5-2022

Detecting The Intensity of Denial-of-Service Cyber Attacks using Supervised Machine Learning

Abigail Hubbard

Detecting The Intensity of Denial-of-Service Cyber Attacks using Supervised Machine Learning

by

Abigail D. Hubbard

An Undergraduate Thesis Submitted in Partial Fulfillment

of the Requirements for the

University Honors College

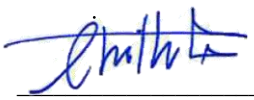
and the

Honors-in Computing Program

College of Business and Technology

East Tennessee State University

Abigail D. Hubbard



Dr. Ghaith Husari, Thesis Advisor

Mr. Matthew Harrison, Reader

Abstract

Denial-of-Service (DoS) attacks are aimed at shutting a machine or network down to block users from accessing it. These attacks can be difficult to detect and can cost millions in damages or lost earnings. Since the first DoS attack occurred in 1999, the way DoS attacks have been launched has become more complicated, making them more elusive and harder to detect. The first step to detect and mitigate a DoS attack is for a system to identify the malicious traffic.

In this experiment, we aim to identify the malicious traffic within ten seconds. To do this the project was divided into 3 phases: data collection, feature extraction and construction of classification. The first phase was to collect malicious and legitimate data using Wireshark. The second phase of the project was to convert the PCAP files into features that are meaningful and easy to read. The third phase of the project is the construction of classification models. We used the Naïve Bayes and decision tree classification models to identify malicious traffic data and differentiate it from legitimate traffic data. This approach yielded an F_1 score average of 92% in detecting DoS attacks and an $F_1\ score$ accuracy range of 37% to 71% to accurately determine the intensity of the DoS attack, a reasonable accuracy for this problem. These results show that it is possible to not only detect DoS attacks, but also, to determine the intensity of such attacks with a reasonable accuracy.

Table of Contents

1 Introduction

Denial-of-Service (DoS) attacks can be detected in many ways. According to [7], one detection technique uses the source IP address; this method is based on a study that showed most IP addresses that are used in attacks are new. In addition to this there are various other network/transport level defense methods including source-based, destination-based, network-based, and hybrid defenses.

This study examines a novel approach to detecting DoS attacks using machine learning to differentiate between malicious and legitimate data traffic collected from Wireshark. Although the main goal of the study is to differentiate between malicious and legitimate traffic, the goal can be further classified as differentiating between various malicious attack intensities and legitimate scenarios. This distinction was added to eventually help aid in determining what reaction should be taken when there is an attack. Attacks with low intensities could mean there is no real threat or damage to the victim while higher intensities can indicate a genuine threat with possible damages. The study is broken down into five phases: collection of malicious traffic, collection of legitimate traffic, feature extraction, classification, and evaluation of model accuracy.

The rest of this report is divided into six chapters. Chapter 2 provides background information on DoS attacks, traffic flow and capturing, machine learning, decision tree classification, naïve bayes classification, and virtual machines. Chapter 3 includes summaries of papers that are relevant to this study. Chapter 4 describes the proposed approach of the experiment including the first four phases. Chapter 5 reports the results of the experiment and discusses the results. Chapter 6 concludes and suggests directions for future work.

.

Commented [HM1]: Help the reader out separately from the abstract—spell acronyms back out for the first time in each section.

Commented [HM2]: Four chapters for the "rest" of the report in your list right now? What about the conclusion listed as Chapter 6 in the ToC?

2 Background

2.1 – Denial-of-Service

Denial-of-Service (DoS) attacks are attacks launched with the aim of shutting down a machine or network. The end goal of a DoS attack is to prevent users from being able to access the network or machine being targeted. Zargar et al. [1] identifies two ways to launch DoS attacks. In the first method, an attacker sends malicious packets to the victim that confuse a protocol or application on the victim's machine. This method exploits known vulnerabilities on a machine. The other more common method is when an attacker disrupts a victim's connection by overwhelming a machine's resources, bandwidth, router processing capacity or network resources [1].

Commented [HM3]: Two different spellings so far—going with this one in the earlier instances.

Commented [HM4]: Is this all considered general knowledge now or do you need a citation?

2.2 – Traffic Flow and Capturing

At any given time, a computer is generating an abundance of network traffic. This traffic consists of data that is broken down into small packets of various types. Thus, traffic flow can be defined as the packets that are moving across a network. There are three different types of traffic on a network: voice traffic, video traffic, and data traffic. For this study, data traffic was the main focus. We focus on data traffic during this study, as the malicious traffic and legitimate traffic generate data traffic and not video or voice traffic. To capture this data traffic, Wireshark was used. Wireshark is an application that captures packets from a network with the purpose of analyzing them [2]. Wireshark captures various data including the following: IP addresses, protocol, flags, and length.

Commented [HM5]: Sentence fragment.

Commented [HM6]: Is this some specific network? Probably best to generalize to "a network."

Commented [HM7]: Why do we care? Briefly remind the reader.

Commented [HM8]: Cite some Wireshark docs here.

Commented [HM9]: Narrow this down to the list of data you care about (and lose the etc.)

2.3 – Machine Learning

Machine learning is a field of study within computer science and artificial intelligence that allows computer to learn and accurately predict outcomes by using data and algorithms. A

user will separate their data into training and testing material. The training data is used by the algorithm to determine what features of the data are important. After this, the computer writes it's own program to return an outcome given features from the testing data. [3].

2.4 – Decision Tree

Decision trees are a type of supervised machine learning classification model that is used to make predictions based on previous data or training data. The decision tree starts with the root node or the base of the decision tree. A decision is made, for example something such as yes or no, that leads to splitting the branch into more nodes. After the decision is made, the branch is split into additional sub nodes called decision nodes that depict additional decision. After making a decision, a leaf node that represents the possible outcomes of the decision tree will be left [4].

Commented [HM10]: Split this into a couple of sentences (comma splicing).

Commented [HM11]: Wordy—no need for "eventually"

2.5 – Naïve Bayes

Naïve Bayes classifiers differ from decision trees as they are a collection of algorithms based on Bayes' Theorem rather than a single algorithm. Having a family of algorithms allows you to utilize different classifiers that are all based on Bayes Theorem. Bayes Theorem finds the probability of an event based on the occurrence of a previous event and its outcome. The Naïve Bayes classifying algorithms assume that all features act independently of each other and that all make an equal contribution to the possible outcomes [5] [6].

Commented [HM12]: Why does this matter in the context of your work? Add the context as to why a collection of algorithms helps or delete.

2.6 – Virtual Machines

A Virtual Machine (VM) is a software that mimics a computer through a virtual environment. It acts just as a physical computer would be able to perform the same operations such as running programs and opening applications. It also has its own separate virtual CPU, memory, network interface, disks to store files, and more. A VM was used to create a sandbox environment. This sandboxed environment allows for a malicious attack to run without affecting

Commented [HM13]: Only highlight the features you need in a VM (e.g., sandboxed filesystem / runtime).

Commented [HM14]: That description would make a VM an actual machine…

the network or local applications. So, by using this sandbox the malicious data was collected ethically, as no real physical system was affected.

Commented [HM15]: Reword to describe how the malicious programs would affect a real machine, i.e., WHY did you use a VM in the first place?

2.7 – Accuracy measures of DoS Detection

Commented [HM16]: I would not define these accuracy measures here, but instead in the background and mention them as part of your experimental design. Focus on your results in this section.

There are various accuracy measures shown in the figures above, but the ones of importance that will be discussed are TP, TN, FP, FN, accuracy, precision, recall, and F_1 score.

2.7.1 – True Positive (TP)

True positive is the outcome that results from the classification model correctly predicting the positive label.

2.7.2 – True Negative (TN)

True negative is the outcome that results from the classification model correctly predicting the negative label.

2.7.3 – False Positive (FP)

False positive is the outcome that results from the classification model incorrectly predicting the positive label.

2.7.4 – False Negative (FN)

False negative is the outcome that results from the classification model incorrectly predicting the negative label.

2.7.5 – Accuracy

Accuracy indicates the overall percentage of the classifier predicting the correct label. Equation 1 shows how to calculate the accuracy.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Commented [HM17]: Label to the side and reference as Equation 1.

2.7.6 – Precision

Precision indicates the ratio the classifier is correct out of all the times it predicted a positive label. Equation 2 shows how to calculate the precision.

$$P = \frac{TP}{TP+FP} \quad (2)$$

2.7.7 – Recall

Recall, also called the true positive rate, is the ratio the classifier correctly chose a positive prediction in relation to the total possible positives. Equation 3 shows how to calculate the recall.

$$R = \frac{TP}{TP+FN} \quad (3)$$

2.7.8 – F_1 score

The F_1 score takes the precision and recall and combines them into a single metric by taking the harmonic mean. The F_1 score is sometimes thought to be a better measure than accuracy, especially when it comes to imbalanced data [7]. Equation 4 shows how to calculate the F_1 score.

$$F_1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

3 Related Works

This section discusses the different methods that have been used in previous research to detect and defend against DoS/DDoS attacks. Distributed Denial-of-Service attacks have the same goal as DoS attacks except for they are executed using multiple machines instead of one. These methods can be broken down into three categories: defense mechanisms, machine learning, and other methods. The first section focuses on the different defense mechanisms or created in previous research. The next section is focused on the different machine learning

Commented [HM18]: Equation 2.

Commented [HM19]: Equation 3.

Commented [HM20]: Is this common knowledge?

Commented [HM21]: Equation 4.

Commented [HM22]: I don't see any definition of DDoS anywhere (since you're defining DoS—define DDoS too).

Commented [HM23]: Avoid the first person here: "This section discusses the different methods used in previous research to detect and defend against DoS/DDoS attacks."?

approaches to detect DoS/DDoS attacks. Lastly, the third section focuses on various non-machine learning detection methods.

3.1 – Defense Mechanisms

Zargar et al. [1] suggests that, typically, by the time a DoS/DDoS attack is detected, the only thing that can be done is to disconnect the victim machine to manually fix the problem. Therefore, the goal of defense mechanisms is to detect an attack as quickly as possible while also stopping the attack as close to the source as possible. Defense mechanisms can be classified into two categories: location-based and time-based mechanisms. Location-based defense mechanisms focus on the location in a network. There are two locations of attack within a network, classified into network/transport level attacks and application-level attacks. Location-based defense mechanisms include source-based, destination-based, network-based and hybrid. The second category, time-based defense mechanisms, can be categorized based on whether the defense mechanism attempts to prevent, detect, or respond to attacks.

Commented [HM24]: Are you classifying both defenses and attacks in this sentence? I *think* I see what you're getting at, but it's confusing to me (as a reader). Probably needs rewording and splitting into a couple of sentences. You have multiple parameters here—each could be its own sentence.

Or, you define the types of defenses at the start of 3.1.1—could delete from here.

3.1.1 – Network/transport level defenses

Network/transport level defenses include source-based, destination-based, network-based and hybrid (i.e. distributed) defenses.

Source-based

Commented [HM25]: Feels like you're missing some citations here for these specific technologies.

Zargar et al. [1] states that source-based defenses are deployed near the source of the attack, including location such as at the edge of the attacker's source network or at the access router of an Autonomous System. The goal of source-based defenses is to prevent the start of a DoS/DDoS attack. Router-based defenses included Ingress/Egress filtering, D-WARD, MULTOPS, TOPS, and MANAnet's reverse firewall. Ingress/egress filtering can help prevent the practice of altering IP source addresses, but this method fails when the attack is using botnets

since those devices are legitimate and have legitimate IP addresses. DWARD monitors the current inbound and outbound traffic and attempts to detect an attack by comparing it to normal traffic patterns. Multi-Level Tree for Online Packet Statistics (MULTOPS) is a data structure used by routers to monitor packets. MULTOPS uses a dynamic tree structure to monitor the packets for each IP address, but this makes this defense method vulnerable to memory exhaustion attacks. TOPS was created to address this vulnerability and instead uses a hashing scheme to overcome the problem. Another problem with TOPS and MULTOPS though is that they are both based on the assumption that the rate of traffic in one direction is proportional to the opposite direction rate, but this is not always true. The last source-based defense mechanism is MANAnet's reverse firewall. While a traditional firewall protects from incoming traffic, MANAnet's reverse firewall "protects the outside from packet flooding attacks that originate from within a network" [1]

Destination-based

Zargar et al. [1] states that destination based defenses are deployed at either the router on the edge of the victim's local network or the access router of the victim's AS. These locations support defense mechanisms such as IP Traceback, packet dropping based on congestion level, management information base (MIB), and packet marking and filtering mechanisms such as hop count filtering and history-based IP filtering. IP Traceback mechanisms find spoofed IP packets true sources. Packet dropping based on congestion is the process of dropping suspicious packets when the network is congested. MIB data includes packet and routing statistics that can be used to identify when a DDoS attack is occurring. Packet marking and filtering mechanisms "aim to mark legitimate packets at each router along their path so that victims' edge routers can filter the attack traffic" [1].

Commented [HM26]: Acronym (could move this earlier in the paragraph, then use the acronym).

Network-based

Zargar et al. [1] states that network-based defenses are deployed inside an AS's network, typically on routers. Network-based defense mechanisms aim to detect attack traffic and stop it at intermediate networks. Major defense mechanisms include route-based packet filtering and detection of and filtering malicious routers. Router-based packet filtering extends ingress filtering "routers at the core of the internet" [1]. Detection of and filtering malicious routers can be done since the creation of specialized anomaly detector protocols that are able to detect malicious routers involved in packet forwarding [1].

Commented [HM27]: General knowledge or part of [5]?

Hybrid (i.e. distributed)

Zargar et al. [1] states that hybrid (i.e. distributed) defenses are distributed over multiple locations and include components for detecting attacks at a victim's router (destination-based) while deterring attacks as the attacker's router (source-based). Modules are placed near the victim that performs filtering close to the source of attack in the hybrid packet marking and throttling/filtering method, while exchange is enabled among defense nodes in the DEFensive Cooperative Overlay Mesh method. Other methods include capability-based mechanisms and Stop-It. Capability-based mechanisms require destinations to tell routers what traffic they are expecting to receive. Stop-It is a filter-based mechanism in which receivers can detect and stop unwanted traffic.

3.1.2 – Application-level attacks

Application-level defenses include destination-based and hybrid (i.e. distributed) defenses.

Destination-based

Zargar et al. [1] states that destination-based defenses observe a network's servers and attempt to model clients' behavior to detect malicious attacks. Types of attacks destination-based

defenses protect against include reflection/amplification attacks and DDoS attacks. Technologies to help detect and mitigate attacks are DDoS-Shield, anomaly detector based on hidden semi-Markov models, and defense against tilts (DAT). Defense against reflection/amplification attacks are deployed at server locations and attempt to detect malicious traffic from protocols using techniques such as machine learning. DDoS-shield aims to "detect characteristics of HTTP sessions and employs rate-limiting as the primary defense mechanism" [1]. This defense method also includes a component that assigns a suspicion rating to each client session. The anomaly detector based on hidden semi-Markov model is able to explain the dynamics of an access matrix which can help to detect DDoS attacks. DAT monitors user's features such as instant traffic volume, session behavior, and more to determine of a user is malicious [1].

Hybrid (i.e. distributed)

Zargar et al. [1] states that hybrid defenses involve collaboration between clients and servers meaning attacks are detected at the server and passed on to the client. Some defense mechanisms include speak-up, defense and offense wall (DOW), admission and congestion control, trust management helmet (TMH), and hybrid detection based on trust and information theory-based metrics. Speak-up is a defense mechanism that requests clients to send more traffic in an attempt to reduce the number of malicious requests. DOW works similar to the speak-up method, but also uses an anomaly detector to detect flooding attacks. Admission control limits the number of concurrent clients on a service using port hiding to make the service appear hidden to unauthorized clients. TMH distinguishes between legitimate and malicious users using trust. When an attack is detected, the system prioritizes "good users" instead of attempting to identify all requests.

Commented [HM28]: Spell out this acronym here instead of below.

Commented [HM29]: I would reword this to "Types of attacks destination-based defenses protect against include reflection/amplification attacks and DDoS attacks." Then, introduce the different technologies that help detect/prevent/mitigate DDoS attacks.

Commented [HM30]: Feels like you're missing a source (or sources) for this section—these specific approaches do not seem like they are common knowledge.

Commented [HM31]: Is this a formal product? Should this be capitalized or is it just the idea of "speaking up" via sending more traffic?

Commented [HM32]: Define DOW acronym here (or lose it).

Commented [HM33]: Define TMH acronym here (or lose it).

Commented [HM34]: Capitalize (or not)—see above.

3.2 – Machine Learning Detection

Liang et al. [8] conducted three experiments to assess the value of machine learning techniques for DDoS detection. The first experiment compared machine learning techniques to the more traditional DWARD technique, and results indicated that although DWARD performed competitively, machine learning techniques outperformed DWARD in all most cases [8]. This experiment also raised doubts as to whether machine learning could effectively detect DDoS attacks. In the second experiment, Liang et al. [8] limited detection to only a portion of the network, but the results indicated that deployment location does not affect how machine learning techniques performed. The third experiment was aimed at assessing the impact of the class imbalance problem, which arises when the "number of observations of one class is far less than the observations for the other class" [8]. The results indicated that the class imbalance problem significantly affects machine learning performance and proposed that careful training must be implemented in order to mitigate this performance loss.

Zhang et al. [9] surveyed some of the most recent machine learning and artificial intelligence techniques used to detect DoS/DDoS attacks. One approach discussed was DeepDefense, a deep learning detection method. DeepDefense uses recurrent neural network (RNN) since it is better at learning features compared to other machine learning models. RNN is better at learning longer historical features that other machine learning techniques. Results from this experiment indicated that RNN performs well and has a better performance that random forest [9]. Another experiment mentioned by Zhang et al. [9] proposed a simple network architecture using a real web server, bait server, and decoy server to distinguish between malicious and legitimate traffic. The results from this study determined that the decision tree model worked the best at distinguishing traffic.

Commented [HM35]: Include IEEE inline citation here.

Commented [HM36]: Be more specific here (but no need to add a lot of text—just briefly state what "best" means in this context). That clarification would help your next sentence make more sense to me.

Commented [HM37]: Another IEEE inline citation here.

Commented [HM38]: This is a good example of results synopsis.

Commented [HM39]: How? Brief explanation, if possible.

Commented [HM40]: Another IEEE inline citation here.

Commented [HM41]: These sentences need rewriting—I'm having trouble following the intent. Clarify "better at learning features compared to other ML models."

Commented [HM42]: IEEE Citation

Al-Eidi et al. [10] propose an innovative image-based solution for the detection of Covert Timing Channels (CTC). They observed that covert channels can be converted to colored images, and that malicious traffic packets can be extracted from these images. Knowing this, the solution is able to convert traffic flows into colored images from which image-based features can be extracted. A classifier was then trained using the extracted features, and the performance results were remarkable with the classifier achieving 95.83% detection accuracy for cautious CTCs [10].

Commented [HM43]: IEEE Citation

3.3. – Other Detection Methods

Shinde et al. [11] propose an approach to detecting DoS attacks using smoothened time-series and wavelet analysis. Their method considers all the traffic in a network as a time-series and then they smooth it using exponential moving average. Their data is then analyzed using wavelet analysis. Their approach significantly shortens the amount of time it takes to detect a DoS attack in addition to having a high accuracy with less false positives [11].

Commented [HM44]: IEEE Citation

RoselinMary et al. [12] propose an approach for early DoS detection in VANET using Attacked Packet Detection Algorithm (APDA). APDA is used to detect DoS attacks before verification time. The algorithm enhances the security of VANET by being applied “before the verification time delay overhead is minimized” [12].

Commented [HM45]: IEEE Citation

4 Proposed Approach

4.1 – Environment

The legitimate and malicious traffic was captured using Wireshark. The malicious attacks were run using a virtual machine, while the legitimate traffic was captured on the regular machine. To set up the virtual machines for the malicious attack, a Kali Linux operating system and a Windows Security XP operating system were set up using VMWare Workstation 16 Player. The Kali machine ran the attack and captured the traffic data while the Windows Security

machine acted as the victim. The legitimate traffic data was captured without using a virtual machine, while still using Wireshark to capture the traffic data.

4.2 – Collection of malicious traffic

To collect malicious data, various syn flood attacks were launched on a virtual machine with Wireshark capturing the data. The attacks were limited to ten seconds. Ten seconds was chosen because it is long enough for the machine learning algorithm to be able to detect the attack yet quick enough to help mitigate the effects DoS attacks can have on a system. The Syn flood attacks were launched at various packet intensities of 50, 100, 250, 500, and 1000 with the increasing packet intensities meaning more syn flood packets are sent each cycle.

Commented [HM46]: Capitalization?

4.3 – Collection of legitimate traffic

To collect legitimate traffic, four scenarios were run to simulate real user data. The first scenario was to browse Google Search results and click on a link to load a page. The second scenario was scrolling through pictures on Instagram. The third scenario was to download a large file from OneDrive. This particular scenario was run in order to generate a large packet flow that could be similar to a malicious user. The last scenario was to play a video on YouTube. These scenarios were also limited to ten seconds to match the data collected from the malicious attacks.

4.4 – Feature extraction

All the malicious and legitimate traffic was captured using Wireshark and saved as PCAP files. An open-source program on GitHub (created by lucadivit [13]) was used to convert the PCAP files into features that the machine learning algorithm can understand [13]. Table 1 shows the features that were extracted from the PCAP files. Note: Delta time is the time between a packet and the following packet, and a payload is considered small if it is 32 Bytes or less.

Commented [HM47]: IEEE citation here.

Table 1: utilized features and their descriptions

Feature Name	Description
Avg_syn_flag	Indicated the average amount of packets whose synchronization flag was active
Avg_urg_flag	Indicated the average amount of packets whose urgent flag was active
Avg_fin_flag	Indicated the average amount of packets whose finish flag was active
Avg_ack_flag	Indicated the average amount of packets whose acknowledgement flag was active
Avg_psh_flag	Indicated the average amount of packets whose push flag was active
Avg_rst_flag	Indicated the average amount of packets whose reset flag was active
Avg_DNS_pkt	The average amount of Domain Name Service packets
Avg_TCP_pkt	The average amount of Transmission Control Protocol packets
Avg_UDP_pkt	The average amount of User Datagram Protocol packets
Avg_ICMP_pkt	The average amount of Internet Control Message Protocol packets

Duration_window_flow	The time between the first and last packet
Avg_delta_time	The average delta times between packets
Min_delta_time	The minimum delta times between packets
Max_delta_time	The maximum delta times between packets
StDev_delta_time	The standard deviation of the delta time
Avg_pkts_length	The average packet length
Min_pkts_length	The minimum packet length
Max_pkts_length	The maximum packet length
StDev_pkts_length	The standard deviation of packet lengths
Avg_small_payload_pkt	The average amount of packets with a small payload
Avg_payload	The average payload size
Min_payload	The minimum payload size
Max_payload	The maximum payload size
StDev_payload	The standard deviation of payload sizes
Avg_DNS_over_TCP	The average number of DNS over TCP packets

The PCAP feature extractor also generates a label 1-9 indicating the nature of the traffic as shown in Table 2.

Table 2: Classification labels and their descriptions

Label	Description
1	Indicates malicious DoS attack traffic with an intensity of 50
2	Indicates malicious DoS attack traffic with an intensity of 100
3	Indicates malicious DoS attack traffic with an intensity of 250
4	Indicates malicious DoS attack traffic with an intensity of 500
5	Indicates malicious DoS attack traffic with an intensity of 1000
6	Indicates legitimate Google traffic
7	Indicates legitimate Instagram traffic
8	Indicates legitimate OneDrive traffic
9	Indicates legitimate YouTube traffic

4.5 – Variance

Variance is the spread of data from the mean. In other terms, variance helps identify what features are most important to the classification model. A high variance indicates that a feature is important for the outcome of the classification model. Figure 1 shows the variance calculated for all the features in the study. The variance indicates that there are three features that were critical in identifying whether the traffic was malicious or legitimate: Max_pkts_length, Max_payload,

and st_dev_delta_time. These three features had an extremely high variance, all higher than 4.00E+05.

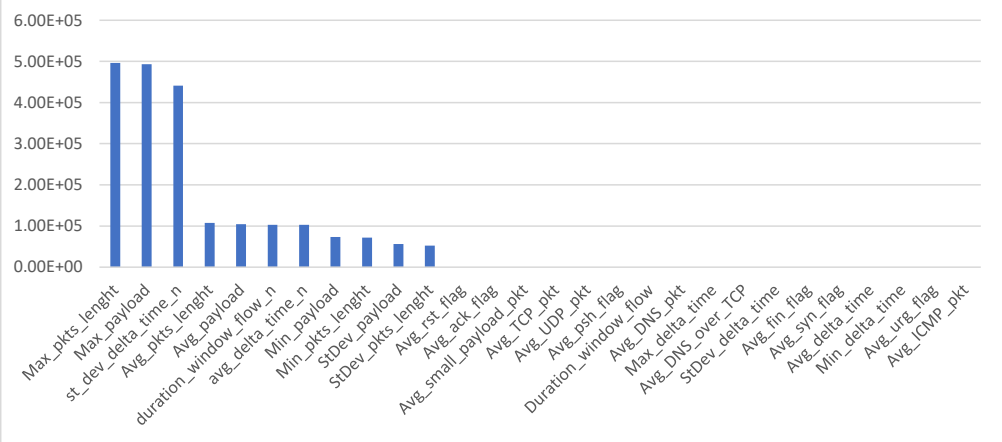


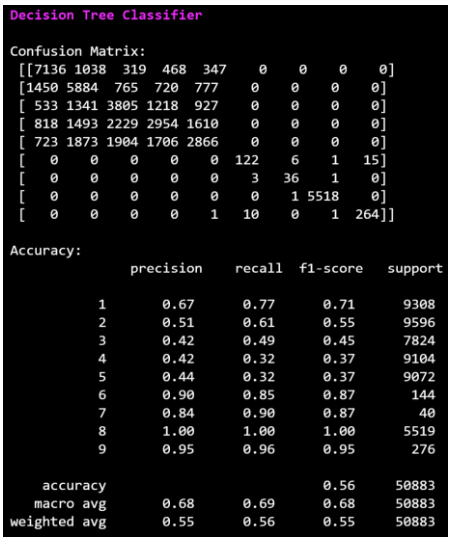
Figure 1: feature variance analysis

4.6 – Classification

We constructed two models to classify our data: a decision tree and a naïve Bayes model.

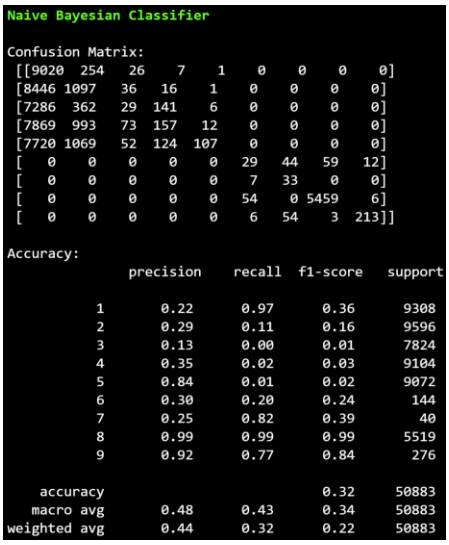
5 Evaluation and Discussion

The evaluation seeks to measure the accuracy of detecting various intensities of malicious attacks and various legitimate scenarios. Figures 2 and 3 show the accuracy measures of DoS detection using the Decision Tree and Naïve Bayes Classifiers.



```
Decision Tree Classifier

Confusion Matrix:
 [[7136 1038  319  468  347    0    0    0    0]
 [1450 5884  765  720  777    0    0    0    0]
 [ 533 1341 3805 1218  927    0    0    0    0]
 [ 818 1493 2229 2954 1610    0    0    0    0]
 [ 723 1873 1904 1706 2866    0    0    0    0]
 [   0    0    0    0    0  122    6    1   15]
 [   0    0    0    0    0    3   36    1    0]
 [   0    0    0    0    0    0    1 5518    0]
 [   0    0    0    0    1   10    0    1  264]]

Accuracy:
              precision    recall  f1-score   support

           1       0.67      0.77      0.71      9308
           2       0.51      0.61      0.55      9596
           3       0.42      0.49      0.45      7824
           4       0.42      0.32      0.37      9104
           5       0.44      0.32      0.37      9072
           6       0.90      0.85      0.87       144
           7       0.84      0.90      0.87        40
           8       1.00      1.00      1.00      5519
           9       0.95      0.96      0.95       276

    accuracy                           0.56     50883
   macro avg       0.68      0.69      0.68     50883
weighted avg       0.55      0.56      0.55     50883
```

Figure 2: Accuracy measures of DoS detection using the Decision Tree Classifier.



```
Naive Bayesian Classifier

Confusion Matrix:
 [[9020  254   26    7    1    0    0    0    0]
 [8446 1097   36   16    1    0    0    0    0]
 [7286  362   29  141    6    0    0    0    0]
 [7869  993   73  157   12    0    0    0    0]
 [7720 1069   52  124  107    0    0    0    0]
 [   0    0    0    0    0   29   44   59   12]
 [   0    0    0    0    0    7   33    0    0]
 [   0    0    0    0    0   54    0 5459    6]
 [   0    0    0    0    0    6   54    3  213]]

Accuracy:
              precision    recall  f1-score   support

           1       0.22      0.97      0.36      9308
           2       0.29      0.11      0.16      9596
           3       0.13      0.00      0.01      7824
           4       0.35      0.02      0.03      9104
           5       0.84      0.01      0.02      9072
           6       0.30      0.20      0.24       144
           7       0.25      0.82      0.39        40
           8       0.99      0.99      0.99      5519
           9       0.92      0.77      0.84       276

    accuracy                           0.32     50883
   macro avg       0.48      0.43      0.34     50883
weighted avg       0.44      0.32      0.22     50883
```

Figure 3: Accuracy measures of DoS detection using the Naïve Bayes Classifier.

5.1 – Experimental results

Commented [HM48]: I would use tables to report the exact numbers: use the text as a place to qualitatively discuss your results and interpret their meanings.

5.1.1 – Decision Tree Classification

In our first experiment, we trained a decision tree model using malicious and legitimate data combined into a single file with 80% used for training and 20% used for testing. Table 3 below shows the accuracy measures of Decision Tree Classification. Figure 4 shows a comparison of the accuracy measures. The Decision Tree model yielded an accuracy of 56%. Overall, the Decision Tree model worked well. Although it had some difficulties distinguishing between some traffic, the model was able to almost entirely distinguish between malicious and legitimate traffic except for one single case. For the malicious traffic, the model was fairly good at predicting the label 1 traffic, but as the intensity of the attacks got higher, the model had more

trouble distinguishing between attacks. For the legitimate data labels 6-9, however, the model did very well at differentiating between the scenarios with the lowest F_1 score being a .87. Also worth noting is the model did a fantastic job distinguishing the OneDrive data as it received 100% precision and recall as well as an F_1 score of 1.

Table 3: accuracy measures of Decision Tree Classification

Label	Precision	Recall	F_1 Score
1	67%	77%	.71
2	51%	61%	.55
3	42%	49%	.45
4	42%	32%	.37
5	44%	32%	.37
6	90%	85%	.87
7	84%	90%	.87
8	100%	100%	1.00
9	95%	96%	.95

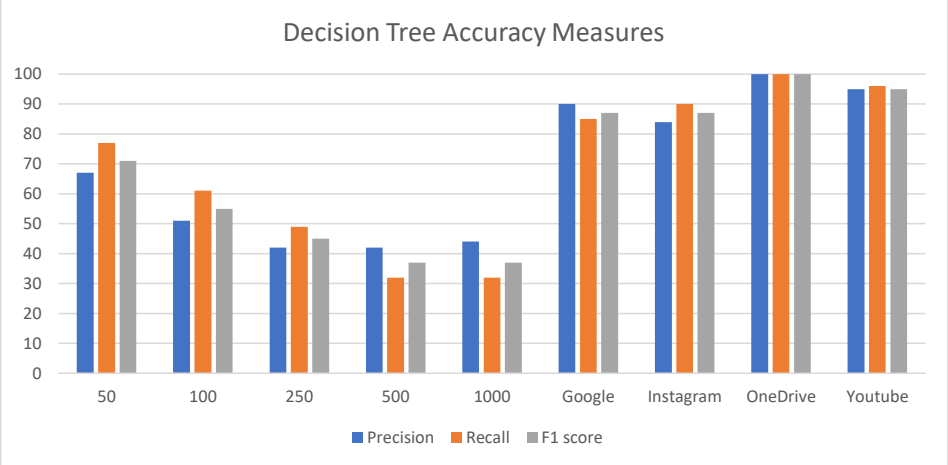


Figure 4: comparison of Decision Tree accuracy measures

5.1.2 – Naïve Bayes Classification

In our next experiment, we trained a Naïve Bayes model using malicious and legitimate data combined into a single file with 80% used for training and 20% used for testing. Table 4 below shows the accuracy measures of Naïve Bayes Classification. Figure 5 shows a comparison of the accuracy measures. The Naïve Bayes model yielded an accuracy of 32%. Overall, this model did not do as well as the decision tree but predictions for certain labels did well. This model was able to completely distinguish between malicious and legitimate traffic but has some troubles when differentiating between the various types of each traffic data. There were a few recall scores that were very high; they included labels 1, 7, 8, and 9. Labels 8 and 9 both had high scores across the board, and similarly to the Decision Tree model, the Naïve Bayes classifier did extremely well with the OneDrive data (label 8).

Table 4: accuracy measures of Naïve Bayes Classification

Label	Precision	Recall	F_1 Score
1	22%	97%	.36
2	29%	11%	.16
3	13%	0%	.01
4	35%	2%	.03
5	84%	1%	.02
6	30%	20%	.24
7	25%	82%	.39
8	99%	99%	.99
9	92%	77%	.84

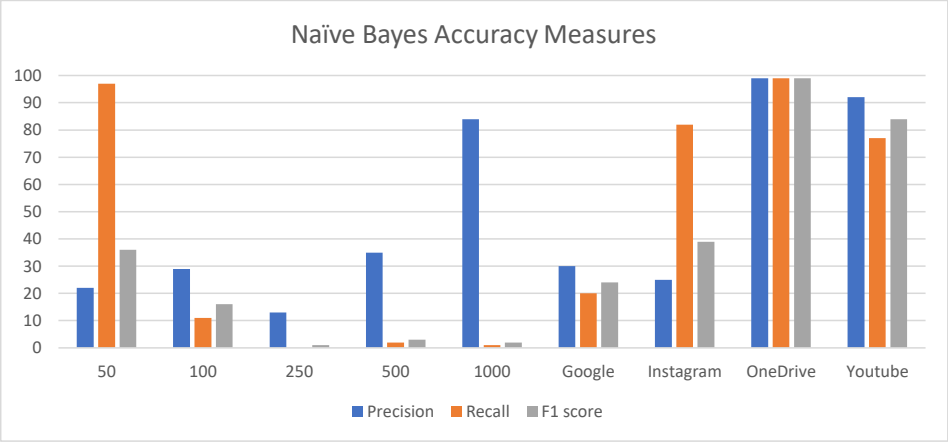


Figure 5: comparison of Naïve Bayes accuracy measures

6 Conclusion

6.1 – Discussion

This experiment aimed to identify malicious traffic from legitimate traffic within ten seconds. To accomplish this Wireshark was used to collect legitimate and malicious data. Data was then turned into features using an open-source feature extractor, and two classification models were constructed. The Decision Tree and Naïve Bayes classification models did fairly well in their accuracies. The Decision Tree had excellent F_1 scores for labels 1, 6, 7, 8, and 9 while Naïve Bayes had excellent recall scores for labels 1, 7, 8, and 9. While both models had some difficulty distinguishing between the intensity levels of the malicious traffic, they were able to successfully distinguish between the legitimate traffic and malicious traffic.

Commented [HM49]: Reword to discuss what was accomplished without the numbering.

Commented [HM50]: Restate qualitatively—save the quantitative measures for the results.

6.2 – Future Work

Commented [HM51]: Consider moving this section to your conclusion chapter.

I might change from the first person to the third person for discussing ideas in this section.

The tone is too conversational.

The future direction of this research is to expand the malicious and legitimate data collection. One way to do this would be to include other forms of legitimate traffic that might look similar to a DoS attack with one idea being to watch an inbox get spammed with messages. The main expansion for future work would be the attacks though. In this experiment the only attacks collected were syn flood attacks, but since attackers can use various other methods of attack, more malicious traffic should be generated using methods such as UDP flooding, ICMP flooding, HTTP flooding, etc. It would also be important to expand the detection to include DDoS attacks, as they are similar and more widely used. Malicious data traffic should also include some popular DoS/DDoS tools such as Slowloris, HULK (HTTP Unbearable Load King), LOIC (Low Orbit Ion Cannon), etc. since tools like these are commonly used to launch DoS and DDoS attacks. In addition to the expansion of data collection, more classification models such as the artificial neuron network, support vector machine, and random forest should

be added. One final idea would be to change the environment the attacks are run in. The attacks were run using a virtual machine meaning they were in an isolated, controlled environment, but the legitimate data was collected while running on a regular machine environment.

References

[1] S. T. Zargar, J. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys & Tutorials,* vol. 15, no. 4, pp. 2046-2069, 2013.

[2] Wireshark, "Chapter 1. Introduction," [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs. [Accessed March 2022].

[3] R. Y. and F. S. J., "The Formation and Use of Abstract Concepts in Design," *Concept Formation,* p. 323–353, 1991.

[4] W. e. a. X, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.,* vol. 14, no. 1, pp. 1-37, 2008.

[5] H. J., J. Lu and C. X. Ling, "Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy," 2003.

[6] R. D. S. Raizada and Y. S. Lee, "Smoothness without Smoothing: Why Gaussian Naive Bayes Is Not Naive for Multi-Subject Searchlight Studies," *PLoS One,* vol. 8, no. 7, p. e69566, July 2013.

[7] J. Korstanje, "The F1 score," Towards Data Science, 31 August 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. [Accessed March 2022].

[8] X. Liang and T. Znati, "An empirical study of intelligent approaches to DDoS detection in large scale networks," in *2019 International Conference on Computing, Networking and Communications (ICNC): Network Algorithms and Performance Evaluation*, 2019.

[9] B. Zhang, T. Zhang and Z. Yu, "DDoS Detection and Prevention Based on Artificial Intelligence Techniques," in *2017 3rd IEEE International Conference on Computer and Communications*, 2017.

[10] S. Al-Eidi, O. Darwish, Y. Chen and G. Husari, "SnapCatch: Automatic Detection of Covert Timing Channels Using Image Processing and Machine Learning," *IEEE Access*, 2020.

[11] P. Shinde and S. Guntupalli, "Early DoS Attack Detection using Smoothened Time-Series and Wavelet Analysis," *Third International Symposium on Information Assurance and Security*, pp. 215-220, 2007.

[12] S. RoselinMary, M. Maheshwari and M. Thamaraiselvan, "Early Detection of DoS Attacks in VANET Using Attacked Packet Detection Algorithm (APDA)," *2013 International Conference on Information Communication and Embedded Systems*, 2013.

[13] Lucadivit, "PCAP Feature Extractor," [Online]. Available: https://github.com/lucadivit/Pcap_Features_Extraction. [Accessed March 2022].