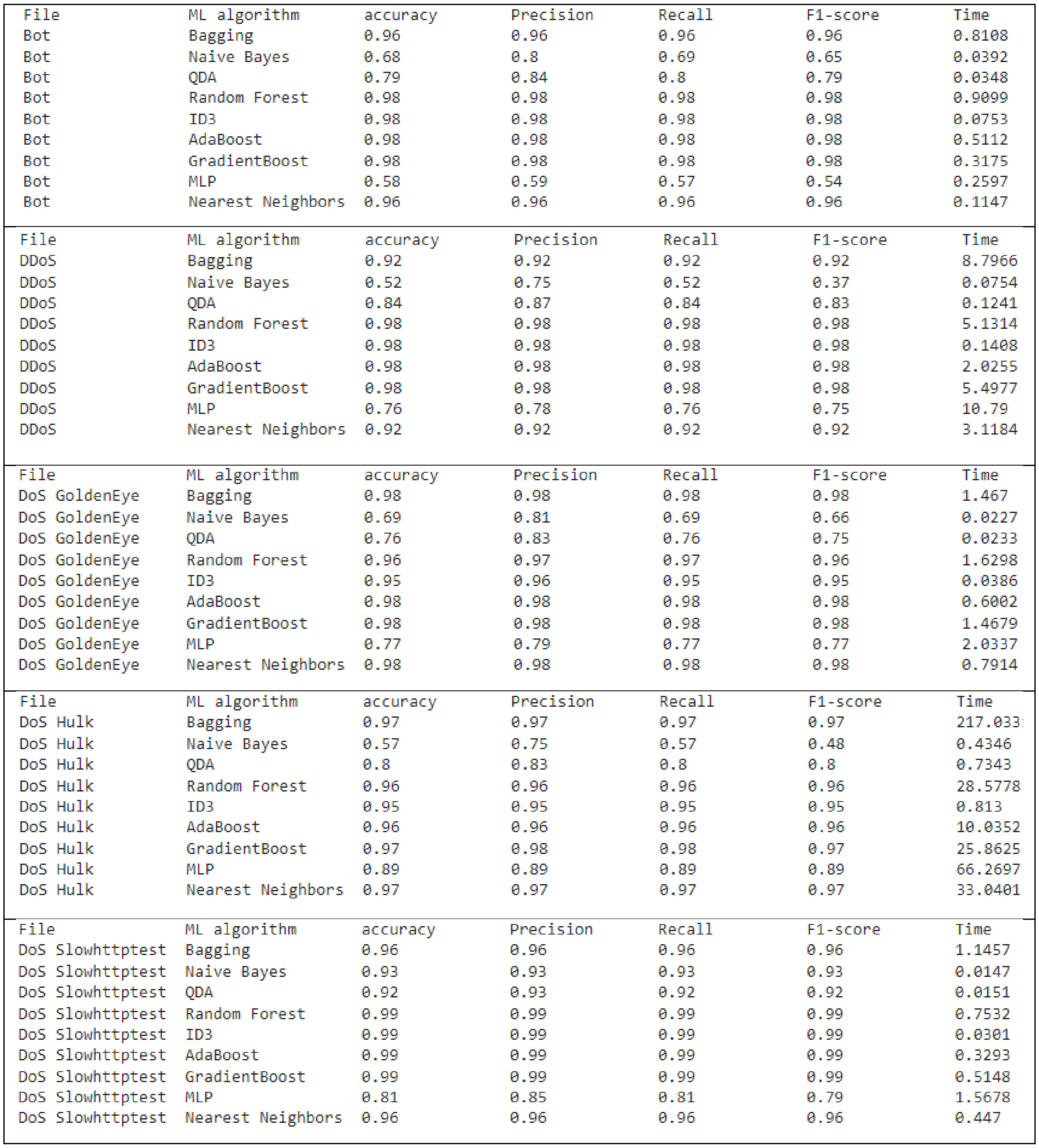
Bot	BwdPacketLengthMean, FlowIATMean, FlowIATStd, Flow IATMin, FlowIATMax
DDoS	FwdPacketLengthMax, TotalLengthofFwdPackets, FwdIAT Total, FlowIATMin, FlowDuration
DoS GoldenEye	FlowIATMax, FlowPackets/s, TotalBackwardPackets, Flow IATMin, FlowIATMean
DoS Hulk	FwdPacketLengthMin, BwdPacketLengthStd, FwdPacket LengthStd, FlowIATMin, FlowBytes/s
DoS Slowhttptest	FlowIATMean, BwdPacketLengthMean, FwdPacketLength Mean, FwdPacketLengthStd, Fwd Packet Length Min
DoS slowloris	Bwd PacketLengthMean, FlowIATStd, FwdPacketLength Min, FwdPacketLengthMax, FwdIATTotal
FTP-Patator	FwdPacketLengthMax, FwdPacketLengthStd, FlowIATMin, BwdPacketLengthStd, BwdPacketLengthMean
Heartbleed	FwdIATTotal, FlowIATStd, FlowIATMax, FlowDuration, Bwd Packet Length Mean
Infiltration	TotalLengthofFwdPackets, FwdPacketLengthStd, FwdPacket LengthMax, FlowDuration, FlowIATMax
PortScan	TotalLengthofFwdPackets, Flow Bytes/s, FlowDuration, Flow IAT Mean, FwdIATTotal
SSH-Patator	FlowIATMin, FwdPacketLengthMax, FlowIATMean, Flow IATMax, FlowPackets/s
Web Attack	BwdPacketLengthStd, BwdPacketLengthMax, FlowIATMin, FlowDuration, FlowIATMax

The provided analysis results show the importance of specific features for different attack types. Each attack type is listed with the corresponding important features. The analysis highlights the specific features that are significant for distinguishing each attack type. These important features can be utilized in developing effective detection and prevention mechanisms for various types of cyber-attacks.

I. Implementation with 50% Benign and 50% attack data with testing and training ratio of 50:50.

In this method, the files and attributes established in the Feature Selection section are used, as well as the attributes retrieved from the same part. Each of these files is labeled after the sort of attack it contains, with just 50% being malicious and 50% being benign. For each form of attack, the nine machine learning approaches are applied with the ten times replications to the same file. Effectiveness and performance of ML approach is the goal of this work.

Table 15. 50% Attack data with 50:50 training and testing data split



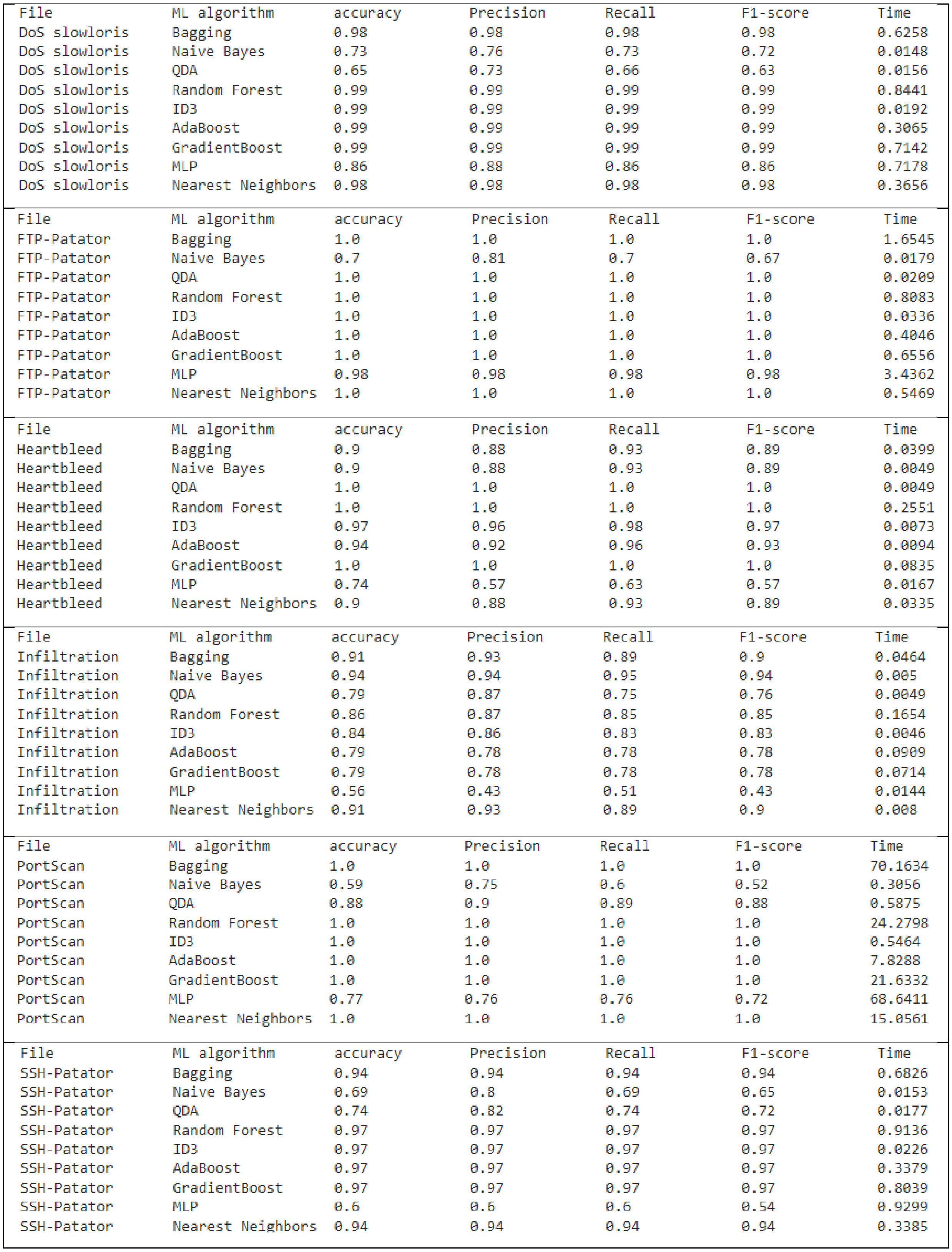
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Bot	Bagging	0.96	0.96	0.96	0.96	0.8108
Bot	Naive Bayes	0.68	0.8	0.69	0.65	0.0392
Bot	QDA	0.79	0.84	0.8	0.79	0.0348
Bot	Random Forest	0.98	0.98	0.98	0.98	0.9099
Bot	ID3	0.98	0.98	0.98	0.98	0.0753
Bot	AdaBoost	0.98	0.98	0.98	0.98	0.5112
Bot	GradientBoost	0.98	0.98	0.98	0.98	0.3175
Bot	MLP	0.58	0.59	0.57	0.54	0.2597
Bot	Nearest Neighbors	0.96	0.96	0.96	0.96	0.1147

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DDoS	Bagging	0.92	0.92	0.92	0.92	8.7966
DDoS	Naive Bayes	0.52	0.75	0.52	0.37	0.0754
DDoS	QDA	0.84	0.87	0.84	0.83	0.1241
DDoS	Random Forest	0.98	0.98	0.98	0.98	5.1314
DDoS	ID3	0.98	0.98	0.98	0.98	0.1408
DDoS	AdaBoost	0.98	0.98	0.98	0.98	2.0255
DDoS	GradientBoost	0.98	0.98	0.98	0.98	5.4977
DDoS	MLP	0.76	0.78	0.76	0.75	10.79
DDoS	Nearest Neighbors	0.92	0.92	0.92	0.92	3.1184

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS GoldenEye	Bagging	0.98	0.98	0.98	0.98	1.467
DoS GoldenEye	Naive Bayes	0.69	0.81	0.69	0.66	0.0227
DoS GoldenEye	QDA	0.76	0.83	0.76	0.75	0.0233
DoS GoldenEye	Random Forest	0.96	0.97	0.97	0.96	1.6298
DoS GoldenEye	ID3	0.95	0.96	0.95	0.95	0.0386
DoS GoldenEye	AdaBoost	0.98	0.98	0.98	0.98	0.6002
DoS GoldenEye	GradientBoost	0.98	0.98	0.98	0.98	1.4679
DoS GoldenEye	MLP	0.77	0.79	0.77	0.77	2.0337
DoS GoldenEye	Nearest Neighbors	0.98	0.98	0.98	0.98	0.7914

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Hulk	Bagging	0.97	0.97	0.97	0.97	217.033
DoS Hulk	Naive Bayes	0.57	0.75	0.57	0.48	0.4346
DoS Hulk	QDA	0.8	0.83	0.8	0.8	0.7343
DoS Hulk	Random Forest	0.96	0.96	0.96	0.96	28.5778
DoS Hulk	ID3	0.95	0.95	0.95	0.95	0.813
DoS Hulk	AdaBoost	0.96	0.96	0.96	0.96	10.0352
DoS Hulk	GradientBoost	0.97	0.98	0.98	0.97	25.8625
DoS Hulk	MLP	0.89	0.89	0.89	0.89	66.2697
DoS Hulk	Nearest Neighbors	0.97	0.97	0.97	0.97	33.0401

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Slowhttptest	Bagging	0.96	0.96	0.96	0.96	1.1457
DoS Slowhttptest	Naive Bayes	0.93	0.93	0.93	0.93	0.0147
DoS Slowhttptest	QDA	0.92	0.93	0.92	0.92	0.0151
DoS Slowhttptest	Random Forest	0.99	0.99	0.99	0.99	0.7532
DoS Slowhttptest	ID3	0.99	0.99	0.99	0.99	0.0301
DoS Slowhttptest	AdaBoost	0.99	0.99	0.99	0.99	0.3293
DoS Slowhttptest	GradientBoost	0.99	0.99	0.99	0.99	0.5148
DoS Slowhttptest	MLP	0.81	0.85	0.81	0.79	1.5678
DoS Slowhttptest	Nearest Neighbors	0.96	0.96	0.96	0.96	0.447



```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
DoS slowloris   Bagging           0.98       0.98        0.98     0.98       0.6258
DoS slowloris   Naive Bayes       0.73       0.76        0.73     0.72       0.0148
DoS slowloris   QDA               0.65       0.73        0.66     0.63       0.0156
DoS slowloris   Random Forest     0.99       0.99        0.99     0.99       0.8441
DoS slowloris   ID3               0.99       0.99        0.99     0.99       0.0192
DoS slowloris   AdaBoost          0.99       0.99        0.99     0.99       0.3065
DoS slowloris   GradientBoost     0.99       0.99        0.99     0.99       0.7142
DoS slowloris   MLP               0.86       0.88        0.86     0.86       0.7178
DoS slowloris   Nearest Neighbors 0.98       0.98        0.98     0.98       0.3656
```

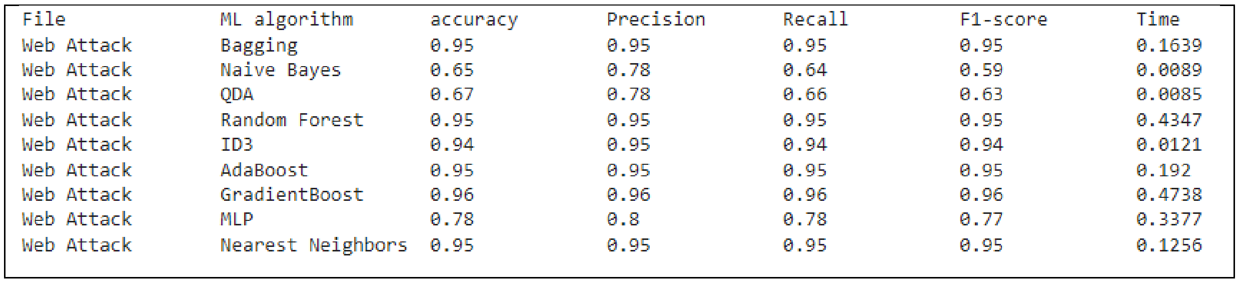
```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
FTP-Patator     Bagging           1.0        1.0         1.0      1.0        1.6545
FTP-Patator     Naive Bayes       0.7        0.81        0.7      0.67       0.0179
FTP-Patator     QDA               1.0        1.0         1.0      1.0        0.0209
FTP-Patator     Random Forest     1.0        1.0         1.0      1.0        0.8083
FTP-Patator     ID3               1.0        1.0         1.0      1.0        0.0336
FTP-Patator     AdaBoost          1.0        1.0         1.0      1.0        0.4046
FTP-Patator     GradientBoost     1.0        1.0         1.0      1.0        0.6556
FTP-Patator     MLP               0.98       0.98        0.98     0.98       3.4362
FTP-Patator     Nearest Neighbors 1.0        1.0         1.0      1.0        0.5469
```

```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
Heartbleed      Bagging           0.9        0.88        0.93     0.89       0.0399
Heartbleed      Naive Bayes       0.9        0.88        0.93     0.89       0.0049
Heartbleed      QDA               1.0        1.0         1.0      1.0        0.0049
Heartbleed      Random Forest     1.0        1.0         1.0      1.0        0.2551
Heartbleed      ID3               0.97       0.96        0.98     0.97       0.0073
Heartbleed      AdaBoost          0.94       0.92        0.96     0.93       0.0094
Heartbleed      GradientBoost     1.0        1.0         1.0      1.0        0.0835
Heartbleed      MLP               0.74       0.57        0.63     0.57       0.0167
Heartbleed      Nearest Neighbors 0.9        0.88        0.93     0.89       0.0335
```

```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
Infiltration    Bagging           0.91       0.93        0.89     0.9        0.0464
Infiltration    Naive Bayes       0.94       0.94        0.95     0.94       0.005
Infiltration    QDA               0.79       0.87        0.75     0.76       0.0049
Infiltration    Random Forest     0.86       0.87        0.85     0.85       0.1654
Infiltration    ID3               0.84       0.86        0.83     0.83       0.0046
Infiltration    AdaBoost          0.79       0.78        0.78     0.78       0.0909
Infiltration    GradientBoost     0.79       0.78        0.78     0.78       0.0714
Infiltration    MLP               0.56       0.43        0.51     0.43       0.0144
Infiltration    Nearest Neighbors 0.91       0.93        0.89     0.9        0.008
```

```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
PortScan        Bagging           1.0        1.0         1.0      1.0        70.1634
PortScan        Naive Bayes       0.59       0.75        0.6      0.52       0.3056
PortScan        QDA               0.88       0.9         0.89     0.88       0.5875
PortScan        Random Forest     1.0        1.0         1.0      1.0        24.2798
PortScan        ID3               1.0        1.0         1.0      1.0        0.5464
PortScan        AdaBoost          1.0        1.0         1.0      1.0        7.8288
PortScan        GradientBoost     1.0        1.0         1.0      1.0        21.6332
PortScan        MLP               0.77       0.76        0.76     0.72       68.6411
PortScan        Nearest Neighbors 1.0        1.0         1.0      1.0        15.0561
```

```
File            ML algorithm      accuracy   Precision   Recall   F1-score   Time
SSH-Patator     Bagging           0.94       0.94        0.94     0.94       0.6826
SSH-Patator     Naive Bayes       0.69       0.8         0.69     0.65       0.0153
SSH-Patator     QDA               0.74       0.82        0.74     0.72       0.0177
SSH-Patator     Random Forest     0.97       0.97        0.97     0.97       0.9136
SSH-Patator     ID3               0.97       0.97        0.97     0.97       0.0226
SSH-Patator     AdaBoost          0.97       0.97        0.97     0.97       0.3379
SSH-Patator     GradientBoost     0.97       0.97        0.97     0.97       0.8039
SSH-Patator     MLP               0.6        0.6         0.6      0.54       0.9299
SSH-Patator     Nearest Neighbors 0.94       0.94        0.94     0.94       0.3385
```



```
File              ML algorithm       accuracy        Precision       Recall          F1-score        Time
Web Attack        Bagging            0.95            0.95            0.95            0.95            0.1639
Web Attack        Naive Bayes        0.65            0.78            0.64            0.59            0.0089
Web Attack        QDA                0.67            0.78            0.66            0.63            0.0085
Web Attack        Random Forest      0.95            0.95            0.95            0.95            0.4347
Web Attack        ID3                0.94            0.95            0.94            0.94            0.0121
Web Attack        AdaBoost           0.95            0.95            0.95            0.95            0.192
Web Attack        GradientBoost      0.96            0.96            0.96            0.96            0.4738
Web Attack        MLP                0.78            0.8             0.78            0.77            0.3377
Web Attack        Nearest Neighbors  0.95            0.95            0.95            0.95            0.1256
```

By analyzing the Table 15 the Naïve Bayes, QDA and MLP algorithm has less accuracy compared to other algorithm. The dataset uses 50% of the random selected data stream with 50% training and 50% testing data split. As there are imbalances in the dataset because each attack is not evenly distributed and not unequal number of instances per attack, so F1-score is best approach to evaluate. Result confirms the data stream selected randomly has even distribution from the result obtained in the accuracy vs. F1-score column. The major differences came in Heartbleed and infiltration attack as total number of attack instance in dataset is only 11 and 36. Based on the given data, here is a summarized result in terms of accuracy, precision, recall, and F1-score:

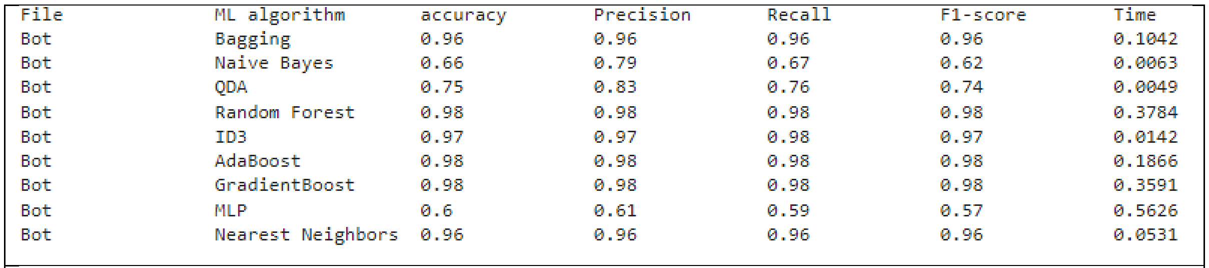
Table 16 Performance and accuracy of ML approach for specific attack

Attack File Considered	**Algorithm Performance**		**Accuracy**	**Precision**	**Recall**	**F1-score**
Bot	Best	AdaBoost	0.983137455	0.983214786	0.983330896	0.983137015
	Worst	Naive Bayes				
DDoS	Best	GradientBoost	0.980777038	0.981257813	0.980750869	0.980771416
	Worst	Naive Bayes				
DoS GoldenEye	Best	GradientBoost	0.982740231	0.982741572	0.982743605	0.982740223
	Worst	Naive Bayes				
DoS Hulk	Best	GradientBoost	0.974996679	0.975051023	0.975092273	0.974996526
	Worst	Naive Bayes				
DoS Slowhttptest	Best	GradientBoost	0.994139194	0.994139194	0.994140255	0.994139191
	Worst	Naive Bayes				
DoS slowloris	Best	GradientBoost	0.994816862	0.994819372	0.994813936	0.994816535
	Worst	QDA				

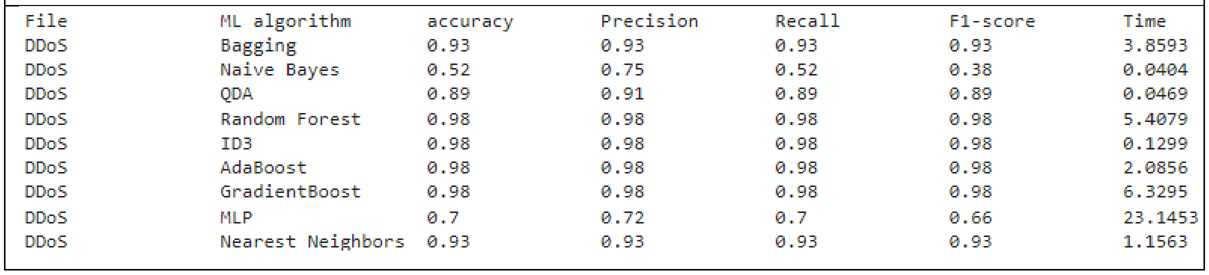
FTP-Patator	Best	Random Forest	0.997764662	0.997762195	0.997767571	0.997764603
	Worst	Naive Bayes				
Heartbleed	Best	Random Forest	1.0	1.0	1.0	1.0
	Worst	MLP				
Infiltration	Best	Naive Bayes	0.939393939	0.9375	0.947368421	0.938888889
	Worst	MLP				
PortScan	Best	Random Forest	0.996501913	0.996522386	0.996476833	0.996498983
	Worst	Naive Bayes				
SSH-Patator	Best	Random Forest	0.972406814	0.972938575	0.97252005	0.972402789
	Worst	MLP				
Web Attack	Best	Bagging	0.949680365	0.949845032	0.949558647	0.949653407
	Worst	Naive Bayes				

II. Implementation with 50% benign and 50% attack data with testing and training ratio of 80:20.

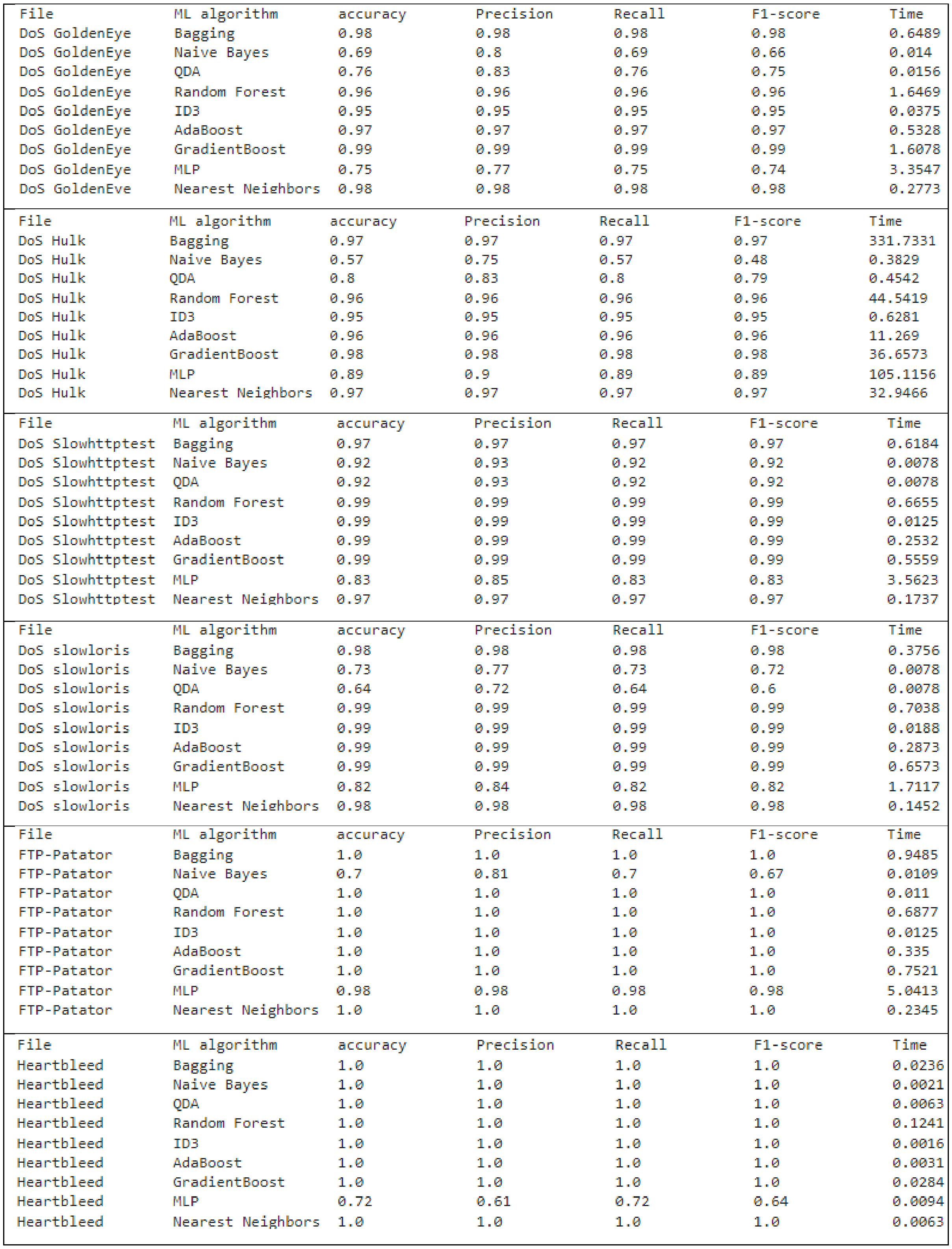
Table 17. 50% Attack data with 80:20 training and testing data split



```
File          ML algorithm       accuracy    Precision    Recall    F1-score    Time
Bot           Bagging            0.96        0.96         0.96      0.96        0.1042
Bot           Naive Bayes        0.66        0.79         0.67      0.62        0.0063
Bot           QDA                0.75        0.83         0.76      0.74        0.0049
Bot           Random Forest      0.98        0.98         0.98      0.98        0.3784
Bot           ID3                0.97        0.97         0.98      0.97        0.0142
Bot           AdaBoost           0.98        0.98         0.98      0.98        0.1866
Bot           GradientBoost      0.98        0.98         0.98      0.98        0.3591
Bot           MLP                0.6         0.61         0.59      0.57        0.5626
Bot           Nearest Neighbors  0.96        0.96         0.96      0.96        0.0531
```



```
File          ML algorithm       accuracy    Precision    Recall    F1-score    Time
DDoS          Bagging            0.93        0.93         0.93      0.93        3.8593
DDoS          Naive Bayes        0.52        0.75         0.52      0.38        0.0404
DDoS          QDA                0.89        0.91         0.89      0.89        0.0469
DDoS          Random Forest      0.98        0.98         0.98      0.98        5.4079
DDoS          ID3                0.98        0.98         0.98      0.98        0.1299
DDoS          AdaBoost           0.98        0.98         0.98      0.98        2.0856
DDoS          GradientBoost      0.98        0.98         0.98      0.98        6.3295
DDoS          MLP                0.7         0.72         0.7       0.66        23.1453
DDoS          Nearest Neighbors  0.93        0.93         0.93      0.93        1.1563
```



```
File            ML algorithm      accuracy    Precision   Recall      F1-score    Time
DoS GoldenEye   Bagging           0.98        0.98        0.98        0.98        0.6489
DoS GoldenEye   Naive Bayes       0.69        0.8         0.69        0.66        0.014
DoS GoldenEye   QDA               0.76        0.83        0.76        0.75        0.0156
DoS GoldenEye   Random Forest     0.96        0.96        0.96        0.96        1.6469
DoS GoldenEye   ID3               0.95        0.95        0.95        0.95        0.0375
DoS GoldenEye   AdaBoost          0.97        0.97        0.97        0.97        0.5328
DoS GoldenEye   GradientBoost     0.99        0.99        0.99        0.99        1.6078
DoS GoldenEye   MLP               0.75        0.77        0.75        0.74        3.3547
DoS GoldenEye   Nearest Neighbors 0.98        0.98        0.98        0.98        0.2773
```

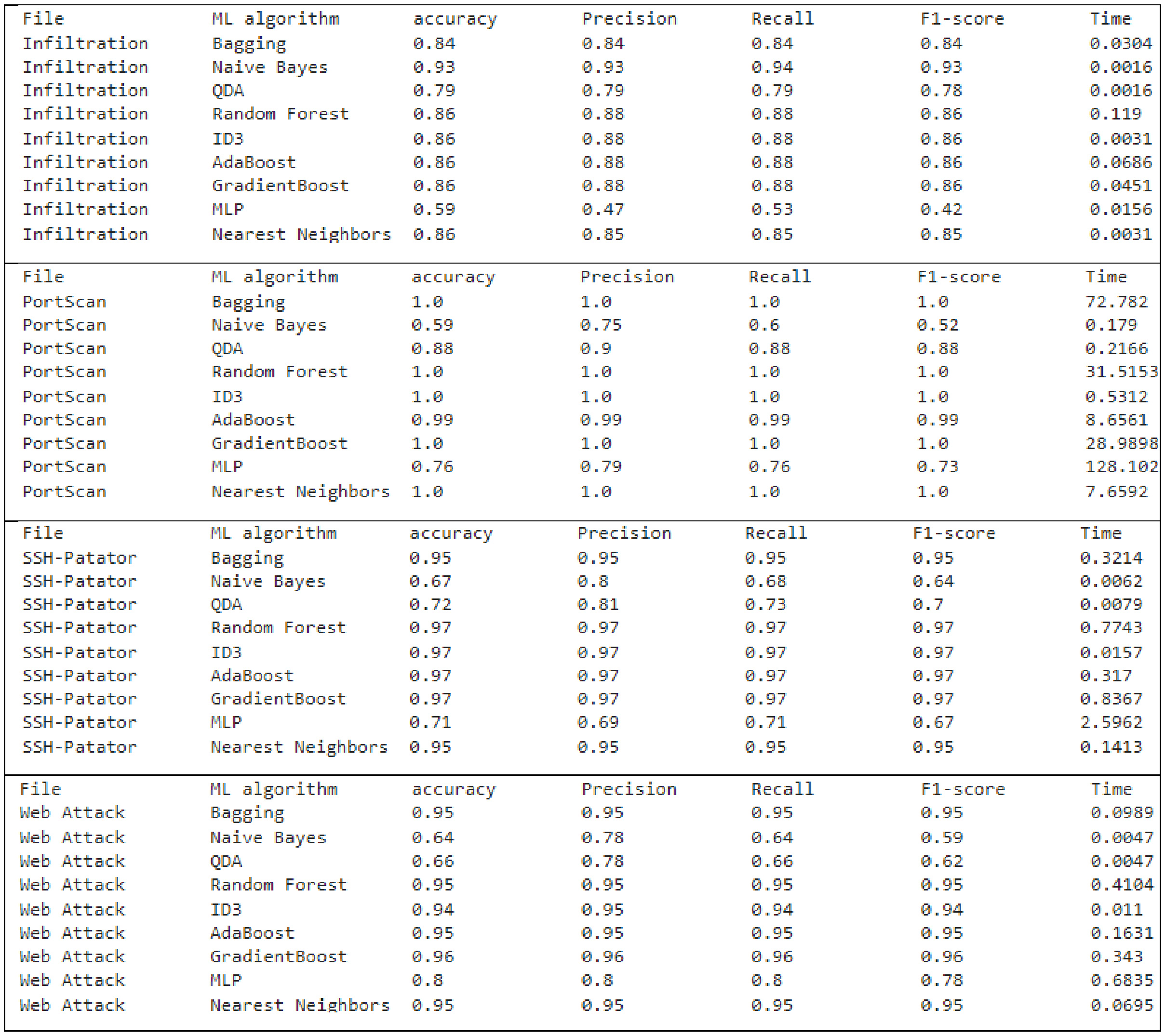
```
File            ML algorithm      accuracy    Precision   Recall      F1-score    Time
DoS Hulk        Bagging           0.97        0.97        0.97        0.97        331.7331
DoS Hulk        Naive Bayes       0.57        0.75        0.57        0.48        0.3829
DoS Hulk        QDA               0.8         0.83        0.8         0.79        0.4542
DoS Hulk        Random Forest     0.96        0.96        0.96        0.96        44.5419
DoS Hulk        ID3               0.95        0.95        0.95        0.95        0.6281
DoS Hulk        AdaBoost          0.96        0.96        0.96        0.96        11.269
DoS Hulk        GradientBoost     0.98        0.98        0.98        0.98        36.6573
DoS Hulk        MLP               0.89        0.9         0.89        0.89        105.1156
DoS Hulk        Nearest Neighbors 0.97        0.97        0.97        0.97        32.9466
```

```
File             ML algorithm      accuracy    Precision   Recall      F1-score    Time
DoS Slowhttptest Bagging           0.97        0.97        0.97        0.97        0.6184
DoS Slowhttptest Naive Bayes       0.92        0.93        0.92        0.92        0.0078
DoS Slowhttptest QDA               0.92        0.93        0.92        0.92        0.0078
DoS Slowhttptest Random Forest     0.99        0.99        0.99        0.99        0.6655
DoS Slowhttptest ID3               0.99        0.99        0.99        0.99        0.0125
DoS Slowhttptest AdaBoost          0.99        0.99        0.99        0.99        0.2532
DoS Slowhttptest GradientBoost     0.99        0.99        0.99        0.99        0.5559
DoS Slowhttptest MLP               0.83        0.85        0.83        0.83        3.5623
DoS Slowhttptest Nearest Neighbors 0.97        0.97        0.97        0.97        0.1737
```

```
File             ML algorithm      accuracy    Precision   Recall      F1-score    Time
DoS slowloris    Bagging           0.98        0.98        0.98        0.98        0.3756
DoS slowloris    Naive Bayes       0.73        0.77        0.73        0.72        0.0078
DoS slowloris    QDA               0.64        0.72        0.64        0.6         0.0078
DoS slowloris    Random Forest     0.99        0.99        0.99        0.99        0.7038
DoS slowloris    ID3               0.99        0.99        0.99        0.99        0.0188
DoS slowloris    AdaBoost          0.99        0.99        0.99        0.99        0.2873
DoS slowloris    GradientBoost     0.99        0.99        0.99        0.99        0.6573
DoS slowloris    MLP               0.82        0.84        0.82        0.82        1.7117
DoS slowloris    Nearest Neighbors 0.98        0.98        0.98        0.98        0.1452
```

```
File             ML algorithm      accuracy    Precision   Recall      F1-score    Time
FTP-Patator      Bagging           1.0         1.0         1.0         1.0         0.9485
FTP-Patator      Naive Bayes       0.7         0.81        0.7         0.67        0.0109
FTP-Patator      QDA               1.0         1.0         1.0         1.0         0.011
FTP-Patator      Random Forest     1.0         1.0         1.0         1.0         0.6877
FTP-Patator      ID3               1.0         1.0         1.0         1.0         0.0125
FTP-Patator      AdaBoost          1.0         1.0         1.0         1.0         0.335
FTP-Patator      GradientBoost     1.0         1.0         1.0         1.0         0.7521
FTP-Patator      MLP               0.98        0.98        0.98        0.98        5.0413
FTP-Patator      Nearest Neighbors 1.0         1.0         1.0         1.0         0.2345
```

```
File             ML algorithm      accuracy    Precision   Recall      F1-score    Time
Heartbleed       Bagging           1.0         1.0         1.0         1.0         0.0236
Heartbleed       Naive Bayes       1.0         1.0         1.0         1.0         0.0021
Heartbleed       QDA               1.0         1.0         1.0         1.0         0.0063
Heartbleed       Random Forest     1.0         1.0         1.0         1.0         0.1241
Heartbleed       ID3               1.0         1.0         1.0         1.0         0.0016
Heartbleed       AdaBoost          1.0         1.0         1.0         1.0         0.0031
Heartbleed       GradientBoost     1.0         1.0         1.0         1.0         0.0284
Heartbleed       MLP               0.72        0.61        0.72        0.64        0.0094
Heartbleed       Nearest Neighbors 1.0         1.0         1.0         1.0         0.0063
```



File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Infiltration	Bagging	0.84	0.84	0.84	0.84	0.0304
Infiltration	Naive Bayes	0.93	0.93	0.94	0.93	0.0016
Infiltration	QDA	0.79	0.79	0.79	0.78	0.0016
Infiltration	Random Forest	0.86	0.88	0.88	0.86	0.119
Infiltration	ID3	0.86	0.88	0.88	0.86	0.0031
Infiltration	AdaBoost	0.86	0.88	0.88	0.86	0.0686
Infiltration	GradientBoost	0.86	0.88	0.88	0.86	0.0451
Infiltration	MLP	0.59	0.47	0.53	0.42	0.0156
Infiltration	Nearest Neighbors	0.86	0.85	0.85	0.85	0.0031

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
PortScan	Bagging	1.0	1.0	1.0	1.0	72.782
PortScan	Naive Bayes	0.59	0.75	0.6	0.52	0.179
PortScan	QDA	0.88	0.9	0.88	0.88	0.2166
PortScan	Random Forest	1.0	1.0	1.0	1.0	31.5153
PortScan	ID3	1.0	1.0	1.0	1.0	0.5312
PortScan	AdaBoost	0.99	0.99	0.99	0.99	8.6561
PortScan	GradientBoost	1.0	1.0	1.0	1.0	28.9898
PortScan	MLP	0.76	0.79	0.76	0.73	128.102
PortScan	Nearest Neighbors	1.0	1.0	1.0	1.0	7.6592

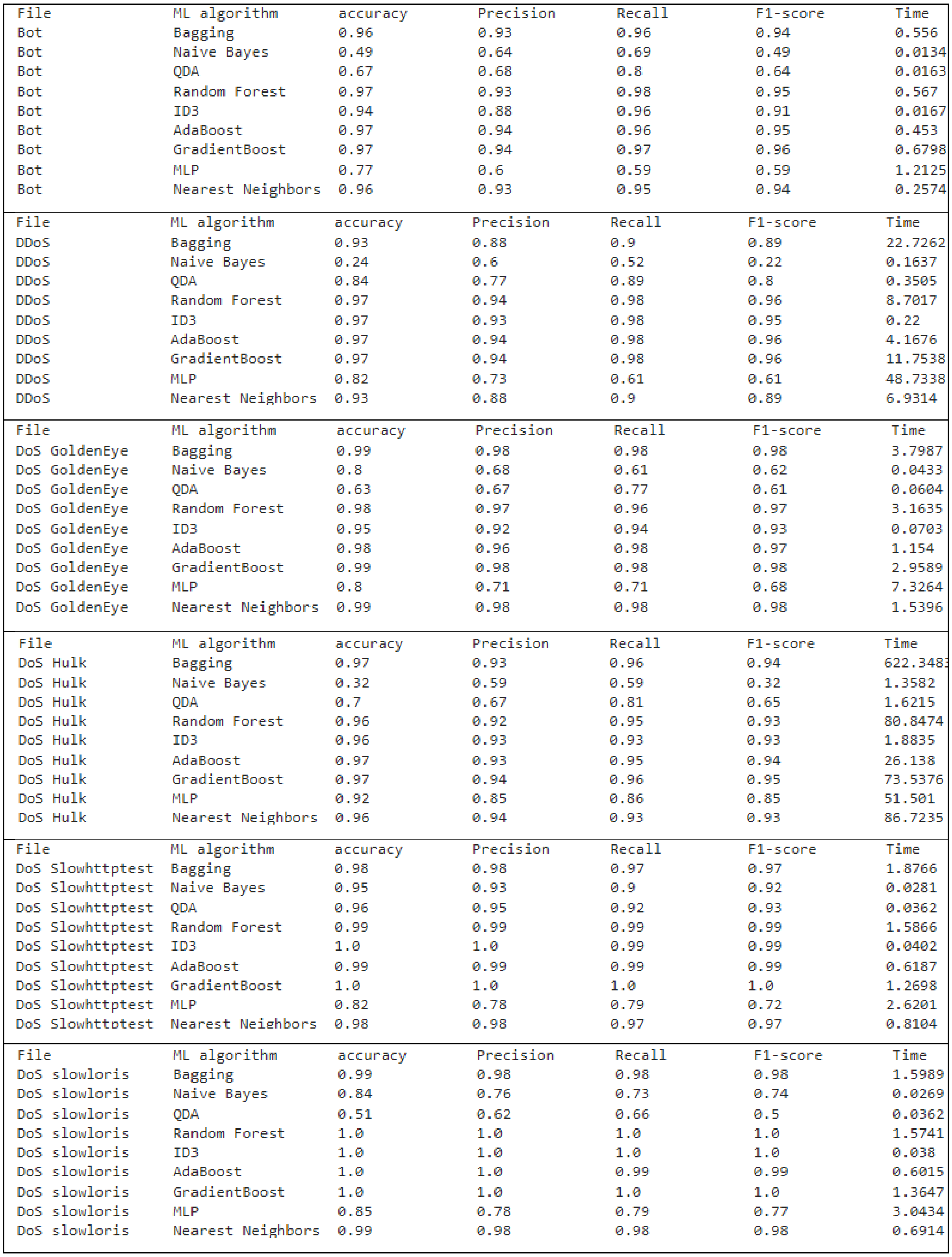
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
SSH-Patator	Bagging	0.95	0.95	0.95	0.95	0.3214
SSH-Patator	Naive Bayes	0.67	0.8	0.68	0.64	0.0062
SSH-Patator	QDA	0.72	0.81	0.73	0.7	0.0079
SSH-Patator	Random Forest	0.97	0.97	0.97	0.97	0.7743
SSH-Patator	ID3	0.97	0.97	0.97	0.97	0.0157
SSH-Patator	AdaBoost	0.97	0.97	0.97	0.97	0.317
SSH-Patator	GradientBoost	0.97	0.97	0.97	0.97	0.8367
SSH-Patator	MLP	0.71	0.69	0.71	0.67	2.5962
SSH-Patator	Nearest Neighbors	0.95	0.95	0.95	0.95	0.1413

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Web Attack	Bagging	0.95	0.95	0.95	0.95	0.0989
Web Attack	Naive Bayes	0.64	0.78	0.64	0.59	0.0047
Web Attack	QDA	0.66	0.78	0.66	0.62	0.0047
Web Attack	Random Forest	0.95	0.95	0.95	0.95	0.4104
Web Attack	ID3	0.94	0.95	0.94	0.94	0.011
Web Attack	AdaBoost	0.95	0.95	0.95	0.95	0.1631
Web Attack	GradientBoost	0.96	0.96	0.96	0.96	0.343
Web Attack	MLP	0.8	0.8	0.8	0.78	0.6835
Web Attack	Nearest Neighbors	0.95	0.95	0.95	0.95	0.0695

Comparing the result of Table 15 and Table 17 it is found that the results are quite similar for all the respective attacks, and it is hardly varying with all the applied machine learning approaches. On analyzing the floating value up to 4 digit it is found that the result on 50:50% training and testing data giving a better result than using 50%:50% training and testing data. Comparing above two table confirms the data stream selected randomly has even distribution from the result obtained in the accuracy vs. F1-score column.

III. Implementation with 80% Benign and 20% attack data with testing and training ratio of 50:50.

Table 18. 80% Attack data with 50:50 training and testing data split



File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Bot	Bagging	0.96	0.93	0.96	0.94	0.556
Bot	Naive Bayes	0.49	0.64	0.69	0.49	0.0134
Bot	QDA	0.67	0.68	0.8	0.64	0.0163
Bot	Random Forest	0.97	0.93	0.98	0.95	0.567
Bot	ID3	0.94	0.88	0.96	0.91	0.0167
Bot	AdaBoost	0.97	0.94	0.96	0.95	0.453
Bot	GradientBoost	0.97	0.94	0.97	0.96	0.6798
Bot	MLP	0.77	0.6	0.59	0.59	1.2125
Bot	Nearest Neighbors	0.96	0.93	0.95	0.94	0.2574

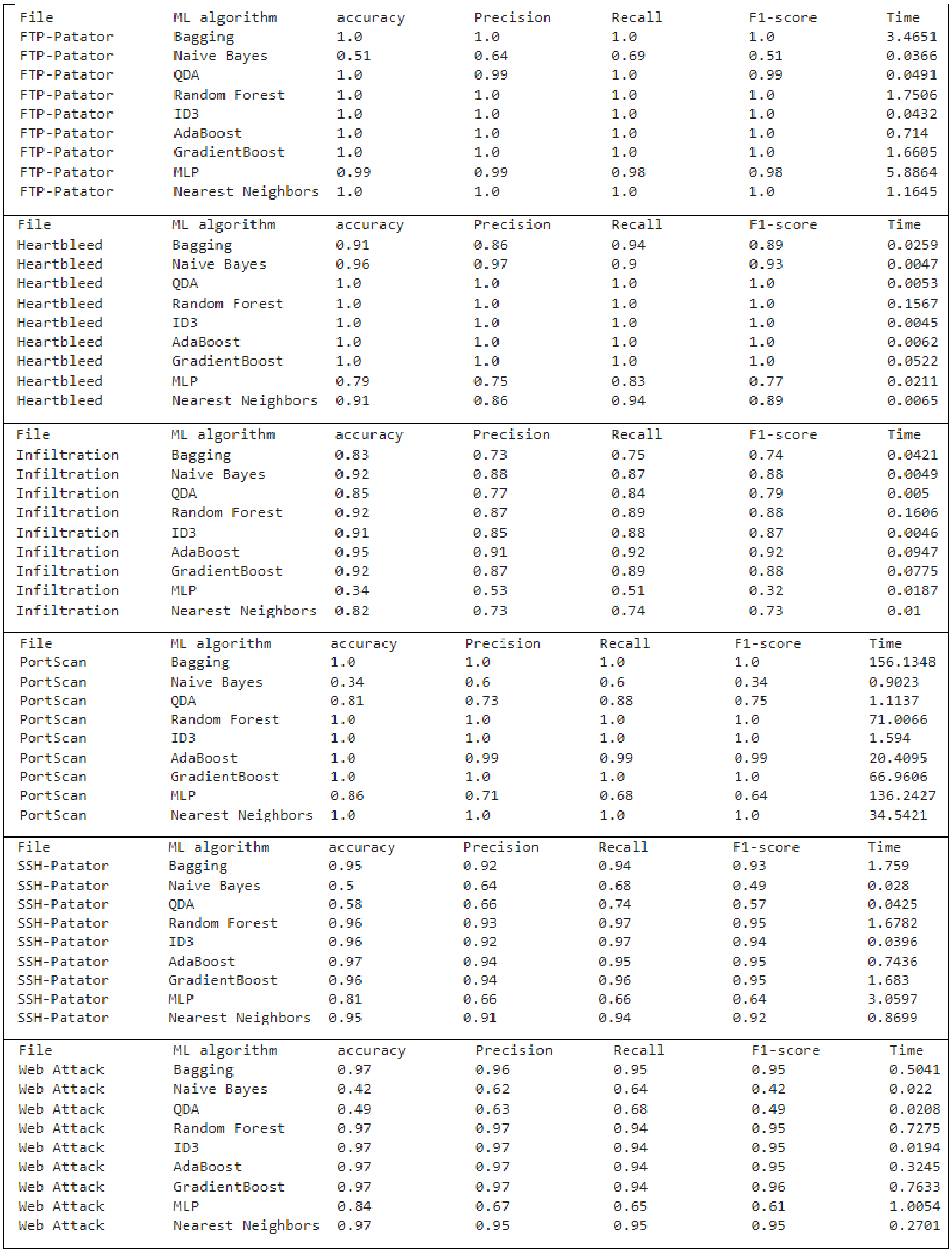
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DDoS	Bagging	0.93	0.88	0.9	0.89	22.7262
DDoS	Naive Bayes	0.24	0.6	0.52	0.22	0.1637
DDoS	QDA	0.84	0.77	0.89	0.8	0.3505
DDoS	Random Forest	0.97	0.94	0.98	0.96	8.7017
DDoS	ID3	0.97	0.93	0.98	0.95	0.22
DDoS	AdaBoost	0.97	0.94	0.98	0.96	4.1676
DDoS	GradientBoost	0.97	0.94	0.98	0.96	11.7538
DDoS	MLP	0.82	0.73	0.61	0.61	48.7338
DDoS	Nearest Neighbors	0.93	0.88	0.9	0.89	6.9314

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS GoldenEye	Bagging	0.99	0.98	0.98	0.98	3.7987
DoS GoldenEye	Naive Bayes	0.8	0.68	0.61	0.62	0.0433
DoS GoldenEye	QDA	0.63	0.67	0.77	0.61	0.0604
DoS GoldenEye	Random Forest	0.98	0.97	0.96	0.97	3.1635
DoS GoldenEye	ID3	0.95	0.92	0.94	0.93	0.0703
DoS GoldenEye	AdaBoost	0.98	0.96	0.98	0.97	1.154
DoS GoldenEye	GradientBoost	0.99	0.98	0.98	0.98	2.9589
DoS GoldenEye	MLP	0.8	0.71	0.71	0.68	7.3264
DoS GoldenEye	Nearest Neighbors	0.99	0.98	0.98	0.98	1.5396

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Hulk	Bagging	0.97	0.93	0.96	0.94	622.348
DoS Hulk	Naive Bayes	0.32	0.59	0.59	0.32	1.3582
DoS Hulk	QDA	0.7	0.67	0.81	0.65	1.6215
DoS Hulk	Random Forest	0.96	0.92	0.95	0.93	80.8474
DoS Hulk	ID3	0.96	0.93	0.93	0.93	1.8835
DoS Hulk	AdaBoost	0.97	0.93	0.95	0.94	26.138
DoS Hulk	GradientBoost	0.97	0.94	0.96	0.95	73.5376
DoS Hulk	MLP	0.92	0.85	0.86	0.85	51.501
DoS Hulk	Nearest Neighbors	0.96	0.94	0.93	0.93	86.7235

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Slowhttptest	Bagging	0.98	0.98	0.97	0.97	1.8766
DoS Slowhttptest	Naive Bayes	0.95	0.93	0.9	0.92	0.0281
DoS Slowhttptest	QDA	0.96	0.95	0.92	0.93	0.0362
DoS Slowhttptest	Random Forest	0.99	0.99	0.99	0.99	1.5866
DoS Slowhttptest	ID3	1.0	1.0	0.99	0.99	0.0402
DoS Slowhttptest	AdaBoost	0.99	0.99	0.99	0.99	0.6187
DoS Slowhttptest	GradientBoost	1.0	1.0	1.0	1.0	1.2698
DoS Slowhttptest	MLP	0.82	0.78	0.79	0.72	2.6201
DoS Slowhttptest	Nearest Neighbors	0.98	0.98	0.97	0.97	0.8104

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS slowloris	Bagging	0.99	0.98	0.98	0.98	1.5989
DoS slowloris	Naive Bayes	0.84	0.76	0.73	0.74	0.0269
DoS slowloris	QDA	0.51	0.62	0.66	0.5	0.0362
DoS slowloris	Random Forest	1.0	1.0	1.0	1.0	1.5741
DoS slowloris	ID3	1.0	1.0	1.0	1.0	0.038
DoS slowloris	AdaBoost	1.0	1.0	0.99	0.99	0.6015
DoS slowloris	GradientBoost	1.0	1.0	1.0	1.0	1.3647
DoS slowloris	MLP	0.85	0.78	0.79	0.77	3.0434
DoS slowloris	Nearest Neighbors	0.99	0.98	0.98	0.98	0.6914



```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
FTP-Patator   Bagging           1.0        1.0         1.0      1.0        3.4651
FTP-Patator   Naive Bayes       0.51       0.64        0.69     0.51       0.0366
FTP-Patator   QDA               1.0        0.99        1.0      0.99       0.0491
FTP-Patator   Random Forest     1.0        1.0         1.0      1.0        1.7506
FTP-Patator   ID3               1.0        1.0         1.0      1.0        0.0432
FTP-Patator   AdaBoost          1.0        1.0         1.0      1.0        0.714
FTP-Patator   GradientBoost     1.0        1.0         1.0      1.0        1.6605
FTP-Patator   MLP               0.99       0.99        0.98     0.98       5.8864
FTP-Patator   Nearest Neighbors 1.0        1.0         1.0      1.0        1.1645
```

```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
Heartbleed    Bagging           0.91       0.86        0.94     0.89       0.0259
Heartbleed    Naive Bayes       0.96       0.97        0.9      0.93       0.0047
Heartbleed    QDA               1.0        1.0         1.0      1.0        0.0053
Heartbleed    Random Forest     1.0        1.0         1.0      1.0        0.1567
Heartbleed    ID3               1.0        1.0         1.0      1.0        0.0045
Heartbleed    AdaBoost          1.0        1.0         1.0      1.0        0.0062
Heartbleed    GradientBoost     1.0        1.0         1.0      1.0        0.0522
Heartbleed    MLP               0.79       0.75        0.83     0.77       0.0211
Heartbleed    Nearest Neighbors 0.91       0.86        0.94     0.89       0.0065
```

```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
Infiltration  Bagging           0.83       0.73        0.75     0.74       0.0421
Infiltration  Naive Bayes       0.92       0.88        0.87     0.88       0.0049
Infiltration  QDA               0.85       0.77        0.84     0.79       0.005
Infiltration  Random Forest     0.92       0.87        0.89     0.88       0.1606
Infiltration  ID3               0.91       0.85        0.88     0.87       0.0046
Infiltration  AdaBoost          0.95       0.91        0.92     0.92       0.0947
Infiltration  GradientBoost     0.92       0.87        0.89     0.88       0.0775
Infiltration  MLP               0.34       0.53        0.51     0.32       0.0187
Infiltration  Nearest Neighbors 0.82       0.73        0.74     0.73       0.01
```

```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
PortScan      Bagging           1.0        1.0         1.0      1.0        156.1348
PortScan      Naive Bayes       0.34       0.6         0.6      0.34       0.9023
PortScan      QDA               0.81       0.73        0.88     0.75       1.1137
PortScan      Random Forest     1.0        1.0         1.0      1.0        71.0066
PortScan      ID3               1.0        1.0         1.0      1.0        1.594
PortScan      AdaBoost          1.0        0.99        0.99     0.99       20.4095
PortScan      GradientBoost     1.0        1.0         1.0      1.0        66.9606
PortScan      MLP               0.86       0.71        0.68     0.64       136.2427
PortScan      Nearest Neighbors 1.0        1.0         1.0      1.0        34.5421
```

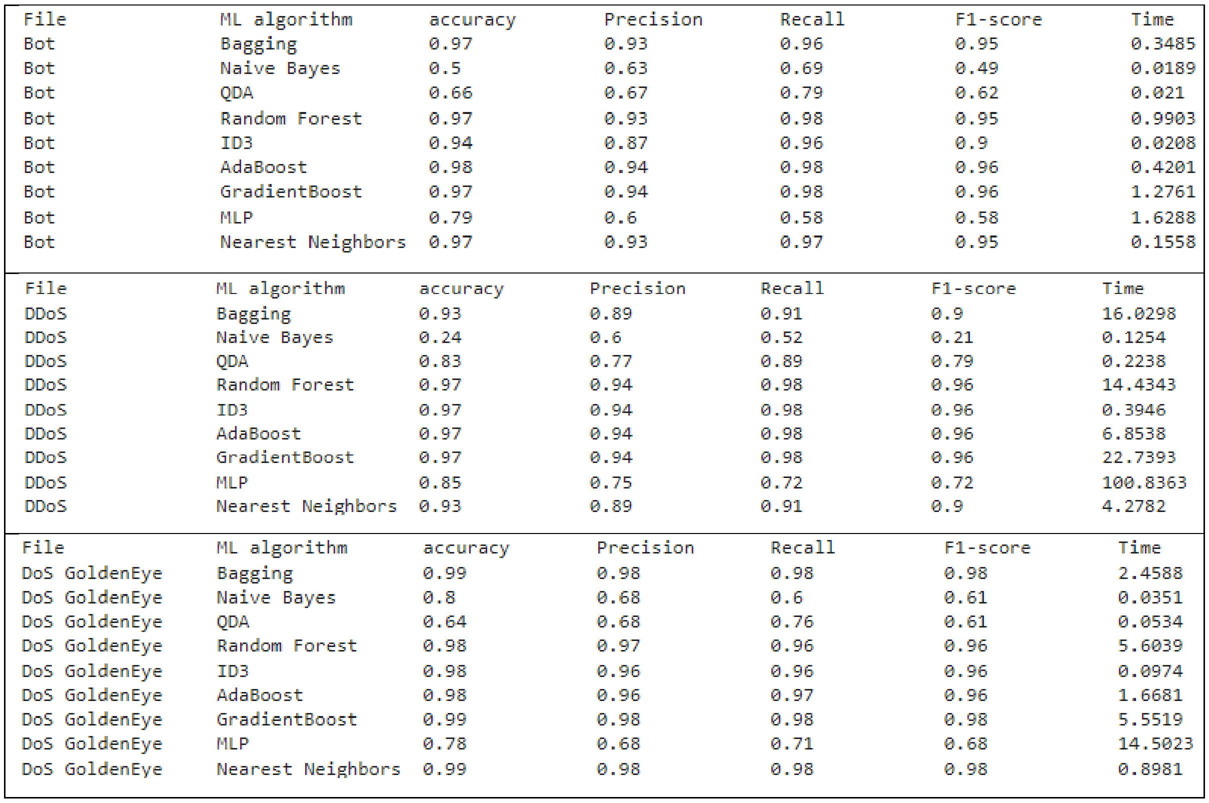
```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
SSH-Patator   Bagging           0.95       0.92        0.94     0.93       1.759
SSH-Patator   Naive Bayes       0.5        0.64        0.68     0.49       0.028
SSH-Patator   QDA               0.58       0.66        0.74     0.57       0.0425
SSH-Patator   Random Forest     0.96       0.93        0.97     0.95       1.6782
SSH-Patator   ID3               0.96       0.92        0.97     0.94       0.0396
SSH-Patator   AdaBoost          0.97       0.94        0.95     0.95       0.7436
SSH-Patator   GradientBoost     0.96       0.94        0.96     0.95       1.683
SSH-Patator   MLP               0.81       0.66        0.66     0.64       3.0597
SSH-Patator   Nearest Neighbors 0.95       0.91        0.94     0.92       0.8699
```

```
File          ML algorithm      accuracy   Precision   Recall   F1-score   Time
Web Attack    Bagging           0.97       0.96        0.95     0.95       0.5041
Web Attack    Naive Bayes       0.42       0.62        0.64     0.42       0.022
Web Attack    QDA               0.49       0.63        0.68     0.49       0.0208
Web Attack    Random Forest     0.97       0.97        0.94     0.95       0.7275
Web Attack    ID3               0.97       0.97        0.94     0.95       0.0194
Web Attack    AdaBoost          0.97       0.97        0.94     0.95       0.3245
Web Attack    GradientBoost     0.97       0.97        0.94     0.96       0.7633
Web Attack    MLP               0.84       0.67        0.65     0.61       1.0054
Web Attack    Nearest Neighbors 0.97       0.95        0.95     0.95       0.2701
```

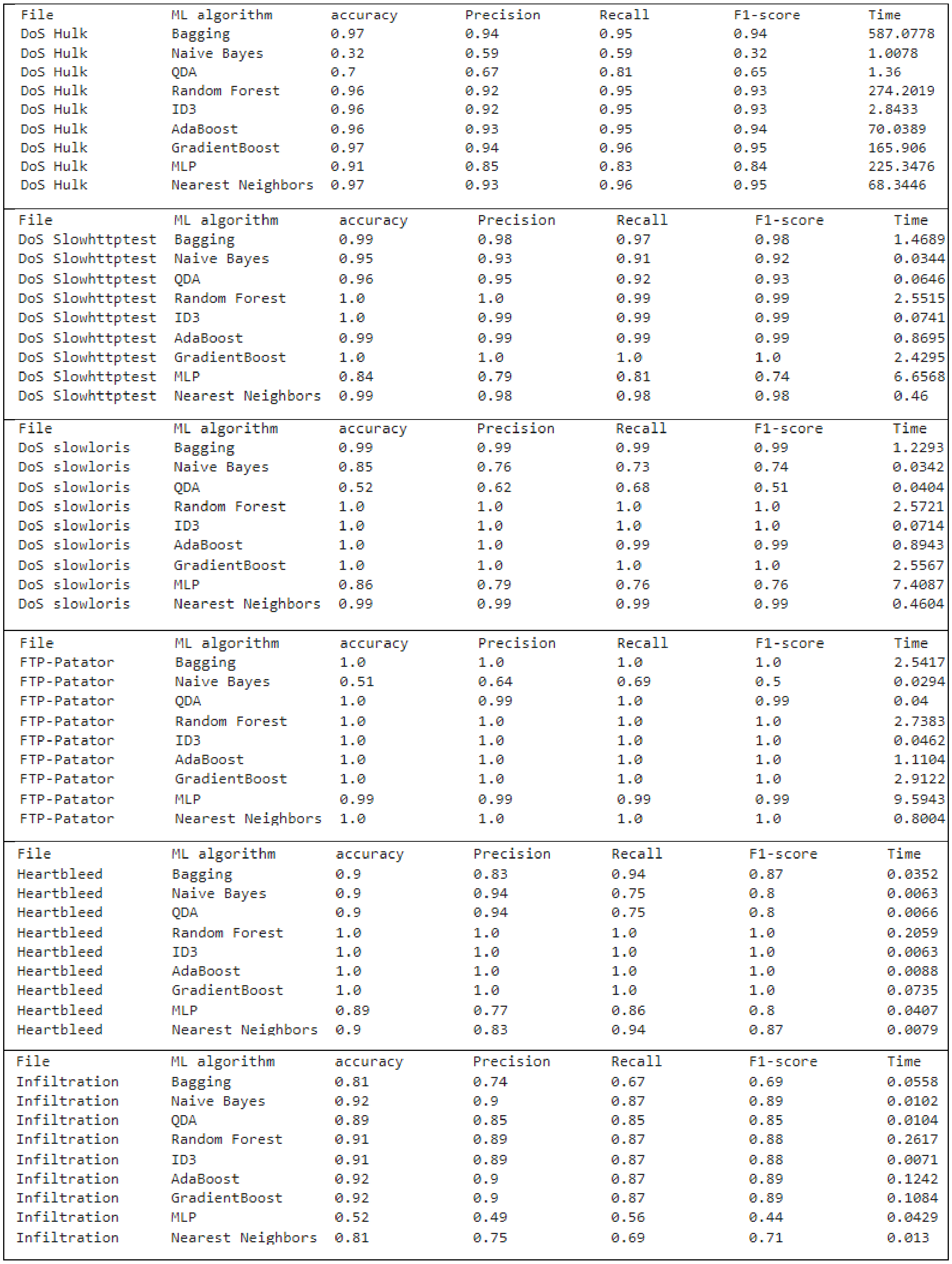
By analyzing the Table 18 similar result is achieved as Naïve Bayes, QDA and MLP algorithm has less accuracy compared to other algorithm. The accuracy of these attacks is calculated based on the considering best 5 features selected for each attack using feature grouping approach. The result obtained outperformed compared with Table 15 and Table 17. The dataset uses 80% of the random selected data stream with 50% training and 50% testing data split. The result confirms the data stream selected randomly has even distribution from the result obtained in the accuracy vs. F1-score column. In this experiment also the major differences came in Heartbleed and infiltration attack. This result can be improved by including a greater number of features for identifying selective attacks, but it will result in degradation in the performance of the system.

IV. Implementation with 80% Benign and 20% attack data with testing and training ratio of 80:20.

Table 19. 80% Attack data with 80:20 training and testing data split



File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Bot	Bagging	0.97	0.93	0.96	0.95	0.3485
Bot	Naive Bayes	0.5	0.63	0.69	0.49	0.0189
Bot	QDA	0.66	0.67	0.79	0.62	0.021
Bot	Random Forest	0.97	0.93	0.98	0.95	0.9903
Bot	ID3	0.94	0.87	0.96	0.9	0.0208
Bot	AdaBoost	0.98	0.94	0.98	0.96	0.4201
Bot	GradientBoost	0.97	0.94	0.98	0.96	1.2761
Bot	MLP	0.79	0.6	0.58	0.58	1.6288
Bot	Nearest Neighbors	0.97	0.93	0.97	0.95	0.1558
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DDoS	Bagging	0.93	0.89	0.91	0.9	16.0298
DDoS	Naive Bayes	0.24	0.6	0.52	0.21	0.1254
DDoS	QDA	0.83	0.77	0.89	0.79	0.2238
DDoS	Random Forest	0.97	0.94	0.98	0.96	14.4343
DDoS	ID3	0.97	0.94	0.98	0.96	0.3946
DDoS	AdaBoost	0.97	0.94	0.98	0.96	6.8538
DDoS	GradientBoost	0.97	0.94	0.98	0.96	22.7393
DDoS	MLP	0.85	0.75	0.72	0.72	100.8363
DDoS	Nearest Neighbors	0.93	0.89	0.91	0.9	4.2782
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS GoldenEye	Bagging	0.99	0.98	0.98	0.98	2.4588
DoS GoldenEye	Naive Bayes	0.8	0.68	0.6	0.61	0.0351
DoS GoldenEye	QDA	0.64	0.68	0.76	0.61	0.0534
DoS GoldenEye	Random Forest	0.98	0.97	0.96	0.96	5.6039
DoS GoldenEye	ID3	0.98	0.96	0.96	0.96	0.0974
DoS GoldenEye	AdaBoost	0.98	0.96	0.97	0.96	1.6681
DoS GoldenEye	GradientBoost	0.99	0.98	0.98	0.98	5.5519
DoS GoldenEye	MLP	0.78	0.68	0.71	0.68	14.5023
DoS GoldenEye	Nearest Neighbors	0.99	0.98	0.98	0.98	0.8981



File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Hulk	Bagging	0.97	0.94	0.95	0.94	587.0778
DoS Hulk	Naive Bayes	0.32	0.59	0.59	0.32	1.0078
DoS Hulk	QDA	0.7	0.67	0.81	0.65	1.36
DoS Hulk	Random Forest	0.96	0.92	0.95	0.93	274.2019
DoS Hulk	ID3	0.96	0.92	0.95	0.93	2.8433
DoS Hulk	AdaBoost	0.96	0.93	0.95	0.94	70.0389
DoS Hulk	GradientBoost	0.97	0.94	0.96	0.95	165.906
DoS Hulk	MLP	0.91	0.85	0.83	0.84	225.3476
DoS Hulk	Nearest Neighbors	0.97	0.93	0.96	0.95	68.3446

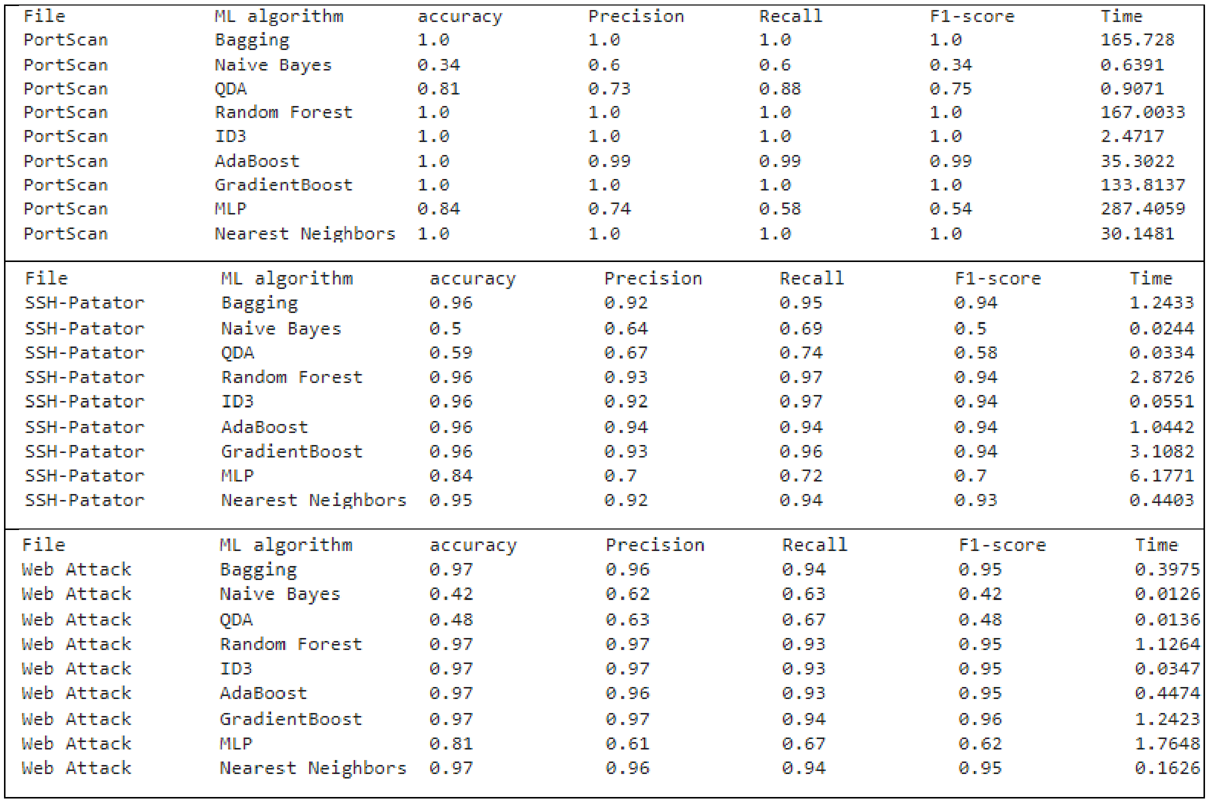
File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS Slowhttptest	Bagging	0.99	0.98	0.97	0.98	1.4689
DoS Slowhttptest	Naive Bayes	0.95	0.93	0.91	0.92	0.0344
DoS Slowhttptest	QDA	0.96	0.95	0.92	0.93	0.0646
DoS Slowhttptest	Random Forest	1.0	1.0	0.99	0.99	2.5515
DoS Slowhttptest	ID3	1.0	0.99	0.99	0.99	0.0741
DoS Slowhttptest	AdaBoost	0.99	0.99	0.99	0.99	0.8695
DoS Slowhttptest	GradientBoost	1.0	1.0	1.0	1.0	2.4295
DoS Slowhttptest	MLP	0.84	0.79	0.81	0.74	6.6568
DoS Slowhttptest	Nearest Neighbors	0.99	0.98	0.98	0.98	0.46

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
DoS slowloris	Bagging	0.99	0.99	0.99	0.99	1.2293
DoS slowloris	Naive Bayes	0.85	0.76	0.73	0.74	0.0342
DoS slowloris	QDA	0.52	0.62	0.68	0.51	0.0404
DoS slowloris	Random Forest	1.0	1.0	1.0	1.0	2.5721
DoS slowloris	ID3	1.0	1.0	1.0	1.0	0.0714
DoS slowloris	AdaBoost	1.0	1.0	0.99	0.99	0.8943
DoS slowloris	GradientBoost	1.0	1.0	1.0	1.0	2.5567
DoS slowloris	MLP	0.86	0.79	0.76	0.76	7.4087
DoS slowloris	Nearest Neighbors	0.99	0.99	0.99	0.99	0.4604

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
FTP-Patator	Bagging	1.0	1.0	1.0	1.0	2.5417
FTP-Patator	Naive Bayes	0.51	0.64	0.69	0.5	0.0294
FTP-Patator	QDA	1.0	0.99	1.0	0.99	0.04
FTP-Patator	Random Forest	1.0	1.0	1.0	1.0	2.7383
FTP-Patator	ID3	1.0	1.0	1.0	1.0	0.0462
FTP-Patator	AdaBoost	1.0	1.0	1.0	1.0	1.1104
FTP-Patator	GradientBoost	1.0	1.0	1.0	1.0	2.9122
FTP-Patator	MLP	0.99	0.99	0.99	0.99	9.5943
FTP-Patator	Nearest Neighbors	1.0	1.0	1.0	1.0	0.8004

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Heartbleed	Bagging	0.9	0.83	0.94	0.87	0.0352
Heartbleed	Naive Bayes	0.9	0.94	0.75	0.8	0.0063
Heartbleed	QDA	0.9	0.94	0.75	0.8	0.0066
Heartbleed	Random Forest	1.0	1.0	1.0	1.0	0.2059
Heartbleed	ID3	1.0	1.0	1.0	1.0	0.0063
Heartbleed	AdaBoost	1.0	1.0	1.0	1.0	0.0088
Heartbleed	GradientBoost	1.0	1.0	1.0	1.0	0.0735
Heartbleed	MLP	0.89	0.77	0.86	0.8	0.0407
Heartbleed	Nearest Neighbors	0.9	0.83	0.94	0.87	0.0079

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Infiltration	Bagging	0.81	0.74	0.67	0.69	0.0558
Infiltration	Naive Bayes	0.92	0.9	0.87	0.89	0.0102
Infiltration	QDA	0.89	0.85	0.85	0.85	0.0104
Infiltration	Random Forest	0.91	0.89	0.87	0.88	0.2617
Infiltration	ID3	0.91	0.89	0.87	0.88	0.0071
Infiltration	AdaBoost	0.92	0.9	0.87	0.89	0.1242
Infiltration	GradientBoost	0.92	0.9	0.87	0.89	0.1084
Infiltration	MLP	0.52	0.49	0.56	0.44	0.0429
Infiltration	Nearest Neighbors	0.81	0.75	0.69	0.71	0.013



File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
PortScan	Bagging	1.0	1.0	1.0	1.0	165.728
PortScan	Naive Bayes	0.34	0.6	0.6	0.34	0.6391
PortScan	QDA	0.81	0.73	0.88	0.75	0.9071
PortScan	Random Forest	1.0	1.0	1.0	1.0	167.0033
PortScan	ID3	1.0	1.0	1.0	1.0	2.4717
PortScan	AdaBoost	1.0	0.99	0.99	0.99	35.3022
PortScan	GradientBoost	1.0	1.0	1.0	1.0	133.8137
PortScan	MLP	0.84	0.74	0.58	0.54	287.4059
PortScan	Nearest Neighbors	1.0	1.0	1.0	1.0	30.1481

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
SSH-Patator	Bagging	0.96	0.92	0.95	0.94	1.2433
SSH-Patator	Naive Bayes	0.5	0.64	0.69	0.5	0.0244
SSH-Patator	QDA	0.59	0.67	0.74	0.58	0.0334
SSH-Patator	Random Forest	0.96	0.93	0.97	0.94	2.8726
SSH-Patator	ID3	0.96	0.92	0.97	0.94	0.0551
SSH-Patator	AdaBoost	0.96	0.94	0.94	0.94	1.0442
SSH-Patator	GradientBoost	0.96	0.93	0.96	0.94	3.1082
SSH-Patator	MLP	0.84	0.7	0.72	0.7	6.1771
SSH-Patator	Nearest Neighbors	0.95	0.92	0.94	0.93	0.4403

File	ML algorithm	accuracy	Precision	Recall	F1-score	Time
Web Attack	Bagging	0.97	0.96	0.94	0.95	0.3975
Web Attack	Naive Bayes	0.42	0.62	0.63	0.42	0.0126
Web Attack	QDA	0.48	0.63	0.67	0.48	0.0136
Web Attack	Random Forest	0.97	0.97	0.93	0.95	1.1264
Web Attack	ID3	0.97	0.97	0.93	0.95	0.0347
Web Attack	AdaBoost	0.97	0.96	0.93	0.95	0.4474
Web Attack	GradientBoost	0.97	0.97	0.94	0.96	1.2423
Web Attack	MLP	0.81	0.61	0.67	0.62	1.7648
Web Attack	Nearest Neighbors	0.97	0.96	0.94	0.95	0.1626

By analyzing the Table 19 it can be seen that Naïve Bayes has outperformed but QDA and MLP algorithm has gained better accuracy by referring Table 15, Table 17 and Table 18. The accuracy of these attacks is calculated based on the considering best 5 features selected for each attack using feature grouping approach. The dataset uses 80% of the random selected data stream with 80% training and 20% testing data split.

Based on this analysis, it is recommended to use the Bagging algorithm for DDoS and Web Attack files, Random Forest for DoS Slowhttptest, ID3 for DoS slowloris, and AdaBoost for FTP-Patator. The selection of algorithms relies on the particular dataset and the trade-off between performance and execution time.

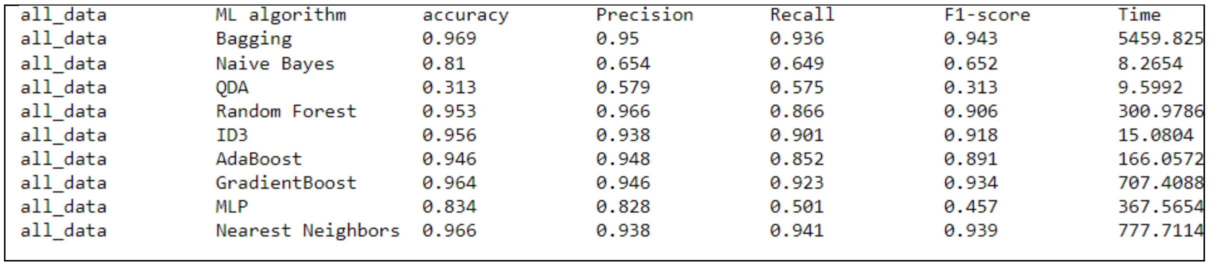
All the experiment is done by selecting individual top 5 features that are obtained from each of the 12 attack types. As each 12 attacks are based on their own individual features so combining all together result in a list of features containing 60 attributes. By removing redundancy, only 17 features represent most important attack attributes as shown in Table 20:

Table 20. 17 top features after aggregating best 5 features from each attack category.

Bwd Packet Length Mean	Total Length of Fwd Packets	Bwd Packet Length Std
Flow IAT Mean	Fwd IAT Total	Fwd Packet Length Std
Flow IAT Std	Flow Duration	Flow Bytes/s
Flow IAT Min	Flow Packets/s	Fwd Packet Length Mean
Flow IAT Max	Total Backward Packets	Bwd Packet Length Max
Fwd Packet Length Max	Fwd Packet Length Min	

The last part of the experiment is to use all these 17 features and apply in the 100% dataset and 80% dataset with training and testing ratio of 80:20. The training and testing ratio is considered based on the best result achieved through 50% and 80% data consideration with respective testing and training ratio.

Table 21. 100% dataset consideration with 50%:50 training and testing data split



all_data	ML algorithm	accuracy	Precision	Recall	F1-score	Time
all_data	Bagging	0.969	0.95	0.936	0.943	5459.825
all_data	Naive Bayes	0.81	0.654	0.649	0.652	8.2654
all_data	QDA	0.313	0.579	0.575	0.313	9.5992
all_data	Random Forest	0.953	0.966	0.866	0.906	300.9786
all_data	ID3	0.956	0.938	0.901	0.918	15.0804
all_data	AdaBoost	0.946	0.948	0.852	0.891	166.0572
all_data	GradientBoost	0.964	0.946	0.923	0.934	707.4088
all_data	MLP	0.834	0.828	0.501	0.457	367.5654
all_data	Nearest Neighbors	0.966	0.938	0.941	0.939	777.7114

This is the overall outcome of the experiment where complete dataset is taken with 50%:50 training and testing data split. Result obtained has much better accuracy compared to conducting experiment using just 5 features for each individual attack. With increasing number of attacks to 17 the result is much better using all the selected algorithm except QDA. Detection of attack using Naïve Bayes and MLP has also increased comparatively. In Table 23 a complete dataset of CICIDS 2017 is taken with 50% training and 50% testing data split.

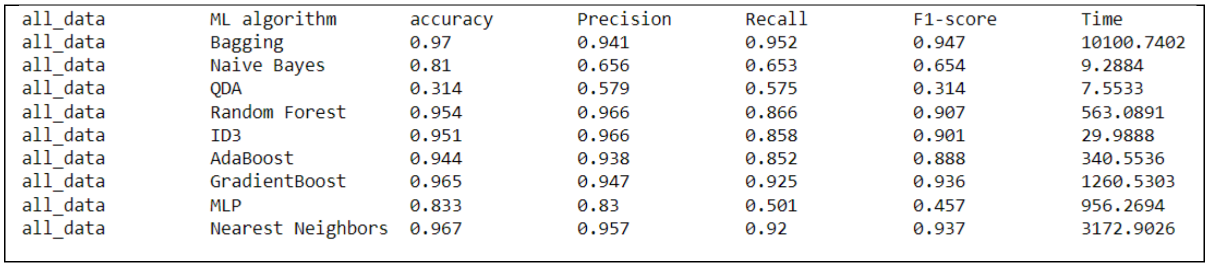
Based on the provided data, the performance metrics for different ML algorithms on complete dataset with the "all_data" file is shown in the table below with respect to the true positive rate and its execution time:

Table 22. Performance on 100% dataset of 50:50 training and testing ratio

ML Algorithm	True Positive Rate	Execution Time
Bagging	High	High
Naïve Bayes	Moderate	Low
QDA	Low	Low
Random Forest	High	High
ID3	High	Moderate
AdaBoost	Moderate	High
GraidentBoost	High	High
MLP	Moderate	High
Nearest Neighbors	High	High

Based on this analysis, the Bagging, Random Forest, and GradientBoost algorithms perform well in terms of accuracy, precision, recall, and F1-score. However, they have higher execution times compared to other algorithms. Naive Bayes and ID3 also perform reasonably well with lower execution times. The QDA algorithm shows low performance across all metrics. The choice of the algorithm depends on the specific requirements of the application, considering the trade-off between performance and execution time.

Table 23. 100% dataset consideration with 80% training and 20% testing data split



```
all_data        ML algorithm       accuracy     Precision    Recall       F1-score     Time
all_data        Bagging            0.97         0.941        0.952        0.947        10100.7402
all_data        Naive Bayes        0.81         0.656        0.653        0.654        9.2884
all_data        QDA                0.314        0.579        0.575        0.314        7.5533
all_data        Random Forest      0.954        0.966        0.866        0.907        563.0891
all_data        ID3                0.951        0.966        0.858        0.901        29.9888
all_data        AdaBoost           0.944        0.938        0.852        0.888        340.5536
all_data        GradientBoost      0.965        0.947        0.925        0.936        1260.5303
all_data        MLP                0.833        0.83         0.501        0.457        956.2694
all_data        Nearest Neighbors  0.967        0.957        0.92         0.937        3172.9026
```

In Table 23 showing the overall outcome of the experiment where complete dataset is taken along with 80%:20% training and testing data split. Result obtained has much better accuracy except the QDA algorithm. Hence the algorithm needs some parameters to estimate. Also, QDA algorithm is not suitable for dimensionality reduction as in our case the data is more dimensionally distributed. The result obtained has much better accuracy compared to conducting experiment using just 5 features for each individual attack. With increasing number of attacks to 17 the result is much better using all the selected algorithm.

Based on the provided data, the performance metrics for different ML algorithms on the "all_data" file:

Table 24. Performance on 100% dataset of 80:20 training and testing ratio

ML Algorithm	True Positive Rate	Execution Time
Bagging	High	High
Naïve Bayes	Moderate	Low
QDA	Low	Low
Random Forest	High	High
ID3	High	Moderate
AdaBoost	Moderate	High
GraidentBoost	High	High
MLP	Moderate	High
Nearest Neighbors	High	High

Based on this analysis, the Bagging, Random Forest, GradientBoost, and Nearest Neighbors algorithms perform well in terms of accuracy, precision, recall, and F1-score. However, they have significantly higher execution times compared to other algorithms. Naive Bayes and ID3 also perform reasonably well with lower execution times. The QDA algorithm shows low performance across all metrics. The choice of the algorithm depends on the specific requirements of the application, considering the trade-off between performance and execution time.

Comparing the two sets of results, it is found that Table 22 and Table 24 performed similar output by changing the training and testing ratio of the dataset. On evaluating with 3 precision point or more, the result was slight varying and giving better result in 80:20 ratio.

Bagging: The accuracy slightly improved from 0.9686 to 0.9699, while precision decreased from 0.9495 to 0.9414. However, recall improved xefrom 0.9359 to 0.9522, resulting in a higher F1-score of 0.9467. The execution time significantly increased from 5459.83 to 10100.74.

Naive Bayes: The accuracy remained almost the same (0.8096 vs. 0.8096). There was a slight increase in precision from 0.6542 to 0.6561. Recall also increased from 0.6492 to 0.6529, leading to a slightly higher F1-score of 0.6545. The execution time slightly increased from 8.27 to 9.29.

QDA: The accuracy showed a slight improvement from 0.3134 to 0.3136. Precision remained similar (0.579 vs. 0.5788), while recall also remained almost the same (0.5754 vs. 0.5751). The F1-score remained stable at around 0.3134. The execution time decreased from 9.60 to 7.55.

Random Forest: There was a marginal increase in accuracy from 0.9534 to 0.9535. Precision improved from 0.9659 to 0.9665, while recall remained similar (0.8655 vs. 0.8657). The F1-score showed a slight improvement from 0.9064 to 0.9067. The execution time increased from 300.98 to 563.09.

ID3: The accuracy decreased from 0.9565 to 0.9511. Precision improved significantly from 0.9385 to 0.9665. Recall decreased from 0.9005 to 0.8576, resulting in a lower F1-score of 0.9010. The execution time increased from 15.08 to 29.99.

AdaBoost: The accuracy decreased from 0.9458 to 0.9435. Precision also decreased from 0.9480 to 0.9379. Recall remained almost the same (0.8520 vs. 0.8523). The F1-score slightly decreased from 0.8911 to 0.8878. The execution time increased from 166.06 to 340.55.

Gradient Boost: The accuracy showed a marginal improvement from 0.9643 to 0.9651. Precision improved from 0.9465 to 0.9473, while recall remained similar (0.9227 vs. 0.9249). The F1-score improved from 0.9340 to 0.9356. The execution time significantly increased from 707.41 to 1260.53.

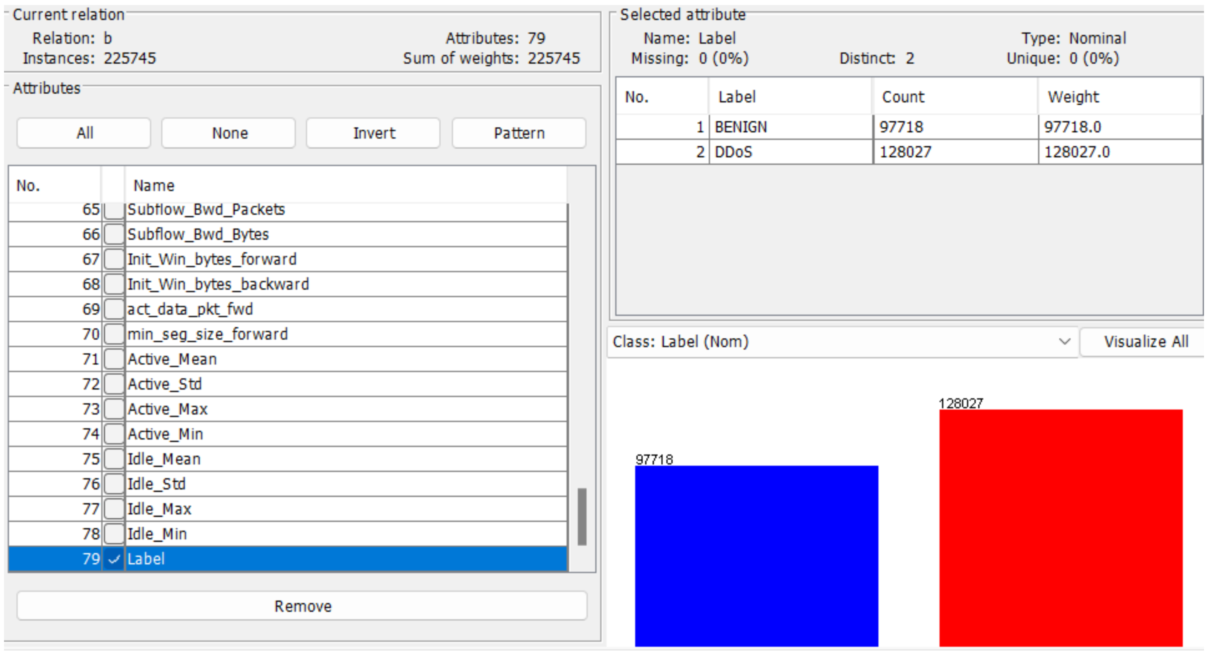
MLP: The accuracy slightly decreased from 0.8337 to 0.8333. Precision remained similar (0.8280 vs. 0.8303), while recall also remained similar (0.5011 vs. 0.5011). The F1-score remained stable at around 0.4570. The execution time significantly increased from 367.57 to 956.27.

Nearest Neighbors: The accuracy showed a slight improvement from 0.9663 to 0.9665. Precision improved from 0.9383 to 0.9574, while recall decreased from 0.9405 to 0.9198. The F1-score remained almost the same (0.9394 vs. 0.9373). The execution time significantly increased from 777.71 to 3172.90.

Based on this comparison:

Bagging and Random Forest consistently performed well in terms of accuracy, precision, recall, and F1-score. However, their execution times significantly increased compared to the previous results. Naive Bayes and QDA showed relatively stable performance, with minor improvements in some metrics. ID3, AdaBoost, and Gradient Boost had mixed results, with some metrics improving and others declining. MLP and Nearest Neighbors had a decrease in accuracy and precision while showing varied results for recall and F1-score. Consider the trade-off between performance metrics and execution time when selecting the ML algorithm for our specific use case. If execution time is a critical factor, we need to prioritize algorithms with faster processing. These results needed cross-validation on different datasets or using additional evaluation techniques to ensure the robustness of the findings.

This works extended by considering another approach in which a random sample of the dataset, which has a total of 225745 cases of BENIGN and DDoS attack, is taken. It also has records with infinite values or values that are missing. In the collected datasets, missing values are filled in with zeros. Lastly, data pre-processing makes sure that all the data are same. The final sampled dataset has a feature called Label that has an DDoS count of 128027 and a BENIGN count of 97718. This sampled data with number of instances, attributes, with total count of BENIGN and DDoS count is shown in Figure 15. Weka 3.9 tool is used for this purpose.


Current relation
Relation: b
Instances: 225745
Attributes: 79
Sum of weights: 225745

Attributes

No.		Name
65	☐	Subflow_Bwd_Packets
66	☐	Subflow_Bwd_Bytes
67	☐	Init_Win_bytes_forward
68	☐	Init_Win_bytes_backward
69	☐	act_data_pkt_fwd
70	☐	min_seg_size_forward
71	☐	Active_Mean
72	☐	Active_Std
73	☐	Active_Max
74	☐	Active_Min
75	☐	Idle_Mean
76	☐	Idle_Std
77	☐	Idle_Max
78	☐	Idle_Min
79	✓	Label

Selected attribute
Name: Label
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	BENIGN	97718	97718.0
2	DDoS	128027	128027.0

Class: Label (Nom)

Figure 15. Attributes with number of BENIGN and DDoS attack count.

In the feature selection and reduction process, three filter methods are utilized. These methods, that are Information-Gain Ratio (IGR), Correlation (CR), and ReliefF (ReF), aim to select the most relevant features. Each feature is assigned a score and weight based on statistical criteria determined by these selection algorithms. To obtain a more refined set of features, the aggregated weight of each approach is calculated. This is done by summing up the scores of all features and dividing it by the total count of features in smaller dataset. Using the average weight for threshold, a new subset of features is generated for each method. Only features that have scores equal to or higher than the threshold is included in the subsets.

Additionally, a Feature Selection Subset (FSS) analysis is conducted on the three subsets obtained. This analysis determines how frequently the features appear across the subsets. Three subsets, namely FSS-1, FSS-2, and FSS-3, are created based on whether a feature appears in at least one, two, or all three subsets, respectively. To train and test the model, FSS-1, FSS-2, and FSS-3 subsets are fed into a chosen classifier. Among these subsets, the one that requires the least amount of time to build the model without compromising its performance is selected. This chosen feature subset consists of weighted selected features that are utilized for identifying DDoS attacks. By using this subset, the model-building time is reduced while enhancing its performance.

The current work employs the PART model with a 90:10 ratio cross-validation approach. The estimation of the model's performance is measured by means of metrics such as accuracy, recognition rate, and timing. The research is divided into three phases: I. Feature reduction, II. The preparation phase, and III. The testing phase.

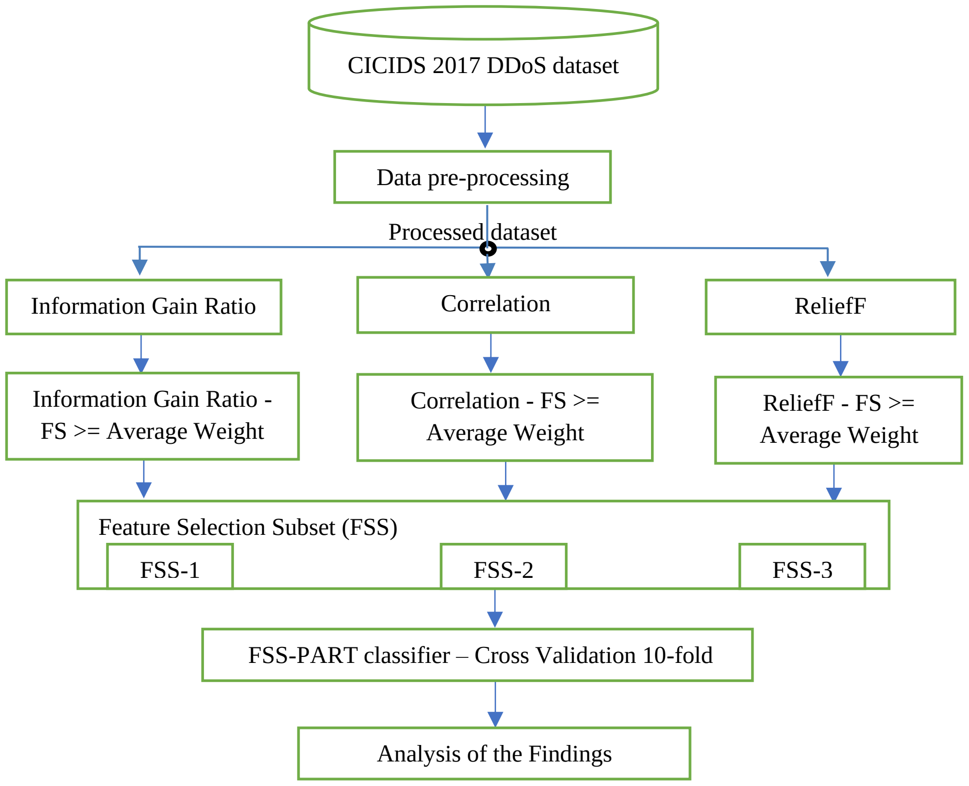

CICIDS 2017 DDoS dataset
Data pre-processing
Processed dataset
Information Gain Ratio
Correlation
ReliefF
Information Gain Ratio - FS >= Average Weight
Correlation - FS >= Average Weight
ReliefF - FS >= Average Weight
Feature Selection Subset (FSS)
FSS-1
FSS-2
FSS-3
FSS-PART classifier – Cross Validation 10-fold
Analysis of the Findings

Figure 16. Integrated IDS models developed for different attack categories.

Complete process is as followed:

Step1: selection of CICIDS 2017 DoS dataset

Step2: dataset pre-processing is done with following approach:

a. data understanding for structure, variable or meaning in context to benign or attack

b. data cleaning for inconsistent or missing values

c. removing duplicate or irrelevant features which is not contributing to model training and attack detection

d. categorize variable such as string to numerical data for label encoding

e. sampling for imbalances in the dataset to prevent outweigh the number of benign and attack instances

Step3: data split of 90 train and 10 test is applied which makes building model for training data

Step4: applying all 79 features on IGR, CR and ReF to get the top weighed subset features

Step5: reducing the number of features to 48, 28 and 14 compared to all 79 features based on step 4 if selected feature >= average weight of the frequency of occurrence in step 4 then accepts those reduced features else discard

Step7: processing of datasets as per reduced features for IGR, CR and ReF

Step 8: apply for best fit attributes selection and use ranker for feature's weightage

Step8: cross validation with 10-fold applied for separation of testing and training

Step9: analysis of the findings for IGR, CR and ReF using FSS-1, FSS-2 and FSS-3 respectively

Step10: best fit attribute selection applied in FSS-PART from step 9

Step11: comparison of result of FSS-PART with all features, Bayesian-Rough Set and AdaBoost predictive methods.

The algorithm uses supervised attribute selection for attribute A_b for 'm' distinct values. A supervised attribute filter that can be used to select attributes uses IG (Information Gain), CR (Correlation) and ReF (ReliefF) for feature selection. Features are selected if their relevance is greater than a threshold τ equivalent to average weight. Raw data captured through libpcap, WinPcap, or Npcap or well-known datasets and considered as T training dataset. These datasets contain Fs features in form of A_b Attribute.

A. Algorithm for Information Gain (IG) attribute selection with attribute weightage using ranker algorithm:

Input:

- A dataset D with n instances and m attributes.
- A set of attribute weights, $T = \{t_1, t_2, ..., t_m\}$, where t_i is weight for attribute i. (1)

Output:

- A ranked list of attributes based on their information gain and weighted importance.

Algorithm:

I. Calculate the information gain of every attribute in the dataset:

- For each attribute i, calculate its entropy using the formula:

$$\text{entropy}(i) = - \text{sum}(p_j * \log_2(p_j)), \quad (2)$$

where p_j is the proportion of instances in D that belong to the j^{th} class for attribute i.

- Calculate the overall entropy of the dataset using the same formula:

$$\text{entropy}(D) = - \text{sum}(p_k * \log_2(p_k)), \quad (3)$$

where p_k is the part of instances in D which belong to the k_{th} class.

- For each attribute i, calculate its information gain using the formula:

$$IG(i) = \text{entropy}(D) - \text{entropy}(i). \quad (4)$$

II. Multiply each attribute's information gain by its corresponding weight, and store the results in a new list, IG_weighted:

- For each attribute i, calculate its weighted information gain using the formula:

$$IG\_weighted(i) = IG(i) * t_i. \quad (5)$$

III. Sort the IG_weighted list in descending order:

- Sort the CR_weighted list in descending order using merge sort.

IV. Return the sorted list as the ranked list of attributes:

- Return the sorted IG_weighted list as the ranked list of attributes, where attribute with the uppermost weighted information gain is ranked first, attribute with second-highest weighted information gain is ranked second, and so on.

V. Apply threshold limit to set limit to attributes, other will be discarded.

B. Algorithm for Correlation (CR) attribute selection with attribute weightage using ranker algorithm:

Input:

- A dataset D with n instances and m attributes.
- A set of attribute weights, T = {t_1, t_2, ..., t_m}, where t_i is weight for attribute i.

Output:

- A ranked list of attributes based on their correlation and weighted importance.

Algorithm:

Here's the step-by-step breakdown of the algorithm:

I. Calculate the correlation coefficient between each attribute and the target attribute in the dataset:

- For each attribute i, calculate its correlation coefficient with the target attribute using the formula:

$$CR(i) = \frac{n*sum(x_i*y_i)-sum(x_i)*sum(y_i)}{\sqrt{(n*sum(x_i{}^2)-sum(x_i)^2-(n*sum(y_i{}^2)-sum(y_i)^2)}} \tag{6}$$

where x_i is the value of attribute i for instance j, y_i is the value of the target attribute for instance j, and n is the number of instances in the dataset.

II. Multiply each attribute's correlation coefficient by its corresponding weight, and store the results in a new list, CR weighted:

- For each attribute i, calculate its weighted correlation coefficient using the formula:

$$CR\_weighted = \{CR_1t_1, CR_2t_2, ..., CR_m*t_m\}. \tag{7}$$

$$CR\_weighted(i) = CR_i * t_i. \tag{8}$$

III. Sort the CR_weighted list in descending order with ranker approach:

- Sort the CR_weighted list in descending order using merge sort.

IV. Return the sorted list as the ranked list of attributes:

- Return the sorted CR weighted list as the ranked list of attributes, where attribute with highest weighted correlation coefficient is ranked first, attribute with the second-highest weighted correlation coefficient is ranked second, and so on.

V. Apply threshold limit to set limit to attributes, other will be discarded.

C. Algorithm for RefleF attribute selection with attribute weightage using ranker algorithm:

I. Initialize the weight vector for each feature to zero.

II. For every instance in dataset:

a. Randomly select another instance from the same class (positive instance) and another instance from a different class (negative instance).

b. Calculate the difference between feature values of present instance and positive and negative instances for each feature.

c. Update the weights of the features as follows:

If feature values of present instance and positive instance are different, increase the weight of the feature.

If feature values of present instance and negative instance are different, decrease the weight of the feature.

III. Repeat Step 2 for a predefined number of iterations or until convergence.

IV. Normalize the weights by dividing each weight by total number of instances.

V. Sort features based on their weights in descending order.

VI. Select top-k features with uppermost weights as concluding feature subset.

Chapter-5

5 Result and Conclusion

5.1 Using FSS-PART feature selection approach on CICIDS-2017 dataset

Execution and analysis are done in Python 3 and used the API of Weka 3.8. The function is selected manually, and content written in Python retains a copy of the input and missing attributes. The Weka widget is open source, and the contains rule-based classifiers. This framework has been tested on the CICIDS-2017 dataset, which contains 79 functions, including label. It ends with a data log with 79 important features which limits the research after removing errored or null records.

Some of the duplicate instances were also removed and the final dataset contains 223112 records, 90% of which are classified according to the training work, and 10% are classified for the testing. The calculations of IG-Feature Selection, CR-Feature Selection and ReF-Feature Selection found to be faster and are used to calculate the rating of every component. In this, all attributes selected and performed a classification test with PART; after that, they were applied with (IGR), Correlation (CR) and ReliefF (ReF). Then reduce the dataset and check whether it gets the same accuracy with fewer features. Using (IGR) with 33 attributes, Association (CR) with 28 attributes, and ReliefF (ReF) with 23 attributes, it gives better accuracy.

Table 25. A subset of 79 features of CICIDS 2017 dataset.

Methods Used	Total Features	Numbers of Features
Information gain -Feature Selection	33	64,5,53,66,6,13,55,1,35,56,11,67,9,54,7,36,24,21,22,68,63,3,41,69,23,42,43,10,14,29,26,38,65
Correlation-Feature Selection	28	15,16,13,55,11,14,1,49,41,53,42,39,40,70,43,9,54,12,7,5,64,26,52,10,29,28,47,45
ReF-Feature Selection	23	15,16,48,11,1,40,14,42,49,43,55,13,41,53,52,26,7,67,54,9,29,2,45

For subsets IG-Feature Selection, CR-Feature Selection, and ReF-Feature choice shown in Table 25, the features are organized by the number of subsets for each event, even if it had some selected set of features.

In Table 26, a Feature Selection Subset-3 is obtained by observing number of feature weightage in the 2 and 3 subsets, respectively. The Feature Selection Subset is composed of 48, 28, and 14 elements. Feature Selection Subset-2 and Feature Selection Subset-3 correspond to the standard half classifier. Machine learning for training and testing, cross-validation of ten times is done. Attribute selection in three subsets is done and got the accuracy for this proposed method. Then, the half classifier prepares the model and tests it ten times. Use half to check Feature Selection Subset-2, and Feature Selection Subset-3 and the result is shown in form of accuracy and time of occurrence.

Table 26. Strategies for combining subsets for feature selection.

Method Used	Total Features	Numbers of Features
FSS -1	48	64,5,53,66,6,13,55,1,35,56,11,67,9,54,7,36,24,21,22,68,63,3,41,69,23,42,43,10,14,29,26,38,65,4,27,40,37,12,28,18,15,19,2,16,17,30,74,25
FSS-2	28	15,16,13,55,11,14,1,49,41,53,42,39,40,70,43,9,54,12,7,5,64,26,52,10,29,28,47,45
FSS-3	14	15,16,48,11,1,40,14,42,49,43,55,13,41,53

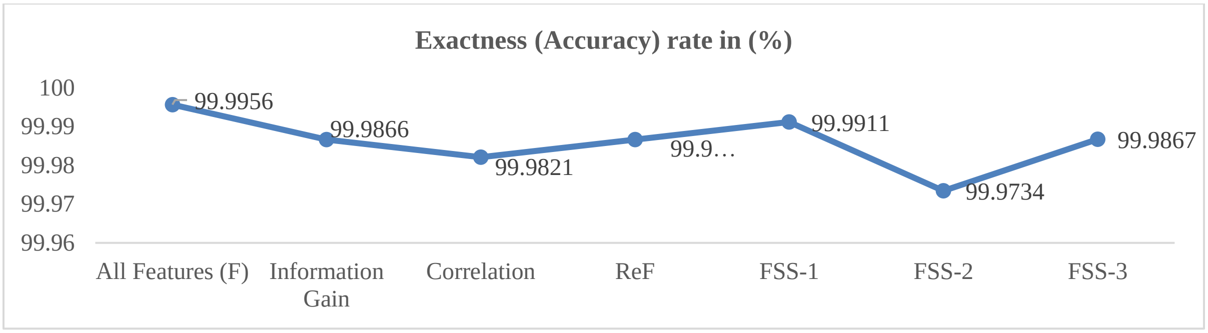


Figure 17. Accuracy rate in (%) for different feature selection subsets.

Feature Selection Subset-2 provides higher accuracy and uses vital chance and half to make models. Finally, Feature Selection Subset-2 was designated as a perfect subset of elements with 28 functions. After analysis for the performance of the 2017 CICIDS dataset, when using all 79 features, it has accuracy rate better than the Information Gain which is using 33 features and has accuracy rate slight lesser than using all features but is better than Correlation which is using 28 features has accuracy difference of 0.0045. ReF with 23 features has better accuracy than Correlation but has similar accuracy compared with Information Gain. With the feature selection subset method, Feature Selection Subset-1 using 48 features has accuracy of 99.9911. Feature Selection Subset-2 with 28 features has accuracy of 99.9734 and Feature Selection Subset-3 with using 14 features has accuracy of 99.9867.

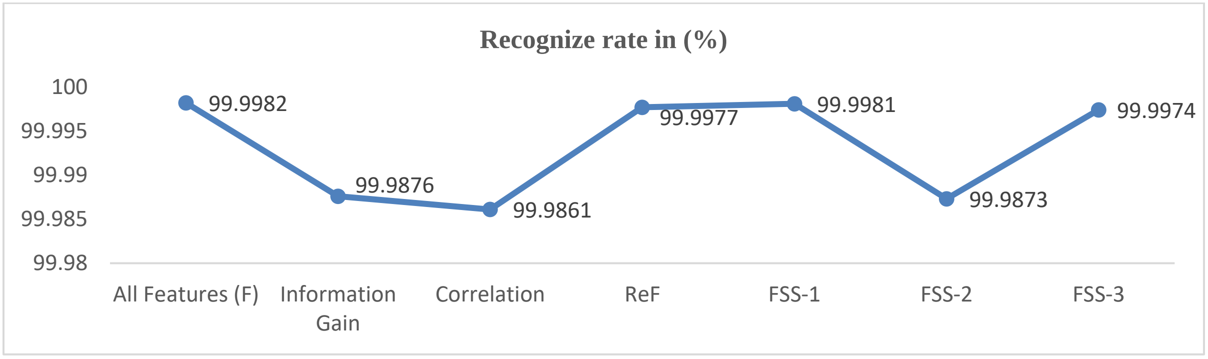


Figure 18. Recognize rate in (%) for different feature selection subsets.

Similarly, in Figure 18 when using all 79 features, it has recognized rate in percentage came out to be 99.9982, similarly for Information Gain using 33 features it has recognized rate of 99.9876, Correlation using 28 features has recognized rate of 99.9821, ReF with 23 features is 99.9977. With the feature selection subset method, Feature Selection Subset-1 using 48 features has recognized rate of 99.9981. Feature Selection Subset-2 with 28 features has recognized rate of 99.9873 and Feature Selection Subset-3 with using 14 features has recognized rate of 99.9974.

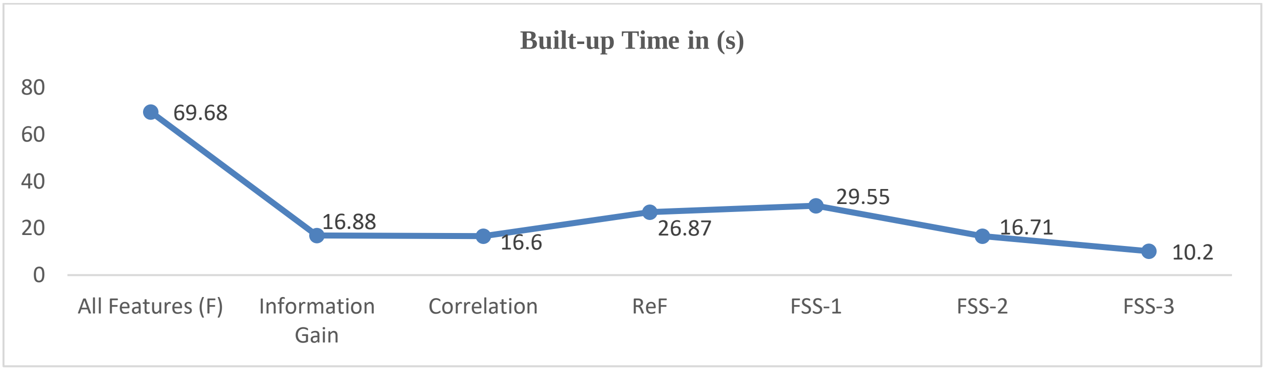


Figure 19. Built-up Time in (s) for different feature selection subsets.

It shows that compared with the first functions such as Feature Selection Subset-1 and Feature Selection Subset-3, the development time of these 14 simplified features is 10.2 seconds in Figure 19, have increase in the accuracy by 99.9867% which has only slight decreased in accuracy when using all 79 features with the accuracy level of 99.9956% considered for the experiment. It can be seen from the graph that using more feature selection accuracy will be better but built-up time also increases. With careful selection of features as taken in Feature Selection Subset-3 with built-up time of 10.2 second with accuracy of 99.9867 and for Feature Selection Subset-1 and Feature Selection Subset-2 accuracy is 99.9911 and 99.9734, built-up time is 29.55 and 16.71, respectively.

Considering the current working state, the relevant research on this test was carried out. Figure 19 and Figure 20 depict similar inspections of existing structures using the proposed strategies introduce the accuracy and timing of applying PART with 10 cross-folds. The accuracy of this model at critical moment is increased by 99.9867% and compared with different algorithm applied.

Table 27. Feature reduction in CICIDS2017

Algorithm Used	Total Features	Features
Bayesian-Rough Set	37	3-13, 15, 20, 26, 28-30, 35, 36, 39, 40, 52, 54, 55,62-73, 75
AdaBoost	25	6, 8, 12, 14, 17, 20, 25-28, 30, 37-39, 43, 47, 48,52-54, 63, 66, 67, 70, 77
FSS-PART	28	15,16,13,55,11,14,1,49,41,53,42,39,40,70,43,9,54,12,7,5,64,26,52,10,29,28,47,45

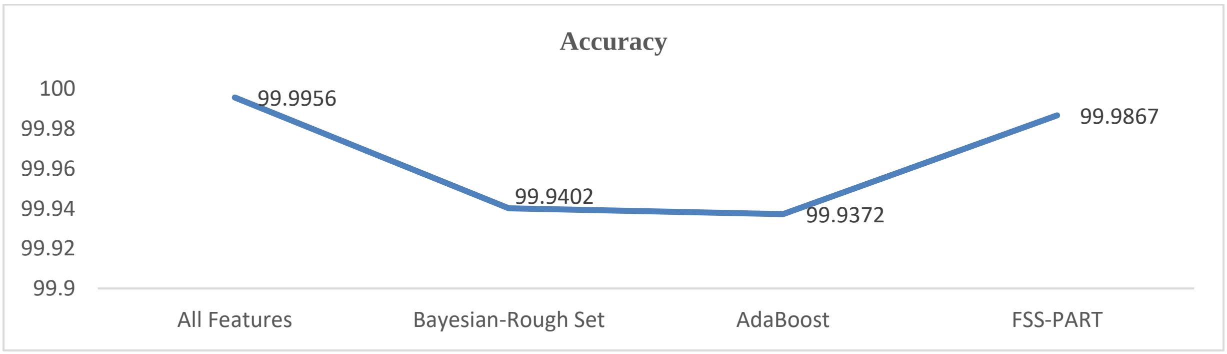


Figure 20. Accuracy Comparison of Bayesian-Rough Set, AdaBoost and FSS-PART.

Using the CICIDS 2017 dataset, using all 79 features, it has accuracy rate in percentage came out to be 99.9956, similarly for Bayesian-Rough Set using 37 features it has recognized rate of 99.9402, AdaBoost using 25 features has accuracy rate of 99.9372, Where as in the proposed method with 28 features is 99.9867 as shown in Figure 20 and number of features selected for respective algorithm as per weightage in Table 27.

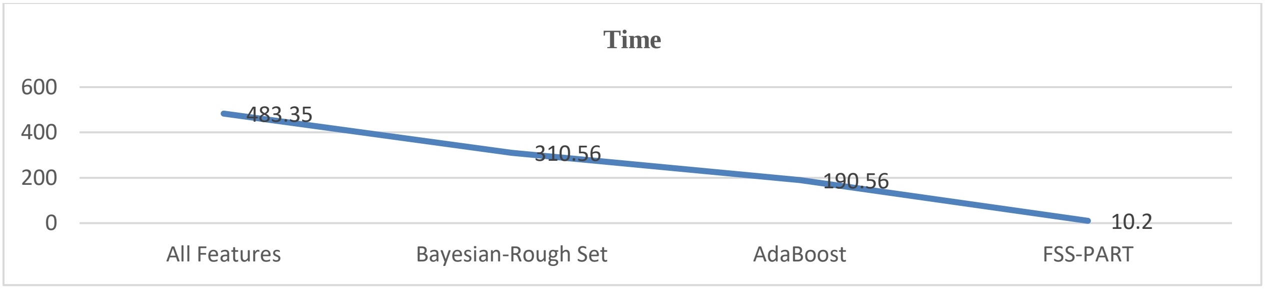


Figure 21. Execution time examination on BRS, AdaBoost and FSS-PART.

When using all 79 features, it has recognized rate came out to be 483.35 seconds, similarly for Bayesian-Rough Set using 37 features it has time execution of 310.56 seconds, AdaBoost using 25 features has accuracy rate of 190.56 seconds, whereas in the proposed method with 28 features is 10.2 seconds. In summary, a total of 79 required features is selected which represent a total of 14 different network attacks in the CICIDS datasets. Other unused features were removed in the data-preprocessing part. Using all the 79 features for detecting the attacks was a time taking task. A feature selection approach is applied with attribute selection for InfoGain, Correlation, ReliefF attribute evaluation and along with the searching of important features using ranker method. This given the important feature weightage as a subset of all the 79 features and the respective features were 33, 28 and 23. These reduced features taken lesser time for training and testing purpose with little degradation in the accuracy compared to considering all the 79 attributes. For reducing the time consumption and attains a similar accuracy, top 48 features are selected from the 79 all features and then reduced to 28 features to implement in FSS-2. Features in FSS-3 is selected by reducing half of the important 14 features from FSS-2. The result obtained in FSS-3 taken lesser time when using supervised approach with Correlation Attribute Evaluation with ranker attribute selection. The time taken in Correlation method is very less compared to Relief, so the final consideration for predicting the accuracy is features used in FSS-2 and applied the same approach for prediction with supervised leaning approach of Bayesian-Rough Set, AdaBoost and this implemented approach of FSS-PART. The FSS-PART uses the Correlation method with ranker approach along with the 28 features to get the near accuracy of 99.9867 compared with the all 79 features 99.9956 with reduction in accuracy of 0.0089% but the most important part is the time taken is much lesser using the FSS-PART method.

Various authors have previously worked on the different datasets and uses machine learning approach that achieved accuracy based on experiment they had done or based on the feature selection they had taken is shown in .

Table 28. Features, datasets and result on various ML techniques

Ref	**ML Technique**	**Dataset**	**Features**	**Results in %**
Cannady [89]	Supervised NN	RealSecure	Payload, header of TCP, IP, and ICMP	Detection Ration: 89-91
Pfahringer [90]	Supervised Ensemble of C5 DTs	KDD Cup	41 features	Normal:99.5 Probe:83.3 DoS:97.1 U2R:13.2 R2L:8.4
Zhi-Song et al. [91]	Supervised NN and C4.5 DT	KDD Cup	41 features	Normal:99.5 DoS:97.3 Satan:95.3 Portsweep: 94.9 U2R:72.7 R2L: 100
Moradi and Zulkernine [92]	Supervised NN	KDD Cup	35 features	MLP:80 ESVM:90 ESVM DR:87
Thomas et al. [93]	Supervised BN and CART	KDD Cup	41 features	Normal:100 Probe:100 DoS:100 U2R:84 R2L:99.47
Elouedi et al. [94]	Supervised NB	KDD Cup	41 features	Normal: 97.68 DoS: 96.65 R2L: 8.66 U2R: 11.84

				Probing: 88.33
Chen et al. [95]	Supervised C4.5 DT	KDD Cup	GA-based Feature Selection	DoS:97.88 Probe: 98.33 R2L: 80.01 U2R: 99.99
Abraham et al. [96]	Supervised Ensemble of SVM, DT, and SVM-DT	KDD Cup	all 41 features	Normal: 99.7 Probe:100 DoS: 99.92 U2R: 68 R2L: 97.16
Sangkatsanee et al. [97]	Supervised C4.5 DT	RLD09	Header of TCP, UPD, and ICMP	Normal: 99.43 DoS: 99.17 Probe: 98.73
Miller and Busby [98]	Supervised Ensemble MPML	NSL-KDD	all 41 features	84.137
Li and Guo [99]	Supervised TCM K-NN Chi-square	KDD Cup	41 features 8 features	99.7 99.6
Kshirsagar et al. [100]	IGR-SCS1 CR-SCS2 ReF-SCS3	CICIDS 2017	48 Features 24 Features 12 Features	DoS:99.9586 DoS:99.9593 DoS:98.8698
Kurniabudi et al. [101]	Random Forest	CICIDS 2017	15 Features	99.81
Kurniabudi et al. [102]	Information Gain + Random Forest	CICIDS 2017	22 Features 28 Features	99.83 99.79
Habtamu [103]	AdaBoost	CICIDS 2017	10 Features	99
Prasad et al. [85]	Bayesian Rough set	CICIDS 2017	40 Features	96.38
Proposed FSS-PART	Supervised IGR, CR and RelifF	CICIDS 2017	77 Features 48 Features 28 Features 14 Features	DDoS: 99.9982 FSS-1: 99.9981 FSS-2: 99.9872 FSS-3: 99.9974

5.2 Using KDD'99 for attack analysis

In another approach, KDD'99 dataset is used as the above dataset was very large and taking a day or more to execute all the instance of machine learning approach.

The most broadly utilized and freely accessible attack dataset is the KDD'99. The informational index is separated into two subsets; the preparation set contains 5 million informational collections, and the test set contains 3 million informational indexes. The accompanying table shows the specific number of assaults of each sort in the KDD'99 record. Also, labels that order association records as typical or extraordinary attack types. The attributes of information records can be isolated into four classifications: Internal qualities e. Association span, convention type (tcp, udp, and so on), network administration (http, telnet, and so on), and so on the quantity of fizzled login endeavors, and so on A similar host work checks the associations set up just now with a similar objective host as the flow association, and computes factual data on convention practices, administrations, etc. The same comparable help capacity will look for associations that have similar assistance as the current association just now.

The KDD'99 informational collection comprises of profoundly repetitive informational collections, which implies that learning calculations are outfitted towards regular informational collections, subsequently keeping them from learning uncommon informational indexes that are generally more ruinous to the organization, for example, U2R and R2L attacks. This eliminates redundant records and reduces the possibility of system errors in the classifier.

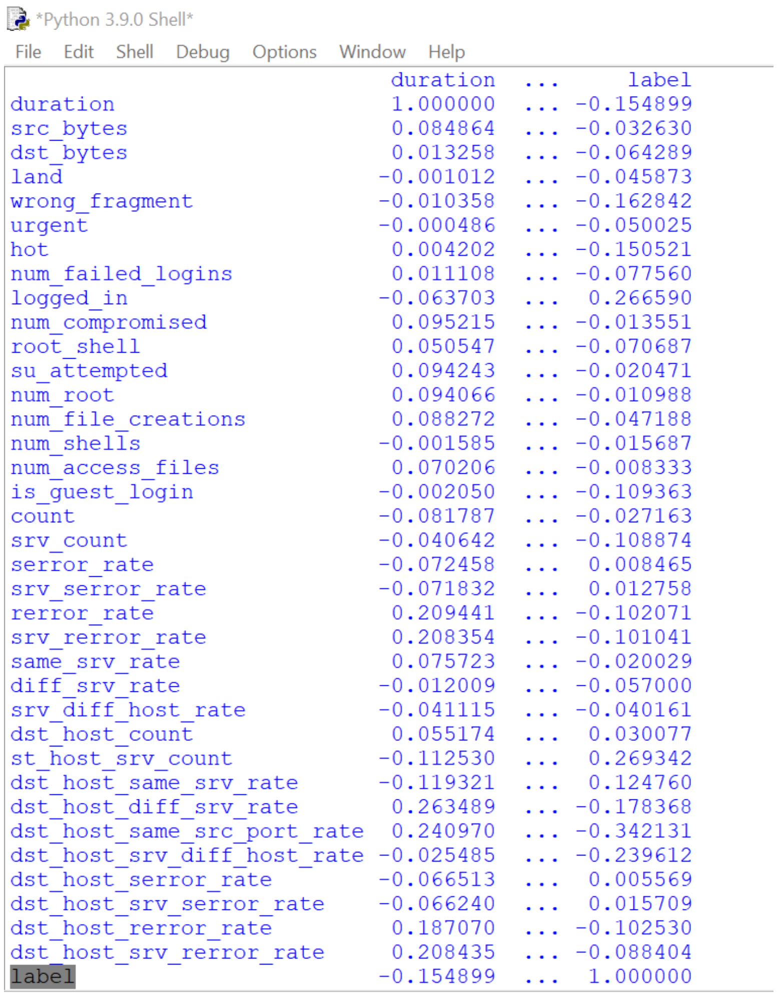
Table 29 below shows the total packet count along with the 10% of the randomly selected dataset. These 10% dataset contains various redundancy, so this further corrected that made the total count of 311029. These datasets are also showing the total number of an attack count.

Table 29. Distribution of attacks in the KDD'99 data set

Dataset	**DoS**	**U2R**	**R2L**	**Probe**	**Normal**	**Total**
10% KDD	391458	4107	52	1126	97277	494020
Corrected KDD	229853	4166	70	16347	60593	311029
Whole KDD	3883370	41102	52	1126	972780	4898430

In each of the three forms of the data index i.e 10% KDD, Corrected KDD and Whole KDD attacks can be categorized as one of four classifications of attack.

Table 30. Attributes ranking for feature selection of KDD'99



```
                               duration  ...     label
duration                       1.000000  ... -0.154899
src_bytes                      0.084864  ... -0.032630
dst_bytes                      0.013258  ... -0.064289
land                          -0.001012  ... -0.045873
wrong_fragment                -0.010358  ... -0.162842
urgent                        -0.000486  ... -0.050025
hot                            0.004202  ... -0.150521
num_failed_logins              0.011108  ... -0.077560
logged_in                     -0.063703  ...  0.266590
num_compromised                0.095215  ... -0.013551
root_shell                     0.050547  ... -0.070687
su_attempted                   0.094243  ... -0.020471
num_root                       0.094066  ... -0.010988
num_file_creations             0.088272  ... -0.047188
num_shells                    -0.001585  ... -0.015687
num_access_files               0.070206  ... -0.008333
is_guest_login                -0.002050  ... -0.109363
count                         -0.081787  ... -0.027163
srv_count                     -0.040642  ... -0.108874
serror_rate                   -0.072458  ...  0.008465
srv_serror_rate               -0.071832  ...  0.012758
rerror_rate                    0.209441  ... -0.102071
srv_rerror_rate                0.208354  ... -0.101041
same_srv_rate                  0.075723  ... -0.020029
diff_srv_rate                 -0.012009  ... -0.057000
srv_diff_host_rate            -0.041115  ... -0.040161
dst_host_count                 0.055174  ...  0.030077
st_host_srv_count             -0.112530  ...  0.269342
dst_host_same_srv_rate        -0.119321  ...  0.124760
dst_host_diff_srv_rate         0.263489  ... -0.178368
dst_host_same_src_port_rate    0.240970  ... -0.342131
dst_host_srv_diff_host_rate   -0.025485  ... -0.239612
dst_host_serror_rate          -0.066513  ...  0.005569
dst_host_srv_serror_rate      -0.066240  ...  0.015709
dst_host_rerror_rate           0.187070  ... -0.102530
dst_host_srv_rerror_rate       0.208435  ... -0.088404
label                         -0.154899  ...  1.000000
```

In the dataset, three protocols TCP, UDP, and ICMP are used to simulate attacks. 10% of corrected KDD'99 dataset contains with the four DoS, U2R, R2L and probe. There are 22 diverse attacks.

Table 31. List of KDD features

Sr. No.	Features	Sr. No.	Features	Sr. No.	Features
1	Duration	15	su_attempted	29	same_srv_rate
2	protocol_type	16	num_root	30	diff_srv_rate
3	Service	17	num_file_creations	31	srv_diff_host_rate
4	Flag	18	num_shells	32	dst_host_count
5	src_bytes	19	num_access_files	33	dst_host_srv_count
6	dst_bytes	20	num_outbound_cmds	34	dst_host_same_srv_rate
7	Land	21	is_host_login	35	dst_host_diff_srv_rate
8	wrong_fragment	22	is_guest_login	36	dst_host_same_src_port_rate

9	Urgent	23	Count	37	dst_host_srv_diff_host_rate
10	Hot	24	srv_count	38	dst_host_serror_rate
11	num_failed_logins	25	serror_rate	39	dst_host_srv_serror_rate
12	logged_in	26	srv_serror_rate	40	dst_host_rerror_rate
13	num_compromised	27	rerror_rate	41	dst_host_srv_rerror_rate
14	root_shell	28	srv_rerror_rate		

Table 30 gives a total portrayal of all the attributes along with its weightage. Some of these weightages with very less values can be removed unless it relates with any attack parameter.

Table 32. Types of attacks in KDD'99 dataset

Category	Attack type
Probe	mscan, portsweep, nmap satan, saint, ipsweep
DoS	apache, Back, land, neptune,teardrop, smurf, mailbomb, teardrop, udpstorm, pod
U2R	rootkit, ps, loadmodule, attack, Perl, buffer overflow, xterm
R2L	imap, ftp_write, multihp, phf, named, warezmaster, worm, xsnoop, snmpgetattack, httptunnel, imap, snmp_guess , Guess_password,

Attacks which can be analyzed with features of this dataset are as below:

Denial of Service (DoS): this type of attack involves back, Neptune, land, pod, smurf and teardrop. Mostly used features are source bytes (5), land (7) and wrong fragment (8) to detect DoS type of attack.

Probe: it involves ipsweep, nmap, satan and portsweep type of attacks. To detect Probe attack diff_srv_rate (30), dst_host_same_src_port_rate (36), source bytes (5), srv_rerror_rate (28) features are used.

Normal: Normal attacks are detected by features of KDD cup dataset but the most preferred feature is same_srv_rate (29). This is used to analysis normal attack.

Remote-to-Local (R2L): R2L is the basic category of these guess_passwd, Imap, Phf, warezmaster, warezclient, multihop and spy attacks. num_failed_logins (11), count (23), service (3), destination bytes (6), dst_host_srv_serror_rate (39) features are used to analyse R2L attacks.

User-to-Root (U2R): Perl, rootkit, load module and buffer_overflow attacks come under U2R attack category. The mostly used features are service (3), dst_host_same_src_port_rate (36), root shell (14), srv_count (24) to detect U2R attack.

Useless features: Information gain is calculated to check whether the features are relevant to intrusion detection or not. Feature 20 (num_outbound_cmds) is always 0 and 21 (is_host_login) do not reflect any change in training set they both are not relevant to intrusion detection technique and their information gain is negligible that is why they are irrelevant features for detection. And other features are 13 (num_compromised), 15 (num_root), 17 (num_file_creations), 22 (is_guest_login), 40 (dst_host_rerror_rate). Information gain from these features is very less. These features are not at all contributing to detection any of attacks. So, above paragraph mentioned have a total of 7 features that have information gain less than 0.0001 which have almost negligible contributing nature.

Table 33. Most useful features for attacks detection in KDD'99 dataset.

Class	Total	Name of features
DoS	11	3,5,6,7,8,23,29,30,32,34,35
Probe	25	1,3,4,5,6,10,12,23,24,25,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41
R2L	23	1,3,5,6,9,10,11,12,19,22,23,28,30,31,32,33,34,35,36,37,38,39,40
U2R	16	3,5,6,11,12,14,16,17,24,32,33,35,36,37,40,41
Normal	25	1, 2, 3, 4, 5, 6, 7,12, 15,16,17,18,19, 23, 24, 14, 15, 19, 20, 21, 23, 25, 26, 27, 28, 30,31, 32, 33, 34, 36, 37, 38

After detail analysis following was the summary table which shows the exact reason responsible for detection of attacks.

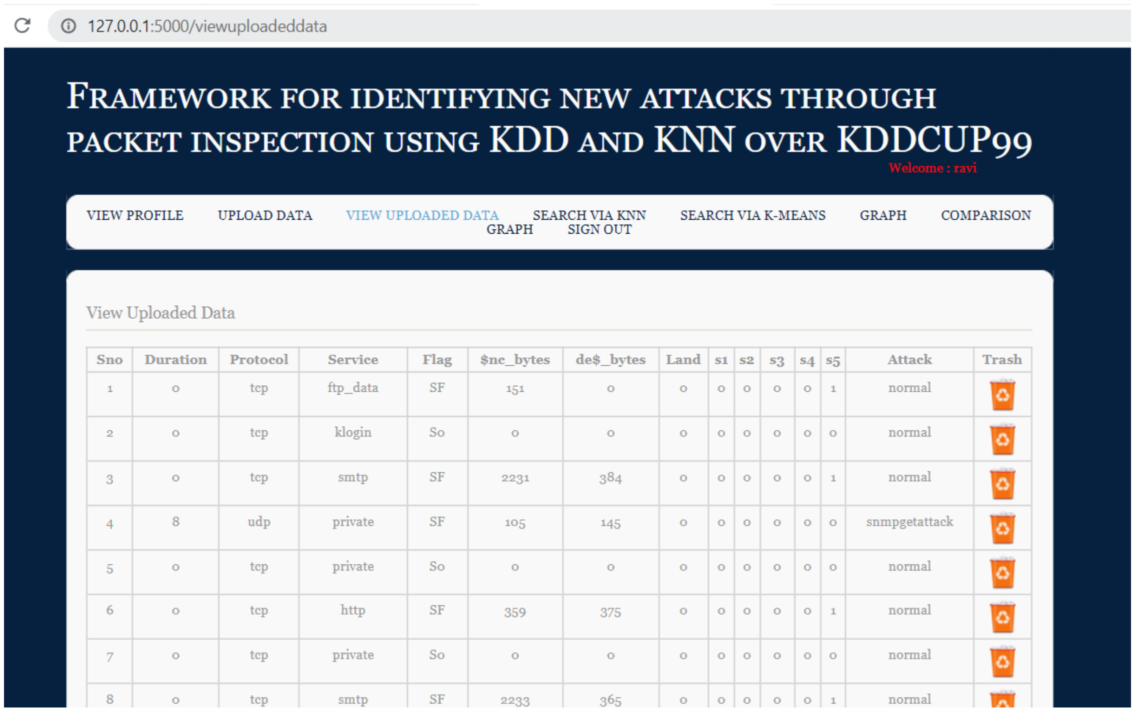
Table 34. Useful features for different attacks in KDD'99

Sr. No.	**Category**	**Attacks**	**Useful Features**
1	DoS	Back	1, 2, 4, 5, **6**, **10**, 11, 12, **13**, 15, 17, 18, 21, 22, 23, 26, 27, 28, 30, 31, 34, 35, 37, 41
2	DoS	Land	1, 2, 3, 4, **7**, 13, 18, 25, 29, 35, 38
3	DoS	Neptune	1, 3, 4, 5, 6, 7, 13, 15, 17, 19, 20,25, 26, 28, 29, **30**, 31, 33, 34, 35, 38, 39
4	DoS	Pod	2, 3, **5**, 7, 8, 9, 10, 11, 17, 19, 21, 23, 26, 33, 34, 39, 40

5	DoS	Smurf	2,3,**5**,8,17,23,24,25,26,27,28,29,33,35,36,38,39
6	DoS	Teardrop	3,4,**5**,6,**8**,10,13,23,24,25,26,32,34,35,36,37,39,40
7	Probe	Satan	1, 3, 5, 11, 15, 19, 23, 24, 25, **27**, 28, 29, 30, 31, 32, 35, 39, 40, 41
8	Probe	Ipsweep	2, 3, 5, 12, 13, 14, 16, 17, 21, 23, 24, 25, 28, 31, 32, 33, **37**, 38
9	Probe	Nmap	1, 2, 3,**4**, 5, 18, 21, 22, 28, 29, 31, 32, 34, 35, 36, 37
10	Probe	Portsweep	3, **4**, 10, 24, 27,28, 29, 34, 35, 36, 37, 41
11	R2L	Guess_passwd	2, 3, 4,**5**, 6, 9, 10, **11**, 13, 14, 17, 21, 23, 24, 37, 38, 39, 40, 41
12	R2L	ftp_write	**5**,**9**,23
13	R2L	Imap	**3**, 4, 5, 6, 10, 12, 20, 23, 25, 27, 29, 30, 32,33, 34, 36, 38, 39, 41
14	R2L	Phf	3, 4, **6**, 8, 9, 10, 13, 14, 19, 28, 29, 36
15	R2L	Multihop	3, 4,**6**, 10, 12, 13, 14, 16, 17, 18, 19, 22, 26, 27, 30, 35, 37
16	R2L	Warezmaster	1, 2, 3, 4, **6**, 12, 13, 14, 16, 17, 19, 22, 23, 24, 31, 35, 36, 37, 39
17	R2L	Warezclient	3, 4, **5**, 6, 10, 12, 14, 16,**22**, 24, 27, 28, 29, 30, 32, 33, 34, 35, 37, 38, 39, 40, 41
18	R2L	Spy	2, 3, 4, 5, 9, 15, 18, 22, 16, **39**
19	U2R	Buffer_overflow	1, 2, 3, 5, **6**, 7, 8, 9, 10, **14**, 21, 23, 29, 30, 31, 32, 33, 36, 38, 39, 40
20	U2R	Load_module	1, 2, 3, 4, **6**,7, 8, 14, 27, 36, 39, 40
21	U2R	Perl	5,14,**16**,18
22	U2R	Rootkit	3, 5,6, 9, 11, 13, 14, 16, 17, 18, 23, 28, 31, 32, 33, 34, 35, 37, 39, 41

5.3 Framework of new attack detection through packet inspection

A Framework for the attack detection is made using a small dataset using KDD'99. The framework only covers the attack at higher layer of TCP/IP model instead of attack at DLL and physical layer due to the nature of an attack.


127.0.0.1:5000/viewuploadeddata

FRAMEWORK FOR IDENTIFYING NEW ATTACKS THROUGH PACKET INSPECTION USING KDD AND KNN OVER KDDCUP99

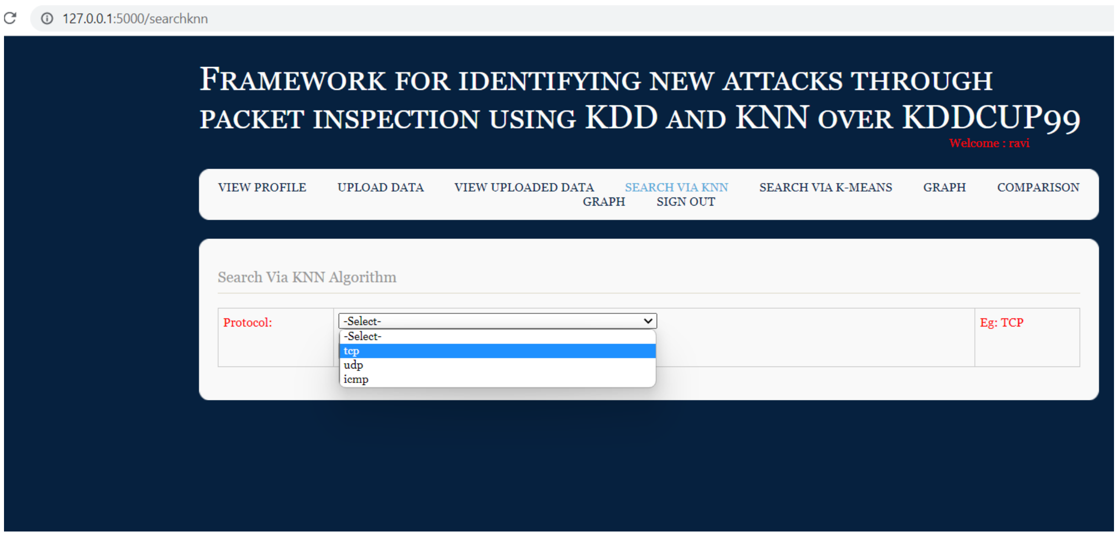
Welcome : ravi

VIEW PROFILE UPLOAD DATA VIEW UPLOADED DATA SEARCH VIA KNN SEARCH VIA K-MEANS GRAPH COMPARISON GRAPH SIGN OUT

View Uploaded Data

Sno	Duration	Protocol	Service	Flag	$nc_bytes	de$_bytes	Land	s1	s2	s3	s4	s5	Attack	Trash
1	0	tcp	ftp_data	SF	151	0	0	0	0	0	0	1	normal	🗑
2	0	tcp	klogin	So	0	0	0	0	0	0	0	0	normal	🗑
3	0	tcp	smtp	SF	2231	384	0	0	0	0	0	1	normal	🗑
4	8	udp	private	SF	105	145	0	0	0	0	0	0	snmpgetattack	🗑
5	0	tcp	private	So	0	0	0	0	0	0	0	0	normal	🗑
6	0	tcp	http	SF	359	375	0	0	0	0	0	1	normal	🗑
7	0	tcp	private	So	0	0	0	0	0	0	0	0	normal	🗑
8	0	tcp	smtp	SF	2233	365	0	0	0	0	0	1	normal	🗑

Figure 22. Framework showing the packet along with the packet label


127.0.0.1:5000/searchknn

FRAMEWORK FOR IDENTIFYING NEW ATTACKS THROUGH PACKET INSPECTION USING KDD AND KNN OVER KDDCUP99

Welcome : ravi

VIEW PROFILE UPLOAD DATA VIEW UPLOADED DATA SEARCH VIA KNN SEARCH VIA K-MEANS GRAPH COMPARISON GRAPH SIGN OUT

Search Via KNN Algorithm

Protocol: -Select- (options: -Select-, tcp, udp, icmp) Eg: TCP

Figure 23. Selection of protocol type for searching attack through KNN algorithm

In the given figure, user can select the particular protocol to detect specific attack category.

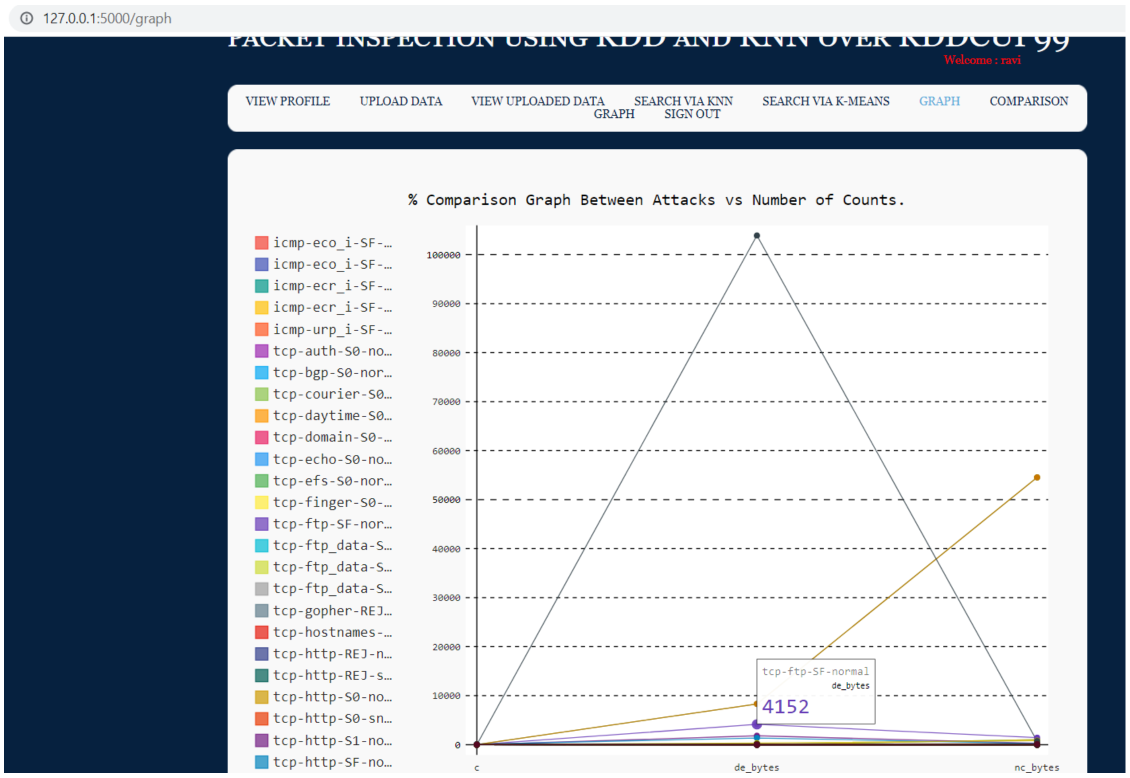


Figure 24. Graph showing the attack and attribute count

This graph shows an attack and number of parameters count of all the packet received that can be used to benchmark the threshold level and detecting any anomaly in the traffic.

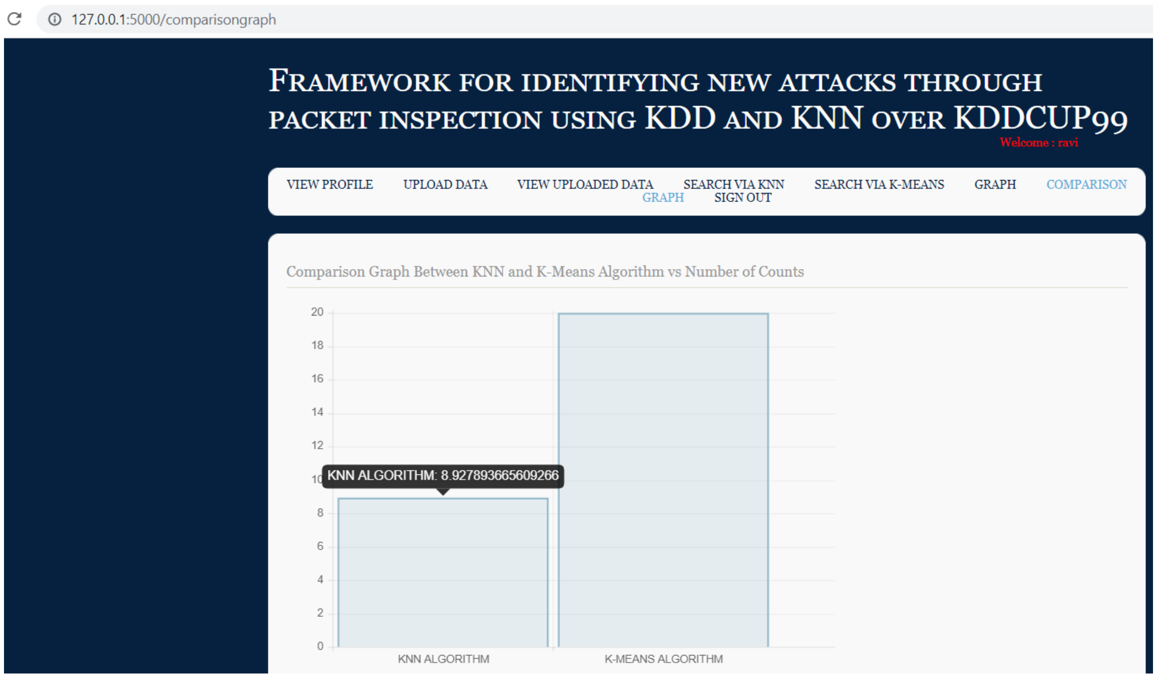

127.0.0.1:5000/comparisongraph
FRAMEWORK FOR IDENTIFYING NEW ATTACKS THROUGH PACKET INSPECTION USING KDD AND KNN OVER KDDCUP99
Welcome : ravi
VIEW PROFILE
UPLOAD DATA
VIEW UPLOADED DATA
SEARCH VIA KNN
SEARCH VIA K-MEANS
GRAPH
COMPARISON GRAPH
SIGN OUT
Comparison Graph Between KNN and K-Means Algorithm vs Number of Counts
KNN ALGORITHM: 8.927893665609266
KNN ALGORITHM
K-MEANS ALGORITHM

Figure 25. Comparison between KNN and K-mean and attack detection count ratio

This graph clearly shows the attack detection by K-mean is better than KNN. On receiving a greater number of packets, the value will vary and can be analyzed easily by an administrator.

5.4 Conclusion

The current research improved the utilization of feature grouping by selecting best features from the CICIDS2017 datasets that include 12 attack categories as Bot, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, FTP-Patator, Heartbleed, Infiltration, PortScan, SSH-Patator and Web Attack. These feature grouping is done with benign and attack ratio of 50:50 and 80:20 percent respectively and top 20 features were taken out whose features were most common. All the top 20 features from complete dataset was then compared with all 12 individual attack categories of 50:50 and 80:20 for the "Benign" vs. “Attack" data stream given only 6 most dominant common features was Bwd Packet Length Std, Flow Bytes/s, Total Length of Fwd Packets, Fwd Packet Length Std, Flow IAT Std and Flow IAT Min. Nine machine learning algorithms were employed for this purpose, namely Bagging, Naïve Bayes, QDA, Random Forest, ID3, AdaBoost, GraidentBoost, MLP and Nearest Neighbors. Bagging and Random Forest consistently performed well in terms of accuracy, precision, recall, and F1-score. Experiment shows that the Naive Bayes and QDA showed relatively stable performance while ID3, AdaBoost, and Gradient Boost had average performance in term of accuracy and precision. MLP and Nearest Neighbors had a decrease in accuracy and precision while showing varied results for recall and F1-score.

The research further extended with applying a novel approach of FSS-PART with efficient feature grouping approach. Using all 79 features of CICIDS2017 dataset, it's attack detection rate was 99.9982. Similarly, using Information Gain using 48 features detection accuracy was 99.9981, Correlation using 28 features was 99.9872 and ReF with 14 features was 99.9974.

In future, we have decided to implement similar methodology to identify attack in distributed environment and at top layer of cloud where authentication and API logs are the only ways to identify attacks while reducing the computational complexity.

Bibliography

[1] Miniwatts Marketing Group, “Internet Growth Statistics.” Accessed: Jan. 03, 2023. [Online]. Available: https://www.internetworldstats.com/emarketing.htm.

[2] S. Ganu, K. Ramachandran, M. Gruteser, I. Seskar, and J. Deng, “Methods for restoring MAC layer fairness in IEEE 802.11 networks with physical layer capture,” in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, New York, NY, USA: ACM, May 2006, pp. 7–14. doi: 10.1145/1132983.1132986.

[3] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa, “Performance of IEEE 802.11 under Jamming,” *Mobile Networks and Applications*, vol. 18, no. 5, pp. 678–696, Oct. 2013, doi: 10.1007/s11036-011-0340-4.

[4] J. T. Chiang and Y.-C. Hu, “Dynamic Jamming Mitigation for Wireless Broadcast Networks,” in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, IEEE, Apr. 2008, pp. 1211–1219. doi: 10.1109/INFOCOM.2008.177.

[5] M. Soroushnejad and E. Geraniotis, “Probability of capture and rejection of primary multiple-access interference in spread-spectrum networks,” *IEEE Transactions on Communications*, vol. 39, no. 6, pp. 986–994, Jun. 1991, doi: 10.1109/26.87188.

[6] N. Borisov, I. Goldberg, and D. Wagner, “Intercepting mobile communications,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, New York, NY, USA: ACM, Jul. 2001, pp. 180–189. doi: 10.1145/381677.381695.

[7] V. Gupta, S. Krishnamurthy, and M. Faloutsos, “Denial of service attacks at the MAC layer in wireless ad hoc networks,” in *MILCOM 2002. Proceedings*, IEEE, 2002, pp. 1118–1123. doi: 10.1109/MILCOM.2002.1179634.

[8] M. Roesch, “Snort-Lightweight Intrusion Detection for Networks,” 1999.

[9] R. Shanker and A. Singh, “Analysis of Network Attacks at Data Link Layer and its Mitigation,” in *2021 International Conference on Computing Sciences (ICCS)*, IEEE, Dec. 2021, pp. 274–279. doi: 10.1109/ICCS54944.2021.00061.

[10] Z. Trabelsi, “Switch’s CAM Table Poisoning Attack: Hands-on Lab Exercises for Network Security Education,” Australia, 2012.

[11] G. Al Sukkar, R. Saifan, S. Khwaldeh, M. Maqableh, and I. Jafar, “Address Resolution Protocol (ARP): Spoofing Attack and Proposed Defense,” *Communications and Network*, vol. 08, no. 03, pp. 118–130, 2016, doi: 10.4236/cn.2016.83012.

[12] X. Hou, Z. Jiang, and X. Tian, “The detection and prevention for ARP Spoofing based on Snort,” in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, IEEE, Oct. 2010, pp. V5-137-V5-139. doi: 10.1109/ICCASM.2010.5619113.

[13] H. Mukhtar, K. Salah, and Y. Iraqi, “Mitigation of DHCP starvation attack,” *Computers & Electrical Engineering*, vol. 38, no. 5, pp. 1115–1128, Sep. 2012, doi: 10.1016/j.compeleceng.2012.06.005.

[14] S. A. Rouiller, “Virtual LAN Security: weaknesses and countermeasures GIAC Security Essentials Practical Assignment Version 1.4b,” 2003.

[15] M. Agarwal, S. Biswas, and S. Nandi, “Detection of De-authentication Denial of Service attack in 802.11 networks,” in *2013 Annual IEEE India Conference (INDICON)*, IEEE, Dec. 2013, pp. 1–6. doi: 10.1109/INDCON.2013.6726015.

[16] J. Milliken, V. Selis, K. M. Yap, and A. Marshall, “Impact of Metric Selection on Wireless DeAuthentication DoS Attack Performance,” *IEEE Wireless Communications Letters*, vol. 2, no. 5, pp. 571–574, Oct. 2013, doi: 10.1109/WCL.2013.072513.130428.

[17] S.-Y. Chang and Y.-C. Hu, “SecureMAC: Securing Wireless Medium Access Control Against Insider Denial-of-Service Attacks,” *IEEE Trans Mob Comput*, vol. 16, no. 12, pp. 3527–3540, Dec. 2017, doi: 10.1109/TMC.2017.2693990.

[18] T. Jamal, M. Alam, and M. M. Umair, “Detection and prevention against RTS attacks in wireless LANs,” in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, IEEE, Mar. 2017, pp. 152–156. doi: 10.1109/C-CODE.2017.7918920.

[19] S. Shetty, M. Song, and L. Ma, “Rogue Access Point Detection by Analyzing Network Traffic Characteristics,” in *MILCOM 2007 - IEEE Military Communications Conference*, IEEE, Oct. 2007, pp. 1–7. doi: 10.1109/MILCOM.2007.4455018.

[20] O. Nakhila and C. Zou, “User-side Wi-Fi evil twin attack detection using random wireless channel monitoring,” in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, IEEE, Nov. 2016, pp. 1243–1248. doi: 10.1109/MILCOM.2016.7795501.

[21] M. A. Chan Aung and K. P. Thant, “Detection and mitigation of wireless link layer attacks,” in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, IEEE, Jun. 2017, pp. 173–178. doi: 10.1109/SERA.2017.7965725.

[22] V. Ramachandran and S. Nandi, “Detecting ARP Spoofing: An Active Technique,” 2005, pp. 239–250. doi: 10.1007/11593980_18.

[23] J. Bi, J. Wu, G. Yao, and F. Baker, "Source Address Validation Improvement (SAVI) Solution for DHCP," May 2015. doi: 10.17487/RFC7513.

[24] J. Zhao, J. Gu, and J. Liu, "Research on Layer 2 Attacks of 802.11-Based WLAN," 2011, pp. 503–509. doi: 10.1007/978-3-642-27503-6_69.

[25] R. O. Verma and S. S. Shriramwar, "Effective VTP Model for Enterprise VLAN Security," in *2013 International Conference on Communication Systems and Network Technologies*, IEEE, Apr. 2013, pp. 426–430. doi: 10.1109/CSNT.2013.95.

[26] Z. Balogh, Š. Koprda, and J. Francisti, "LAN security analysis and design," in *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*, IEEE, Oct. 2018, pp. 1–6. doi: 10.1109/ICAICT.2018.8746912.

[27] A. Makanju, P. LaRoche, and A. N. Zincir-Heywood, "A Comparison Between Signature and GP-Based IDSs for Link Layer Attacks on WiFi Networks," in *2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications*, IEEE, Apr. 2007, pp. 213–219. doi: 10.1109/CISDA.2007.368156.

[28] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems," *IEEE Trans Dependable Secure Comput*, vol. 10, no. 4, pp. 198–211, Jul. 2013, doi: 10.1109/TDSC.2013.8.

[29] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities," *IEEE Trans Dependable Secure Comput*, vol. 11, no. 1, pp. 30–44, Jan. 2014, doi: 10.1109/TDSC.2013.24.

[30] M. Kallitsis, S. A. Stoev, S. Bhattacharya, and G. Michailidis, "AMON: An Open Source Architecture for Online Monitoring, Statistical Analysis, and Forensics of Multi-Gigabit Streams," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1834–1848, Jun. 2016, doi: 10.1109/JSAC.2016.2558958.

[31] B. Bose, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting Insider Threats Using RADISH: A System for Real-Time Anomaly Detection in Heterogeneous Data Streams," *IEEE Syst J*, vol. 11, no. 2, pp. 471–482, Jun. 2017, doi: 10.1109/JSYST.2016.2558507.

[32] V. Matta, M. Di Mauro, M. Longo, and A. Farina, "Cyber-Threat Mitigation Exploiting the Birth–Death–Immigration Model," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 3137–3152, Dec. 2018, doi: 10.1109/TIFS.2018.2838084.

[33] A. F. Taha, J. Qi, J. Wang, and J. H. Panchal, “Risk Mitigation for Dynamic State Estimation Against Cyber Attacks and Unknown Inputs,” *IEEE Trans Smart Grid*, vol. 9, no. 2, pp. 886–899, Mar. 2018, doi: 10.1109/TSG.2016.2570546.

[34] R. S. M. Carrasco and M.-A. Sicilia, “Unsupervised intrusion detection through skip-gram models of network behavior,” *Comput Secur*, vol. 78, pp. 187–197, Sep. 2018, doi: 10.1016/j.cose.2018.07.003.

[35] Kuypers, Marshall A, Maillart, and Thomas, “An empirical analysis of cyber security incidents at a large organization,” *Department of Management Science and Engineering, Stanford University, School of Information*, vol. 30, 2016.

[36] R. Boutaba *et al.*, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Dec. 2018, doi: 10.1186/s13174-018-0087-2.

[37] J. Li *et al.*, “Feature Selection,” *ACM Comput Surv*, vol. 50, no. 6, pp. 1–45, Nov. 2018, doi: 10.1145/3136625.

[38] R.-T. Liu, N.-F. Huang, C.-H. Chen, and C.-N. Kao, “A fast string-matching algorithm for network processor-based intrusion detection system,” *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 3, pp. 614–633, Aug. 2004, doi: 10.1145/1015047.1015055.

[39] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, “Cyber intrusion detection by combined feature selection algorithm,” *Journal of Information Security and Applications*, vol. 44, pp. 80–88, Feb. 2019, doi: 10.1016/j.jisa.2018.11.007.

[40] K. El-Khatib, “Impact of Feature Reduction on the Efficiency of Wireless Intrusion Detection Systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 8, pp. 1143–1149, Aug. 2010, doi: 10.1109/TPDS.2009.142.

[41] J. Song, Z. Zhu, and C. Price, “Feature Grouping for Intrusion Detection Based on Mutual Information,” *Journal of Communications*, vol. 9, no. 12, pp. 987–993, 2014, doi: 10.12720/jcm.9.12.987-993.

[42] O. Y. Al-Jarrah, Y. Al-Hammdi, P. D. Yoo, S. Muhaidat, and M. Al-Qutayri, “Semi-supervised multi-layered clustering model for intrusion detection,” *Digital Communications and Networks*, vol. 4, no. 4, pp. 277–286, Nov. 2018, doi: 10.1016/j.dcan.2017.09.009.

[43] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[44] E.-S. M. El-Alfy and M. A. Alshammari, “Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algorithm in MapReduce,” *Simul Model Pract Theory*, vol. 64, pp. 18–29, May 2016, doi: 10.1016/j.simpat.2016.01.010.

[45] Y. Peng, Z. Wu, and J. Jiang, “A novel feature selection approach for biomedical data classification,” *J Biomed Inform*, vol. 43, no. 1, pp. 15–23, Feb. 2010, doi: 10.1016/j.jbi.2009.07.008.

[46] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, “Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset,” *Expert Syst Appl*, vol. 38, no. 5, pp. 5947–5957, May 2011, doi: 10.1016/j.eswa.2010.11.028.

[47] S. Mohammadi, H. Mirvaziri, and M. Ghazizadeh-Ahsaee, “Multivariate correlation coefficient and mutual information-based feature selection in intrusion detection,” *Information Security Journal: A Global Perspective*, vol. 26, no. 5, pp. 229–239, Sep. 2017, doi: 10.1080/19393555.2017.1358779.

[48] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine Learning for Networking: Workflow, Advances and Opportunities,” *IEEE Netw*, vol. 32, no. 2, pp. 92–99, Mar. 2018, doi: 10.1109/MNET.2017.1700200.

[49] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, NY, USA: ACM, Jun. 2005, pp. 50–60. doi: 10.1145/1064212.1064220.

[50] Jun Zhang, Chao Chen, Yang Xiang, Wanlei Zhou, and Yong Xiang, “Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 5–15, Jan. 2013, doi: 10.1109/TIFS.2012.2223675.

[51] M. Soysal and E. G. Schmidt, “Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison,” *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, Jun. 2010, doi: 10.1016/j.peva.2010.01.001.

[52] P. Bermolen and D. Rossi, “Support vector regression for link load prediction,” *Computer Networks*, vol. 53, no. 2, pp. 191–201, Feb. 2009, doi: 10.1016/j.comnet.2008.09.018.

[53] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, "Inter-data-center network traffic prediction with elephant flows," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2016, pp. 206–213. doi: 10.1109/NOMS.2016.7502814.

[54] T. Bakhshi and B. Ghita, "On Internet Traffic Classification: A Two-Phased Machine Learning Approach," *Journal of Computer Networks and Communications*, vol. 2016, pp. 1–21, 2016, doi: 10.1155/2016/2048302.

[55] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Netw*, vol. 26, no. 1, pp. 35–40, Jan. 2012, doi: 10.1109/MNET.2012.6135854.

[56] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS," in *Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data - MineNet '05*, New York, New York, USA: ACM Press, 2005, p. 197. doi: 10.1145/1080173.1080183.

[57] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA: ACM, Oct. 2006, pp. 313–326. doi: 10.1145/1177080.1177123.

[58] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "KISS: Stochastic Packet Inspection Classifier for UDP Traffic," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1505–1515, Oct. 2010, doi: 10.1109/TNET.2010.2044046.

[59] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos, "Digging into HTTPS," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA: ACM, Nov. 2010, pp. 322–327. doi: 10.1145/1879141.1879184.

[60] P. Bermolen, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Abacus: Accurate behavioral classification of P2P-TV traffic," *Computer Networks*, vol. 55, no. 6, pp. 1394–1411, Apr. 2011, doi: 10.1016/j.comnet.2010.12.004.

[61] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015, doi: 10.1109/TNET.2014.2320577.

[62] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian Neural Networks for Internet Traffic Classification," *IEEE Trans Neural Netw*, vol. 18, no. 1, pp. 223–239, Jan. 2007, doi: 10.1109/TNN.2006.883010.

[63] A. Este, F. Gringoli, and L. Salgarelli, “Support Vector Machines for TCP traffic classification,” *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, Sep. 2009, doi: 10.1016/j.comnet.2009.05.003.

[64] N. Jing, M. Yang, S. Cheng, Q. Dong, and H. Xiong, “An efficient SVM-based method for multi-class network traffic classification,” in *30th IEEE International Performance Computing and Communications Conference*, IEEE, Nov. 2011, pp. 1–8. doi: 10.1109/PCCC.2011.6108074.

[65] Y. Liu, W. Li, and Y. Li, “Network Traffic Classification Using K-means Clustering,” in *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, IEEE, Aug. 2007, pp. 360–365. doi: 10.1109/IMSCCS.2007.52.

[66] W. Yang, W. Zuo, and B. Cui, “Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Network,” *IEEE Access*, vol. 7, pp. 29891–29900, 2019, doi: 10.1109/ACCESS.2019.2895751.

[67] S. Huda *et al.*, “Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data,” *Inf Sci (N Y)*, vol. 379, pp. 211–228, Feb. 2017, doi: 10.1016/j.ins.2016.09.041.

[68] M. H. Kamarudin, C. Maple, T. Watson, and N. S. Safa, “A LogitBoost-Based Algorithm for Detecting Known and Unknown Web Attacks,” *IEEE Access*, vol. 5, pp. 26190–26200, 2017, doi: 10.1109/ACCESS.2017.2766844.

[69] Z. Guo, D. Shi, K. H. Johansson, and L. Shi, “Optimal Linear Cyber-Attack on Remote State Estimation,” *IEEE Trans Control Netw Syst*, vol. 4, no. 1, pp. 4–13, Mar. 2017, doi: 10.1109/TCNS.2016.2570003.

[70] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *2015 Military Communications and Information Systems Conference (MilCIS)*, IEEE, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.

[71] S. and B. S. D. Hettich, “The UCI KDD Archive,” University of California, Department of Information and Computer Science.

[72] K. Cho, K. Mitsuya, and A. Kato, “Traffic Data Repository at the WIDE Project,” San Diego, California: USENIX Association, 2000, p. 51.

[73] S. IMPACT, “Information Marketplace for Policy and Analysis of Cyber-Risk & Trust.” Accessed: Mar. 04, 2023. [Online]. Available: https://www.impactcybertrust.org/

[74] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," Dec. 2002. doi: 10.17487/rfc3411.

[75] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Computer Networks*, vol. 55, no. 5, pp. 1083–1099, Apr. 2011, doi: 10.1016/j.comnet.2010.11.002.

[76] Benoît Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," Jan. 2008. doi: 10.17487/rfc5101.

[77] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016, doi: 10.1109/TC.2016.2519914.

[78] V. Jyothsna and V. V. Rama Prasad, "FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale," *ICT Express*, vol. 2, no. 3, pp. 103–116, Sep. 2016, doi: 10.1016/j.icte.2016.08.003.

[79] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput Secur*, vol. 70, pp. 255–277, Sep. 2017, doi: 10.1016/j.cose.2017.06.005.

[80] Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Expert Syst Appl*, vol. 88, pp. 249–257, Dec. 2017, doi: 10.1016/j.eswa.2017.07.005.

[81] T. H. Divyasree and K. K. Sherly, "A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach," *Procedia Comput Sci*, vol. 143, pp. 442–449, 2018, doi: 10.1016/j.procs.2018.10.416.

[82] K. Selvakumar *et al.*, "Intelligent temporal classification and fuzzy rough set-based feature selection algorithm for intrusion detection system in WSNs," *Inf Sci (N Y)*, vol. 497, pp. 77–90, Sep. 2019, doi: 10.1016/j.ins.2019.05.040.

[83] K. Selvakumar, L. Sairamesh, and A. Kannan, "Wise intrusion detection system using fuzzy rough set-based feature extraction and classification algorithms," *International Journal of Operational Research*, vol. 35, no. 1, p. 87, 2019, doi: 10.1504/IJOR.2019.099545.

[84] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset," *J Phys Conf Ser*, vol. 1192, p. 012018, Mar. 2019, doi: 10.1088/1742-6596/1192/1/012018.

[85] M. Prasad, S. Tripathi, and K. Dahal, “An efficient feature selection based Bayesian and Rough set approach for intrusion detection,” *Appl Soft Comput*, vol. 87, p. 105980, Feb. 2020, doi: 10.1016/j.asoc.2019.105980.

[86] S. Hosseini and H. Seilani, “Anomaly process detection using negative selection algorithm and classification techniques,” *Evolving Systems*, vol. 12, no. 3, pp. 769–778, Sep. 2021, doi: 10.1007/s12530-019-09317-1.

[87] S. Alabdulwahab and B. Moon, “Feature Selection Methods Simultaneously Improve the Detection Accuracy and Model Building Time of Machine Learning Classifiers,” *Symmetry (Basel)*, vol. 12, no. 9, p. 1424, Aug. 2020, doi: 10.3390/sym12091424.

[88] I. F. Kilincer, F. Ertam, and A. Sengur, “Machine learning methods for cyber security intrusion detection: Datasets and comparative study,” *Computer Networks*, vol. 188, p. 107840, Apr. 2021, doi: 10.1016/j.comnet.2021.107840.

[89] J. CANNADY, “Artificial neural networks for misuse detection,” *Proc. of the 1998 National Information Systems Security Conference*, pp. 443–456, 1998, Accessed: Apr. 15, 2023. [Online]. Available: https://cir.nii.ac.jp/crid/1572261551239315584.bib?lang=en

[90] B. Pfahringer, “Winning the KDD99 classification cup,” *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 65–66, Jan. 2000, doi: 10.1145/846183.846200.

[91] Zhi-Song Pan, Song-Can Chen, Gen-Bao Hu, and Dao-Qiang Zhang, “Hybrid neural network and C4.5 for misuse detection,” in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, IEEE, May 2003, pp. 2463–2467. doi: 10.1109/ICMLC.2003.1259925.

[92] Mehdi Moradi and Mohammad Zulkernine, “A neural network-based system for intrusion detection and classification of attacks,” *IEEE Lux-embourg-Kirchberg, Luxembourg*, pp. 15–18, Nov. 2004.

[93] S. Chebrolu, A. Abraham, and J. P. Thomas, “Feature deduction and ensemble design of intrusion detection systems,” *Comput Secur*, vol. 24, no. 4, pp. 295–307, Jun. 2005, doi: 10.1016/j.cose.2004.09.008.

[94] N. Ben Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs decision trees in intrusion detection systems,” in *Proceedings of the 2004 ACM symposium on Applied computing*, New York, NY, USA: ACM, Mar. 2004, pp. 420–424. doi: 10.1145/967900.967989.

[95] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, “Decision tree classifier for network intrusion detection with GA-based feature selection,” in *Proceedings of the 43rd annual*

Southeast regional conference - Volume 2, New York, NY, USA: ACM, Mar. 2005, pp. 136–141. doi: 10.1145/1167253.1167288.

[96] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 114–132, Jan. 2007, doi: 10.1016/j.jnca.2005.06.003.

[97] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Comput Commun*, vol. 34, no. 18, pp. 2227–2235, Dec. 2011, doi: 10.1016/j.comcom.2011.07.001.

[98] S. T. Miller and C. Busby-Earle, "Multi-Perspective Machine Learning a Classifier Ensemble Method for Intrusion Detection," in *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, New York, NY, USA: ACM, Jan. 2017, pp. 7–12. doi: 10.1145/3036290.3036303.

[99] Y. Li and L. Guo, "An active learning based TCM-KNN algorithm for supervised network intrusion detection," *Comput Secur*, vol. 26, no. 7–8, pp. 459–467, Dec. 2007, doi: 10.1016/j.cose.2007.10.002.

[100] D. Kshirsagar and S. Kumar, "An efficient feature reduction method for the detection of DoS attack," *ICT Express*, vol. 7, no. 3, pp. 371–375, Sep. 2021, doi: 10.1016/j.icte.2020.12.006.

[101] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020, doi: 10.1109/ACCESS.2020.3009843.

[102] K. Kurniabudi, D. Stiawan, D. Darmawijoyo, M. Y. Bin Idris, B. Kerim, and R. Budiarto, "Important Features of CICIDS-2017 Dataset For Anomaly Detection in High Dimension and Imbalanced Class Dataset," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 9, no. 2, May 2021, doi: 10.52549/ijeei.v9i2.3028.

[103] G. T. Habtamu, "Development of a Method for Detecting Network Attack on Machine Learning Algorithms," *SSRN Electronic Journal*, 2020, doi: 10.2139/ssrn.4106481.

[104] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Comput Surv*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[105] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Syst Appl*, vol. 41, no. 4, pp. 1690–1700, Mar. 2014, doi: 10.1016/j.eswa.2013.08.066.

[106] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit*, vol. 36, no. 1, pp. 229–243, Jan. 2003, doi: 10.1016/S0031-3203(02)00026-2.

[107] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput Secur*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.

[108] A. Thakkar and R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets," *Procedia Comput Sci*, vol. 167, pp. 636–645, 2020, doi: 10.1016/j.procs.2020.03.330.

[109] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Comput Stat Data Anal*, vol. 143, p. 106839, Mar. 2020, doi: 10.1016/j.csda.2019.106839.

[110] Al Tobi and Duncan, "Improving Intrusion Detection Model Prediction by Threshold Adaptation," *Information*, vol. 10, no. 5, p. 159, Apr. 2019, doi: 10.3390/info10050159.

[111] R. Vaarandi, "Real-time classification of IDS alerts with data mining techniques," in *MILCOM 2009 - 2009 IEEE Military Communications Conference*, IEEE, Oct. 2009, pp. 1–7. doi: 10.1109/MILCOM.2009.5379762.

[112] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, IEEE Comput. Soc, pp. 209–216. doi: 10.1109/SAINT.2003.1183050.

[113] G. C. Tjhai, M. Papadaki, S. M. Furnell, and N. L. Clarke, "The Problem of False Alarms: Evaluation with Snort and DARPA 1999 Dataset," in *Trust, Privacy and Security in Digital Business*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 139–150. doi: 10.1007/978-3-540-85735-8_14.

[114] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An Evaluation Framework for Intrusion Detection Dataset," in *2016 International Conference on Information Science and Security (ICISS)*, IEEE, Dec. 2016, pp. 1–6. doi: 10.1109/ICISSEC.2016.7885840.

[115] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[116] M. S. Al-Daweri, K. A. Zainol Ariffin, S. Abdullah, and M. F. E. Md. Senan, “An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System,” *Symmetry (Basel)*, vol. 12, no. 10, p. 1666, Oct. 2020, doi: 10.3390/sym12101666.

[117] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, “KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research,” *Computer (Long Beach Calif)*, vol. 52, no. 2, pp. 41–51, Feb. 2019, doi: 10.1109/MC.2018.2888764.

[118] M. A. Siddiqi and W. Pak, “Optimizing Filter-Based Feature Selection Method Flow for Intrusion Detection System,” *Electronics (Basel)*, vol. 9, no. 12, p. 2114, Dec. 2020, doi: 10.3390/electronics9122114.

[119] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, “Towards a Reliable Intrusion Detection Benchmark Dataset,” *Software Networking*, vol. 2017, no. 1, pp. 177–200, Jan. 2017, doi: 10.13052/jsn2445-9739.2017.009.

[120] N. Meti, D. G. Narayan, and V. P. Baligar, “Detection of distributed denial of service attacks using machine learning algorithms in software defined networks,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Sep. 2017, pp. 1366–1371. doi: 10.1109/ICACCI.2017.8126031.

[121] Z. Chen *et al.*, “A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures,” *Big Data Research*, vol. 3, pp. 10–23, Apr. 2016, doi: 10.1016/j.bdr.2015.11.002.

[122] C. Birkinshaw, E. Rouka, and V. G. Vassilakis, “Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks,” *Journal of Network and Computer Applications*, vol. 136, pp. 71–85, Jun. 2019, doi: 10.1016/j.jnca.2019.03.005.

[123] M. Almseidin, M. Al-Kasassbeh, and S. Kovacs, “Detecting Slow Port Scan Using Fuzzy Rule Interpolation,” in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, IEEE, Oct. 2019, pp. 1–6. doi: 10.1109/ICTCS.2019.8923028.

[124] K. Bicakci and B. Tavli, “Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks,” *Comput Stand Interfaces*, vol. 31, no. 5, pp. 931–941, Sep. 2009, doi: 10.1016/j.csi.2008.09.038.

[125] T. Shorey, D. Subbaiah, A. Goyal, A. Sakxena, and A. K. Mishra, “Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools,” in *2018 International Conference on Advances in Computing, Communications and*

Informatics (ICACCI), IEEE, Sep. 2018, pp. 318–322. doi: 10.1109/ICACCI.2018.8554590.

[126] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, “Machine Learning for Detecting Brute Force Attacks at the Network Level,” in *2014 IEEE International Conference on Bioinformatics and Bioengineering*, IEEE, Nov. 2014, pp. 379–385. doi: 10.1109/BIBE.2014.73.

[127] S. Tayama and H. Tanaka, “Analysis of Slow Read DoS Attack and Communication Environment,” in *Mobile and Wireless Technologies 2017*, Springer, Singapore., 2018, pp. 350–359. doi: 10.1007/978-981-10-5281-1_38.

[128] A. Dimitriadis, J. L. Flores, B. Kulvatunyou, N. Ivezic, and I. Mavridis, “ARES: Automated Risk Estimation in Smart Sensor Environments,” *Sensors*, vol. 20, no. 16, p. 4617, Aug. 2020, doi: 10.3390/s20164617.

[129] P. Likarish, E. Jung, and I. Jo, “Obfuscated malicious javascript detection using classification techniques,” in *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE, Oct. 2009, pp. 47–54. doi: 10.1109/MALWARE.2009.5403020.

[130] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, “Heartbleed 101,” *IEEE Secur Priv*, vol. 12, no. 4, pp. 63–67, Jul. 2014, doi: 10.1109/MSP.2014.66.

List of publications:

1. "*Analysis of information security service for internet application"* Ravi Shanker, Dr. Kavita, Dr. Sahil Verma, International Journal of Engineering & Technology 7 (12), 58-62

2. *"To Enhance the Security in Wireless Nodes using Centralized and Synchronized IDS Technique"* Ravi Shanker, et al., Indian Journal of Science and Technology 9 (32)

3. *"Efficient Feature Grouping for IDS Using Clustering Algorithms in Detecting Known/Unknown Attacks"* in Information Security Handbook 1st Edition by CRC press.

4. *"Analysis of Network Attacks at Data Link Layer and its Mitigation."*, Ravi Shanker et al. 2021 International Conference on Computing Sciences (ICCS). IEEE, 2021.

5. *"Framework for identifying network attacks through packet inspection using machine learning"*, in Nonlinear Engineering. Modeling and Application (Degruyter), 2023.

6. *"FSS-PART: Feature Grouping Subset Model for Predicting Network Attacks"*, in SN Computer Science.

7. *"Machine Learning enabled Security Parameter selection to Identify Attacks on Cloud and Host"*, in ICCET 2023.