P ISSN : 2618-1630
E ISSN : 2909-6130

Analyzing ML-based IDS over Real-Traffic

Shafqat Ali Siyyal[1]*, Faheem Yar Khuawar[1], Erum Saba[2], Abdul Latif Memon[1], Muhammad Raza Shaikh[1]

[1]Department of Telecommunication, Mehran University of Engineering and Technology Jamshoro, Pakistan.

[2]Information Technology Center, Sindh Agriculture University, Tandojam, Pakistan

* Corresponding: Shafqat Ali Siyyal, Email: mshafqat00@gmail.com

Citation | M. R. S. Shafqat Ali Siyyal, Faheem Yar Khuawar, Erum Saba, Abdul Latif Memon, "Analyzing ML-based IDS over Real-Traffic," Int. J. Innov. Sci. Technol., vol. 4, no. 3, pp. 621–640, 2022

Received | May 28, 2022; Revised | June 20, 2022; Accepted | June 28, 2022; Published |June 30, 2022.

The rapid growth of computer networks has caused a significant increase in malicious traffic, promoting the use of Intrusion Detection Systems (IDSs) to protect against this ever-growing attack traffic. A great number of IDS have been developed with some sort of weaknesses and strengths. Most of the development and research of IDS is purely based on simulated and non-updated datasets due to the unavailability of real datasets, for instance, KDD '99, and CIC-IDS-18 which are widely used datasets by researchers are not sufficient to represent real-traffic scenarios. Moreover, these one-time generated static datasets cannot survive the rapid changes in network patterns. To overcome these problems, we have proposed a framework to generate a full feature, unbiased, real-traffic-based, updated custom dataset to deal with the limitations of existing datasets. In this paper, the complete methodology of network testbed, data acquisition and attack scenarios are discussed. The generated dataset contains more than 70 features and covers different types of attacks, namely DoS, DDoS, Portscan, Brute-Force and Web attacks. Later, the custom-generated dataset is compared to various available datasets based on seven different factors, such as updates, practical-to-generate, realness, attack diversity, flexibility, availability, and interoperability. Additionally, we have trained different ML-based classifiers on our custom-generated dataset and then tested/analyzed it based on performance metrics. The generated dataset is publicly available and accessible by all users. Moreover, the following research is anticipated to allow researchers to develop effective IDSs and real traffic-based updated datasets.

Keywords: Anomaly-based, IDS, Dataset Generation Method, Attack Traffic, Normal Traffic Machine Learning

Acknowledgment. The author would like to express my special thanks of gratitude to my supervisor Dr Faheem Yar Khuawar who helped me throughout this research and I came to know about so many new things.

Project details. Nil

CONFLICT OF INTEREST: The author(s) declare that the publication of this article has no conflict of interest.

Author's Contribution. All the authors contributed equally

Introduction

IDS plays a significant role as a defense tool for networks and systems that forewarn security administrators for abnormal behaviors such as intrusions or malware traffic [1]. Due to the ever-increasing intrusion activities, many researchers have drawn attention to improving the performance of IDS. Over the years, work in this domain has flourished, and many researchers have proposed Machine-Learning based IDS; however, there is still a big gap for the researchers to find a valid, comprehensive dataset. Before deploying the IDS in a real environment, it must be trained and analyzed using a real, updated, labeled dataset, which should contain the intensive range of attacks or intrusions. This task is challenging itself as not many such datasets are present [2]. As a consequence, IDSs are depending on these publicly available datasets that do not reflect the ground truth, current trends, or lack of attack diversity or updated patterns. For these reasons, a perfect benchmark dataset is required [3]. Moreover, generating a perfect dataset is not enough due to continuous changes in traffic patterns and malware evolutions. So, to cope with it, this paper provides a complete method that users can follow to generate a new, updated dataset each time they need it.

The research is based on two parts. Firstly, we worked on generating a new benchmark dataset that contains a different range of actual attacks such as Distributed Denial of Service (DDoS) [4], Denial of Service (DoS) [5], Portscan [6], Brute-force [7], and web attacks [8]. We also have separately captured the normal traffic, which is based on day-to-day activities. In the second part of our work, we have trained three different Machine Learning (ML)-based classifiers, such as Support Vector Machines (SVM) [9], Decision Tree [10], and Naïve Bayes [11] using our custom-generated dataset. Later, we tested and evaluated the performance of mentioned ML-based classifiers.

This paper is organized as follows: An overview of different datasets has been discussed in the section "Available Dataset" which provides detailed information about previous works and datasets based on their popularity and flaws to understand the need for reliable and authentic datasets. In section "Material & Methods", we have discussed the dataset creation methodology, including network configurations, attack scenarios, processing of data, and their related tools. Section "Results & Discussion" represents the performance analysis of different classifiers as well as provides information about comparative analysis of different available datasets versus our custom-generated dataset. Finally, section "Conclusion & Future work" discusses the conclusion of the whole research and future work.

Dataset

The good quality of the dataset give researchers the ability to focus on improving the performance of IDS. Although there are many datasets available, which are valuable for the research community in developing algorithms, performance comparison or finding relevant features [12]. However, our objectives are different and cannot be achieved with these available datasets. Following is the list of datasets publicly available.

1. DARPA 98 & 99. It [13] was generated by MIT's lab of Lincoln with expectations for offline detection of intrusions; despite its age, their dataset is still widely used by the community. The dataset contains a different range of attacks, from privilege access to network scans. It contains five weeks of traffic with two weeks of regular traffic that is simulated that does not prove ground truths and contains irregularities. The dataset contains one week of labeled attacks and two weeks of unlabeled mixed traffic. Dataset is aged enough that it cannot cope in terms of both network infrastructures and attack types [14].

2. KDD '99. It [15] was developed in 1999, known as an updated version of DARPA as it contains the same extracted flows of the DARPA dataset by processing it through TCPdump. Dataset's size is around 4 gigabytes of TCP flows of normal and malicious traffic collected in 7 weeks. Attack traffic consists of 24 types of attacks and falls into four categories such as Probing, DoS, U2L, and R2L. According to a report [16], KDD'99 is the most used dataset for IDS evaluation during the years 2010-2015; around 125 published papers used this dataset but if we analyze it carefully, this dataset comes with many flaws, such as the dataset contains a significant portion of the redundant values; moreover, the dataset is not updated with the current trends. In comparison, many researchers argued that this dataset is perfect for benchmarking and limited their work specific to the KDD99 [17].

3. Kyoto. The dataset [18] was created at Kyoto University in 2006, and different versions of the dataset were released till 2015. The Kyoto dataset used the honeypot technique to capture the attack traffic and used the Zeek tool to extract the features. This dataset is based on 24 features, including 14 features same as KDD '99. They simulate the normal traffic and restrict their work to mailing and DNS traffic data, which seems insufficient. Moreover, no information is mentioned about the payload and how the labeling of the dataset is conducted [19].

4. ISCX 2012. It [20] was created by New Brunswick University in 2012. The dataset contains two parts, the alpha, which includes benign traffic generated using tor, and realistic networks. It includes a variety of traffic such as HTTP, SSH, SMTP, POP3, IMAP, and FTP protocols with their payloads. However, the dataset has not been updated since then and is missing nearly 70% of today's traffic. Furthermore, the attacks included in the dataset are simulated that do not prove the real-world statistics [21].

5. The UNSW-NB15. The dataset [22] was developed by the Australian Centre for Cyber Security (ACCS). The dataset was created using different commercial tools for simulating normal and malicious traffic. Traffic was captured using Tcpdump, with a total duration of 31 hours. Moreover, nine different types of attacks are included with labels. Further, the dataset includes around 49 features divided into five categories: basic features, flow features, time features, content features, and some other additional features. The dataset is available in .pcap format as well as in .csv format in the same style as KDD '99.

6. CIC-IDS-2018. The dataset [12] was developed by the Canadian Institute for Cyber Security at the University of Brunswick. The dataset was mainly created for intrusion detections that consist of a wide range of attack scenarios. Different behaviors of multiple users on the internet were captured, such as HTTP, HTTPS, SSH, FTP, and SMTP, which show traffic diversity and different attack ranges such as DoS, Probing attack, User to Root (U2R), and Root to Local (R2L) were generated using different tools which are publicly available. The dataset is generally preferred by researchers to apply for feature engineering as it proves authentic traces of real traffic [23]. However, the major limitation of the dataset is they have not updated it since its publication; moreover, its generating methods are not defined clearly.

7. Requirements for Suitable Dataset. Different datasets and their flaws were discussed in the previous section that represents different objectives and scopes. By reviewing the existing work, different requirements have been derived that should be fulfilled to cover the existing gap and add complement to the research of the IDS domain. The requirements are listed below with some descriptions, such as:

i. Ground Truth. The dataset must contain the realistic traffic captured from actual networks compared to synthetically generated and ensures that all data points must be correctly labeled [24].

ii. Practical to generate. The dataset should be publicly available with its complete methodology as it will allow users to generate their new dataset based on their requirements [25].

iii. Updates. The majority of datasets lack this feature as network traffic is not static; it continuously changes with time, so updating the dataset periodically with newer traffic will improve the performance of IDS [26].

iv. Attack Diversity. Attacks are evolving rapidly with time. Therefore, working with a wide range of newer attacks is the top priority of researchers [25].

v. Flexibility. The dataset should contain a variety of traffics such as HTTP, HTTPS, FTP, SSH, DNS, SMTP, and so on since analysts use it for the different objectives and scopes. So, the dataset should be flexible to be used for different scenarios [27].

vi. Interoperability. The dataset should be available in a widespread format such as a .pcap file or .csv file for analyst's ease.

vii. Publicly Present. The dataset should be publicly available without any formal requirements from the authors. Most of the available datasets based on real scenarios are not easily accessible by researchers due to privacy concerns and require some formalities [28].

MATERIAL AND METHODS

This section of the paper provides detailed information on how a dataset is created, which includes methods, tools, codes and network configuration. Figure. 1 shows the diagram of flow of methodology that we have followed to achieve our objectives.

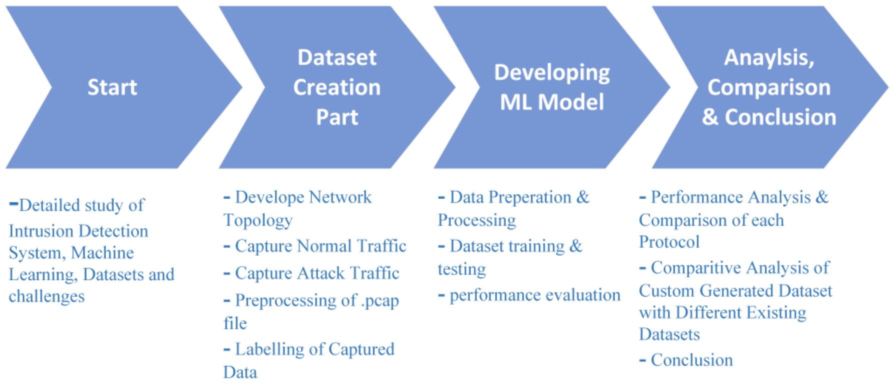

Start
Dataset Creation Part
Developing ML Model
Anaylsis, Comparison & Conclusion
-Detailed study of Intrusion Detection System, Machine Learning, Datasets and challenges
- Develope Network Topology
- Capture Normal Traffic
- Capture Attack Traffic
- Preprocessing of .pcap file
- Labelling of Captured Data
- Data Preperation & Processing
- Dataset training & testing
- performance evaluation
- Performance Analysis & Comparison of each Protocol
- Comparitive Analysis of Custom Generated Dataset with Different Existing Datasets
- Conclusion

Figure 1. Flow Diagram of Methodology

The total duration of the experiment is five days, as shown in Table. 1, which starts from Monday to Friday and each scenario such as: capturing the normal and attack traffic is divided into different days. We have used the Wireshark tool [29] for capturing network traffic in .pcap format at the attacker's side. Further detail for each scenario is described in the following sections.

Table 1. Distribution of Normal and Attack Traffic

Days	Labels
Monday	Normal Traffic
Tuesday	DoS Attack: GoldenEye, Hulk, Slowloris, SlowHttpTest
Wednesday	Brute Force: SSH, FTP & Normal Traffic

Thursday	DDoS: Synflood & LOIC Attack
Friday	Portscan Attack: Nmap & Web Attack: Burp Suite & Normal traffic

1. Network Configuration. The process starts with the infrastructure of the network, Figure. 2 shows the complete configuration of the network. We have used 5 machines: 2 Kali Linux machines, 1 windows machine, 1 Ubuntu-based Metasploitable 2, and a web server. Each machine is connected using a switch. Both Kali machines are chosen for performing attacks as they provide over 600 penetration tools [30]. While remaining machines are considered as the victim. Victim 1 is Metasploitable 2 [31], an intentionally vulnerable virtual machine that comes with 3 security levels low, medium, and impossible. While the web server is being set up on Metasploitable 2, which provides different login pages and some application layer services such as HTTP, HTTPS, FTP, SSH, etc. The complete topology shown is configured using VirtualBox [32].

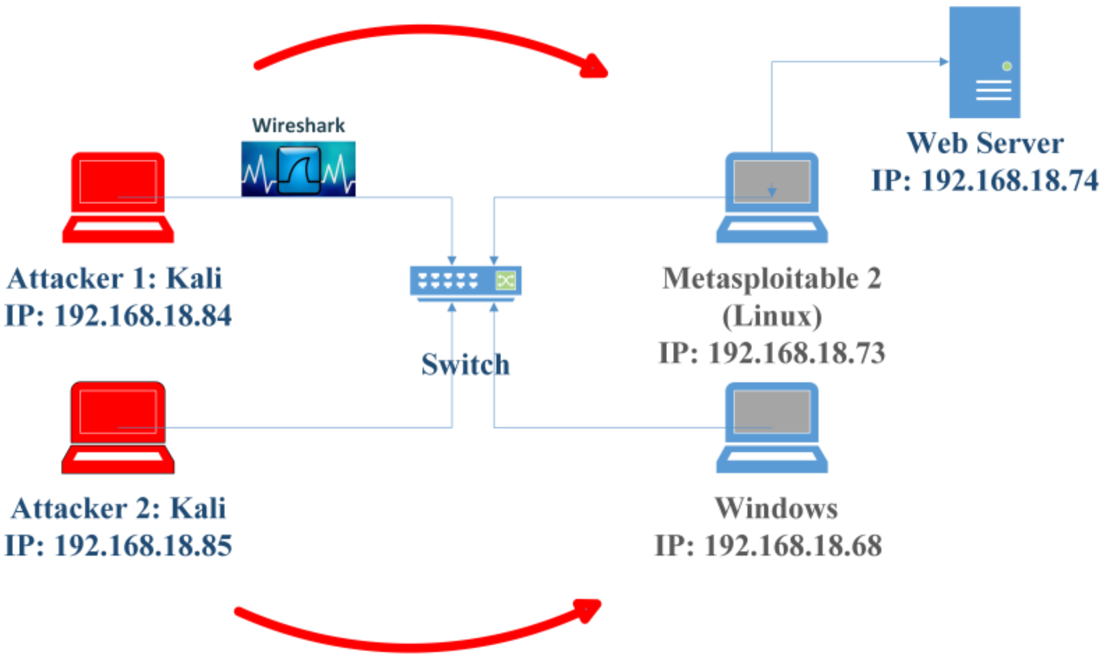
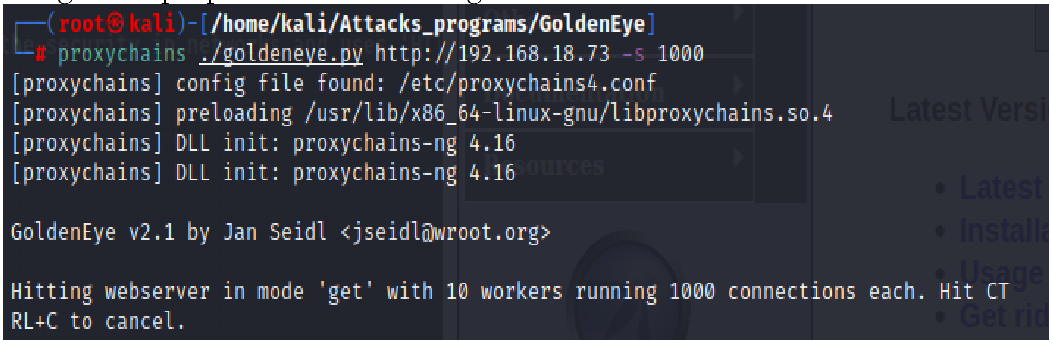


Figure 2. Network Topology of Virtual Testbed

2. Normal Profile. Working with realistic traffic is one of the priorities of this research. To achieve it, we have captured the complete network traffic of a user on a windows machine for three different days at different times. Captured traffic includes routine-based activities, such as surfing the internet, attempting logins on different web pages, transferring files, or sending emails that show the variety of traffic, for instance, HTTP, HTTPS, FTP, or mailing protocols. We have set up an antivirus and IDS tool to ensure that normal traffic does not contain any intrusions or malicious traffic.

3. Attack Profiles. Since the paper intends to provide network security and intrusion detection, it should provide a diverse range of attacks. Below, we have defined the list of common attacks, their related tools, and the codes to execute them. Each attack is performed using Kali Linux, so most attacking tools are pre-installed, or they can easily be found on GitHub [33]. Mainly, these attacks are based on the CLI method and are easier to use.

i. DoS Attack. For generating the DoS attack, we have used 4 different tools based on their specification such as GoldenEye, Hulk, SlowHttptest, and Slowloris. These tools can be easily accessible from GitHub. Using GoldenEye [34], a single machine is enough to put down another machine, which tries to flood legitimate HTTP traffic to overwhelm web resources by frequently requesting multiple URLs. Before generating this attack, we have started the Tor service to anonymize the attacker and simply typing ./goldeneye.py -h gives you the detail of parameters to be inserted. Figure. 3 shows that we have started the attack on IP: 192.168.18.73 with 10 workers generating 1000 requests each time. While proxy chain command is used to go through multiple proxies to avoid being identified.



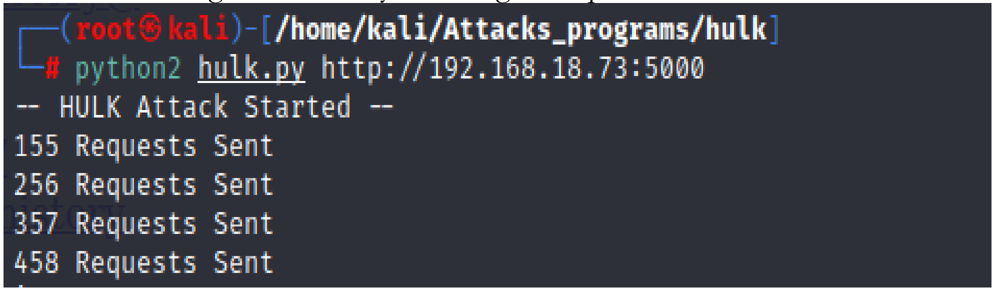
```
┌──(root㉿kali)-[/home/kali/Attacks_programs/GoldenEye]
└─# proxychains ./goldeneye.py http://192.168.18.73 -s 1000
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>

Hitting webserver in mode 'get' with 10 workers running 1000 connections each. Hit CT
RL+C to cancel.
```

Figure 3. GoldenEye DoS Attack

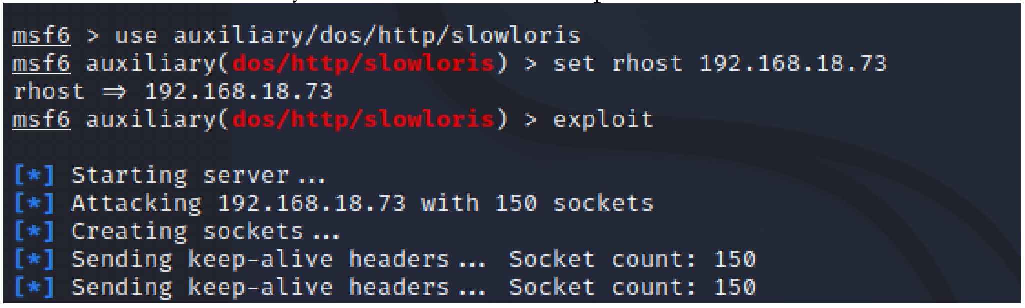
Hulk [35] Attack differs from goldeneye because it generates unique patterns on each request which helps in avoiding its detection. Figure. 4, shows the hulk attack generated on server 192.168.18.73:5000 by simply running the command hulk.py using python and each time attacker is stressing the victim by doubling its request volume.



```
┌──(root㉿kali)-[/home/kali/Attacks_programs/hulk]
└─# python2 hulk.py http://192.168.18.73:5000
-- HULK Attack Started --
155 Requests Sent
256 Requests Sent
357 Requests Sent
458 Requests Sent
```

Figure 4. Hulk DoS Attack

SlowLoris [36] DoS is an attack known for its lower bandwidth consumption with higher impact. The tool starts the partial requests to a server and tries to maintain connection as long as possible. We have used the slowloris module provided by the Metasploit application, built-in Kali Linux. Figure. 5 shows how we have set up the target for an attack, where the socket count shows the number of sockets used during an attack. After each interval, these keep-alive headers are sent by the attacker to make a persistent connection with the host.

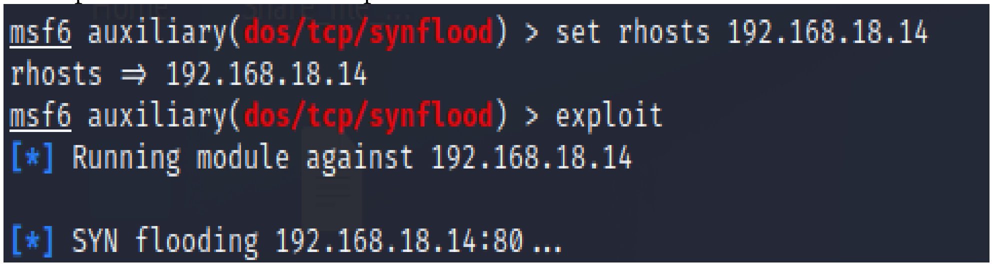


```
msf6 > use auxiliary/dos/http/slowloris
msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.18.73
rhost => 192.168.18.73
msf6 auxiliary(dos/http/slowloris) > exploit

[*] Starting server...
[*] Attacking 192.168.18.73 with 150 sockets
[*] Creating sockets...
[*] Sending keep-alive headers... Socket count: 150
[*] Sending keep-alive headers... Socket count: 150
```

Figure 5. SlowLoris DoS Attack

ii. DDoS Attack. Involves multiple systems, collectively called botnets that try to overwhelm the target by simultaneously attacking at the same time. SynFlood tool [37] is used simultaneously on both kali machines that bombards thousands of TCP connection requests without replying to corresponding acknowledgement. We have used the synflood module provided by the Metasploit tool. Attached Figure. 6 shows an attack from a single source, while the impact on the victim is dependent on the number of attackers.

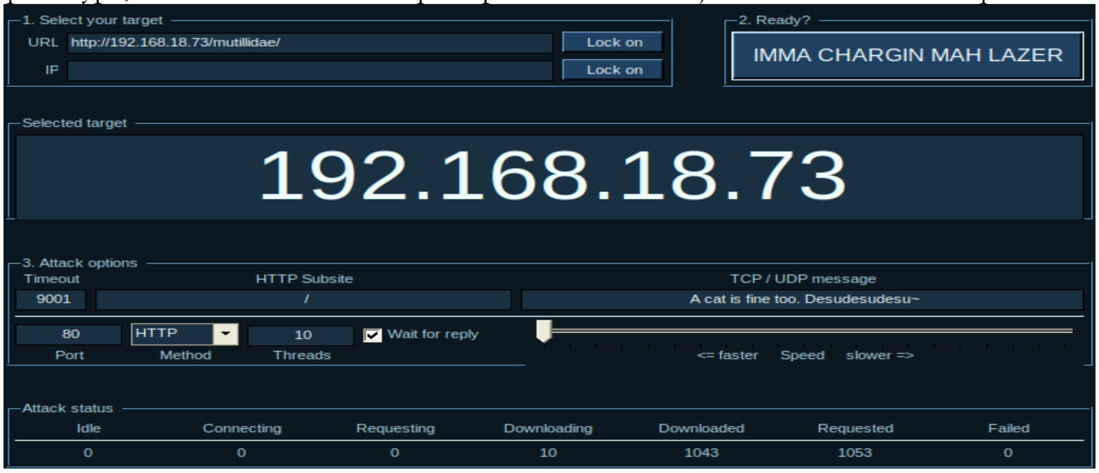


```
msf6 auxiliary(dos/tcp/synflood) > set rhosts 192.168.18.14
rhosts ⇒ 192.168.18.14
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.18.14

[*] SYN flooding 192.168.18.14:80 ...
```

Figure 6. SynFlood DDoS Attack

LOIC [38], short for low orbit ion cannon, is another GUI-based tool used for DDoS attacks that are capable of generating three different types of requests: TCP, UDP, and HTTP. In Figure. 7, target is set for an attack by sending HTTP requests. Parameters such as port no., request type, and transfer rate of request packets can be adjusted based on the requirements.

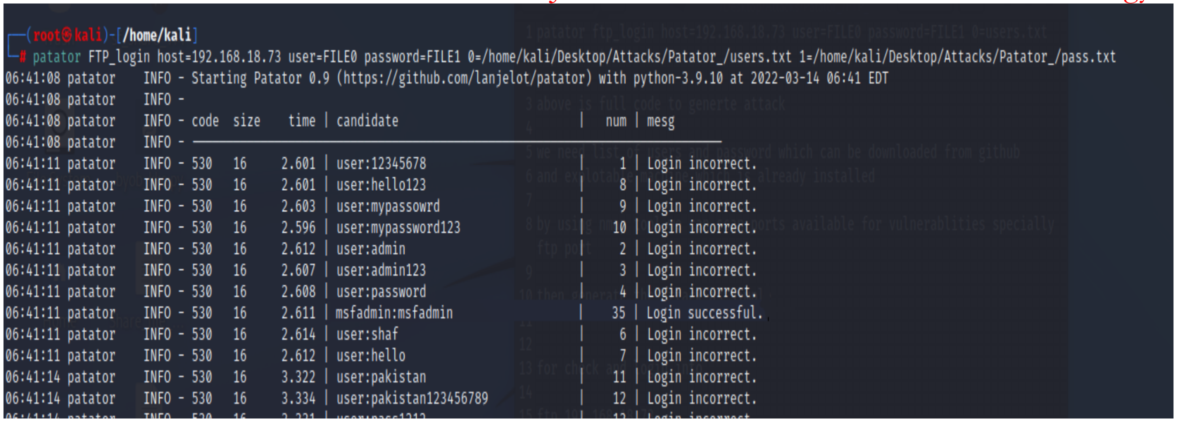


1. Select your target
URL http://192.168.18.73/mutillidae/
IP
2. Ready?
Selected target
192.168.18.73
3. Attack options
Timeout 9001
HTTP Subsite /
TCP / UDP message A cat is fine too. Desudesudesu~
Port 80
Method HTTP
Threads 10
[✓] Wait for reply
Attack status

Idle	Connecting	Requesting	Downloading	Downloaded	Requested	Failed
0	0	0	10	1043	1053	0

Figure 7. LOIC DDoS Attack

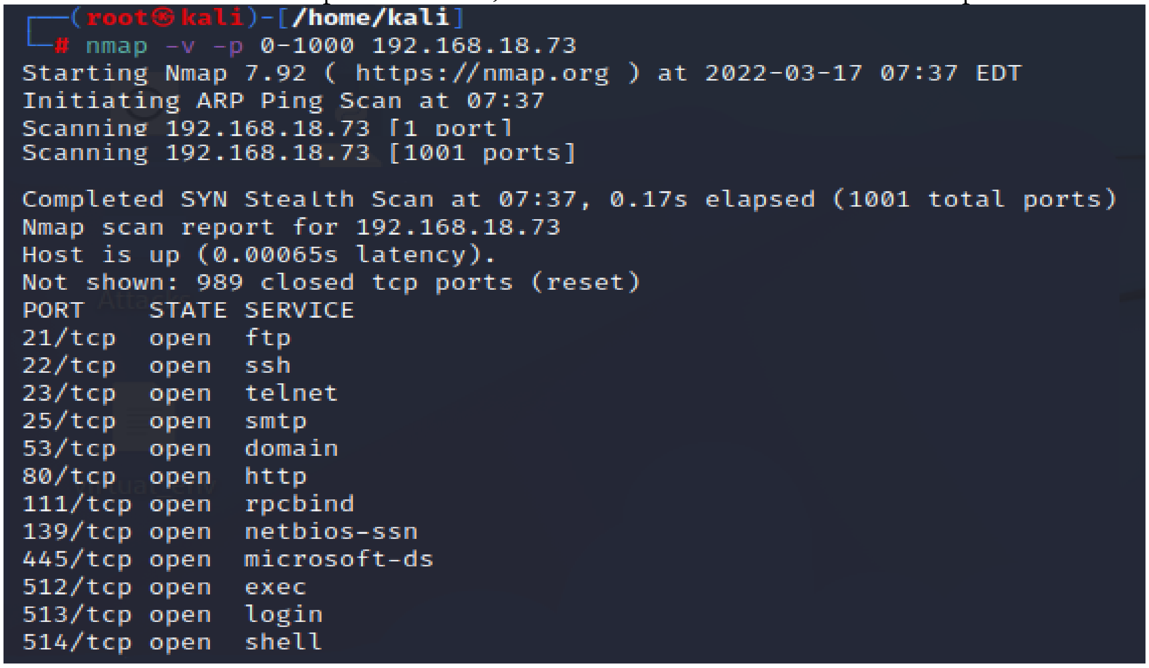
iii. Brute Force Attack. In this attack, attacker tries to guess the login information using the hit and try the method. According to [39], most people prefer to choose simpler and more common passwords such as their names, date-of-births or "12345", "passwords," "admin," etc., which can be guessed easily. Several tools are available to perform Brute force attacks, such as Patator, Hydra, Ncrack, Medusa, Nmap NSE scripts, and Metasploit modules. We have used Patator [40] because of its simplicity and reliability, as it provides a separate log file for each response that can be viewed later. Moreover, the Patator tool can be used on more than 30 various applications such as SSH, FTP, Telnet, SMTP, and so on. In our case, we have set up an FTP & SSH vulnerability on our Metasploitable 2 machine; we have executed the Patator shown in Figure. 8. Before generating an attack, a list of common usernames & passwords is provided separately in text format.



```
┌─(root㉿kali)-[/home/kali]
└─# patator FTP_login host=192.168.18.73 user=FILE0 password=FILE1 0=/home/kali/Desktop/Attacks/Patator_/users.txt 1=/home/kali/Desktop/Attacks/Patator_/pass.txt
06:41:08 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.9.10 at 2022-03-14 06:41 EDT
06:41:08 patator    INFO -
06:41:08 patator    INFO - code  size    time | candidate                          |   num | mesg
06:41:08 patator    INFO - ----------------------------------------------------------------------
06:41:11 patator    INFO - 530   16     2.601 | user:12345678                      |     1 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.601 | user:hello123                      |     8 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.603 | user:mypassowrd                    |     9 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.596 | user:mypassword123                 |    10 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.612 | user:admin                         |     2 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.607 | user:admin123                      |     3 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.608 | user:password                      |     4 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.611 | msfadmin:msfadmin                  |    35 | Login successful.
06:41:11 patator    INFO - 530   16     2.614 | user:shaf                          |     6 | Login incorrect.
06:41:11 patator    INFO - 530   16     2.612 | user:hello                         |     7 | Login incorrect.
06:41:14 patator    INFO - 530   16     3.322 | user:pakistan                      |    11 | Login incorrect.
06:41:14 patator    INFO - 530   16     3.334 | user:pakistan123456789             |    12 | Login incorrect.
```

Figure 8. Brute Forcing: FTP Attack

iv. Portscan Attack. It is a common technique used by hackers to scan and find vulnerabilities in the target machine. Nmap [41] is one of the famous tools used for scanning, pre-installed in Kali Linux. It helps identify the users on a network, their open ports, and services. Figure. 9 shows that scanning is performed on a victim within the specified ports from 0-1000, and the result shows different port numbers, and their services are available for exploitation.

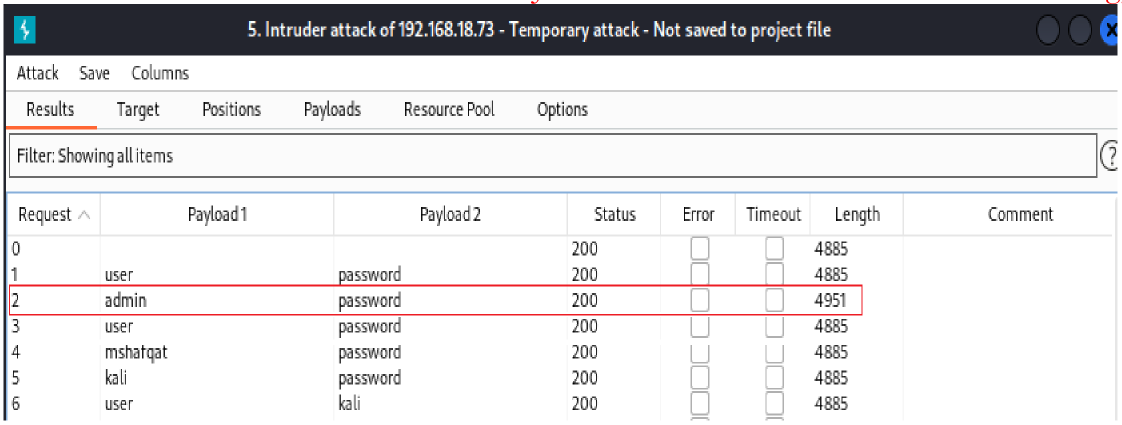


```
┌─(root㉿kali)-[/home/kali]
└─# nmap -v -p 0-1000 192.168.18.73
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-17 07:37 EDT
Initiating ARP Ping Scan at 07:37
Scanning 192.168.18.73 [1 port]
Scanning 192.168.18.73 [1001 ports]

Completed SYN Stealth Scan at 07:37, 0.17s elapsed (1001 total ports)
Nmap scan report for 192.168.18.73
Host is up (0.00065s latency).
Not shown: 989 closed tcp ports (reset)
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
23/tcp  open  telnet
25/tcp  open  smtp
53/tcp  open  domain
80/tcp  open  http
111/tcp open  rpcbind
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
512/tcp open  exec
513/tcp open  login
514/tcp open  shell
```

Figure 9. Portscan Attack using Nmap

v. Web Attack. Brute force is the first technique an attacker tries before proceeding to other attacks. Burp suite [42]**,** which is used for penetration tests and analysis of web attacks; has been applied here to perform an attack on the web pages. Different sample login web pages can be accessed using Metasploitable 2. Figure. 10 shows the trials of attempting different passwords, and highlighted area shows the correct password with an authentic username has been found.



5. Intruder attack of 192.168.18.73 - Temporary attack - Not saved to project file

Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
0			200	☐	☐	4885	
1	user	password	200	☐	☐	4885	
2	admin	password	200	☐	☐	4951	
3	user	password	200	☐	☐	4885	
4	mshatqat	password	200	☐	☐	4885	
5	kali	password	200	☐	☐	4885	
6	user	kali	200	☐	☐	4885	

Figure 10. Web Attack using Burp Suite

4. Dataset Processing. Traffic was captured using the Wireshark tool that produces packet capture files in .pcap format. As .pcap files are not enough to directly feed to ML-models, for its further processing, we have used the CIC-Flow meter [43]. Figure. 11 shows the flowchart of how processing of the .pcap file is done. The CIC-Flow meter was developed by the Canadian Institute of cyber security and used as a traffic analyzer that can extract more than 70 network features from a .pcap file such as Table. 2 shows the list of features extracted. The output file of the CIC-Flow meter is the .csv file, which can easily be used for machine learning models.

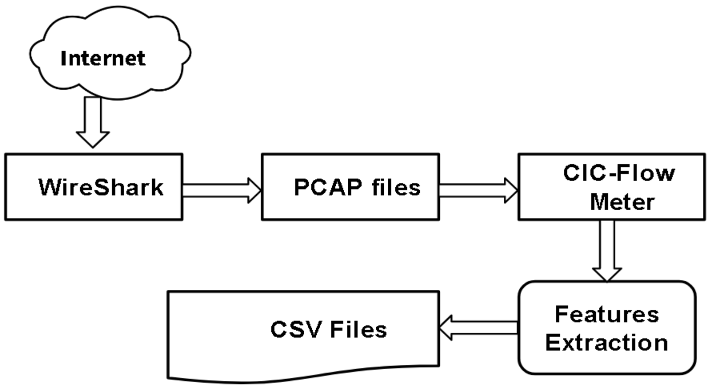


Figure 11. Preprocessing of pcap Files

Table 2. Features Extracted using CIC-Flow Meter

No	Features	No	Features	No	Features
1	Destination_port	26	Fwd_PSH_Flags	51	Fwd_Avg_bytes_bulk
2	Flow_duration	27	BWD_PSH_Flags	52	Fwd_Avg_pkts_bulk
3	Total_fwd_pkts	28	FWD_URG_Flags	53	FWD_Avg_bulk_rate
4	Total_bwd_pkts	29	BWD_URG_Flags	54	BWD_Avg_bulk_rate
5	Total_len_of_fwd_pkt	30	FWD_Header_len	55	BWD_AVG_pkts_blk
6	Total_len_of_bwd_pkt	31	BWD_Header_len	56	BWD_AVG_bulk_rat
7	Fwd_pkt_len_max	32	FWD_pkts_s	57	Subflow_FWD_pkts
8	Fwd_pkt_len_min	33	BWD_pkts_s	58	Subflow_FWD_bytes
9	Fwd_pkt_len_mean	34	Min_pkt_len	59	Subflow_BWD_pkts
10	Fwd_pkt_len_std	35	Max_pkt_len	60	Subflow_FWD_bytes

11	Flow_bytes	36	Pkt_Len_Mean	61	Init_Win_bytes_FWD
12	Flow_pkts	37	Pkt_len_Std	62	Init_Win_bytes_BWD
13	Flow_IAT_Mean	38	Pkt_len_Variance	63	act_data_pkt_fwd
14	Flow_IAT_Std	39	FIN_Flags_count	64	Min_Seg_size_FWD
15	Flow_IAT_Max	40	SYN_Flags_count	65	Act_Mean
16	Flow_IAT_Min	41	RST_Flags_count	66	Act_Std
17	Fwd_IAT_Total	42	PSH_Flags_count	67	Act_Max
18	Fwd_IAT_Mean	43	ACK_Flags_count	68	Act_Min
19	Fwd_IAT_Std	44	URG_Flags_count	69	Idle_Mean
20	Fwd_IAT_Max	45	CWE_Flags_count	70	Idle_Std
21	Fwd_IAT_Min	46	ECE_Flags_count	71	Idle_Max
22	Bwd_IAT_Total	47	Down_Up_ratio	72	Idle_Min
23	Bwd_IAT_Mean	48	Avg_FWD_seg_siz	73	Label
24	Bwd_IAT_Std	49	Avg_BWD_seg_siz		
25	Bwd_IAT_Max	50	Fwd_Head_len		

5. Labeling. It is the last part of the dataset creation method, which is the process of identifying raw data points. Labeling was performed manually on each .csv file according to its scenario. Figure. 12 the total distribution of each traffic type, while Figure. 13 shows the total classes and their samples that we have feed to our ML-models.

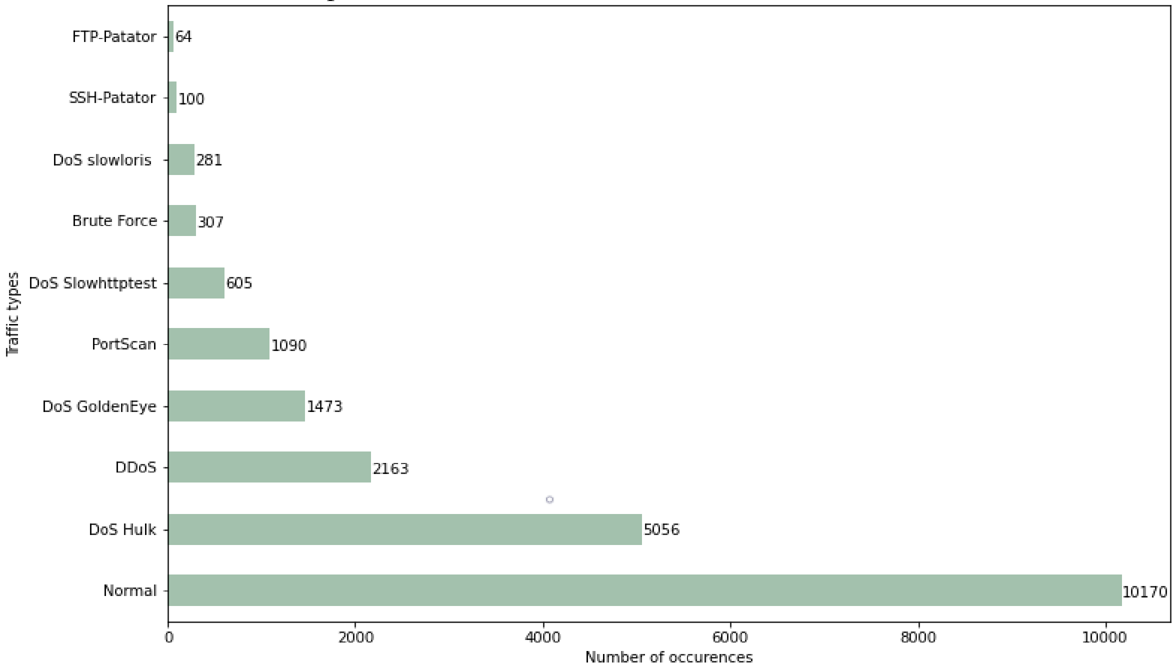


Figure 12. Distribution of Traffic Types

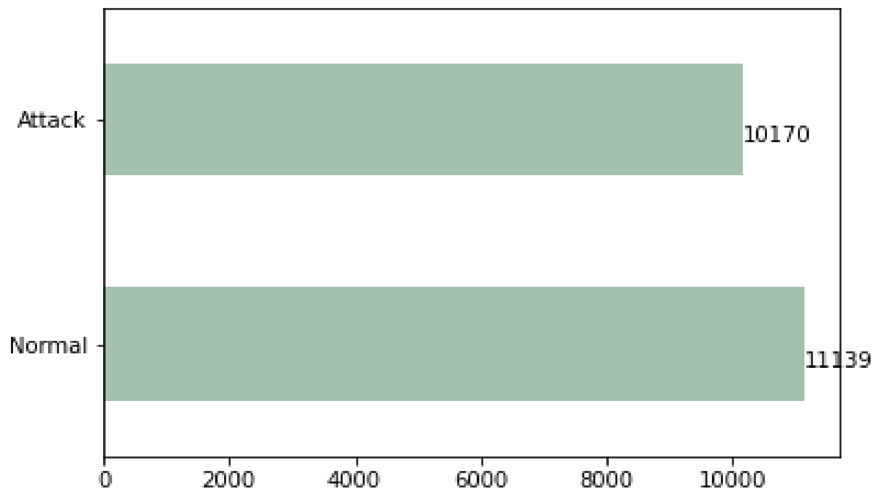


Figure 13. Total Classes and their Samples

6. Developing ML-based IDS. After generating the desired dataset, we used it for our ML-based IDS. Figure. 14 shows the complete methodology.

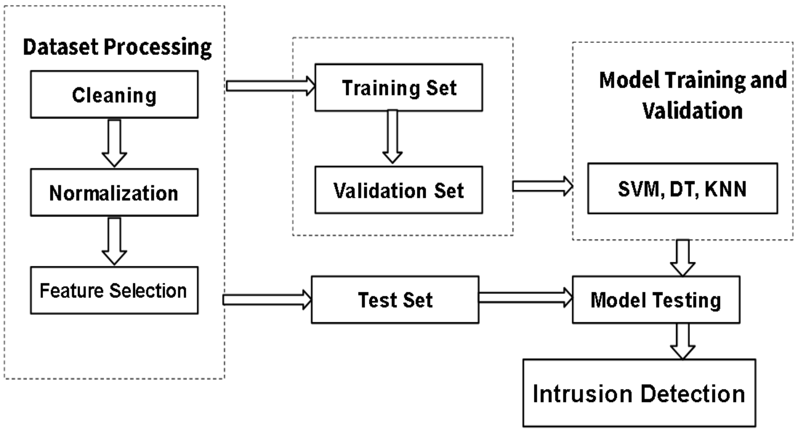


Figure 14. Methodology of ML-based IDS

Further processing of the dataset such as cleaning, normalization, and feature selection is performed before forwarding the dataset to the ML models. The generated dataset may contain null or infinite values that may affect the final results [44], so, we have eliminated them.

Furthermore, independent variables of the dataset contain highly varying magnitudes, so for feature scaling, we have used the Normalization method that translates all the independent values within the range of [0-1] [45].

A dataset contains more than 70 independent features and applying all of them is not feasible because it may cost computational power and efficiency variations, so we have applied the chi-squared test for feature selection [46]. In Figure. 15, each feature variable has a score which is representing correlation with the labels. By analyzing the graph, we have found that around 99% of the information is found in 40 features.

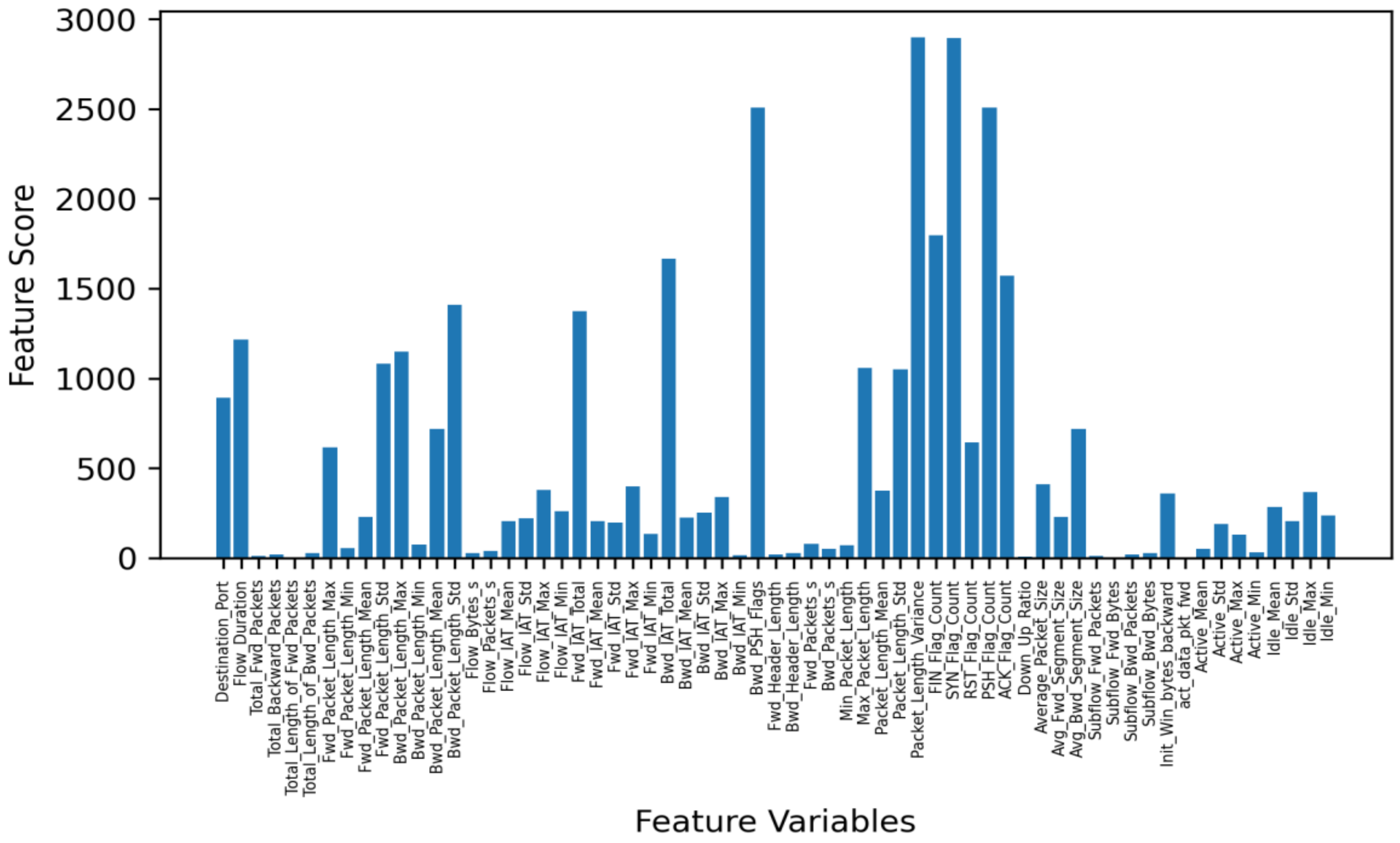


Figure 15. Feature Selection using Chi2 Test

Further, Figure. 16 shows the cutoff point that specifies whether to include or eliminate the feature, so for better accuracy and results we have eliminated the remaining features that fall after the cutoff point.

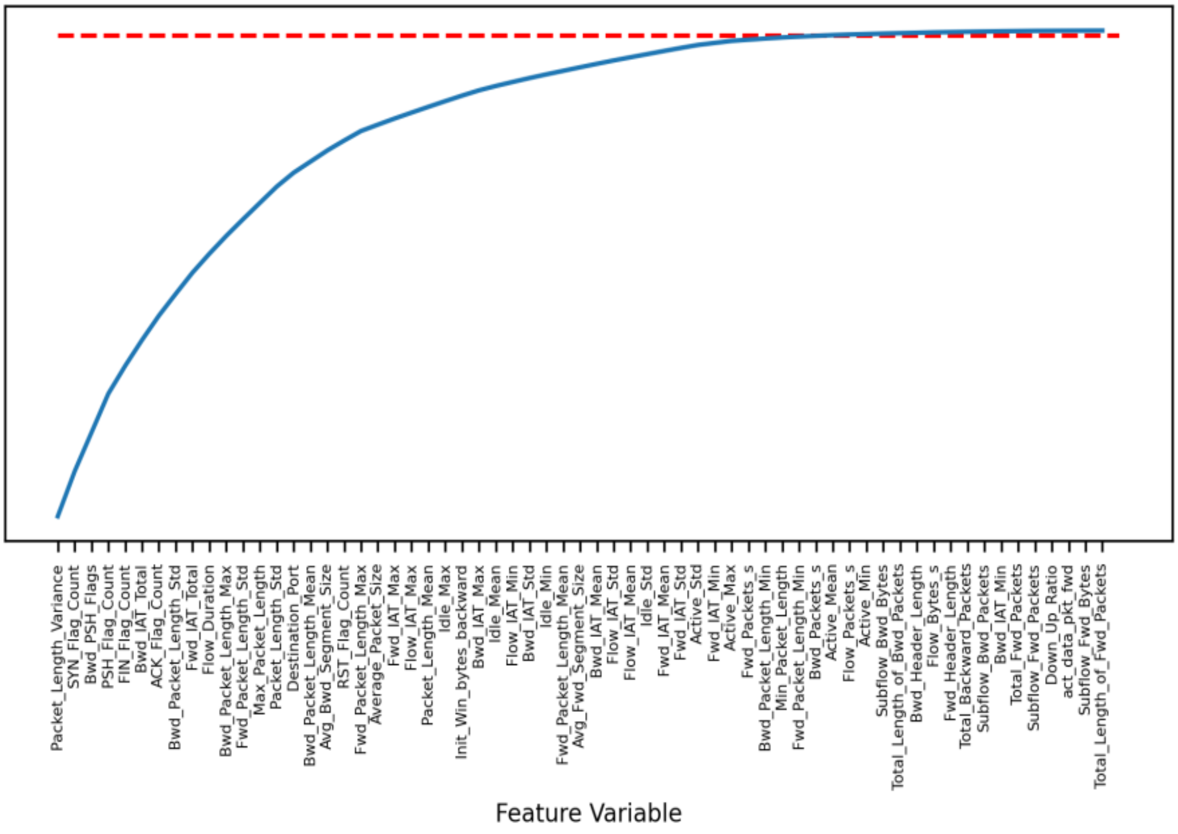


Figure 16. Cutoff Point of Features

After completing the data processing, we have used the split-validation method and divided the dataset into three proportions; training, validation and testing set with a ratio of 60:20:20. We have selected 3 different algorithms based on their performances such as Support Vector Machine (SVM), Decision Tree (DT) and Naive Bayes with their default paraments (Hyper Parameters) such as LinearSVC, Decision Tree Classifier(random_state = 0), and MultinomialNB(), respectively [12] [47] [48] and analyzed their performance based on evaluation metrics, such as Accuracy, Precision, Recall and F-measure which can be determined by using values of confusion matrix, for instance, True Positive (TP), True Negative (TN), False Positive ((FP) and False Negative (FN) [49].

i. Accuracy. It is a measurement that is the ratio of the numbers of all correct predictions to the total no. of predictions and can be calculated by:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

ii. Precision. It is known as the Positive Prediction value and can be calculated by total correct positive predictions divided by predicted positives.

$$\frac{TP}{TP + FP}$$

iii. Recall. It is known as sensitivity or true positive rate and is calculated by dividing the correct positive prediction by the total positive samples.

$$\frac{TP}{TP + FN}$$

iv. F-Measure (f1 Score). It is the harmonic mean of recall and precision and can be calculated by:

$$2\frac{Precision * Recall}{Precision + Recall}$$

RESULTS AND DISCUSSION

1. Performance Analysis of Different ML Models. In this research we have analyzed different well known classification algorithms and evaluated them using original unbalanced custom dataset. The performance of each ML model on unbalanced dataset is shown and discussed below:

i. Evaluating SVM. Tables. 3 and 4 depict the results of the Support Vector Machine (SVM). By analyzing the confusion matrix shown in Table. 4 clarifies that around 1957 and 2206 instances are correctly identified by the model, whereas 77 and 22 instances are falsely detected. Moreover, the classification report of SVM is shown in Table.3, illustrates better results as all figures are above 90%.

Table 3. Classification report of SVM

Class	Precision	Recall	F1 Score
Normal	0.966	0.990	0.9780
Malicious	0.988	0.9621	0.9753

Table 4. Confusion Matrix of SVM

	Actual Malicious	Actual Normal
Predicted Malicious	1957	77
Predicted Normal	22	2206

ii. Evaluating Decision Tree. Table. 5 represents the classification report of DT that shows higher positive numbers of each metric that is above 90% in each case. While Table. 6 depicts the confusion matrix that shows the better results compare to SVM with the highest number of correct predictions that is 2029 and 2219 and very few instances are falsely predicted by the model.

Table 5. Classification report of DT

Class	Precision	Recall	F1 Score
Normal	0.997	0.995	0.996
Malicious	0.995	0.997	0.996

Table 6. Confusion Matrix of DT

	Actual Malicious	Actual Normal
Predicted Malicious	2029	5
Predicted Normal	9	2219

iii. Evaluating Naïve Bayes. Tables. 7 and 8 illustrate the results of the NB model, which show lowest performance compared to SVM and DT. The confusion matrix in Table. 8 depicts the highest number of false positives and false negatives which are 826 and 122 respectively. Whereas 1208 and 2106 instances are accurately predicted by the model.

Table 7. Classification report of NB

Class	Precision	Recall	F1 Score
Normal	0.718	0.945	0.816
Malicious	0.90	0.593	0.718

Table 8. Confusion Matrix of NB

	Actual Malicious	Actual Normal
Predicted Malicious	1208	826
Predicted Normal	122	2106

By analyzing the Figure. 17 Decision Tree's overall performance outperforms Support Vector Machine and Naive Bayes with a percentage of around 99% in all metrics. While SVM is the second-best performer with a percentage greater than 90% in all metrics. However, Naive Bayes overall performance shows the lowest percentage throughout metrics.

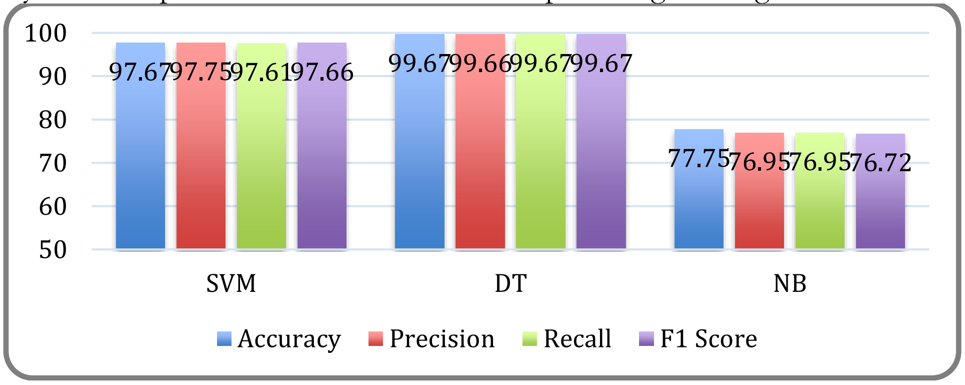


Figure 17. Comparative Analysis of SVM, DT and NB

2. Comparative Analysis of Different Datasets. Table. 5 shows the comparative analysis between DARPA, KDD'99, Kyoto, ISCX2012, UNSW-NB15, CIC-IDS-18, and the proposed dataset. All the datasets mentioned are chosen based on their popularity. For comparison, we have chosen seven parameters such as realistic traffic, practical to generate, updates, publicly available, attack diversity, flexibility, and interoperability.

By analyzing the real-traffic column, it's obvious that most of the datasets are based on simulated traffic; either they have generated synthetic normal traffic, or their attacks are replicating real attacks thus we have tried to provide a dataset based on realistic scenarios by capturing the actual normal traffic that flows through the network. While capturing the real attacks, we have worked on manually generating each attack.

Another noticeable problem with this dataset is that no proof or document clearly defines how this particular dataset is generated or what tools and methods have been used. Researcher's favorite dataset: KDD '99 & DARPA, which is 22 years old, cannot be reproduced due to the unavailability of detailed methodology. Whereas, in addition to the dataset, we have provided the complete methodology that includes complete information about the scenarios, tools, methods and processes which a user can apply to generate their new dataset based on their requirements.

While the Update column depicts that most datasets are non-updated. In comparison, Kyoto datasets had released their updates till 2015, but since then, no update has appeared.

Further, CIC-IDS-18 only releases partial updates, such as their last dataset released in 2022 that only contains obfuscated malware traffic. In our case, we have tried to provide an updated dataset whose traffic is based on current trends and patterns.

By analyzing the attack diversity column, most used datasets such as KDD '99, DARPA, and UNSW-NB15 had used a wide range of attacks, but most of these attacks are no longer in use. Over time different techniques and tools have evolved, so we have tried to adopt the latest tools in our dataset.

The flexibility column depicts that most of the dataset contains the traffic diversity except for Kyoto and ISCX-12, which had worked on a specific scenario. For such a case, we have tried to include the wide range of traffic patterns that enabled the dataset to adjust to different objectives and scenarios.

We have tried to cover the research gap by providing a complete dataset based on realistic scenarios. Moreover, we have provided the complete methodology that enabled the community to reproduce new datasets based on their needs.

Table 9. Performance Analysis on All Labels

Datasets	**Real Traffic**	**Practical to Generate**	**Updates**	**Publicly Available**	**Attack Diversity**	**Flexible**	**Interoperability**
DARPA	✗	✗	✗	✓	✓	✓	✓
KDD'99	✗	✗	✗	✓	✓	✓	✓
Kyoto	✗	✗	✗	✗	✗	✗	✗
ISCX12	✗	✗	✗	✓	✗	✗	✓
UNSW-NB15	✗	✓	✗	✓	✗	✓	✗
CIC-IDS-15	✗	✓	✗	✓	✓	✓	✓
Custom Dataset	✓	✓	✓	✓	✓	✓	✓

CONCLUSION & FUTURE WORK. To improve the performance and accuracy of IDSs, a reliable, authentic dataset is essential. In this paper, we have discussed different datasets since 1998, which appear insufficient in the sense of unavailability of traffic diversity, ground truths, updated versions, and lack of diverse and updated attacks. The problem with these static, one-time generated datasets is that they cannot adjust to the ongoing changes of networks.

Based on the research gap and requirements, we have worked on generating a new feasible dataset that fulfills the requirement of a researcher who wants to test their IDS on a realistic dataset. The generated dataset is publicly available for users on GitHub [50][51]. Moreover, we have provided complete detailed methods which help analysts to generate their new dataset based on their needs and objectives. Further, we have developed three different ML models on custom generated unbalanced dataset and compared performance of each model. In our case, Decision Tree (DT) has shown much better results than the Support

Vector Machine and Naïve Bayes with higher accuracy that is around 99.67%. Various studies on machine learning have been conducted that show the emergence of ML in daily dynamics [52, 53, 54, 55, 56, 57, 58,59].

In future work, we can extend our research by using our custom dataset as a benchmark dataset, where we will train different machine learning-based models on different available datasets and test them using our custom dataset. Moreover, we will extend this research by training these ML-models using same custom dataset where we will apply different balancing techniques. Later we will analyze and compare how a balancing technique affects the performance of each model.

Acknowledgement. The author would like to express my special thanks of gratitude to my supervisor Dr Faheem Yar Khuawar who helped me throughout this research, and I came to know about so many new things.

Author's Contribution. All the authors contributed equally.

Conflict of interest. We declare no conflict of interest for publishing this manuscript in IJIST.

Project details. NIL

Availability of Dataset. The generated dataset is publicly available for the researchers and can be accessed on https://github.com/mshafqat00/IDS-Dataset

REFERENCES

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

[2] M. Al-kasassbeh, G. Al-naymat, and E. Al-hawari, "Towards Generating Realistic SNMP-MIB Dataset for Network Anomaly Detection," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. December, p. 1162, 2016.

[3] V. R. Varanasi and S. Razia, "Intrusion Detection using Machine Learning and Deep Learning," *Int. J. Recent Technol. Eng.*, vol. 8, no. 4, pp. 9704–9719, 2019, doi: 10.35940/ijrte.d9999.118419.

[4] A. Chadd, "DDoS attacks: past, present and future," *Netw. Secur.*, vol. 2018, no. 7, pp. 13–15, 2018.

[5] S. Wankhede and D. Kshirsagar, "DoS attack detection using machine learning and neural network," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–5.

[6] Q. A. Al-Haija, E. Saleh, and M. Alnabhan, "Detecting Port Scan Attacks Using Logistic Regression," in *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 2021, pp. 1–5.

[7] K. Trieu and Y. Yang, "Artificial intelligence-based password brute force attacks," 2018.

[8] R. Singh, H. Kumar, R. K. Singla, and R. R. Ketti, "Internet attacks and intrusion detection system: A review of the literature," *Online Inf. Rev.*, 2017.

[9] S. V. M. Vishwanathan and M. N. Murty, "SSVM: a simple SVM algorithm," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, 2002, vol. 3, pp. 2393–2398.

[10] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Trans. Geosci. Electron.*, vol. 15, no. 3, pp. 142–147, 1977.

[11] I. Rish and others, "An empirical study of the naive Bayes classifier," in *IJCAI 2001*

workshop on empirical methods in artificial intelligence, 2001, vol. 3, no. 22, pp. 41–46.

[12] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2018, no. 1, pp. 177–200, 2018.

[13] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000.

[14] S. Hossen and A. Janagam, "Analysis of network intrusion detection system with machine learning algorithms (deep reinforcement learning Algorithm)," no. October, pp. 1–63, 2018.

[15] R. P. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, 2000, vol. 2, pp. 12–26.

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

[17] A. Mishra and P. Yadav, "Anomaly-based IDS to detect attack using various artificial intelligence machine learning algorithms: A review," *2nd Int. Conf. Data, Eng. Appl. IDEA 2020*, 2020, doi: 10.1109/IDEA49133.2020.9170674.

[18] R. Chitrakar and C. Huang, "Anomaly based intrusion detection using hybrid learning approach of combining k-medoids clustering and naive bayes classification," in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, 2012, pp. 1–5.

[19] M. Al-Fawa'reh and M. Al-Fayoumiy, "Detecting stealth-based attacks in large campus networks," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 4, pp. 4262–4277, 2020, doi: 10.30534/ijatcse/2020/15942020.

[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. \& Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[21] M. H. Abdulraheem and N. B. Ibraheem, "A detailed analysis of new intrusion detection dataset," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 17, pp. 4519–4537, 2019.

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

[23] A. Thakkar and R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 636–645, 2020, doi: 10.1016/j.procs.2020.03.330.

[24] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. \& Secur.*, vol. 86, pp. 147–167, 2019.

[25] D. Stiawan, M. Y. Bin Idris, A. M. Bamhdi, R. Budiarto, and others, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[26] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE access*, vol. 9, pp. 22351–22370, 2021.

[27] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and S. Nadjm-Tehrani, "On generating network traffic datasets with synthetic attacks for intrusion detection," vol. 0, no. 0, 2019.

[28] G. Brogi and G. Brogi, "Sharing and replaying attack scenarios with Moirai To cite this version :," no. June, 2017.

[29] "Wireshark." .

[30] "Kali LINUX." .

[31] "Metasploitable." .

[32] "VirtualBox." .

[33] "GitHub."

[34] "GoldenEye." .

[35] "Hulk DoS Attack." .

[36] "SlowLoris."

[37] "SynFlood."

[38] "LOIC." .

[39] R. Damasevicius *et al.*, "Litnet-2020: An annotated real-world network flow dataset for network intrusion detection," *Electron.*, vol. 9, no. 5, 2020, doi: 10.3390/electronics9050800.

[40] "Patator." .

[41] "Nmap." .

[42] PortSwigger Ltd, "Burp Suite," 2015. .

[43] "CIC-Flow-Meter." .

[44] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *J. Big Data*, vol. 7, no. 1, pp. 1–20, 2020.

[45] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study," *Comput. Networks*, vol. 188, p. 107840, 2021.

[46] C. J. Ugochukwu, E. O. Bennett, and P. Harcourt, *An intrusion detection system using machine learning algorithm*. LAP LAMBERT Academic Publishing, 2019.

[47] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 228–233.

[48] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, Oct. 2018, pp. 1–8, doi: 10.1109/CCCS.2018.8586840.

[49] S. Zwane, P. Tarwireyi, and M. Adigun, "Performance analysis of machine learning classifiers for intrusion detection," *2018 Int. Conf. Intell. Innov. Comput. Appl. ICONIC 2018*, pp. 1–5, 2019, doi: 10.1109/ICONIC.2018.8601203.

[50] S. A. Siyyal, "Custom Generated IDS Dataset," 2022. .

[51] Abdul Malik, & Muhammad Shumail Naveed. (2022). Analysis of Code Vulnerabilities in Repositories of GitHub and Rosettacode: A comparative Study. International Journal of Innovations in Science & Technology, 4(2), 499–511. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/289
[52] Khan, M. I., Imran, A., Butt, A. H., & Butt, A. U. R. . (2021). Activity Detection of Elderly People Using Smartphone Accelerometer and Machine Learning Methods. International Journal of Innovations in Science & Technology, 3(4), 186–197. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/96
[53] Muhammad Asad Arshed, Jabbar, M. A. ., Liaquat, F., Chaudhary, U. M.- ud-D. ., Karim, D. ., Alam, H. ., & Mumtaz, S. . (2022). Machine Learning with Data Balancing Technique for IoT Attack and Anomalies Detection. International Journal of Innovations in Science & Technology, 4(2), 490–498. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/277
[54] Malik, Z. A., Siddique, M. ., Zahir Javed Paracha, Imran, A., Yasin, A., & Butt, A. H. (2022). Performance Evaluation of Classification Algorithms for Intrusion Detection on NSL-KDD Using Rapid Miner . International Journal of Innovations in Science & Technology, 4(1), 135–146. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/101
[55] Farman Hassan, Muhammad Hamza Mehmood, Babar Younis, Nasir Mehmood, Talha Imran, & Usama Zafar. (2022). Comparative Analysis of Machine Learning Algorithms for Classification of Environmental Sounds and Fall Detection. International Journal of Innovations in Science & Technology, 4(1), 163–174. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/188
[56] Irfan Qutab, Malik, K. I., & Hira Arooj. (2022). Sentiment Classification Using Multinomial Logistic Regression on Roman Urdu Text. International Journal of Innovations in Science & Technology, 4(2), 323–335. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/217
[57] Shahrukh Hussain, Usama Munir, & Chaudhry, . M. S. (2022). Visualizing Impact of Weather on Traffic Congestion Prediction: A Quantitative Study. International Journal of Innovations in Science & Technology, 3(4), 210–222. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/125
[58] Asad Ur Rehman, Madiha Liaqat, Ali Javeed, & Farman Hassan. (2022). HealthConsultantBot: Primary Health Care Monitoring Chatbot for Disease Prediction. International Journal of Innovations in Science & Technology, 4(1), 201–212. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/193
[59] Sohail Manzoor, Huma Qayyum, Farman Hassan, Asad Ullah, Ali Nawaz, & Auliya Ur Rahman. (2022). Melanoma Detection Using a Deep Learning Approach. International Journal of Innovations in Science & Technology, 4(1), 222–232. Retrieved from https://journal.50sea.com/index.php/IJIST/article/view/191