Naif Arab University for Security Sciences
Journal of Information Security & Cybercrimes Research
مجلة بحوث أمن المعلومات والجرائم السيبرانية
https://journals.nauss.edu.sa/index.php/JISCR
JISCR

DDOS Botnets Attacks Detection in Anomaly Traffic : A Comparative Study

Ahmed Elsherif [*], **Arwa A. Aldaej**
Forensic Sciences Department, College of Criminal Justice, Naif Arab University for Security Sciences, Riyadh, Saudi Arabia.

Received 27 Aug. 2020; Accepted 21 Oct. 2020; Available Online 20 Nov. 2020

Abstract

One of the major challenges that faces the acceptance and growth rate of business and governmental sites is a Botnet-based DDoS attack. A flooding DDoS strikes a victim machine by means of sending a vast amount of malicious traffic, causing a significant drop in the service quality (QoS) in IoT devices. Nonetheless, it is not that easy to detect and tackle flooding DDoS attacks, owing to the significant number of attacking machines, the usage of source-address spoofing, and the common areas shared between legitimate and malicious traffic. New kinds of attacks are identified daily, and some remain undiscovered, accordingly, this paper aims to improve the traffic classification algorithm of network traffic, that hackers use to try to be ambiguous or misleading. A recorded simulated traffic was used for both samples; normal and DDoS attack traffic, approximately 104.000 cases of each, where both datasets -which were created for this study- represent the input data in order to create a classification model, to be used as a tool to mitigate the risk of being attacked.

The next step is putting datasets in a format suitable for classification. This process is done through preprocessing techniques, to convert categorical data into numerical data. A classification process is applied to capture datasets, to create a classification model, by using five classification algorithms which are; Decision Tree, Support Vector Machine, Naive Bayes, K-Neighbours and Random Forest. The core code used for classification is the python code, which is controlled by a user interface. The highest prediction, precision and accuracy are obtained using the Decision Tree and Random Forest classification algorithms, which also have the lowest processing time.

I. Introduction

The growing number of botnets attacks on the net has made it vital to develop tougher advanced techniques to deal with them, since human intervention is not enough to examine and provide the necessary response to such attacks. Moreover, the nature and techniques of recent online attacks have changed drastically, particularly after the appearance of intelligent agents such as computer variant DDoS attacks and worms [1]. That is why, the need for combating them intelligently is increasing. In chapter one we are going to investigate a general description of the problem, objectives and a brief of the proposed solution.

Online platforms today need to be checking whether the user is human or not to avoid brute force and flooding attacks, as these are the most common vulnerabilities, due to the availability of enhanced computerized power and network speed [2].

The problem in taking counter measures against attacks is that the HTTP DDoS attack acquires legitimacy

Keywords: *Cybercrimes, DDos, Botnet, Anomaly traffic, Machine learning*.

Production and hosting by NAUSS

* Corresponding Author: Ahmed Elsherif
Email: aelsherif@nauss.edu.sa.

doi: 10.26735/ZRXN1433

by representing legitimate behavior, and thus is ready to penetrate the available security and protection measures, to affect services provided and cause a negative impact on clients. The negative impact is to affect service availability and to stop providing the service in proper time. In case of attacker success penetrating through the security measures, the system is subject to other types of attacks or malware infection.

Because of the different environment of network devices and its architectures, the traditional attack detection systems cannot be competently used in the detection process. Additionally, the potential incidents or attacks might be different from the attacks that are observed on the conventional network devices.

The Internet of things (IoT) devices provide businesses with a number of advantages, so they can; monitor all business operations, progress a client's trial, save both time and money, increase employee efficiency, combine and modify models; improve the decisions of administrators and improve the movement of sales [3].

Recently, the IOT has become very sophisticated. It has also been inserted into many daily applications. It has become the direction of the future internet, providing many facilities to users, whether on a personal level, or for a range of manufacturing. Researchers have become interested in developing multiple technologies to apply it for all uses [4].

It is expected that the volume of IoT devices will increase from 8 billion in 2017 to 20 billion in 2020 [5]. However, a lot of IOT devices are essentially exposed to hacking. By analyzing the number of attacks on IOT devices, it was found that for 10 appliances connected to the internet, 250 vulnerabilities were exposed, including; open Telnet ports, old Linux firmware, unencrypted transmission of critical data [6].

Smart devices are very costly, however, their adoption and invasion aren't as high, and this can somehow be related to the presence of several suppliers and salesmen, although standardization is the key to decreasing the price of these devices and guaranteeing their interoperability [7]. Various cybercrimes can be committed by utilizing IoT devices in different fields, for example online transportation [8] and medical records [9].

The Distributed Denial of Service (DDoS) attack occurs when services stop being purposefully delivered by the action of an attacker, or attackers. This happens by foiling entry to all kinds of services such as internet, servers, appliances, applications, and the interaction of data through applications. The DDoS is composed of a source which transmits malicious data or demands. That source could result from several systems.

Mostly, these attacks occur by flooding a system with information requirements. This allows a network server to transmit a huge number of requests for a page that has been disabled as needed, or a database may be subject to a large number of queries. This results in causing the web bandwidth, the amplitude, the CPU and the main memory becoming saturated [10]. In spite of this, DDoS attacks provide simple attack methods in comparison with other cyber-attack methods, but they spread in more severe and developed ways.

In this paper, a comparative study of the machine learning techniques is studied to find the best algorithm that should be utilized in order to detect anomalous attacks in network traffic. According to the findings of this study, the proposed system should mitigate attacks and be applicable to all devices in Saudi Arabia.

This paper is organized as follows: the next section provides an overview of previous works and a discussion of author contributions to the main aspects of traffic classification in detecting DDoS attacks. Section three presents; the dataset, techniques and methodology used in attack detection. Section four shows the results obtained and indicates the best algorithm to mitigate attacks. Finally, the conclusions are introduced.

II. Related Work

This section details different studies that indicate the detection process of DDoS attacks using machine learning techniques, along with a summary of results and conclusions. The study outputs enlighten our research towards appropriate ML techniques used, tools, and data sets applied, as deduced by the paper's findings.

Authors in [11] suggested executing machine learning classifiers, to discover HTTP botnets. They utilized the functions of the TCP packet in order to extract the dataset from network traffic. They also worked on discovering the most effective machine learning classifier for eliciting best results. Their suggested experiment is based on classifying the HTTP botnet in the network flow, by utilizing the preferable classifier they found during the experiment, with a rate of accuracy that could reach 92.93%.

Botnet has adjusted itself towards various types of attacks and utilizes different forms of web protocols to carry out malicious actions. Take the model of peer to peer (p2p) botnet, which utilizes the P2P program to execute the command and control (C&C) server order. Nevertheless, P2P botnet has its disadvantages as to the intricacy in running bots on decentralized network architecture, consequently the HTTP botnet was introduced to overcome this issue. HTTPBotnet is working in the centralized network architecture, like the IRC botnet with the advantages of detection and avoiding. For example, DNS rapid flow and utilizing HTTP protocol leads to obstacles in the detection of the HTTP botnet in charge of carrying out DDoS attacks [11].

Authors in [12] suggested botnet discovery types that depend on the ML by utilizing DNS query data. This paradigm comes from the idea that the bots that are sent from the botnets, regularly transmit the search queries to DNS in order to discover the IP addresses and command and control servers (C&C,) by utilizing the name of the domains that have been automatically created.

This paradigm has been executed at two stages: a. the training stage. b. the detection stage. At the training stage, it gathers the query data of the DNS, thereafter, it takes out the domain names that are present at the DNS queries. Then, the group of group of domain names have extracted the characteristics to be used at the training stage. During the training stage, the ML algorithms are utilized to know the classifiers. Over the rating operations, they had the chance to be able to evaluate the highest ML algorithms that will implement the highest precision. Throughout the detection stage of the paradigm, the DNS queries are being monitored, and come out of the operation of excluding the domain names. The ML algorithm classifiers check the legitimacy of the domain name.

Sheriff Saad et. al. [13] suggest a novel method of description and disclosure of robots utilizing web traffic actions. It concentrates on the latest discovery and the most difficult robot types before they start Attack. There were several machine learning mechanisms to encounter the demands of detecting robots on the Internet, the ability of modification, disclosure of novelties and precocious disclosure. Results of empirical valuation, indicate from the dataset displayed, that there is a chance in an effective way to recognize Botnets through the botnet Command-and Control (C & C) Stage, prior to starting their attacks with simply traffic behaviour.

Authors in [14] examined a PCAP file and studied the DoS attack using the Decision Tree Data Mining Tool. They utilized a classifier sample at the WEKA intrusion disclosure tool. By decision tree algorithms, several bases were displayed to show if a SYN torrent exists or not. The decision tree ultimately showed that this is not the case. SYN packets from an origin identical to the same destination are larger than a packet considered to be a threat. Otherwise, it is normal. For example; Tcp.flags.syn <=0: Normal and Tcp.flags.syn >0: Threat.

III. Methodology

Different machine learning methodologies discussed botnet detection such as in [15]. In this research, the experiments are implemented by simulating every part of the desired environment, to avoid critical effects on other network components. The Test environment is composed of several virtual machines to represent the side of the botnets, and other virtual elements serve as victim and attack machines used by an attacker to control attacks by botnets. The code's implementation is designed to satisfy the required performance by various parts of the test environment, in which the attacker part of the code is used to recruit the botnets. The code installed on botnets is capable of generating a DDoS attack with different forms of attack. In order to be able to detect a DDoS attack and to ensure suitable actions to mitigate attacks we have implemented a code that can monitors analyzes network traffic, for the purpose of attack detection and mitigation.

A. Environment Preparation

The methodology followed in this study is based on utilizing a virtual environment running Linux (Ubuntu 18.2), to ensure a safe test environment and a HTTP response that is directed towards the botnet that targets a http attack.

The attack environment consists of 6 virtual Ubuntu working as botnets; the 6 botnets are used to attack the 7th virtual Ubuntu which acts as a victim. Fig. 1 shows a DDoS attack test environment and data-control flow.

Each one of the 6 botnets are supplied with a botnet message generator (DDoS Botnet Traffic Simulator) to simulate Http flood attack.

Bonesi [16] botnet traffic simulators can produce different protocol attacks (ICMP, UDP and TCP (HTTP)) used to simulate a flood attack. It can send different packet sizes, packet counts, packet rates and more parameters can be configured.

The victim is supplied with a Web server (Linux Apache MySQL PHP/Perl web server - LAMPP) used to produce web services and to host web pages, that will be used as a target for a http flood attack. This structure is used to ensure that the attack message penetrates through to layer 7 (application layer), rather than layer 3 or 4 (Network Layer Attack). In other words, the Http flood message uses the web service to permit attack messages to penetrate through to layer 7 of the victim communication model.

We also supply the victim with a network sniffer (network traffic capture/analysis tool), to capture traffic for the purpose of analysis (training and mitigation process).

B. Botnet Preparation Methodology

The proposed methodology focuses on building and controlling a Java application that performs all required operations needed to recruit botnets, in order to commit HTTP DDoS attacks on a victim machine.

Our Java application procedure can be summarized as follows:

1. Selecting IP range to scan for suitable devices to be recruited as botnets.
2. Scan each IP in the given range for corrupted usernames and passwords.
3. Store all pairs of IP addresses, usernames and passwords.
4. Store other IP addresses that cannot be discovered, for the purpose of login data to be used as a victim.
5. Monitor all discovered data.
6. Allow users to select as many recruited botnets as required in order to start attack.
7. Select the attack protocol (UDP – TCP – ICMP – HTTP … etc.).
8. Select the attack packet count.
9. Start attack.
10. Reboot selected botnets.

Fig. 2 shows botnet recruiting and attack process.

C. Attack Methodology

This methodology can be summarized in 2 steps, the first one centers on capturing simulated attack traffic in order to create the training model, the second one focuses on using this model to examine traffic, to alert users about real attack traffic. Fig. 3 and Fig. 4 show this process.

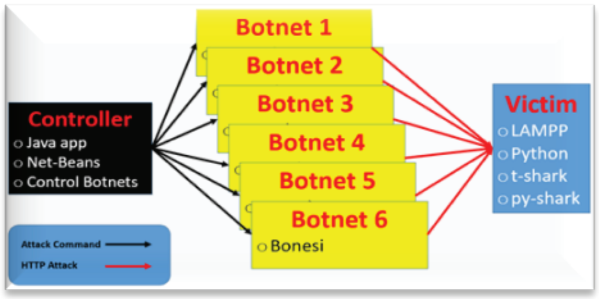


Fig. 1 DDoS attack environment.

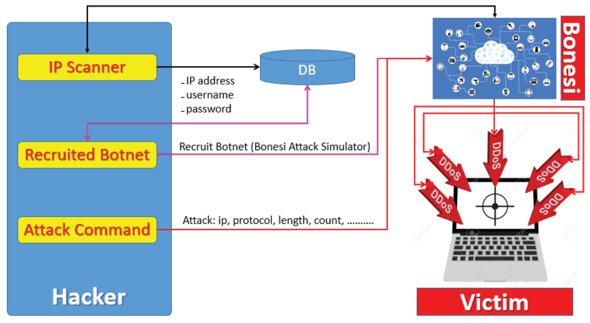


Fig. 2 Botnet recruiting and attack process.

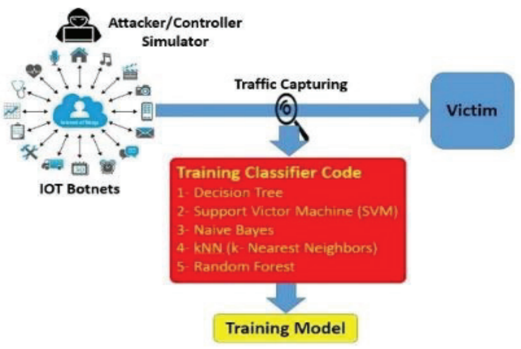


Fig. 3 Learning Process.

The following points illustrate our Java application procedure in detail:

1. Search for selected range of IP addresses (from xxx.xxx.xxx.1 to xxx.xxx.xxx.254) that represent the botnets that will be recruited as zombies to attack victims. One range can be searched at a time, but more than one IP range can be checked to collect as many botnets as possible to use for attack
2. For each responding IP address, the Java application will continue to check usernames and pass-

words by applying a list of standard usernames and password.

3. As soon as the Java application succeeds in logging in to one of the botnets, the botnet login information (IP address – username - password) will be stored into the SQLite database as a botnet for future use.
4. When discovering IPs that cannot be logged into, they will be added to the victim list.
5. Alongside collecting log in information for a sufficient number of botnets, Java code plays a second important role by recruiting botnets. This operation is summarized by downloading the BoNeSi (DDoS botnet traffic simulator) into the botnet. The next step is setting up BoNeSi on the botnet, ready to act as a zombie to be used to attack victims. This step is done by downloading a shell script file (file.sh) that contains a series of commands for adding the BoNeSi botnet simulator, as well as all necessary libraries, in order for BoNeSi to achieve the required performance.
6. The role of Java code is not limited to downloading and the setting up of BoNeSi, but rather it extends to running various configurations of BoNeSi attributes to simulate UDP, TCP, ICMP and other traffic protocols used for flood attacks.
7. Other factors can be controlled such as packet size, packet count, send rate and others.
8. Finally, the Java code is capable of restarting the botnet if required.
9. The 7th virtual run would be used to play the role of the victims in our environment. It's equipped with t-shark. T-Shark is the Linux version of wire-shark used for network traffic sniffing. The main purpose of T-shark is to capture the traffic exchange between botnets and victims, then export captured messages into CSV files. These files contain clean traffic and DDoS attack traffic information.
10. The sniffing process is completed on the host machine by supplying one line.
11. Written commands in the CLI (Ubuntu terminal Command Line Interpreter) are used to capture data passing into or from the host machine. A capturing filter is used to determine the exact required information and store extracted data onto a csv file for the purpose of data training. The sniffing filter

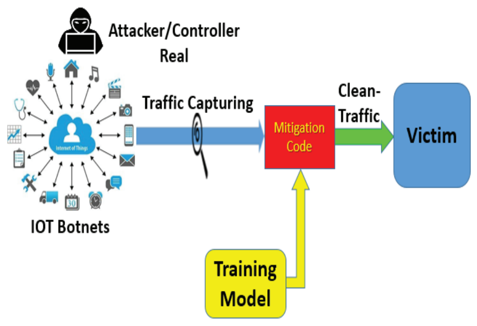
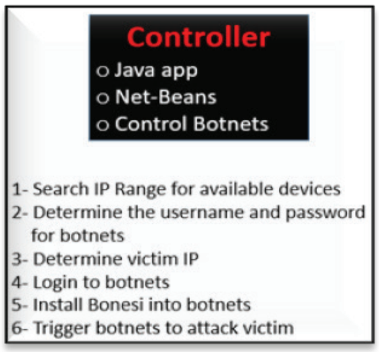


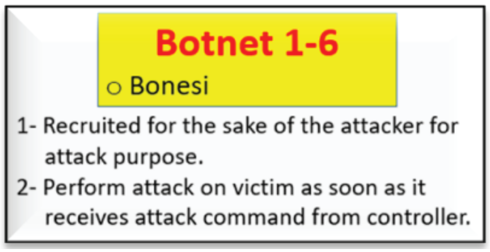
Fig. 4 Mitigation Process.



Controller
- Java app
- Net-Beans
- Control Botnets

1- Search IP Range for available devices
2- Determine the username and password for botnets
3- Determine victim IP
4- Login to botnets
5- Install Bonesi into botnets
6- Trigger botnets to attack victim

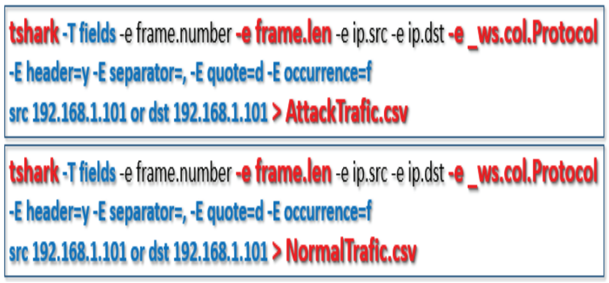
Fig. 5 Controller Tasks.



Botnet 1-6
- Bonesi

1- Recruited for the sake of the attacker for attack purpose.
2- Perform attack on victim as soon as it receives attack command from controller.

Fig. 6 Botnets Tasks.



```
tshark -T fields -e frame.number -e frame.len -e ip.src -e ip.dst -e _ws.col.Protocol
-E header=y -E separator=, -E quote=d -E occurrence=f
src 192.168.1.101 or dst 192.168.1.101 > AttackTrafic.csv
```

```
tshark -T fields -e frame.number -e frame.len -e ip.src -e ip.dst -e _ws.col.Protocol
-E header=y -E separator=, -E quote=d -E occurrence=f
src 192.168.1.101 or dst 192.168.1.101 > NormalTrafic.csv
```

Fig. 7 T-Shark captures and filter commands for normal and attack traffic.

extract requires features using many data collections. Other properties used to format the sniffed data allow the T-Shark to extract target features and store it for future use. Fig. 5 describes attack control tasks and Fig. 6 describes the tasks that are provided by botnet software. Fig. 7 show network traffic capture command used to monitor network traffic.

12. Once the data is captured, we carry out data pre-processing (remove unneeded data, fill missed data and feature selection).
13. We prepared a Python code to process captured data (clean DDoS and normal traffic) for the purpose of machine learning.
14. We used different algorithms (Nearest Neighbourhood, Random Forest [17] and other classification algorithms) to provide training for captured data (normal and attack traffic). A different classifier is used to secure the best prediction for unknown traffic. The result of this step is a Pickles file. Each Pickles file contains all required information for classification (prediction of traffic/packet type) of unknown traffic for the purpose of DDoS mitigation.
15. Besides using Pickles file for classification, we used other Pickles for data preparation, such as replacing empty spaces with 0 values and converting categorical data to feature numerical data.
16. The Python code is used to detect a HTTP Flood attack; the attack status is declared to inform the user about the presence of an attack.
17. We selected a protocol name (categorical) and a frame size for the purpose of traffic training to produce a training model. The reason behind selecting a protocol name and a frame size, is that this amount of information is sufficient for detecting the presence of a DDoS attack, whilst at the same time, the attack data pattern is characterized by its repetitive and periodic pattern. These two features are repeated in a special pattern that helps in attack detection. Another reason for only selecting these two features is that it is preferable to complete the job using as minimal resources as possible, to avoid wasting system resources.

D. DDOS Analysis

The basic idea of DDoS analysis is to use supervised classification algorithms to create a model to identify attack traffic based on the data concluded from the training process of well-known attack traffic (simulated), with respect to normal traffic, using different algorithms.

1. Feature used:
   a. Frame size - numerical.
   b. Protocol – categorical.
   c. Traffic type – numerical.
2. Packet count 100.000
3. Traffic types count: 2
   a. Normal = 0.
   b. DDoS Attack = 1.

We selected a protocol name (categorical) and a frame size for the purpose of traffic training to produce a training model. The reason behind selecting the protocol name and frame size, is that this amount of information is sufficient for detecting the presence of a DDoS attack, whilst at the same time, the attack data pattern is characterized by its repetitive and periodic pattern. These two features are repeated in a special pattern that helps in attack detection. Another reason for only selecting these two features is that it is preferable to complete the job using as minimal resources as possible, to avoid wasting system resources.

Data Sample: Fig. 8 shows captured data features of concern, for analysis.

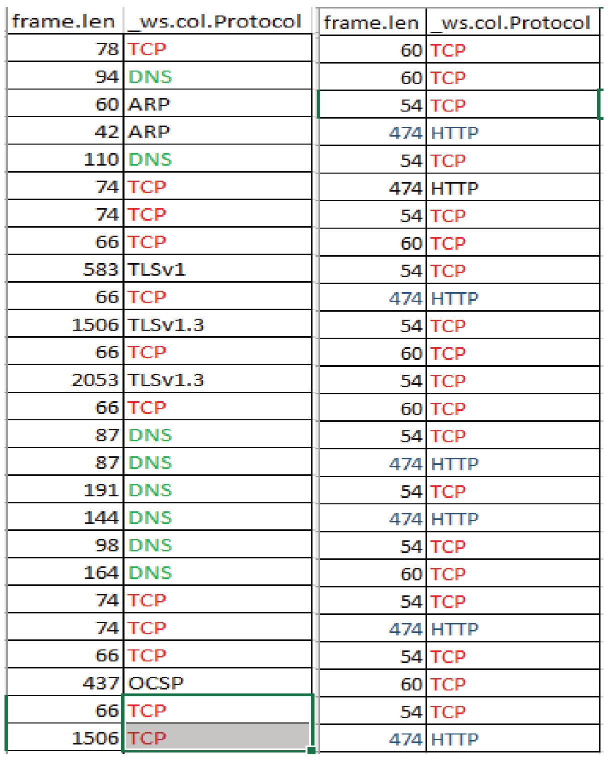
E. Selected classifiers

We shall apply more than one classifier to predict the traffic type. We select five classifiers to study the traffic classification, to discover the best classifier, that has the highest performance, when detecting traffic type.

1. Decision Tree.
2. Support Vector Machine.
3. Naive Bayes.
4. K-Neighbours Classifier K=3.
5. Random Forest Classifier.

1) Measuring classification output:

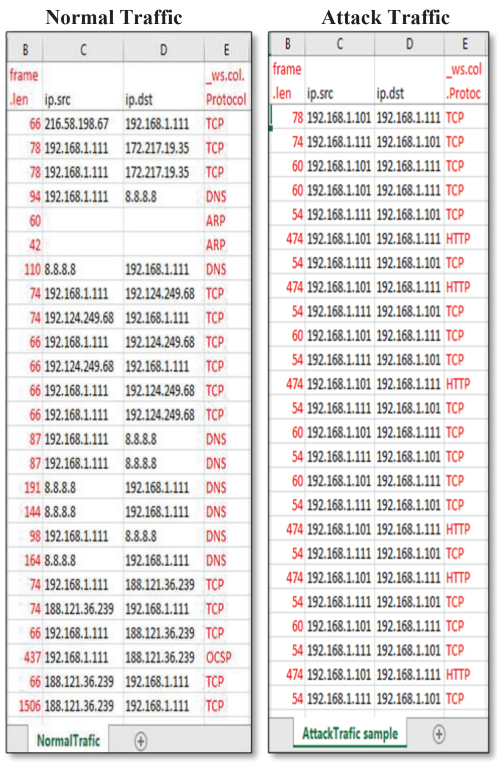
The purpose of classification is to predict the type of unknown traffic, using a model created during the training process. The training process results are summarized in order to produce a confusion matrix that calculates ac-



frame.len	_ws.col.Protocol
78	TCP
94	DNS
60	ARP
42	ARP
110	DNS
74	TCP
74	TCP
66	TCP
583	TLSv1
66	TCP
1506	TLSv1.3
66	TCP
2053	TLSv1.3
66	TCP
87	DNS
87	DNS
191	DNS
144	DNS
98	DNS
164	DNS
74	TCP
74	TCP
66	TCP
437	OCSP
66	TCP
1506	TCP

frame.len	_ws.col.Protocol
60	TCP
60	TCP
54	TCP
474	HTTP
54	TCP
474	HTTP
54	TCP
60	TCP
54	TCP
474	HTTP
54	TCP
60	TCP
54	TCP
60	TCP
54	TCP
474	HTTP
54	TCP
474	HTTP
54	TCP
60	TCP
54	TCP
474	HTTP
54	TCP
60	TCP
54	TCP
474	HTTP

Fig. 8 Captured Data Sample.



Normal Traffic

B	C	D	E
frame .len	ip.src	ip.dst	_ws.col. Protocol
66	216.58.198.67	192.168.1.111	TCP
78	192.168.1.111	172.217.19.35	TCP
78	192.168.1.111	172.217.19.35	TCP
94	192.168.1.111	8.8.8.8	DNS
60			ARP
42			ARP
110	8.8.8.8	192.168.1.111	DNS
74	192.168.1.111	192.124.249.68	TCP
74	192.124.249.68	192.168.1.111	TCP
66	192.168.1.111	192.124.249.68	TCP
66	192.124.249.68	192.168.1.111	TCP
66	192.168.1.111	192.124.249.68	TCP
66	192.168.1.111	192.124.249.68	TCP
87	192.168.1.111	8.8.8.8	DNS
87	192.168.1.111	8.8.8.8	DNS
191	8.8.8.8	192.168.1.111	DNS
144	8.8.8.8	192.168.1.111	DNS
98	192.168.1.111	8.8.8.8	DNS
164	8.8.8.8	192.168.1.111	DNS
74	192.168.1.111	188.121.36.239	TCP
74	188.121.36.239	192.168.1.111	TCP
66	192.168.1.111	188.121.36.239	TCP
437	192.168.1.111	188.121.36.239	OCSP
66	188.121.36.239	192.168.1.111	TCP
1506	188.121.36.239	192.168.1.111	TCP

NormalTrafic

Attack Traffic

B	C	D	E
frame .len	ip.src	ip.dst	_ws.col .Protoc
78	192.168.1.101	192.168.1.111	TCP
74	192.168.1.111	192.168.1.101	TCP
60	192.168.1.101	192.168.1.111	TCP
60	192.168.1.101	192.168.1.111	TCP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP
60	192.168.1.101	192.168.1.111	TCP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP
60	192.168.1.101	192.168.1.111	TCP
54	192.168.1.111	192.168.1.101	TCP
60	192.168.1.101	192.168.1.111	TCP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP
60	192.168.1.101	192.168.1.111	TCP
54	192.168.1.111	192.168.1.101	TCP
474	192.168.1.101	192.168.1.111	HTTP
54	192.168.1.111	192.168.1.101	TCP

AttackTrafic sample

Fig. 9 Normal and attack Captured data at victim position.

curacy, and other parameters, that ensure the accuracy of the training process. For unknown traffic, only predictions can be obtained, as a result of the classification process. If the prediction value is positive, the python code alerts the user about a detected attack.

2) Results Calculation:

The Accuracy Score would be evaluated to find the best classifier. "Time consumed" represents the classification process performance measure.

To calculate the Accuracy Score we have to obtain the confusion matrix. The Confusion matrix represents the count of true/false predictions for each class. We can represent the confusion matrix as follows:

1. Count of true detected packets that are DDoS attack packet (T/T).
2. Count of true detected packets that are not DDoS attack packet (T/F).
3. Count of False detected packets that are DDoS attack packet (F/T).
4. Count of False detected packets that are not DDoS attack packet (F/F).

The Accuracy Score can be calculated from the following Equation:

$$\text{Accuracy Score} = \frac{\text{Number of Elements Correctly Classified}}{\text{Total Count of Elements}}$$

$$= \frac{(T/T)+(F/F)}{\text{Total Count of Elements}} \quad (1)$$

IV. Results and Discussions

The results are achieved by running the implemented code during the various steps, which includes; creating the test environment, detecting the botnet IP, identifying usernames and passwords, storing botnet information on the database, recruiting the botnet (deploying Bonesi in the botnet), detecting the victim IP and sending attack commands to the recruited botnets. The attack packets are captured at the victim's network card. The attack packets are captured for the purpose of attack dataset training to produce the classification model. This model is used to classify runtime traffic to judge traffic for the DDoS attack. Collecting test results is necessary in order to evaluate how much we've succeeded in our research, and to provide evidence for test result discussions, therefore enriching the way of thinking in regard to detecting and mitigating DDoS attacks.

Results archived in our research are collected in 3 steps, such as:

1. Data collection.
2. Production of attack traffic model.
3. Runtime traffic analysis.

Each step produces the input data for the next step.

A. Data Collection Results:

Collected traffic can be described as shown in Fig. 9, as follows:

- Frame Length: The TCP length field is the length of the TCP header and data (measured in octets).
- IP Source: The IP (v4) address of the sender of the packet.
- IP Destination: The IP (v4) address of the receiver of the packet.
- Protocoled: the protocol used in the data portion of the packet.

The result of this step are CSV files named (NormalTrafic.csv and AttackTrafic.csv). These two files are saved in a dataset folder, resulting in a sample of the 2 files.

In light of the above, we notice that:

- Normal traffic has a sequence of packets which contain different protocols such as TCP, ARP, DNS, HTTP and more. This sequence is repeated on a regular basis.
- Attack traffic is composed of a long sequence of pure TCP packets followed by a sequence of TCP and HTTP for a short period, then it goes back to a long sequence of TCP packets. Another feature is that the HTTP packet has a constant length.

B. Dataset Training and Cross Validation Results

The training process is conducted by applying different classification algorithms to generate a model to be used in calculating predictions of unseen data.

1. Decision Tree: The Decision Tree Classifier presents the highest prediction precision, accuracy and performance. As shown in Table I.
2. Support Vector Machine: The Support Vector Machine Classifier presents a good Accuracy Score, but with much lower performance. As shown in Table II.
3. Naive Bayes: The Naive Bayes Classifier presents a lower Accuracy Score, as shown in Table III.
4. K-Neighbours Classifier K=3: The K-Neighbours Classifier presents a good Accuracy Score, but with lower performance, as shown in Table IV.
5. Random Forest Classifier: The Random Forest classifier provides good performance and prediction precision, as shown in Table V.

Fig. 10 shows the output test evaluation parameters results. Unlike other methodology used to detect DDoS attacks, our methodology doesn't depend on the application under attack or attacking an IP, it detects any DDoS attack that can penetrate and reach layer 7 in the OSI network communication model. In order to have the best classifier, 5 classifiers should be tested to select the best classifier for prediction and cross validation processing time.

Rudy et. al. [10], suggested executing machine learning classifiers, to discover HTTP botnets, in order to predict HTTP DDoS attacks within different bot families. The values were as follows:

- Accuracy: (51.84%-97.88%).
- Precision: (71.02%-98.66%).
- Recall: (43.58%-99.99%).
- FPR: (3.06%-97.12%).

Xuan el. at. [11] used 4 machine learning algorithms on the T1, T2, and T3 training datasets, consequently the most similar training dataset to ours. We can summarize the results as follows:

- Accuracy: (90.2%-90.8%).
- PVR: (83.1%-90.7%).
- FPR: (86.5%-90.8%).
- TPR: (90.2%-91.2%).
- F1: (86.5%-90.8%).

We used A sufficient amount of traffic packets (101400 packets for normal traffic and 101400 packets for attack traffic).

We've selected five classifiers in order to produce a suitable model, using well-known traffic, and used it to predict the traffic type. We've succeeded in obtaining the following results:

- Accuracy (81.378%-97.61%).
- F1 score (80.68%-97.61%).
- Recall (81.37%-97.61%).

TABLE I
DECISION TREE CLASSIFIER RESULT SUMMERY

Accuracy	0.9760796751341257
F1 score	0.9761284983757427
Recall	0.9760796751341257
Precision	0.9762409084974778
Confusion Matrix	[[32431, 937], [1484, 66359]]
Time Consumed	0:0:2

TABLE II
SUPPORT VECTOR MACHINE CLASSIFIER RESULT

Accuracy	0.9744785129400514
F1 score	0.9744778762903109
Recall	0.9744785129400514
Precision	0.9744964320164035
Confusion Matrix	[[32342, 959], [749, 32874]]
Time Consumed	0:2:41

TABLE III
NAIVE BAYES CLASSIFIER RESULT SUMMERY

Accuracy	0.8748653802452303
F1 score	0.8655517847719487
Recall	0.8748653802452303
Precision	0.8945507526205738
Confusion Matrix	[[20703, 12665], [0, 67843]]
Time Consumed	0:0:2

TABLE IV
K-NEIGHBOURS CLASSIFIER RESULT SUMMERY

Accuracy	0.9744187436495129
F1 score	0.974418088463661
Recall	0.9744187436495129
Precision	0.9744373625841419
Confusion Matrix	[[32338, 963], [749, 32874]]
Time Consumed	0:1:18

TABLE V
RANDOM FOREST CLASSIFIER RESULT SUMMERY

Accuracy	0.974493455262686
F1 score	0.9744928232060458
Recall	0.974493455262686
Precision	0.9745112014551831
Confusion Matrix	[[32343, 958], [749, 32874]]
Time Consumed	0:0:2

Note: Processing time depends on hardware limitations.

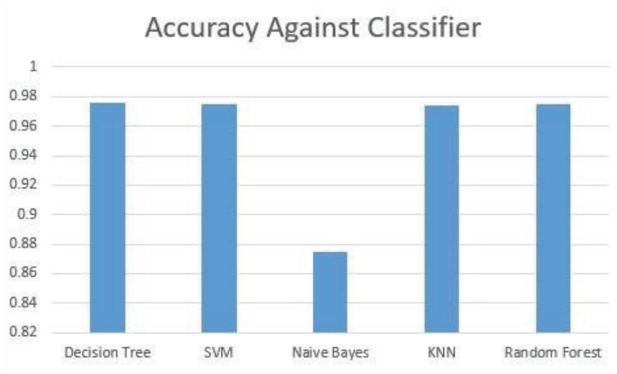


Fig. 10 Accuracy against classifier result.

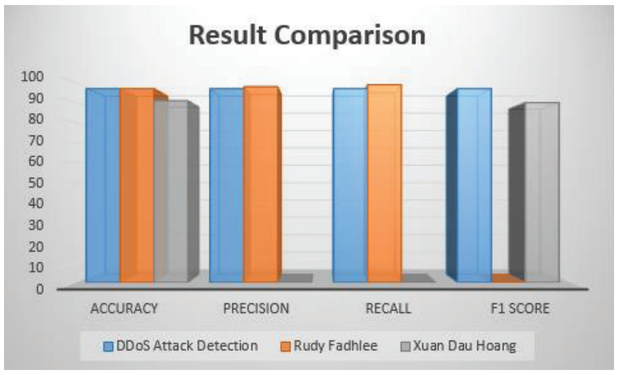


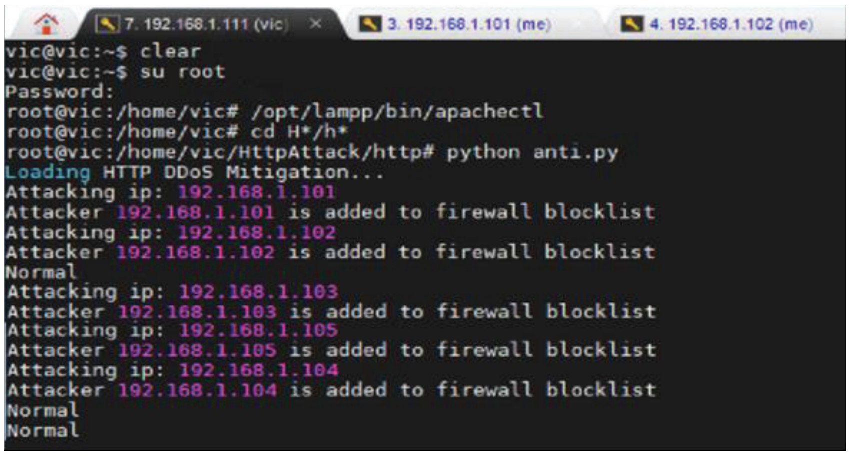
Fig. 11 Result comparison for our results and other efforts to enhancement to detect DDoS attack.

- Precision (86.41%-97.62%).

Fig. 11 shows a comparison of our results, and [10] and [8] efforts to enhancement the detection of a DDoS attack.

C. Mitigation Results

The mitigation results are summarized in blocking HTTP DDoS attacks on IPs by adding the botnet IP fire-



```
vic@vic:~$ clear
vic@vic:~$ su root
Password:
root@vic:/home/vic# /opt/lampp/bin/apachectl
root@vic:/home/vic# cd H*/h*
root@vic:/home/vic/HttpAttack/http# python anti.py
Loading HTTP DDoS Mitigation...
Attacking ip: 192.168.1.101
Attacker 192.168.1.101 is added to firewall blocklist
Attacking ip: 192.168.1.102
Attacker 192.168.1.102 is added to firewall blocklist
Normal
Attacking ip: 192.168.1.103
Attacker 192.168.1.103 is added to firewall blocklist
Attacking ip: 192.168.1.105
Attacker 192.168.1.105 is added to firewall blocklist
Attacking ip: 192.168.1.104
Attacker 192.168.1.104 is added to firewall blocklist
Normal
Normal
```

Fig. 12 py_anti.java prediction output and firewall action.

wall. Fig. 12 shows prediction and mitigation module outputs.

V. Conclusion

The types of DDoS attacks, and the difficulties in detecting them, require us to carry out further research, in order to obtain an algorithm that is capable of detecting attacks, for the purpose of attack mitigation. We've worked on designing codes that implement classification algorithms for analyzing network traffic. The process depends on analyzing well-known attack traffic, then using this experience when detecting real attacks.

The classification results show that the Decision Tree algorithm has the highest accuracy and precision, while Random Forest and SVM come second, with slightly lower accuracy and precision. The decision tree and Naive Bayes algorithms have the lowest classification time, K-nearest neighbour and SVM have a much higher classification time.

As for machine learning models, SVM, among others, achieved the highest score, in terms of accuracy. This score is 97.37%, while all other algorithms achieved acceptable false negative scores.

References

[1] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," *IECON 2011 - 37th Annu. Conf. IEEE Ind. Electron. Soc.,* Melbourne, VIC, 2011, pp. 4490-4494, doi: 10.1109/IECON.2011.6120048.

[2] N. Kheshaifaty and A. Gutub, "Preventing Multiple Accessing Attacks via Efficient Integration of Captcha Crypto Hash Functions," in *Int. J. Comput. Sci. Netw. Secur. (IJCSNS),* vol. 20, no. 9, pp. 16-28, Sept. 2020, doi: 10.22937/IJCSNS.2020.20.09.3

[3] M. Rouse. "nternet of things (IoT)." internetofthingsagenda.techtarget.com. https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[4] P. Gokhale, O. Bhat and S. Bhat, "Introduction to IOT," in *Int. Adv. Res. J. Sci. Eng. Technol.,* vol. 5, no. 1, pp. 41-44, Jan. 2018, doi: 10.17148/IARJSET.2018.517

[5] J. Manyika et al. "Unlocking the potential of Internet of Things." Mckinsey.com. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world#

[6] Broadband Internet Technical Advisory Group, "Internet of Things Security and Privacy Recommendation," BITAG, Nov. 2016. [Online]. Available: https://www.bitag.org/documents/BITAG_Report_-_Internet_of_Things_(IoT)_Security_and_Privacy_Recommendations.pdf

[7] N. Farooqi, A. Gutub and M. O. Khozium, "Smart Community Challenges: Enabling IoT/M2M Technology Case Study," in *Life Sci. J.,* vol. 16, no. 7, pp. 11-17, July 25, 2019, doi: 10.7537/marslsj160719.03.

[8] A. Alsaidi, A. Gutub and T. Alkodaidi, "Cybercrime on Transportation Airline," in *J. Forensic Res.,* vol. 10, no. 4, Nov. 19, 2019.

[9] H. Samkari and A. Gutub, "Protecting Medical Records against Cybercrimes within Hajj Period by 3-layer Security," in *Recent Trends Inf. Technol. Appl.,* vol. 2, no. 3, 2019.

[10] G. V. Hulme. "DDoS explained: How distributed denial of service attacks are evolving." Csoonline.com. https://www.csoonline.com/article/3222095/ddos-explained-how-denial-of-service-attacks-are-evolving.html

[11] R. Fadhlee, M. Dollah, M. A. Faizal, M. Z. Mas'ud and L. K. Xin, "Machine Learning for HTTP Botnet Detection Using Classifier Algorithms," in *J. Telecommun. Electron. Comput. Eng.,* vol. 10, no. 1-7, Jan./Mar. 2018.

[12] X. D. Hoang and Q. C. Nguyen, "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data," in *Future Internet,* vol. 10, no. 5, May 18, 2018. doi: 10.3390/fi10050043

[13] S. Saad et al., "Detecting P2P botnets through network behavior analysis and machine learning," *2011 Ninth Annu. Int. Conf. Priv. Secur. Trust,* Montreal, QC, 2011, pp. 174-180, doi: 10.1109/PST.2011.5971980.

[14] N. Sharma, A. Mahajan and V. Mansotra, "Identification and analysis of DoS attack Using Data Analysis tools," in *Int. J. Innov. Res. Comput. Commun. Eng.,* vol. 4, no. 6, June, 2016, doi: 10.15680/IJIRCCE.2016. 0406208.

[15] M. S. Gadelrab, M. Elsheikh, M. A. Ghoneim and M. Rashwan, "BotCap: Machine Learning Approach for Botnet Detection Based on Statistical Features," in *Int. J. Commun. Netw. Inf. Secur. (IJCNIS),* vol. 10, no. 3, pp. 563-579, Dec. 2018.

[16] M. Goldstein. "BoNeSi – the DDoS Botnet Simulator." Github.com. https://github.com/Markus-Go/bonesi

[17] N. Sirikulviriya and S. Sinthuponyo, "Integration of rules from a random forest," *Int. Conf. Inf. Electron. Eng.,* Bangkok, Thailand, May 28-29, 2011.