Information Security Journal: A Global Perspective

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/uiss20

Adversarially robust and real-time DDoS detection and classification framework using AutoML

Sambhrant Maurya, Anand Handa, Nitesh Kumar & Sandeep K. Shukla

To cite this article: Sambhrant Maurya, Anand Handa, Nitesh Kumar & Sandeep K. Shukla (04 Apr 2024): Adversarially robust and real-time DDoS detection and classification framework using AutoML, Information Security Journal: A Global Perspective, DOI: 10.1080/19393555.2024.2332955

To link to this article: https://doi.org/10.1080/19393555.2024.2332955

INFORMATION SECURITY JOURNAL: A GLOBAL PERSPECTIVE
https://doi.org/10.1080/19393555.2024.2332955

Taylor & Francis
Taylor & Francis Group

Adversarially robust and real-time DDoS detection and classification framework using AutoML

Sambhrant Maurya, Anand Handa, Nitesh Kumar, and Sandeep K. Shukla

C3i Center, Department of Computer Science and Engineering, IIT Kanpur, Kanpur, UP, India

ABSTRACT

Denial of Service (DoS) attacks target the availability part of the CIA triad (Confidentiality, Availability, and Integrity). A special category of these attacks is the Distributed DoS (DDoS) attack, where the attacker uses a network of compromised systems called a botnet to flood a target server with requests and refuses to serve legitimate users. DDoS attacks can cost an organization millions of dollars in terms of lost revenue, remediation costs, and damage to brand reputation. Hence, all organizations need speedy real-time detection of DDoS attacks. This work presents a DDoS detection and classification framework using the flow-based approach for feature engineering and the AutoML technique. Our detection system is trained on the latest DDoS datasets – CIC-DDoS 2019 and CIC-IDS 2017, which contain various categories of DDoS attacks. We use various tools to perform adversarial attacks on our trained model. We retrain our models using adversarially crafted network packet captures and then test our models for robustness against practical adversarial attacks that an attacker might use to evade detection. Finally, we deploy our model in real-time using a GUI-based tool. Our model achieves a validation accuracy of 99.9% and a low false positive rate of 0.05%.

KEYWORDS

Adversarial attack; adversarial retraining; AutoML; DDoS attack detection; flow based analysis

1. Introduction

In the present world, most modern devices are connected to the internet. According to statistics (Key internet statistics to know in 2022, 2022), approximately 5.25 billion individuals access and utilize the internet today. This equates to 66.2% of the entire world's population. Individuals utilize the internet for a variety of purposes, including communication, banking, transactions, information gathering, and recreation. It is used by businesses to connect with their customers, partners, and suppliers, among other things. The widespread use of the Internet has ushered in a new era of cybercrime. The National Crime Records Bureau (NCRB) reported that while only 12,317 incidences of cybercrime were reported in India in 2016, this number rose to 50,035 in 2020, hence accounting for a 306% increase in cybercrime from 2016 to 2020 (The Hindu: Rise in Cyber crime in India, 2020). This implies that in 2020, India had on average 136 cybercrime incidents recorded per day.

DoS (Denial of Service) attacks appear to be one of the most common types of cyberattacks. The aim of these attacks is to prevent legitimate users from accessing the services or resources of the target system. The attacked system becomes overburdened with requests and refuses to serve legitimate users, resulting in the loss of resources and services for the affected clients. In DoS attacks, the attacker typically floods the target system with large amount of traffic or sends it some data that causes it to stop responding and crash. DoS attacks usually reported by media focus on those launched against high profile companies and government organizations. The victims of the DoS attack may have to spend a significant amount of time and money dealing with the incident, while the users are denied access to certain services and resources. However, Industrial control systems that help keep critical infrastructure running can also be attacked by DoS attacks. These attacks have more severe consequences, which may include the inability to monitor and control remote sites or control critical processes, which may cause a loss of essential services such as health-care and transportation (Bela & Siaterlis, 2013). A targeted DDoS attack on critical infrastructure can also lead to other serious consequences like extended supply shortages, power failure and public disorder.

CONTACT Anand Handa anand@c3ihub.iitk.ac.in C3i Center, Department of Computer Science and Engineering, IIT Kanpur, Kanpur, UP, India

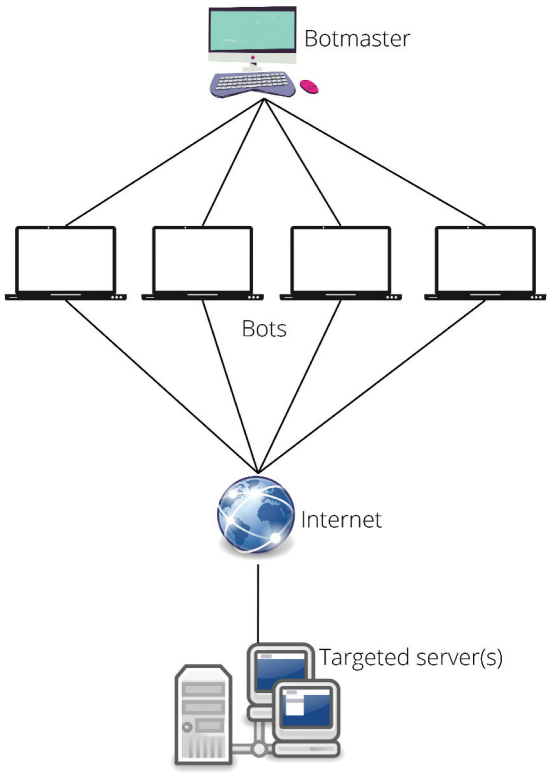


Figure 1. A schematic of DDoS attacks.

DDoS is an abbreviation for **Distributed Denial of Service**, which is a special variant of DoS attacks. Figure 1 illustrates a schematic of DDoS attacks. The threat actor uses a network of machines from multiple, often remote locations, called a **botnet** to launch an attack on the target server or network. Each individual machine in such a botnet is called a **bot**, and the attacker controlling them is called the **botmaster**. Once a botnet has been established, the botmaster sends remote commands to each bot in the botnet. Each bot then sends requests to the target server or network, aiming to overload or crash it, subsequently causing a denial-of-service to benign traffic. Distinguishing malicious traffic from benign traffic is often difficult because each bot is a legitimate Internet device. DDoS attacks are treated as a high-risk threat by enterprises.

1.1. Types of DDoS attacks

Mahjabin et al (Mahjabin et al., 2017). proposed a DDoS classification based on how the attacks impact the target system's network or services. DDoS attacks can be broadly classified into 4 types:

1.1.1. Bandwidth depletion attacks

The goal of the attacker is to use up the entire network bandwidth of the targeted system. This causes the victim to deny service to legitimate users until the attack is dealt with. Bandwidth depletion attacks are of two types – protocol exploited attacks and amplification attacks.

1.1.2. Resource depletion attacks

Resource depletion attacks overload or crash the system's memory, sockets, and CPU. This attack can be done two ways. The first method uses network, transport, and application layer protocols. Second-method attacks use malformed packets.

1.1.3. Infrastructure attacks

This attack aims to harm critical Internet components. It targets network bandwidth as well as system resources (memory, CPU) of the target. Infrastructure attacks target DNS, especially root DNSs, which serve all Internet users globally. Because the DNS is hierarchical, an attack on the root name servers wouldn't affect global Internet service. DNS flooding is a frequently used tactic, where a botnet is used to send a large volume of UDP requests to the DNS server. Such queries overwhelm the system, exhausting all available resources. This causes denial of all requests from legitimate users for a certain amount of time.

1.1.4. Zero-day DDoS attacks

These attacks make use of undiscovered loopholes or vulnerabilities in the system's security. The name "zero day" refers to the fact that the vulnerabilities in the system are only identified the day after they have been exploited. The signature and impact of these kinds of attacks are unknown until the attack has been carried out.

In February 2020, Amazon Web Services (AWS) faced yet another large DDoS attack where the incoming traffic peaked at 2.3 Tbps (DDoS attacks history by Radware, 2020). On June 14, 2022, Cloudflare mitigated the largest HTTPS DDoS attack detected to date, peaking at 26 million requests per second (*News Article: Cloudflare mitigates record-breaking HTTPS DDoS attack* 2022b). According to StationX, Q1 2023 saw a 47% surge in attacks compared to the same period in 2022 (DDoS Attack Patterns Statistics, 2023).

The frequency of DDoS attacks has significantly increased over the last few years. There has been an increase in the number of ransom DDOS attacks between 2020 and 2021, and by 75% in the fourth quarter (Q4) of 2021 compared to the previous quarter (DDoS 2021-q4 stats, 2021). The month of December 2021 saw a greater number of network-layer DDoS attacks than the first two quarters of 2021 combined. A wave of large-scale DDoS attacks spread across New Zealand in Q3 2021, according to Yandex and Qrator Labs (DDoS q3–2021 stats, 2022). In Q3, ransom DDoS attacks on VoIP providers impacted businesses in the United Kingdom, Canada, and the United States. Manufacturing, Business Services, and Gaming/Gambling were the most targeted industries in Q4 of 2021. The report also demonstrates that over three out of every thousand HTTP requests from Chinese IP addresses were part of a DDoS attack. The United States came in second, followed by Brazil and India.

Motivation of this research work - Due to such a large number of DDoS attacks, there is a dire need for an automated approach to detect such attacks. Hence, in this work, we develop a framework to detect and classify various categories of DDoS attacks. As we know that ML algorithms can learn to distinguish between normal network traffic and malicious DDoS attack traffic, thereby reducing false positives and minimizing unnecessary disruptions to legitimate user activities and services. By accurately identifying and filtering out malicious traffic, ML-based DDoS detection systems help maintain service availability while minimizing the impact on user experience. Hence, we develop a ML-based DDoS detection model. We utilize the AutoML technique to automate the process of model training and hyperparameter tuning. The result shows that our model can detect various DDoS attacks with high accuracy. The result also shows that our ML-based DDoS detection systems can scale to handle large and diverse network environments with varying traffic patterns and volumes.

1.2. Our contributions

The following is an outline of our contributions to this work:

(1) We develop a tool for parsing packet capture (pcap) files, which constructs bidirectional flows in 5-tuples, extracts features for each flow engineered based on DDoS traffic characteristics, and saves the output in the CSV format. Our tool performs this process at an average rate of 3000 packets per second.

(2) We develop an ML-based DDoS detection and classification framework. One can deploy our framework in real-time on a real network. We have tested the performance of our model on live network traffic from a real network and a virtual network.

(3) Unlike most previous works that perform adversarial attacks on the feature level, we have simulated practical adversarial attacks by perturbing the problem space itself (i.e., at the network traffic level) to test the robustness of the model. We then perform adversarial training of the model to improve its robustness. We further test the retrained model to prove that it is indeed robust.

(4) We also develop a simple GUI application to demonstrate the deployment of our detection system in a real environment.

This section provides a concise introduction to Distributed Denial of Service (DDoS) attacks, the various types of DDoS attacks, and the contributions of our work. The relevant background information and previous research carried out in the field are presented in Section 2. Section 3 breaks down our proposed approach for DDoS detection in great detail. The outcomes of our tests and experiments with the proposed method are presented in Section 4. Section 5 concludes our work along with some recommendations for potential future work.

2. Related work

In this section, we discuss DDoS detection techniques, and the adversarial attacks on these detection techniques. Unexpected variations in network traffic are encountered when a system is under a DDoS attack. DDoS detection systems aim to automatically monitor and analyze abrupt network changes (Gyanchandani & Rana, 2012). These detection systems may be hardware or software-based. The detection approaches can be divided into signature-based, anomaly-based, and hybrid detection categories (Agarwal & Mittal, 2012).

Signature based detection – The Signature based approach for attack detection is also referred to as rule-based detection or misuse detection. In this approach, the incoming traffic is matched with signatures or patterns of known attacks to identify the attack instances (Agarwal & Mittal, 2012). These signatures usually consist of IP fields like source and destination IPs, protocol, ports and information extracted from the packet payload (Xia et al., 2005). This approach is not effecting for detecting new attacks or slight modifications of existing attacks, as signatures of these are not available to the detection system (Lee & Xiang, 2001). Snort (*SNORT: Network Intrusion Detection & Prevention System*, 2022) is a popular open-source Intrusion Detection & Prevention System based on signature-based detection. Thomas et al. (2003) (Thomas et al., 2003) presented a detection system called 'NetBouncer.' A database of legitimate users is prepared. A series of legitimacy tests must be performed if the received packets do not come from an authorized client. After passing the tests, if the new client manages to send enough packets before the session window times out, this client will be added to the database. Y.-C. Wu et al. (2011) (Y.-C. Wu et al., 2011) presented a DDoS detection method involving traffic pattern matching and decision trees. After examining the forward and backward packet transmission rates, SYN and ACK flag rates, the decision trees can distinguish between DDoS and normal traffic. It uses an algorithm for pattern matching to identify traffic patterns that resemble those of an attack. Limwiwatkul and Rungsawang (2004) (Limwiwatkul & Rungsawang, 2004) differentiated attack and normal traffic by analyzing the TCP/IP packets through a set of defined rules and conditions. Thapngam et al. (2014) (Thapngam et al., 2014) used packet transmission rate to detect attack traffic. The authors assumed that the transmission rate of attack traffic is higher than of normal traffic. This method of detection is inefficient because the attackers can easily use flash events to send traffic to the target in order to disguise the attack.

For signature based DDoS detection, various techniques may be used to construct the attack signatures. These include Expert Systems, Adept Systems, State Transition Analysis, Petri Nets and Description Languages. Signature based DDoS detection systems can be deployed either on source end, victim end or core end and can detect known attacks efficiently in real time, but these approaches are unable to detect new attacks. Foreknowledge of attacks and their signatures is essential, and the signatures also need to be updated frequently.

Anomaly based detection – This approach is also called behavior based detection or outlier detection. This method models the typical operation of the network and then evaluates how well it matches up with the incoming data instances. An anomaly alarm is generated by the detection system whenever the deviation between the actual behavior and anticipated behavior exceeds a certain limit. This results in the disclosure of an attack (García-Teodoro et al., 2009).

Data mining approaches involve finding patterns in the data (Garg & Chawla, 2011). There are many useful patterns and relationships hidden in large data sets that can be discovered using a statistical model (Aggarwal & Gupta, 2015). When used in conjunction with AI or machine learning, this method has a high rate of detection accuracy. Detection using Associative rule mining, clustering, and classification are all data mining approaches. Sumathi and Karthikeyan (2018) (Sumathi & Karthikeyan, 2018) analyzed the performance of several data mining algorithms for detection of DDoS attacks, and reported the Fuzzy c-means clustering algorithm as the most effective.

Machine Learning based approaches for DDoS detection are similar to data mining or statistical approaches, both of which aim to discover the rules that lead to new data being generated (García-Teodoro et al., 2009). AI-based methods include Machine Learning and Soft Computing techniques. Linear Discrimination, Multivariate Methods, Neural Networks, Classification Trees, Gradient Boosting Machine and Hidden Markov Models are all examples of machine learning techniques that are used in AI-based detection (S.-Y. Wu & Yen, 2009). She et al. (2017) (She et al., 2017) proposed a way to detect DDoS attacks at application layer using One class Support Vector Machine (SVM). Imamverdiyev and Abdullayeva (2018) (Imamverdiyev & Abdullayeva, 2018) proposed an approach based on Gaussian-Bernoulli restricted Boltzmann machines to the detection of

DoS attacks. Ateş et al. (2019) (Ateş et al., 2019) proposed a non-parametric community clustering-based DDoS detection based on the relationship between the source and destination sides of packet headers, and tested the proposed algorithm on real data. The application of Deep Neural Networks to DDoS detection has been explored in the works (Asad et al., 2019; Shieh et al., 2021; Virupakshar et al., 2020) (Muraleedharan & Janet, 2021), and (Sbai & El Boukhari, 2020).

Hybrid detection – As the name suggests, this approach employs a blend of signature based and anomaly based detection approaches. A combination of multiple signature based approaches or multiple anomaly based approaches can also be used. This approach is well suited for detection of known attacks as well as novel attacks (Bhuyan et al., 2014). Smart Detection (Filho et al., 2019), NFBoost (Arun Raj Kumar & Selvakumar, 2013), RST-SVM (Chen et al., 2010), EMERALD (Patcha & Park, 2007) are examples of hybrid detection systems. Nadiammai et al. (2013) showed that the signature based SNORT IDS can be combined with anomaly based detection methods to develop Although the detection system becomes more robust by combining different methods, the results are not always as good as they could be. Developing a detection system based on the hybrid approach that is both potent and efficient is, in fact, a difficult endeavor practically (Patcha & Park, 2007). Developing and deploying a hybrid system can also turn out to be complex and costly.

Adversarial attacks on DDoS detection systems – Adversarial attacks aim to fool a Machine Learning based DDoS detection system by making perturbations in the input features of the model. Several works in the past have explored adversarial attacks on DDoS detection systems and on Network Intrusion Detection Systems in general. Works (Abdelaty et al., 2022; Chauhan & Shah Heydari, 2020; Mustapha et al., 2023; Novaes et al., 2021; Nugraha et al., 2021; Zhang et al., 2024; Zhao et al., 2021) and (Lin et al., 2018) explore the employment of Generative Adversarial Networks (GAN) to simulate adversarial scenarios. Alhajjar et al. (2020) (Alhajjar et al., 2020) presented two novel techniques for the generation of adversarial examples. These techniques make use of evolutionary computation using Particle Swarm Optimization and Genetic Algorithm. Additionally, they presented a genetic adversarial network (GAN) that considers constrained feature sets in order to avoid traffic failure. Many works have utilized some other methods to introduce perturbations. Papadopoulos et al. (2021) (Papadopoulos et al., 2021) used Fast Gradient Sign Method (FGSM), while Ayub et al. (2020) (Ayub et al., 2020) used Jacobian Saliency Map Attack (JSMA) to perturb the feature space. Some works (Ayub et al., 2020) have also explored poisoning attacks where adversarial examples are injected into the training phase itself.

Apruzzese et al. (2021) (Apruzzese et al., 2021) studied the feasibility and realism of the existing adversarial adversarial attacks on ML-based Intrusion Detection Systems. They observed that existing literature makes the assumption that the attacker knows everything about the target detection system or is able to interact with it, and hence they are unrealistic. They presented the idea of "power," which indicates the level of command that the adversary has over the targeted detection system, and they identified five elements that can be controlled by the attacker are training data, feature set, detection model, Oracle, and manipulation depth. According to the authors' argument, poisoning attacks during the training stage can be carried out only if the attacker possesses some form of control (specifically, write access) over the dataset used for training. Else, the attacker is restricted to launching attacks during the inference phase. The attacks perturbing the feature space assume partial or full knowledge of the feature space. In a realistic scenario, the attacker can only perturb the problem space and not the feature space. The attacker has partial/no knowledge of the feature space and can neither access the training data nor obtain feedback from the output of the NIDS. In such a scenario, the attacker is neither required to compromise the target detection system prior to the attack, nor acquire any sensitive information about the target detection system.

(Aiken & Scott-Hayward, 2020) demonstrated a practical adversarial attack on a Syn-Flood DDoS detection system where the attacker has minimal power over the target system. It is only presumed that, the attacker has partial knowledge of the feature space of the detection system. No

access to its training data, no oracle power on its output and no knowledge about its internal configuration is assumed. Their adversarial attack modifies the problem space itself rather than the feature space, where modifications are made to the packets in the malicious network traffic itself. This induces perturbations in the feature space of this traffic, which may cause the target detection system to misclassify. Aiken et al (Aiken & Scott-Hayward, 2020). showed that the accuracy of some commonly used classification algorithms against these attacks dropped from 99.9% to 70%. However, the dataset of their target detection system was limited to Syn-Flood attacks. Their attack was not tested on more recent classification algorithms like Gradient Boosting classifiers and Neural Networks. The full list of features used by their target detection system has not been disclosed. Bukac and Matyas (2015) (Bukac & Matyas, 2015) thoroughly examined the traffic generated by some of the widely used DDoS tools. They showed that, of the various DDoS tools they tested, no two of them generate DDoS traffic with the same characteristics. Hence, it is possible that a detection system trained on attack data from one DDoS tool may not perform well on attack data from some other DDoS tool.

For the detection of DDoS attacks, numerous signature based and anomaly based approaches have been proposed in the past. However, the majority of detection methods do not offer DDoS detection in real-time that is characterized by a high detection accuracy and a low number of false positives. Also, most of the previous work done on adversarial attacks on Intrusion Detection Systems focuses on perturbations in the feature space and not the problem space, which is impractical. In this work, we build a DDoS detection system trained on 8 types of DDoS attacks. We experiment state-of-the-art algorithms like Random Forest, Gradient Boosting Classifiers, Generalized Linear Models and Neural Networks for our detection system. Our detection system achieves an accuracy of 99.9% and a false positive rate of 0.05%. We perform realistic adversarial attacks by modifying the problem-space itself. We then perform adversarial training to improve robustness, and illustrate the deployment of the resulting model on a real network.

3. Proposed approach

In this work, we propose a framework to detect DDoS traffic using flow-based approach in 5-tuple format, [Protocol, Source IP, Source port, Destination IP, Destination port]. We parse the collected pcap data to extract bidirectional network flows. We then extract 32 statistical flow-based features based on flow duration, inter arrival time of packets, packet transmission rate, data transmission rate, packet size, flags and count of packets. Feature selection is applied to select the best features. We then apply AutoML to employ several Machine Learning and Deep Learning algorithms on the obtained data. We test the robustness of the trained model using practical adversarial attacks in a real environment to ensure that an attacker is not able to evade the Intrusion Detection System (IDS) and get past the defense easily. We then perform adversarial training to enhance the robustness of our detection system. Finally, we present a simple GUI application to enable any user to deploy our system over his network easily in real time.

3.1. Dataset

The datasets that are publicly available for performing DDoS attack detection are as follows:

- DARPA dataset (2000) (DARPA 2000Intrustion Detection Scenario Specific Data Sets, 2000)
- CAIDA DDoS dataset (2007) (*CAIDA UCSD "DDoS Attack" 2007 Dataset* , 2007)
- CIC-IDS dataset (2017) (Sharafaldin et al., 2018)
- CIC-IDS dataset on AWS (2018) (*CSE-CIC IDS 2018Dataset on AWS* , 2018)
- CIC-DDoS dataset (2019) (Sharafaldin et al., 2019)

Among these datasets, the DARPA dataset (2000) and the CAIDA dataset (2007) are quite outdated. The CIC-IDS 2017 and 2018 datasets contain the same categories of DDoS attacks. However, handling the CIC-IDS 2017 dataset is easier due to its smaller size compared to the 2018 version (which is over 789 GB). Hence, for this work, we employ the CIC-DDoS (2019) dataset and the DDoS data present in the CIC-IDS 2017 dataset.

Table 1. Attacks present in CIC-DDoS 2019.

Attacks	Attack Times	Attacks	Attack Times
NTP	10:35–10:45	SSDP	12:27–12:37
DNS	10:52–11:05	UDP	12:45–13:09
LDAP	11:22–11:32	UDP-Lag	13:11–13:15
MSSQL	11:36–11:45	WebDDoS	13:18–13:29
NetBIOS	11:50–12:00	SYN Flood	13:29–13:34
SNMP	12:12–12:23	TFTP	13:35–17:15

3.1.1. CIC DDoS 2019 dataset

The CICDDoS2019 dataset mimics the data in a real attack scenario, and contains packet captures of recent and most widespread DDoS attacks. The dataset contains packet capture files (pcaps) of two days. Day 1 has been called as the "Testing day" and Day 2 has been called as the "Training day." In this work, we employ day 2 data, which comprises 1514 pcap files with a total size of about 190 GB. The attack types present in the data and their attack times (GMT-3) are mentioned in Table 1.

3.1.2. CIC IDS 2017 dataset

This dataset contains packet captures of several days from July 3, 2017 to July 7, 2017. July 3 data contains only benign traffic where as data of other days contains captures of various attacks. The purpose of using this dataset apart from the CIC DDoS 2019 dataset, is to collect data on WebDDoS attacks (which is insufficient in the latter) and benign traffic data which is not present in the CIC DDoS 2019 dataset. The data of July 3 (Benign traffic) and July 5 and July 7 (DDoS captures) has been collected. The attacks fetched from this dataset and their attack times is described in Table 2. All these attacks have been labeled as WebDDoS attacks.

3.2. Feature engineering

In data processing, we filter and process the collected packet capture files to generate flow-based features. The data processing steps are as follows:

3.2.1. Packet filtering

As we are dealing with only Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) based DDoS attacks in this work, we filter out all packets containing either TCP or UDP layer. This included ICMP packets and ARP packets. Other protocols which get filtered using this rule are: Spanning Tree Protocol (STP), Open Shortest Path First (OSPF), Dynamic Trunk Protocol (DTP) and Cisco Discovery Protocol (CDP).

Table 2. Attacks fetched from CIC IDS 2017.

Attacks	Attack Date	Attack Times
DoS Hulk	July 5, 2017	10:43–11:00
DoS GoldenEye	July 5, 2017	11:10–11:23
DDoS LOIC	July 7, 2017	15:56–16:16

3.2.2. Flow extraction and feature extraction

The following are some of the characteristics of DDoS attacks that we use as a basis for our features:

(1) **Packet length**: In a DDoS attack, attackers use tools which usually generate fixed length packets from a particular source. However, benign traffic usually has packets of variable lengths.

(2) **Packet transmission rate**: In a DDoS attack, packet transmission rate is very high compared to that of normal traffic.

(3) **Inter-arrival time**: Inter-arrival time is the time interval between the timestamps of 2 consecutively transmitted unidirectional packets. In a DDoS attack, the inter-arrival time between two packets is expected to be lesser as compared to that of benign traffic.

(4.) **Flags**: DDoS attacks like TCP flood rely on setting some very specific flags in the packet to carry out the attack. These flag patterns can be helpful to identify DDoS traffic.

We extract the features in 5-tuple flow-based format (Protocol, Source IP, Source Port, Destination IP, Destination Port). The 2-tuple format was not possible as the dataset does not contain sufficient source and destination IP addresses for attacks. In fact, even though multiple source IPs from attack network were used for DDoS attacks, the packet captures of the attacks contain a single source IP address. We use the **scapy** (scapy, 2022) library in Python to extract the features from.pcap files. The scapy library contains an inbuilt function called *sessions()* which can be utilized to generate bi-directional flows. The classes and the number of flows obtained for each class is described in Table 3. A total of 38 flow-based features were extracted, as illustrated in Table 4.

3.2.3. Feature selection

We extract 38 features based on DDoS behavior. However, some of these features might be correlated. Correlated features must be removed as some

Table 3. Number of flows for each DDoS attack type.

Class	Number of flows	Class	Number of flows
TFTP	13984791	Syn-Flood	1506161
SNMP	4805717	NTP	1184288
NetBIOS	3809469	Normal	205450
MSSQL	3414788	WebDDoS	132107
UDP-Flood	3068899	DNS	14253
SSDP	2540355	UDP-Lag	3699
LDAP	2032022		

models tend to become unstable in the presence of high feature correlations (Toloşi & Lengauer, 2011). Having correlated features also causes misleading feature ranking. We use Pearson's correlation matrix to find highly correlated features. Each cell of this matrix represents the value of Pearson's correlation coefficient between two variables. We remove the length based features – "max_fpktl," "min_fpktl," "mean_fpkptl" because during experiments it is found that, as the dataset contains packets of fixed length for an attack type, the Machine Learning model tends to overfit on the length of the packets in the forward direction.

The Pearson's correlation matrix for the extracted features is shown in Figure 2. We identify the following pairs of features which have very high positive correlation ($>98\%$):

- total_fiat and max_fiat (99.2%)
- total_fiat and mean_fiat (98.3%)
- total_biat and max_biat (98.3%)
- total_biat and max_biat (98.6%)
- flowBytesPerSecond and flowPktsPerSecond (99.5%)
- flowBytesPerSecond and fPktsPerSecond (98.4%)

The features *total_fiat*, *total_biat* and *flowBytesPerSecond* are removed. We also observe that the correlation values of the feature *flow_urg* have null values. This is because this feature has

Table 4. Extracted features.

Feature name	Description
proto	Protocol
service	Type of Service (tos) field
flduration	Duration of the flow in microseconds
total_fpackets	Number of forward packets in the flow
total_bpackets	Number of backward packets in the flow
min_fpktl	Minimum length of a forward packet
min_bpktl	Minimum length of a backward packet
max_fpktl	Maximum length of a forward packet
max_bpktl	Maximum length of a backward packet
mean_fpktl	Mean length of forward packets
mean_bpktl	Mean length of backward packets
std_fpktl	Standard deviation of lengths of forward packets
std_bpktl	Standard deviation of lengths of backward packets
total_fiat	Sum of inter arrival times of forward packets
total_biat	Sum of inter arrival times of backward packets
min_fiat	Minimum inter arrival time between two forward packets
min_biat	Minimum inter arrival time between two backward packets
max_fiat	Maximum inter arrival time between two forward packets
max_biat	Maximum inter arrival time between two backward packets
mean_fiat	Mean of inter arrival times of forward packets
mean_biat	Mean of inter arrival times of backward packets
flow_psh	Count of packets with PSH flag set
flow_urg	Count of packets with URG flag set
flow_syn	Count of packets with SYN flag set
flow_ack	Count of packets with ACK flag set
flow_fin	Count of packets with FIN flag set
flow_rst	Count of packets with RST flag set
flow_cwr	Count of packets with CWR flag set
flow_ece	Count of packets with ECE flag set
fwd_df	Number of forward packets with DF flag set
fwd_mf	Number of forward packets with MF flag set
total_fhlen	Sum of header lengths in the forward packets
total_bhlen	Sum of header lengths in the backward packets
fPktsPerSecond	Packet transmission rate in the forward direction
bPktsPerSecond	Packet transmission rate in the backward direction
flowPktsPerSecond	Packet transfer rate (Forward + Backward)
flowBytesPerSecond	Data transmission rate (Forward + Backward)
downUpRatio	Ratio of total length of forward packets to total length of backward packets

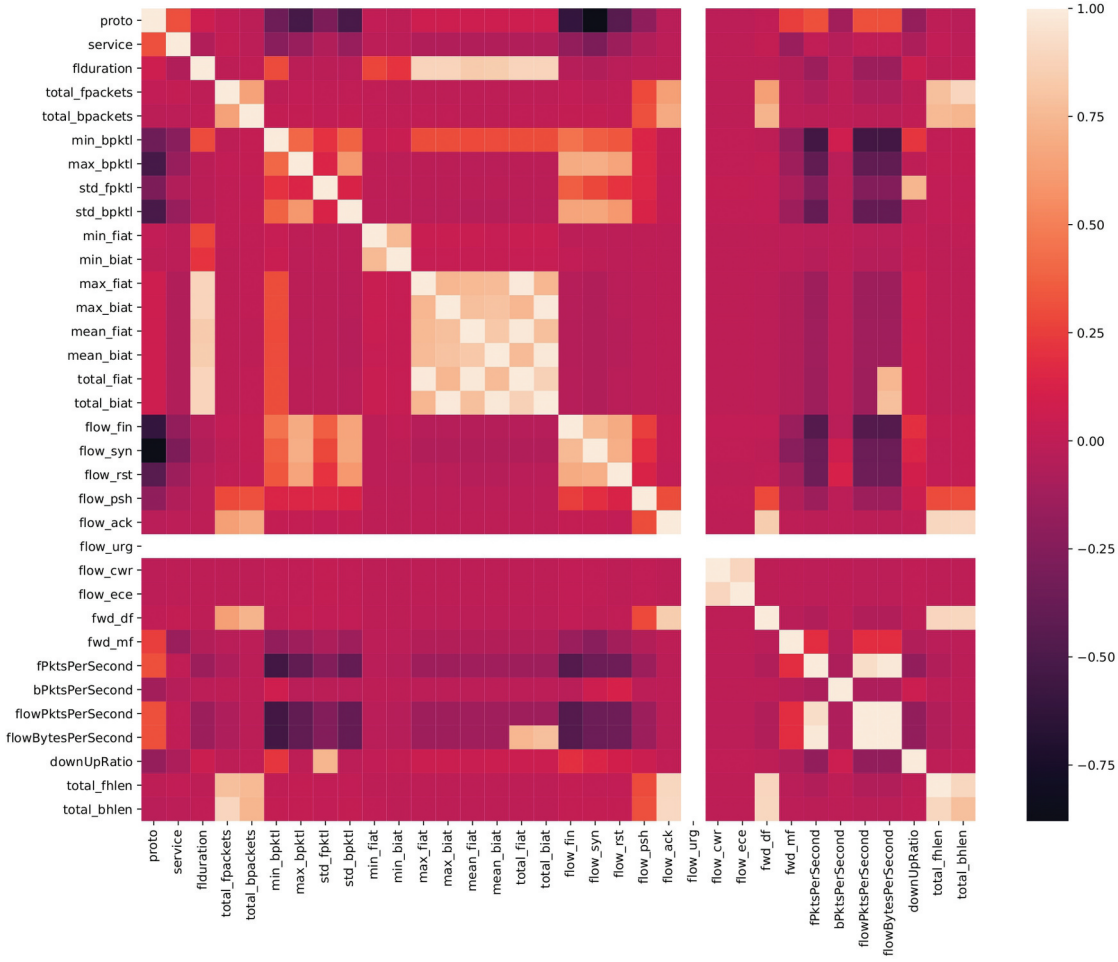


Figure 2. Pearson's correlation matrix for extracted features.

a single value of 0 throughout the dataset, implying that the URG flag is not used at all in the dataset. The feature *flow_urg* is also removed. The final set of features after feature selection is illustrated in Table 5.

3.3. Class filtering and class balancing

Table 3 shows that there are 13 classes of DDoS attacks present in our pre-processed dataset. However, we observe that the features of some of these classes have very little variation from each other, hence confusing the classifier. An example of such classes is SNMP and SSDP. We hereby remove classes on which the model performs poorly. These classes are SSDP, SNMP and NetBIOS. We remove the classes "DNS" and "UDP-Lag" as data of these classes is sparse compared to other classes.

Table 3 shows an unequal distribution of classes in the dataset. A classification dataset is called imbalanced if it has skewed class proportions. Classes that make up a large proportion of the dataset are called majority classes, while those making up a smaller proportion are called minority classes. In our case, TFTP is a majority class while UDP-Lag is a minority class. Having an unbalanced dataset is problematic because the model will spend a large amount of time training on the majority classes and not enough time training on the minority classes, leading to a bias in the model's predictions. A way to fix this issue is by undersampling the majority classes. We under sample the classes "TFTP," "MSSQL," "UDP-Flood" and "LDAP" to 200,000 samples using random sampling. The classes and their number of samples after class balancing is illustrated in Table 6.

3.4. Classification using machine learning

We employ AutoML to find the best classification algorithm for our data along with the best performing

Table 5. Selected features.

Feature name	Description
proto	Protocol
service	Type of Service (tos) field
flduration	Duration of the flow in microseconds
total_fpackets	Number of forward packets in the flow
total_bpackets	Number of backward packets in the flow
min_bpktl	Minimum length of a backward packet
max_bpktl	Maximum length of a backward packet
std_fpktl	Standard deviation of lengths of forward packets
std_bpktl	Standard deviation of lengths of backward packets
min_fiat	Minimum inter arrival time between two forward packets
min_biat	Minimum inter arrival time between two backward packets
max_fiat	Maximum inter arrival time between two forward packets
max_biat	Maximum inter arrival time between two backward packets
mean_fiat	Mean of inter arrival times of forward packets
mean_biat	Mean of inter arrival times of backward packets
flow_psh	Count of packets with PSH flag set
flow_syn	Count of packets with SYN flag set
flow_ack	Count of packets with ACK flag set
flow_fin	Count of packets with FIN flag set
flow_rst	Count of packets with RST flag set
flow_cwr	Count of packets with CWR flag set
flow_ece	Count of packets with ECE flag set
fwd_df	Number of forward packets with DF flag set
fwd_mf	Number of forward packets with MF flag set
total_fhlen	Sum of header lengths in the forward packets
total_bhlen	Sum of header lengths in the backward packets
fPktsPerSecond	Packet transmission rate in the forward direction
bPktsPerSecond	Packet transmission rate in the backward direction
flowPktsPerSecond	Packet transfer rate (Forward + Backward)
downUpRatio	Ratio of total length of forward packets to total length of backward packets

Table 6. Data obtained after class filtering and balancing.

Class	Number of flows	Class	Number of flows
Normal	205450	Syn-Flood	200000
NTP	200000	TFTP	200000
UDP-Flood	200000	LDAP	200000
MSSQL	200000	WebDDoS	132107

hyperparameters. Automated machine learning (AutoML) is an emerging field that seeks to select, compose, and parameterize ML models automatically (Waring et al., 2020). AutoML eliminates the hassle of manually trying out various Machine Learning/Deep Learning algorithms and performing hyperparameter tuning, as it does all of these automatically.

We use the AutoML functionality of the H2O library (H2O automl (2022a)) in Python to perform this task. The algorithms available in H2O AutoML are – **DRF** (Distributed Random Forest), **XRT** (Extremely Randomized Trees), **GLM** (Generalized Linear Model with regularization), **GBM** (Gradient Boosting Machine), **XGB** (eXtreme Gradient Boosting), **ANN** (Artificial Neural Networks), and **Stacked Ensembles**.

To keep things simple and reduce training time, we exclude Stacked Ensembles from our analysis. We split the training and validation sets in the ratio 80:20.

3.5. Real-time testing

We develop a real-time detection module, which captures network traffic in promiscuous mode using *tcpdump* (Tcpdump, 2022) and queries the ML model to make predictions for every x packets captured. We find the optimal value of the parameter x to be 10,000. For a value less than this, the model will be queried too frequently, leading to high resource consumption. For higher values of x, the time taken by the model to make predictions increases significantly, and may even yield captures with reuse of port numbers for some DDoS attacks, giving unusual values of some features like flow duration, minimum inter-arrival time and packet transmission rate. We first try to replicate the attacks present in the training data itself and report the detection performance of our model against these attacks. Then, we test the performance of the model against adversarial attacks. Finally, we perform adversarial training of the model to check it's robustness and report the detection performance after adversarial training.

For easy deployment of our detection system, we develop a Tkinter (Tkinter library in Python, 2022) based GUI application, which demonstrates attack detection in real time. Figure 3 shows an illustration of our GUI. The application accepts the network interface to sniff as the input. On clicking "Start sniffing," the detection system starts making predictions as explained. The two status bars at the bottom display the prediction statistics and the type of traffic detected respectively. The packets captured in real time and the flows generated are also saved as pcap and csv files respectively for manual inspection, if needed. As the application is Python based, it can easily run on any Operating System with python installed.

4. Experiments and results

This section describes our experiments in details. We divide our experiments in two phases – Phase 1 includes Modelling using Automated Machine

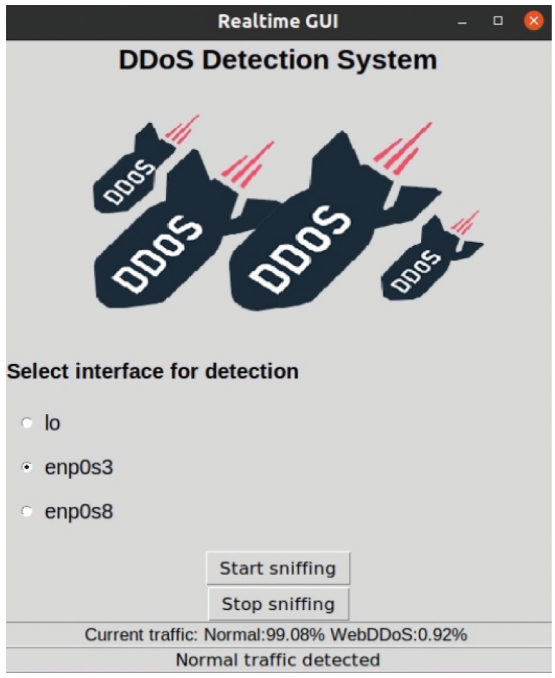


Figure 3. GUI of the proposed detection system.

Learning (AutoML). Phase 2 includes adversarial attacks, adversarial training and demonstration of model deployment using GUI.

4.1. Phase 1

All experiments carried out for training and testing in Phase-1 of experiments are performed on a Ubuntu 20.04 LTS cloud server with 128GB of RAM and Intel Xeon E5–2670 CPU with 20 cores. The various tools used in Phase 1 are Wireshark (Wireshark, 2022), tcpdump (Tcpdump, 2022), sklearn (sklearn., 2022), scapy (scapy, 2022), and H20 (2022a).

The dataset is divided into train and validation sets in the ratio 80:20. We apply AutoML to train models using various Machine Learning algorithms. Hyperparameters are also tuned automatically for each algorithm. Table 7 shows the performance of each algorithm at validation time, where FPR is False Positive Rate. Note that the values of precision, recall, FPR and accuracy are macro averaged values. It is evident that XGBoost, GBM and DRF give the best performance in terms of accuracy and false positive rate, with XGBoost and GBM performing equally good. We consider XGBoost as the winner here, as it performs better than GBM in terms of prediction speed. We discard GLM from further analysis.

Table 7. Performance of various algorithms using AutoML.

Algorithm	Mean per class error	Precision	Recall	FPR	Accuracy
XGBoost	0.0036	0.9966	0.9965	0.0005	0.9991
GBM	0.0036	0.9966	0.9965	0.0005	0.9991
DRF	0.0059	0.9956	0.9955	0.0006	0.9988
XRT	0.0049	0.9954	0.9953	0.0007	0.9987
ANN	0.0849	0.9355	0.8933	0.0158	0.9724
GLM	0.2138	0.8081	0.8045	0.0272	0.9525

4.2. Phase 2

To perform the experiments for Phase 2 of our work, we use a Ubuntu 20.04 with 32 GB RAM and Intel i7 octa core processor as a host machine. We use one Ubuntu 20.04 VM as the server with 8 GB RAM and access to the 8 cores of the host machine, 3 Ubuntu 20.04 VMs and 1 Windows 10 VM as the attackers, each with 2 GB RAM and access to 4 cores of the host machine. For real network simulation, we use 3 Ubuntu 20.04 machines and 1 Windows 10 machine, each with 32 GB RAM and Intel i7 octa core processors; one Ubuntu machine as the victim and 4 other machines as the attackers in our lab environment. Table 8 gives a description of the tools used in Phase 2 of our experiments.

Table 8. Tools used in phase 2.

Name	Version	Attacks
hping3	3.0.0	SYN-Flood
Metasploit framework	6.1.40	SYN-Flood
High Orbit Ion Cannon (HOIC)	1.1	HTTP (Get+Post)
Low Orbit Ion Cannon (LOIC)	1.0.6	HTTP (Get)
Hulk	1.0	HTTP
Pyflooder	1.0	HTTP (Get)
Doser	1.0	HTTP (Get, Post)
GoldenEye	1.2	HTTP (Get, Post)
DoS	5.0	HTTP
Fireflood	1.2	HTTP

Table 9 shows the performance of the top 5 algorithms against benign traffic and attacks performed using tools whose data is present in the training set. While it is not known which tool is used to perform Syn-Flood attacks in the dataset, we assume it is performed using Metasploit-framework. We perform further testing against other Syn-Flood tools.

Table 9. Accuracy of various algorithms against attacks using tools whose data is present in the dataset.

Attack Type	Syn-Flood	GoldenEye	GoldenEye	Hulk	LOIC	Benign
\Classifier	(%)	(Get) (%)	(Post) (%)	(%)	(%)	(%)
ANN	99.8	42.4	40.19	23.11	96.0	35.0
DRF	94.3	99.1	91.2	43.16	99.8	93.3
GBM	95.3	96.4	93.8	50.2	99.5	96.9
XGB	96.9	98.6	98.0	50.4	99.8	98.7
XRT	83.6	70.8	70.3	60.5	99.3	97.7

From the Table 9, one can observe that ANN performs well on Syn-Flood and LOIC attacks, but poorly against other attacks. The XGB algorithm performs well against all attacks, slightly better than GBM. This is followed by DRF and XRT, where DRF performs slightly better than XRT for most attacks. The ANN is the least performing algorithm – it performs well on Syn-Flood, LOIC and benign traffic but poorly against others. Strangely, all the classifiers gave average performance against the Hulk tool. This is possibly caused by difference in versions of the tool used for training and testing. Further, we evaluate the adversarial performance of the top 3 performers – XGB, GBM and DRF.

We use the hping3 tool to make changes in the IP/TCP fields of a packet, making sure that the attack is still potent. The Syn-flood detection accuracy is the fraction of Syn-flood instances predicted in the total predictions made by the classifier.

(1) **Packet size**: Figure 4a shows the variation in Syn-flood detection accuracy of the three classifiers with the packet size. For DRF, as the packet size is increased from 0 B to 10 B, there is a drop in accuracy from 98.23% to 96.83%. After this there is a slight increase in accuracy, followed by a constant trend. GBM and XGB algorithms show robustness to a change in packet size, and their accuracy remains almost constant as the packet size increases.

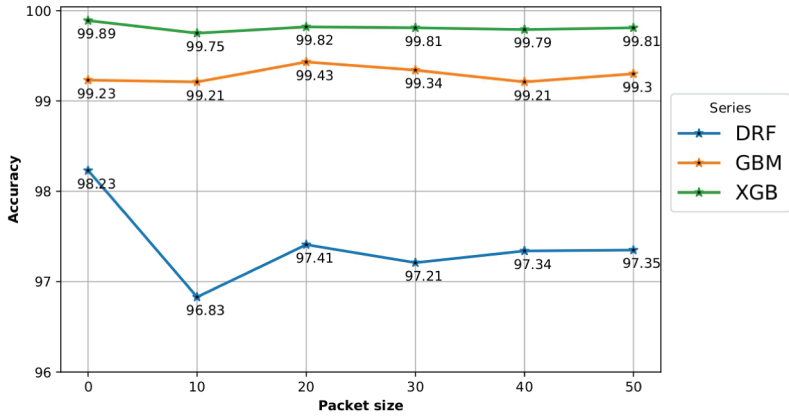
(2) **Inter-arrival time**: Figure 4b shows the variation in Syn-Flood detection accuracy of the three classifiers with the inter-arrival time between packets. Note that, changing the inter-arrival time also changes the transmission rate of packets. For DRF, the highest accuracy is achieved between 50–100 μs. Any increase or decrease in the inter-arrival time causes a drastic reduction in the accuracy. GBM and XGB follow a similar trend. However, XGB is maintain a constant trend in a longer range of 50–400 μs, after which there is a slight decrease in accuracy with increase in inter-arrival time.

(3) **Flags, TOS field and combinations**: We define the following 5 combinations:

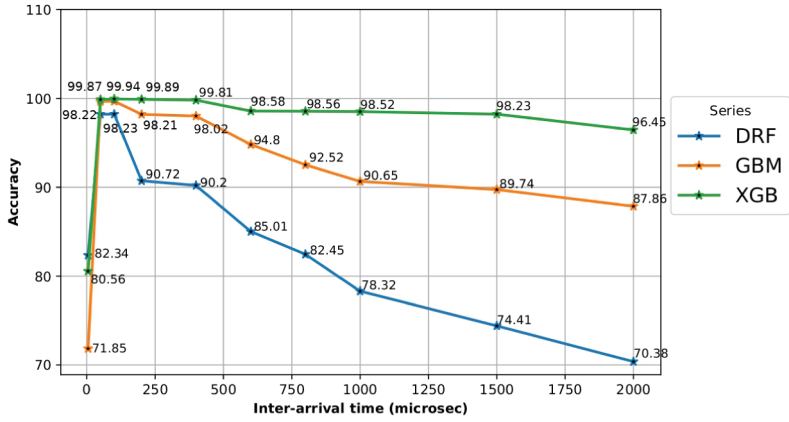
(a) PSH, URG flags set.
(b) TOS (type of service) field changed to 121.
(c) A combination of flags and TOS fields set as defined in points a and b.
(d) A combination of flags, TOS fields set as defined in points c + packet length set to 50 B.
(e) A combination of flags, TOS fields and packet length set to 50 B set as defined in point d + inter-arrival time set to 1500 μs.

As evident, the classifiers GBM and XGB do not experience any significant drop in accuracy for the first 4 combinations, but for the 5^{th} combination, there is a drastic fall in the accuracy. The DRF classifier experiences a slight fall in accuracy at the 4^{th} combination, with a drastic fall at the 5^{th} combination. XGB classifier comes out to be the best performing and most robust among the three classifiers.

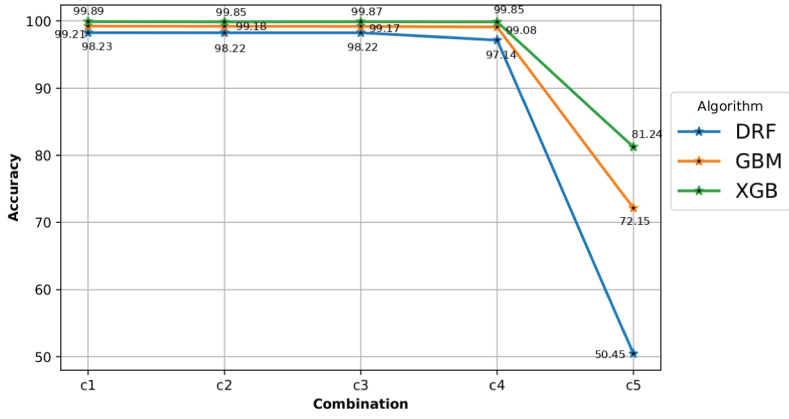
Table 10 illustrates the performances of the algorithms DRF, GBM and XGB against new HTTP-flood tools (whose data is not present in the dataset). For the tools Fireflood, Doser and Pyflooder, all the classifiers show poor performance. HOIC attack traffic is detected best by the GBM classifier with 99.5% accuracy, while DOS 5.0 traffic is detected with a decent accuracy by all the classifiers. Except for HOIC, the XGB classifier again outperforms the other two classifiers.



(a) Variation of accuracy against packet sizes



(b) Variation of accuracy against inter-arrival time



(c) Variation of accuracy against different combinations of perturbations

Figure 4. Accuracy of various algorithms against adversarial syn-flood attacks.

Table 10. Accuracy of various algorithms against new HTTP-flood tools.

Attack Type	Fireflood	HOIC	DOS 5.0	Doser	Doser	Pyflooder
\Classifier	(%)	(%)	(%)	(Get) (%)	(Post) (%)	(%)
DRF	1.0	67.8	82.3	3.6	4.7	2.8
GBM	1.5	99.5	76.6	3.6	3.5	5.5
XGB	10.4	70.8	81.5	10.8	13.5	34.4

All experiments reveal the vulnerabilities in the detection system, which can be leveraged by an attacker to get past the detection system and avoid detection. We choose XGB as the best performing algorithm. In the preceding section, we perform adversarial training of this classifier with

some additional data of packet captures of tools against which this classifier performed poorly.

4.2.1. Adversarial training

We perform adversarial training of the XGB model, with the following additional packet captures of the following:

(1) **Syn-Flood**: Packet captures from traffic generated by hping3 for the following settings:

(a) Inter-arrival time 100 μs, packet sizes 0, 10 and 20.
(b) Packet size 0, inter-arrival times 200, 400, 600, 800, 1000, 1500 and 2000 μs.
(c) Packet size 25/50, PSH and URG flags set, TOS set to 121/192, inter-arrival time set to 1500/2000 μs.
(d) **HTTP-Flood**: Packet captures for the tools DOS 5.0, Doser, Fireflood, HOIC, Pyflooder and Hulk.

The dataset used for adversarial training is described in Table 11. The learning curve of the retrained model is illustrated in Figure 5. The figure shows the variation of log loss against the number of trees (which is a hyperparameter for this model). The curves for training and cross validation perfectly coincide. At number of trees = 82, the log loss ceases to decrease further, hence the AutoML stops growing more trees at this point and takes 82 as the number of trees for the XGB model.

Table 12 shows the derive mean per class accuracy, weighted average precision, recall and the false positive rate.

4.2.2. Testing the retrained model

We test the retrained XGB model against the attacks from all the tools we have experimented till now. Figure 6 highlights the performance of the retrained model. Here c5 is the same combination of packet modifications as previously mentioned in the *Flag, TOS, and combinations* section.

The model performs well against all kinds of attack traffic and benign traffic. The performance against the Hulk tool improved, but is still not perfect. We examine the cause behind this and find that the some of the flows generated from the traffic of the Hulk tool actually exhibits benign traffic behavior. These flows have the values of the features $total\_fpackets = 1$ and $total\_bpackets = 0$. In this case, the values of the features related to the backward direction of traffic will have default values, i.e. 0. The features related to inter-arrival time will also have the default values. Hence, classifying these flows as benign is indeed, not a misclassification.

Table 11. Data used for adversarial training.

Class	Number of flows	Class	Number of flows
Syn-Flood	250000	UDP-Flood	200000
WebDDoS	216301	LDAP	200000
Normal	205450	MSSQL	200000
TFTP	200000	NTP	200000

Table 12. Performance metrics of the retrained XGB model.

Accuracy	Precision	Recall	FPR
0.9991	0.9967	0.9969	0.0005

4.2.3. Model deployment

We demonstrate the deployment of our detection system on a real network. Our GUI integrates the retrained XGB model, and Figure 7 shows how our GUI displays the statistics of the predictions made by the model in real time. Figure 7a represents the state under a Syn-Flood attack, performed using Metasploit on 3 other systems on the network. Syn-Flood is detected with high accuracy with a slight false-positive of WebDDoS. Figure 7b shows the state under a WebDDoS attack, performed using Pyflooder on 3 other systems on the network. WebDDoS has been detected with high accuracy.

5. Conclusion and future work

DDoS attacks are a real threat in today's cyber space, and their detection is crucial to mitigate them. In this work, we address the research gaps present in the previous work done in this domain. We collect data from the CIC-DDoS 2019 and CIC-IDS 2017 datasets. We present a robust DDoS detection and classification framework using a 5-tuple bidirectional flow-based approach and Machine Learning. We develop a Python-based tool to parse pcap files, construct bidirectional traffic flows, extract features for each flow and save the output in csv format. We use AutoML to perform automatic model training and

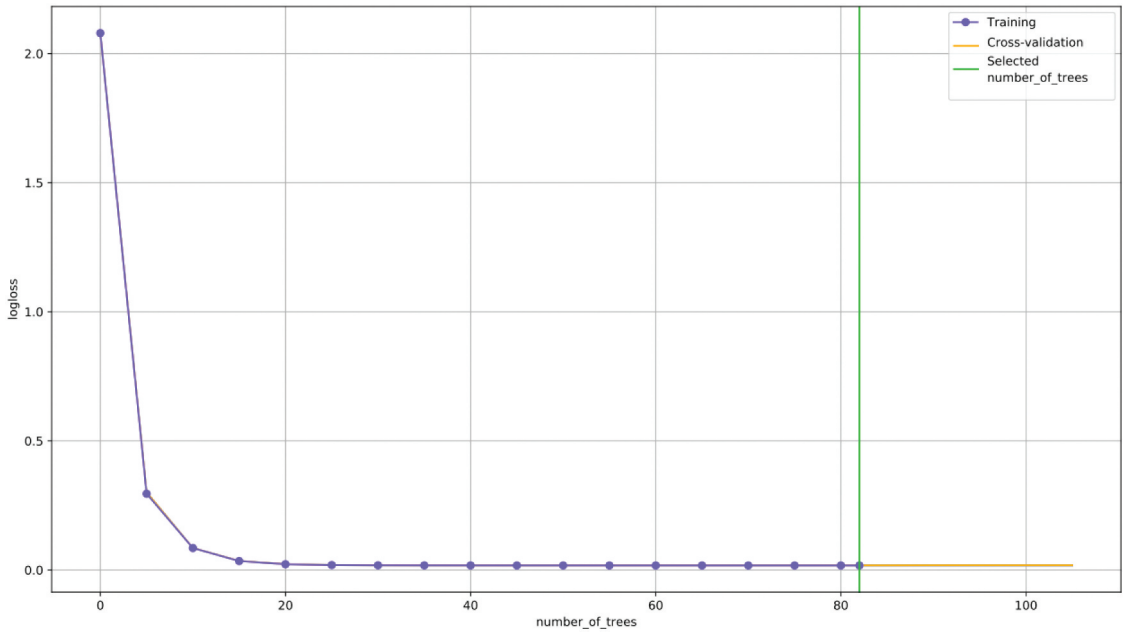


Figure 5. Learning curve of XGB model.

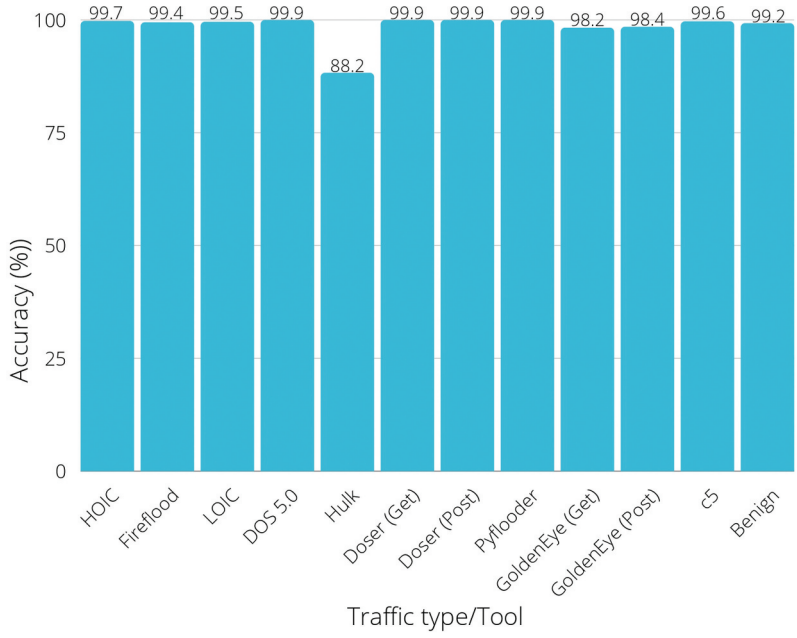
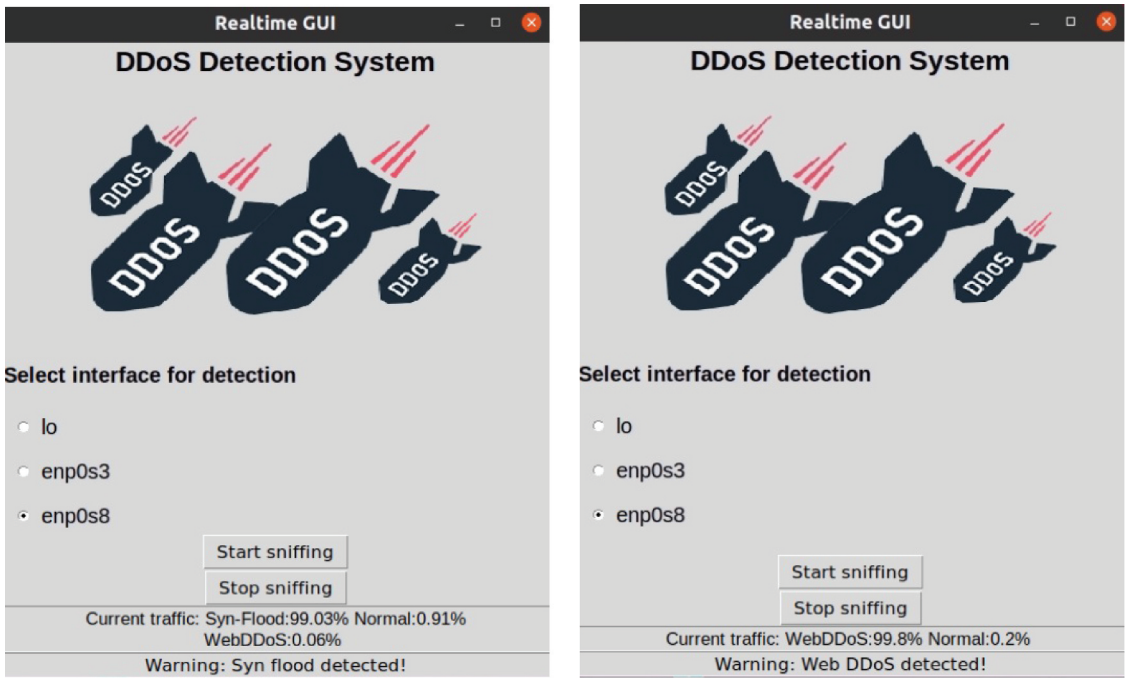


Figure 6. Performance of model after adversarial training.

hyperparameter tuning. In our work, eXtreme Gradient Boosting (XGB) classifier performs as the best classifier. We test the performance of our detection system on live network traffic. Practical adversarial attacks are performed by perturbing the problem space itself to test the robustness of our model against evasion attacks. We also perform adversarial training to improve the robustness of our model. We test the retrained model to prove that, it is indeed robust to adversarial attacks. We demonstrate the real-time deployment of our model through GUI. We achieve the best accuracy of 99.9% and a low False Positive Rate (FPR) of 0.05%.



(a) Current traffic: Syn-Flood attack (b) Current traffic: WebDDoS attack

Figure 7. Demonstration of the GUI of our DDoS detection system.

In future, one can work on the detection of slow DDoS attacks (Cambiaso et al., 2013), more informative features can be derived by diving deep into DDoS attack traffic, and detection and evasion performance against UDP-Flood and reflection-based DDoS categories can also be tested.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Abdelaty, M., Scott-Hayward, S., Doriguzzi-Corin, R., & Siracusa, D. (2022). Gadot: Gan-based adversarial training for robust ddos attack detection. arXiv. https://arxiv.org/abs/2201.13102

Agarwal, B., & Mittal, N. (2012, 12). Hybrid approach for detection of anomaly network traffic using data mining techniques. *Procedia Technology*, *6*, 996–1003. https://doi.org/10.1016/j.protcy.2012.10.121

Aggarwal, A., & Gupta, A. (2015). Survey on data mining and ip traceback technique in ddos attack. *International Journal of Engineering and Computer Science*, *4*(6), 12595–12598.

Aiken, J., & Scott-Hayward, S. (2020, March 19). Investigating adversarial attacks against network intrusion detection systems in SDNs. In *IEEE conference on network functions virtualization and software defined networks 12/11/2019 → 14/11/2019 dallas, united states*. IEEE. Conference on Network Functions Virtualization and Software Defined Networks, IEEE NFV-SDN; Conference date: 12-11-2019 Through 14-11-2019. https://nfvsdn2019.ieee-nfvsdn.org/

Alhajjar, E., Maxwell, P., & Bastian, N. D. (2020). Adversarial machine learning in network intrusion detection systems. *CoRR, Abs/2004.11898*. https://arxiv.org/abs/2004.11898

Apruzzese, G., Andreolini, M., Ferretti, L., Marchetti, M., & Colajanni, M. (2021). Modeling realistic adversarial attacks against network intrusion detection systems. *CoRR, Abs/2106.09380*. https://arxiv.org/abs/2106.09380

Arun Raj Kumar, P., & Selvakumar, S. (2013). Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. *Computer Communications*, *36*(3), 303–319. https://doi.org/10.1016/j.comcom.2012.09.010

Asad, M., Asim, M., Javed, T., Beg, M. O., Mujtaba, H., & Abbas, S. (2019, 07). Deep- detect: Detection of distributed denial of service attacks using deep learning. *The Computer Journal*, *63*(7), 983–994. https://doi.org/10.1093/comjnl/bxz064

Ateş, C., Ozdel, S., & Anarim, E. (2019, 10). Clustering based ddos attack detection using the relationship between packet headers. *In*, 1–6.

Ayub, M. A., Johnson, W. A., Talbert, D. A., & Siraj, A. (2020). Model evasion attack on intrusion detection systems using adversarial machine learning. In *2020 54th annual conference on information sciences and systems (ciss)* (p. 1–6).

Bela, G., & Siaterlis, C. (2013, 06). Analysis of the effects of distributed denial-of- service attacks on mpls networks. *International Journal of Critical Infrastructure Protection*, *6*(2), 87–95. https://doi.org/10.1016/j.ijcip.2013.04.001

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, *16*(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046

Bukac, V., & Matyas, V. (2015). *Analyzing traffic features of common standalone dos attack tools*. Springer-Verlag. Retrieved from. https://doi.org/10.1007/978-3-319-24126-52

Caida ucsd "ddos attack"2007 dataset. (2007). http://www.caida.org/data/passive/ddos-20070804dataset.xml

Cambiaso, E., Papaleo, G., Chiola, G., & Aiello, M. (2013). Slow dos attacks: Definition and categorisation. *International Journal of Trust Management in Computing and Communications*, *1*(3–4), 300–319. https://www.inderscienceonline.com/doi/abs/10.1504/IJTMCC.2013.056440 (PMID:56440)

Chauhan, R., & Shah Heydari, S. (2020). Polymorphic adversarial ddos attack on ids using gan. In *2020 international symposium on networks, computers and communications (isncc)* (p. 1–6).

Chen, R.-C., Cheng, K.-F., & Hsieh, C.-F. (2010, 04). Using rough set and support vector machine for network intrusion detection. *International Journal of Network Security & Its Applications*, *1*.

Cse-cic ids 2018 dataset on aws. (2018). https://www.unb.ca/cic/datasets/ids-2018.html

Darpa 2000 intrustion detection scenario specific data sets. (2000). https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-cenario-specific-data-sets

Ddos 2021-q4 stats. (2021). https://radar.cloudflare.com/notebooks/ddos-2021-q4

Ddos attack patterns statistics. (2023). https://www.stationx.net/ddos-statistics/#:~:text=DDoS%20Attack%20Patterns%20Statistics,-1.&text=There%20has%20been%20a%20807,high%20benchmark%20for%20attack%20frequency

Ddos attacks history by radware. (2020). https://www.radware.com/security/ddos-knowledge-center/ddos-chronicles/ddos-attacks-history/

Ddos q3-2021 stats. (2022). https://securelist.com/ddos-attacks-in-q3-2021/104796/

Filho, F., Silveira, F., Junior, A., Vargas Solar, G., & Silveira, L. (2019, 10). Smart detection: An online approach for dos/ddos attack detection using machine learning. *Security and Communication Networks*, *2019*, 1–15. https://doi.org/10.1155/2019/1574749

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009 Feb). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Garg, K., & Chawla, R. (2011). Detection of ddos attacks using data mining. *International Journal of Computing and Business Research (IJCBR)*, *2*(1), 2226–6166.

Gyanchandani, M., & Rana, J. L. (2012). Taxonomy of anomaly based intrusion detection system: A review.

H2o automl. (2022a). https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

The hindu: Rise in cyber crime in india. (2020). https://www.thehindu.com/news/national/india-reported-118-rise-in-cyber-crime-in-2020-578-incidents-of-fake-news-on-social-media-data/article36480525.ece

Imamverdiyev, Y., & Abdullayeva, F. (2018). Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big Data*, *6*(2), 159–169. https://doi.org/10.1089/big.2018.0023

Key internet statistics to know in 2022. (2022). https://www.broadbandsearch.net/blog/internet-statistics

Lee, W., & Xiang, D. (2001, 02). Information-theoretic measures for anomaly detection. *In*, 130–143.

Limwiwatkul, L., & Rungsawang, A. (2004). Distributed denial of service detection using tcp/ip header and traffic measurement analysis. In *Ieee international symposium on communications and information technology* (Vol. 1, p. 605–610).

Lin, Z., Shi, Y., & Xue, Z. (2018). IDSGAN: Generative adversarial networks for attack generation against intrusion detection. *CoRR*. http://arxiv.org/abs/1809.02077

Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial- of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, *13*(12), 1550147717741463. https://doi.org/10.1177/1550147717741463

Muraleedharan, N., & Janet, B. (2021). A deep learning based http slow dos classification approach using flow data. *ICT Express*, *7*(2), 210–214. https://doi.org/10.1016/j.icte.2020.08.005

Mustapha, A., Khatoun, R., Zeadally, S., Chbib, F., Fadlallah, A., Fahs, W., & El Attar, A. (2023). Detecting ddos attacks using adversarial neural network. *Computers & Security*, *127*, 103117. https://doi.org/10.1016/j.cose.2023.103117

Nadiammai, G. V., & Hemalatha, M. 2013. Effective approach toward Intrusion Detection System using data mining techniques. *Egyptian Informatics Journal*, *15*(1), 37–50. https://doi.org/10.1016/j.eij.2013.10.003

News article: Cloudflare mitigates record-breaking https ddos attack. (2022). https://www.bleepingcomputer.com/news/security/cloudflare-itigates-record-breaking-https-ddos-attack/

Novaes, M. P., Carvalho, L. F., Lloret, J., & Proença, M. L. (2021). Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, *125*, 156–167. https://doi.org/10.1016/j.future.2021.06.047

Nugraha, B., Kulkarni, N., & Gopikrishnan, A. (2021). Detecting adversarial ddos attacks in software- defined networking using deep learning techniques and adversarial training. In *2021 ieee international conference on cyber security and resilience (csr)* (p. 448–454).

Papadopoulos, P., Thornewill von Essen, O., Pitropakis, N., Chrysoulas, C., Mylonas, A., & Buchanan, W. J. (2021). Launching adversarial attacks against network intrusion

detection systems for iot. *Journal of Cybersecurity and Privacy*, *1*(2), 252–273. https://doi.org/10.3390/jcp1020014

Patcha, A., & Park, J.-M. J. (2007, 08). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, *51*(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

Sbai, O., & El Boukhari, M. (2020). *Data flooding intrusion detection system for manets using deep learning approach*. Association for Computing Machinery. https://doi.org/10.1145/3419604.3419777

scapy. (2022). https://scapy.net.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Icissp*.

Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 international carnahan conference on security technology (iccst)* (p. 1–8).

She, C., Wen, W., Lin, Z., & Zheng, K. (2017, 01). Application-layer ddos detection based on a one-class support vector machine. *International Journal of Network Security & Its Applications*, *9*(1), 13–24. https://doi.org/10.5121/ijnsa.2017.9102

Shieh, C.-S., Lin, W.-W., Nguyen, T.-T., Chen, C.-H., Horng, M.-F., & Miu, D. (2021, 06). Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Applied Sciences*, *11*(11), 5213. https://doi.org/10.3390/app11115213

sklearn. (2022). https://scikit-learn.org.

Snort: Network Intrusion Detection & Prevention System. (2022b). https://www.snort.org

Sumathi, S., & Karthikeyan, N. (2018). Search for effective data mining algorithm for network based intrusion detection (nids)-ddos attacks. In *2018 international conference on intelligent computing and communication for smart world (i2c2sw)* (p. 41–45).

Tcpdump. (2022). https://www.tcpdump.org.

Thapngam, T., Yu, S., Zhou, W., & Makki, S. (2014, 12). Distributed denial of service (ddos) detection by traffic pattern analysis. *Peer-To-Peer Networking and Applications*, *7*(4), 346–358. https://doi.org/10.1007/s12083-012-0173-3

Thomas, R., Mark, B., Johnson, T., & Croall, J. (2003, 01). Netbouncer: Client- legitimacy-based high-performance ddos filtering. *In*, *1*, 14–25.

Tkinter library in python. (2022). https://docs.python.org/3/library/tkinter.html

Tolo̧şi, L., & Lengauer, T. (2011, 05). Classification with correlated features: Unreliability of feature ranking and solutions. *Bioinformatics*, *27*(14), 1986–1994. https://doi.org/10.1093/bioinformatics/btr300

Virupakshar, K. B., Asundi, M., Channal, K., Shettar, P., Patil, S., & Narayan, D. (2020). Distributed denial of service (ddos) attacks detection system for OpenStack-based private cloud. *Procedia Computer Science (International Conference on Computational Intelligence and Data Science)*, *167*, 2297–2307. https://doi.org/10.1016/j.procs.2020.03.282

Waring, J., Lindvall, C., & Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*, *104*, 101822. https://doi.org/10.1016/j.artmed.2020.101822

Wireshark. (2022). https://www.wireshark.org

Wu, Y. C., Tseng, H. R., Yang, W., & Jan, R. H. (2011, 03). Ddos detection and traceback with decision tree and grey relational analysis. *International Journal of Ad Hoc and Ubiquitous Computing*, *7*(2), 121–136. https://doi.org/10.1504/IJAHUC.2011.038998

Wu, S.-Y., & Yen, E. (2009). Data mining-based intrusion detectors. *Expert Systems with Applications*, *36*(3, Part 1), 5605–5612. https://doi.org/10.1016/j.eswa.2008.06.138

Xia, T., Qu, G., Hariri, S., & Yousif, M. (2005). An efficient network intrusion detection method based on information theory and genetic algorithm. *PCCC 2005 24th IEEE International Performance, Computing, and Communications Conference, 2005*, 11–17.

Zhang, H., Han, D., Liu, Y., Wang, Z., Sun, J., Zhuang, S., & Dong, J. (2024). Explainable and transferable adversarial attack for ml-based network intrusion detectors. arXiv preprint arXiv:2401.10691.

Zhao, S., Li, J., Wang, J., Zhang, Z., Zhu, L., & Zhang, Y. (2021). attack-gan: Adversarial attack against black-box ids using generative adversarial networks. *Procedia Computer Science*, *187*, 128–133. (2020 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI2020). https://doi.org/10.1016/j.procs.2021.04.118