5-2022

A Machine Learning Approach for Reconnaissance Detection to Enhance Network Security

Rachel Bakaletz
East Tennessee State University

A Machine Learning Approach for Reconnaissance Detection to Enhance Network Security

---

A thesis

presented to

the faculty of the Department of Computing

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Computer and Information Systems, Information Technology

---

by

Rachel Bakaletz

May 2022

Dr. Ghaith Husari, Chair

Dr. Brian Bennett, Committee Member

Dr. Mohammad Khan, Committee Member

Keywords: machine learning, reconnaissance, intrusion detection, cybersecurity analytics, classification

ABSTRACT

A Machine Learning Approach for Reconnaissance Detection to Enhance Network Security

by

Rachel Bakaletz

Before cyber-crime can happen, attackers must research the targeted organization to collect vital information about the target and pave the way for the subsequent attack phases. This cyber-attack phase is called reconnaissance or enumeration. This malicious phase allows attackers to discover information about a target to be leveraged and used in an exploit. Information such as the version of the operating system and installed applications, open ports can be detected using various tools during the reconnaissance phase. By knowing such information cyber attackers can exploit vulnerabilities that are often unique to a specific version.

In this work, we develop an end-to-end system that uses machine learning techniques to detect reconnaissance attacks on cyber networks. Successful detection of such attacks provides the target the time to devise plans on how to evade or mitigate the cyber-attack phases that supervene the reconnaissance phase.

DEDICATION

Dedicated to my boyfriend, Michael. Thank you for your unwavering love and support over the last 2 years. I could not have done this without you.

ACKNOWLEDGEMENTS

Thank you to my committee, Dr. Husari, Dr. Bennett, and Dr. Khan for supporting me and helping to guide me through the thesis process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1. Introduction

Before cyber-crime can happen, criminals must do research on the target organization they are planning to attack. This research is called enumeration or reconnaissance, and it allows for a criminal to discover information about a target that can be used in an exploit. Information such as version detection, open ports, and operating systems can be detected during reconnaissance. By knowing the version of an application or service, criminals can exploit vulnerabilities that are unique to a specific version. Criminals can also use known vulnerabilities in operating systems to exploit their target.

When criminals do reconnaissance operations on an organization, there is data sent over the network in the form of packets. These packets contain information and data that is unique to the service that created them. Based on the contents of the packets, the initiating application or software can be determined by looking at calculations such as the duration of the window flow, average delta time, and standard deviation delta time. The duration of the window flow is how long it took from the time the first packet sent to when the last packet was delivered. Average delta time is the average from one packet to the next packet in a window of packets. Standard deviation delta time is the standard deviation from one packet to the next packet in a window of packets. Standard deviation is how dispersed a data set is in relation to the average of a data set. Since the data in the packets is unique to a program, each type of reconnaissance will have unique qualities. These unique qualities can be used to train a computer, in the form of supervised machine learning, to recognize when an organization is about to be the victim of a cyber-attack. Machine learning techniques use data and algorithms to mimic learning in human brains. Supervised machine learning techniques use training sets help the models learn. As a machine learns more over time, the accuracy of the machine improves.

1.1 Cyber Reconnaissance Attack Phase

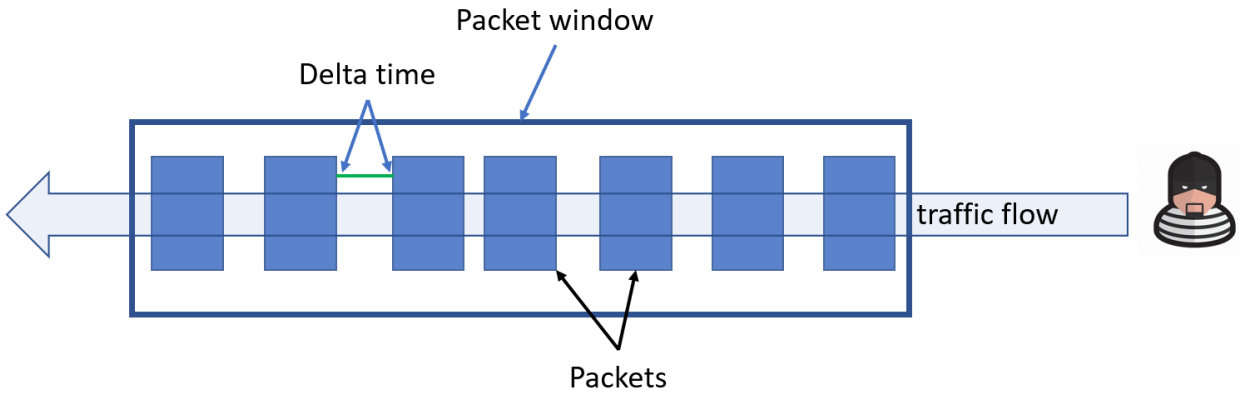
The first step for cyber attackers is launch a reconnaissance attack with the objective of collection information about the software (and sometimes the hardware) of the victim. This is an essential step for attackers to discover potential weaknesses and software vulnerabilities in these systems and subsequently leverage them to gain access to the victim's machine, steal sensitive information, or disrupt the victim's system and potential make unavailable by conducting Denial of Service (DoS) attacks.

1.2 Malicious Traffic Flow

On the cyber network plan, the reconnaissance phase is conducted using a series of packets (traffic flow) that is engineered to interrogate the victim's system and discover technical information that will enable the attackers to use malicious tools and exploiting scripts to gain access to the targeted machine.

Figure 1 illustrates the malicious traffic flow generated during a reconnaissance attack. As shown in the Figure, these packets that are sent from the attacker's machine to the victim's machine flow during a certain period of time (packet window). With this window containing the malicious packets, cyber defenders can use the characteristics of such packet flows to detect these attempts and devise a plan to respond to such attacks using Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Responding to such attacks may contain various countermeasures such as changing the virtual IP address of the targeted machine, block the source IP address (malicious machine) from accessing the network, or force update the software of the targeted machine using the forced reboot operating system functionality.

Figure 1: Reconnaissance Traffic Flow



1.3 Virtual Machines

A virtual environment is the emulation, or reproduction, of a specific software system. Virtual machines are part of a virtual environment and allow the user to run a different operating system than what is on their current machine. For the experiments that I ran, I used virtual machines as the target hosts for each attack. It is illegal to launch any attack against a machine or service that you do not have permission to attack, or that you do not own. By using a virtual machine, I was able to run attacks against targets without breaking any laws.

1.3.1 Kali Linux

Kali Linux is an operating system used for penetration and security testing. Kali Linux contains hundreds of tools and programs that can be used to the security of a target or system. All attacks used in this thesis were from Kali Linux.

1.3.1 Metasploitable

Metasploitable is a virtual machine that has purposeful vulnerabilities integrated in it. Metasploitable can be used by penetration testers to test tools and conduct training. The web application DVWA is housed in Metasploitable.

1.3.3 DVWA

Damn Vulnerable Web Application (DVWA) is a web application that is made to be purposely vulnerable. DVWA offers security professionals a chance to try their penetration testing tools in a controlled environment. It also offers web developers a chance to see all the ways a web application may be unsecure.

1.4 Machine Learning

Supervised machine learning algorithms use labeled datasets to train models to correctly classify data [1]. In supervised learning, models learn by using training sets of data. These training sets teach the models to produce the proper output. The models learn over time from the dataset, which include the correct outputs. The accuracy for the algorithms is measured in loss function, which is the difference between the current output and the expected output. The model will try and minimize this number as much as possible, to make the model as accurate as possible.

1.4.1 Decision Tree

A decision tree is a type of supervised learning model that classifies data by transforming that data into tree structure or representation [2]. When a decision tree learns, it continuously splits the data based on specific parameters defined by the model. Some advantages of decision tree models are that they require less data preparation, and missing values do not affect the tree-building process.

1.4.2 Naïve Bayes

A Naïve-Bayes is type of supervised learning model that classifies data by using the Bayes-Theorem [3]. The Bayes-Theorem uses conditional probability to predict outcomes. Some advantages to using the Naïve-Bayes theorem is that it is easy to implement and gives

good results because prior training and knowledge can be combined with new data, which improves the accuracy [4].

1.4 Remaining Organization of Thesis

The remainder of this thesis is organized as follows. In section 2, there is a discussion of literature that inspired this thesis. In section 3, an overview of the design and implementation of this architecture is presented. In section 4 the results of that architecture are evaluated by various machine learning models. A conclusion is drawn and avenues for potential future work are offered in section 5.

Chapter 2. Background and Literature Review

2.1 Background

As mentioned earlier, a cyber attacker must generate a series of packets engineered in a certain way to interrogate the victim's system and discover its technical information such as the version of its operating system, installed applications, open ports, etc. Constructing these packets manually is a tedious- and time-consuming process. To avoid this, cyber attackers use automated tools that can generate and control these packets and achieve the objective in rapid manner. This subsection introduces some of the popular (and malicious) tools that are used in this domain.

2.1.1 Aggressive Scanning

In nmap, aggressive scanning is a type of scan against a target that provides information such as operating system detection, version detection, script scanning and traceroute—the path that network packets takes when data is transmitted in a system. The argument -A can be used when initializing an aggressive nmap scan; however, it does have a higher rate of detection than other forms of scans.

2.1.2 Stealth Scanning

In nmap, stealth scanning is a type of scan against a target that provides information such as open ports. Based on this information, an attacker can find ways to exploit certain vulnerabilities, or they could find out what services are running on specific ports. The argument -s can be used when initializing a silent nmap scan, and it has a lower rate of detection than other forms of scans like an aggressive scan.

2.1.3 NMAP

Nmap is network scanner that is available in Kali Linux. Nmap supplies many features that can be used to discover information about a host. Nmap works by sending packets to a target and analyzing the responses. Table 1 depicts some functionalities available to use in nmap. This is not a complete list.

Table 1: Nmap functionalities and descriptions [5]

Functionality	Description
-sS	Scans TCP SYN ports
-sn	Disables port scans, and only discovers hosts
-p	Scans a specified port
-sV	Determines the version of running services on a port
-A	Detects operating systems, software versions, running scripts, and traceroute

2.1.4 Nikto

Nikto is similar to Nmap, but instead of computer systems, it scans web applications. Nikto can check for a variety of issues in a vulnerable web application, like dangerous files and outdated software or services. Tables 3 and 4 depict some functionalities available to use in Nikto. This is not a complete list.

Table 2: Nmap functionalities and descriptions

Scan Options	Description
-h	Scans host
-h -port	Scans a specific port on a host
-h -ssl	Scans using SSL
-h -dbcheck	Checks database of host
-h -nolookup	Stops DNS lookups for a host

Table 3: Nmap tuning options and descriptions [6]

Tuning Options	Description
0	Uploads a file to host
1	Allows user to view a specific file in a log
2	Default file misconfiguration
3	Displays information disclosure
4	Allows for XSS, Script, and HTML injection

2.1.5 Metasploit

Metasploitable is a penetration testing framework available in Kali Linux. In Metasploitable, the user can customize attacks against a target. Metasploitable offers a comprehensive set of tools that can be used to help test the security of a system or target.

2.2 Literature Review

This subsection outlines published research on cybersecurity analytics, as well as research on using machine learning and network traffic to predict attacks. Topics such as intrusion detection and image classification using machine learning are also discussed in this section.

Fang et al. [7] propose a method involving machine learning that enhances intrusion detection in computer systems. They use Elman neural networks and support vector machines (SVM) to develop an intrusion detection system that ensures the safety of information systems and computing devices connected to a network. Fang et al. [5] assert that their neural network reduces missing text information by using a clustering algorithm and helps to further detect

abnormal behavior between packets. Their SVM helps to improve intrusion detection and lower the false alarm rate of the models.

AL-Eidi et al. [8] propose a method that uses machine learning to detect malicious traffic flows from legitimate traffic flows in Covert Timing Channels (CTC). The authors present an automated CTC detection solution that converts traffic to colored images. Their solution detects and locates malicious segments of traffic flows. Since their solution locates only malicious parts of traffic flows, the quality of service is less impacted by the dropping of only the malicious flow because entire channels are not blocked.

Anderson et al. [9] propose a machine learning framework that can detect even advanced malware that has tried to obfuscate itself. The authors state that most malware detection systems use classification algorithms along with static or dynamic features, but not both. Their proposed framework uses classification algorithms along with both static and dynamic features for the most complete detection model. The authors plan to do this by using kernels to assign a similarity metric to each view which produces the most accurate support vector machine classifier. [9]

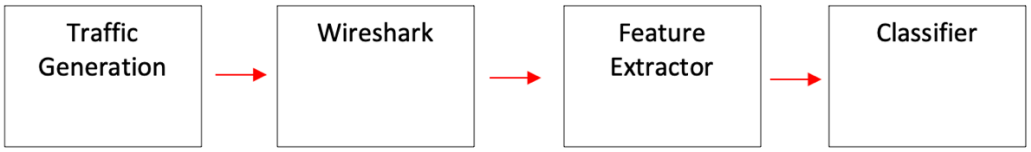
Shon and Moon [10] propose a machine learning method using a Support Vector Machine (SVM) classifiers to detect anomalies in network traffic. The authors assert that signature-based detection programs are not sufficient given the swift evolution of viruses and cyberattacks. This is due to the fact that signature-based tools can detect malicious tools using characteristics can be easily changed by attackers such as the malicious file name and size. Their method uses a hybrid approach of enhanced SVM's to solve this issue. Along with using SVM's, Shon and Moon also use "a profile of normal packets using Self-Organized Feature Map (SOFM), … a packet filtering scheme based on Passive TCP/IP Fingerprinting (PTF), … a

feature selection technique using a Genetic Algorithm (GA)…[and they] use the flow of packets based on temporal relationships during data preprocessing" [10].

Chapter 3. Design and Implementation

This section presents a malicious traffic detection approach while detailing the necessary steps for detecting malicious and non-malicious traffic using machine learning. This approach can be categorized into 3 main sections: 1) Traffic generation from malicious and legitimate traffic; 2) Feature extraction from packet captures; 3) Classification using Decision Tree and Naïve Bayesian machine learning models. I used Kali Linux, Windows 7, Windows 10 and Metasploitable environments. All environments except Windows 10 were virtual. Figure 2 shows a diagram of the proposed design and implementation.

Figure 2: Proposed design and implementation overview



3.1 Approach Design and Architecture

To construct a machine learning-based IDS, first, a dataset of packets (or traffic flows) needed to be acquired. In this thesis, I generated two types of traffic: Malicious and Legitimate (non-malicious) traffic with the objective of constructing a machine learning model to distinguish between these types of traffic and detect the malicious ones.

3.1.1 Generating Malicious Traffic

The malicious traffic in this thesis is a series of packets sent to a targeted machine to perform different types of reconnaissance attacks. To execute these attacks and generate the malicious traffic, I used different malicious and popular tools. These tools are as follows:

1- Metasploit

2- Nmap

3- Nikto

These tools are all publicly available on Kali Linux, which is a virtual machine used for the penetration testing of systems. The target machine was a virtual machine running Windows 7.

The first attack used Metasploit against Windows 7. The Metasploit framework is a command line interface available in Kali Linux that is used for penetration testing. In the Metasploit console, I used the module '*auxiliary/scanner/portscan/syn*' against the target machine, Windows 7. This module works by attempting to initiate Transmission Control Protocol (TCP) connections with the target machine's ports. TCP facilitates message exchanges between devices that are connected to a network. Port scanning attempts to investigate a range of ports on a target machine to determine if those ports are open or closed. Based on this determination, certain exploits could then be used to attack the target machine using the open ports that were found during the scan. Also during this module, a SYN packet is sent and if the target machine replies with SYN/ACK packet, then the port is assumed to be open, and thus possibly meaning it could be exploited. A SYN packet is generated when a machine attempts to connect to another machine via TCP. When the SYN packet is received by the target machine, the target machine sends a SYN/ACK packet back to the original machine. To gather the data from this attack, I used Wireshark, which is a network packet capturing tool. I started Wireshark, then ran the scanner/portscan/syn module in Metasploit, then stopped Wireshark and saved the packet capture.

The next attack performed was an aggressive nmap scan. Nmap is network mapping tool that can be used to discover services running on a target machine. Nmap can also find hosts that are live and connected to a specific network. For this attack, I used the -A option to attack my

target. -A is an option available in nmap that aggressively scans a target machine. This option enables operating system (OS) detection, version detection, script scanning, and traceroute. By knowing what OS and version a target machine is running, an attacker can better curate exploits for that machine. Script scans might detect vulnerabilities in certain services, malware, or database information from a target machine. Aggressive scans provide more information than regular nmap scans. The more information an attacker has, the better chance they might at a successful exploit. To gather the data for this attack, I started Wireshark, ran the nmap scan, then stopped Wireshark and saved the packet capture.

The next attack was a stealthy nmap scan. Stealthy scans usually bypass firewall or other anti-virus detection mechanisms. I used the command *'nmap - -stats-every 1m -sS -P0 -T sneaky -p 1-100*' to generate this attack. The -sS flag denotes a syn scan, which is a quick and unaggressive way to scan ports. When doing a syn scan, nmap can go undetected by most anti-virus and firewalls because it does not fully connect to ports. The *-P0* flag means that Nmap will not ping the target system, which would open a connection and could be detectable by a firewall. The -T flag denotes the speed of the scan, with 0 being the slowest and 5 being the fastest. The faster a tool scans a target, the more likely it is to be detected by a firewall. For this experiment, I chose a speed of 1 or 'sneaky' so that this scan would fall below the threshold of danger to an intrusion detection system. Because I chose a much slower scan rate, I only scanned ports 1-100 to save time. To gather the data for this attack, I started Wireshark, ran the nmap scan, then stopped Wireshark and saved the packet capture.

The last attack used Nikto. Nikto is a webserver vulnerability scanner. Nikto scans for issues like dangerous files, outdated versions of software, and problems with software versions. For the target, I used a VM running Metasploitable. Metasploitable is virtual machine that has

known, intentional vulnerabilities. Another tool available in Metasploitable is DVWA, which is a vulnerable web application. I used the command *nikto -h http://192.168.182.135/dvwa/index.php*. This is the default scan in nikto, which looks for dangerous files, outdated software, and lists any known problems with versions running on a webserver. To gather the data for this attack, I started Wireshark, ran the nmap scan, then stopped Wireshark and saved the packet capture.

3.1.2 Generating Legitimate (Non-malicious) Traffic

With the objective of detecting reconnaissance attacks using machine learning, the second part of data must contain legitimate traffic that is normally generated by the users to browse the Internet, download files, stream videos, etc. For the generation of the non-malicious traffic, I used the websites Google, Amazon and YouTube. I started Wireshark, searched for a term on Google, and then stopped Wireshark. For Amazon, I used the same process, but searched for a specific product on their website. For YouTube, I searched for a certain video and then streamed 30 seconds of that video. Each example's data was captured separately using Wireshark.

3.1.2 Feature Extraction

After the generated data was captured using WireShark, I used a tool to extract traffic features from the data, Lucadivit Feature Extractor [11]. The feature extractor takes in pcap files, the filetype generated by Wireshark, and extracts those files' features. This feature extractor can extract a total of 26 features from files. These features represent detailed specifics about a given traffic flow such as the time between the first packet and the last packet in a given windown. All 26 features were used to classify the data points in both models. Table 4 shows a summary of the features used in the classifiers.

Table 4: List of 26 features extracted from traffic flows and their description [11]

Feature	Description
Avg_syn_flag	The average of packets with syn flag active in a window of packets.
Avg_urg_flag	The average of packets with urg flag active in a window of packets.
Avg_fin_flag	The average of packets with fin flag active in a window of packets.
Avg_ack_flag	The average of packets with ack flag active in a window of packets.
Avg_psh_flag	The average of packets with psh flag active in a window of packets.
Avg_rst_flag	The average of packets with rst flag active in a window of packets.
Avg_DNS_pkt:	The average of DNS packets in a window of packets.
Avg_TCP_pkt	The average of TCP packets in a window of packets.
Avg_UDP_pkt	The average of UDP packets in a window of packets.
Avg_ICMP_pkt	The average of ICMP packets in a window of packets.
Duration_window_flow	The time from the first packet to last packet in a window of packets.
Avg_delta_time	The average of delta times in a window of packets. Delta time is the time from a packet to the next packet.
Min_delta_time:	The minimum delta times in a window of packets.
Max_delta_time:	The maximum delta times in a window of packets.
StDev_delta_time	The Standard Deviation of delta time in a window of packets
Avg_pkts_lenght:	The average of packet lengths in a window of packet.
Min_pkts_lenght	The minimum of packet lengths in a window of packet.
Max_pkts_lenght	The maximum of packet lengths in a window of packet.
StDev_pkts_lenght	The standard deviation of packet lengths in a window of packet.
Avg_small_payload_pkt:	The average of packet with a small payload. A payload is considered small if his size is lower than 32 Byte.

Avg_payload	The average of payload size in a window of packets.
Min_payload	The minimum of payload size in a window of packets.
Max_payload	The maximum of payload size in a window of packets.
StDev_payload	The standard deviation of payload size in a window of packets.
Avg_DNS_over_TCP	The average of ration DNS/TCP in a window of packets.
Label	1-7 respectively if pcap is legitimate or malware.

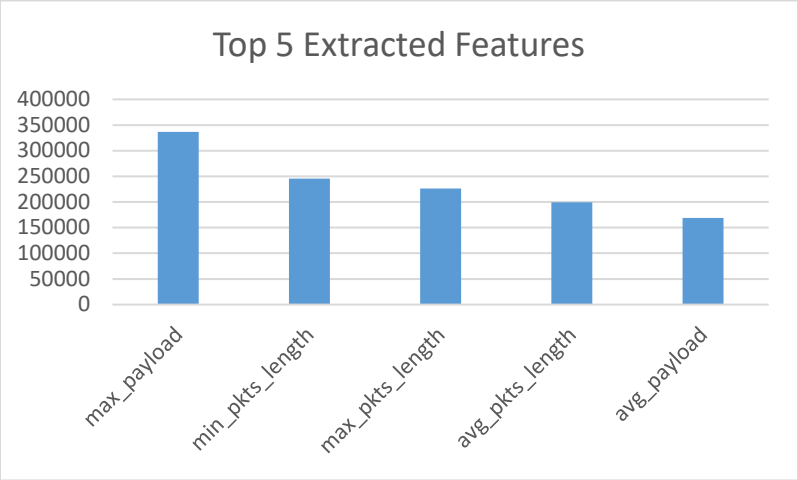
3.1.3 The Variance of the Features

Variance is a popular statistical measurement of the spread between numbers in the data. This spread indicates a higher data quality in terms of machine learning. Machine learning models tend to be accurate when the data has higher variance with spread numbers rather than very similar numbers or near constants.

To measure the spread of numbers for the 25 features (without the label column), I calculated the variance of these features. As shown in Figure 3, the top five features with the highest variance in decending order were as follows:

1. max_payload
2. min_pkts_length
3. max_pkts_length
4. avg_pkts_length
5. avg_payload

Figure 3: Top 5 features extracted from traffic flows



For the machine learning models, we use all 25 features to construct the model. While some of the features have low variance, the machine learning algorithms that we used, namely, decision tree, and naïve bayes will reduce the weight and importance of these features if they do not increase the accuracy of classification. This process is explained in details the next subsection.

3.1.4 Constructing the Classification Model

In machine learning, classification is the process of having the model predict categories, or labels, for a certain dataset. The model usually has testing and training data, which helps it to correctly predict outcomes for datapoints.

The model construction process is often comprised by two steps:

1- **Training**: Training data is used to train the model on which datapoints go in a certain category.

2- **Testing**: Testing data is used to test the model and see if it can correctly identify which datapoints go in each category.

For the classification of the data, I used a Naïve-Bayesian classifier, and a Decision Tree classifier. I chose these classifiers over other models because they are easy to implement and I are models that I had the most experience implementing and using. Decision Tree and Naïve-Bayesian classifiers also classify data differently, and I wanted 2 models that were not similar. Decision Tree models classify data based on pre-defined features that the model selects, while Naïve-Bayesian models have to be told which features to use to classify data. Classifiers use algorithms to categorize data based on labels that the data is given, which trains the models to correctly identify data. Classifiers work in two steps, with the first step being learning, and the last step being predicting. In the learning step, a model is developed using sets of training data. In the prediction step, the model is used to classify new data by using what it's learned in the first step. Naïve-Bayesian models work by using the Bayes Theorem. The Bayes Theorem uses conditional probability to predict outcomes. Conditional probability is the probability that something will happen, given that something else has already occurred. [12] Decision Tree models work by using if-then-else rules to classify data. Table 5 describes each label used in the classifier and it's description.

Table 4: Top 5 features extracted from traffic flows

Label	Description
1	This label denotes the data that was collected during a Metasploit attack.
2	This label denotes the data that was collected during an aggressive nmap scan.
3	This label denotes the data that was collected during a stealthy nmap scan.

4	This label denotes the data that was collected during a Nikto scan.
5	This label denotes the data that was collected during an Amazon product search.
6	This label denotes the data that was collected during a Google search.
7	This label denotes the data that was collected during a Youtube video stream.

Chapter 4. Evaluation

4.1 Supervised Learning

The final step in this thesis is to construct a machine learning model to classify the generated traffic into either a reconnaissance attempt or legitimate traffic. For this step, I trained two models using the following machine learning algorithms:

•Decision Tree (DT).

•Naïve Bayes (NB).

I constructed each machine learning model using the three features extracted from each traffic flow, namely, the duration window flow, the average delta time, and the standard deviation delta time. The data was split into 80% and 20%. The 80% portion of data was used for training and 20% part was used for the testing part.

For both the Decision Tree and the Naïve-Bayesian classifiers, supervised learning methods were used in both algorithms.

4.2 Precision, Recall, and F score

Both models had output in the form of a confusion matrix, which shows a visualization of specific model's performance. Other indicators of a model's performance are precision, recall and F-score. The primitive accuracy measures that were used in this evaluation were True Postive (TP), True Negative (TN), False Positive (FP), False Negative (FN).

True positives tell how many times the classifier correctly identified a positive case. True negatives tell how many times the classifier correctly identified a negative case. False positives tell how many times the classifier identified a negative result as positive. False negatives tell how many times the classifier identified a positive result as negative.

4.2.1 Recall

Recall (R) is the number of samples that were correctly predicted as positive by the classifier. Recall can be calculated with the following formula:

$$\mathbf{R = TP / (TP + FN)}$$

4.2.2 Precision

Precision (P) is the number of positive classifications in the True Positive class that were actually positive. Precision can be calculated with the following formula:

$$\mathbf{P = TP / (TP + FP)}$$

4.2.3 F-score

F-score combines precision and recall by showing the mean of each measurement combined; F-score can be calculated with the following formula:

$$\mathbf{F = 2 \times \frac{Precision * Recall}{Precision + Recall}}$$

4.3 Experimental Results

As previously mentioned, I used 2 classifiers in my experiment which were a Decision Tree classifier and Naïve-Bayesian classifier. Two datasets were given to the classifiers, with the first dataset having 2 labels, malicious and non-malicious traffic. The second dataset had separate labels for each of the generated traffic types, Metasploit, nmap -aggressive, nmap -stealth, nikto, Amazon, Google, and YouTube. For the first experiment, the Decision Tree classifier had an average precision of 99% and an average recall of 99%. The Naïve-Bayesian classifier had an average precision of 98% and an average recall of 98%. For the second experiment, the Decision Tree classifier had an average precision of 90%, and an average recall

of 84%. The Naïve-Bayesian classifier had an average precision of 82%, and an average recall of 84%. Figures 3 and 4 show the precision and recall results of the Decision Tree classifier and the Naïve-Bayesian classifier.

Figure 4: Precision Results

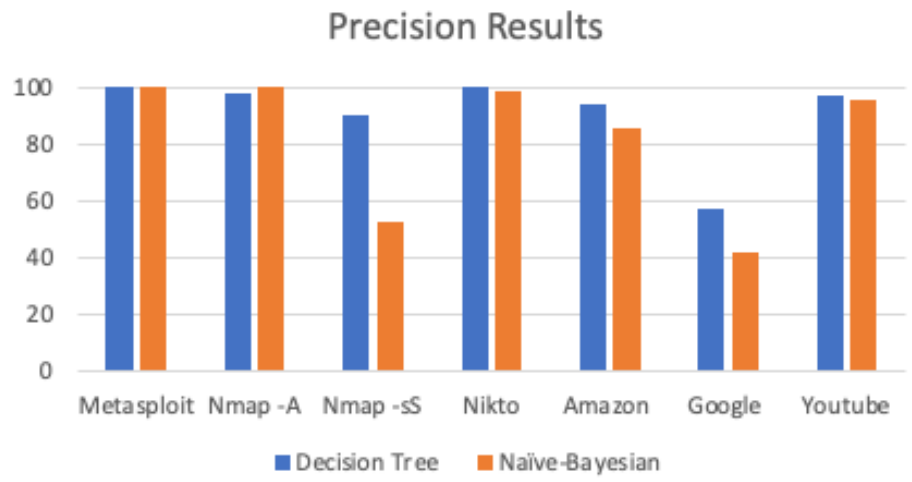


Figure 5: Recall Results

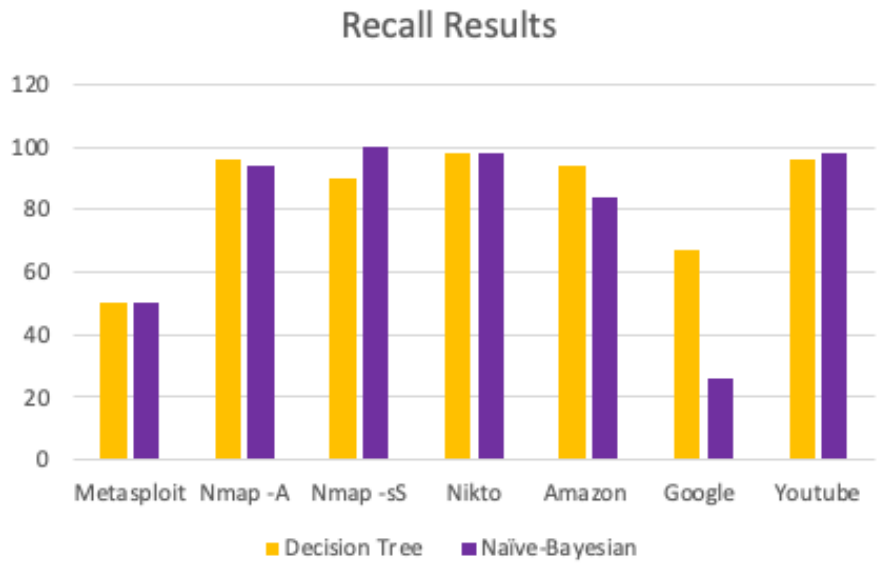


Figure 3: F-Score Decision Tree Results

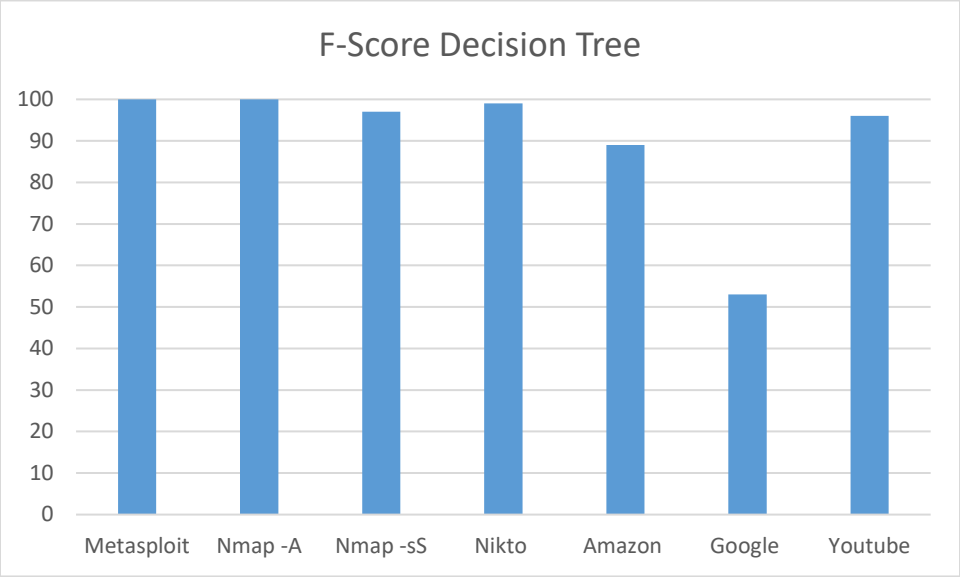
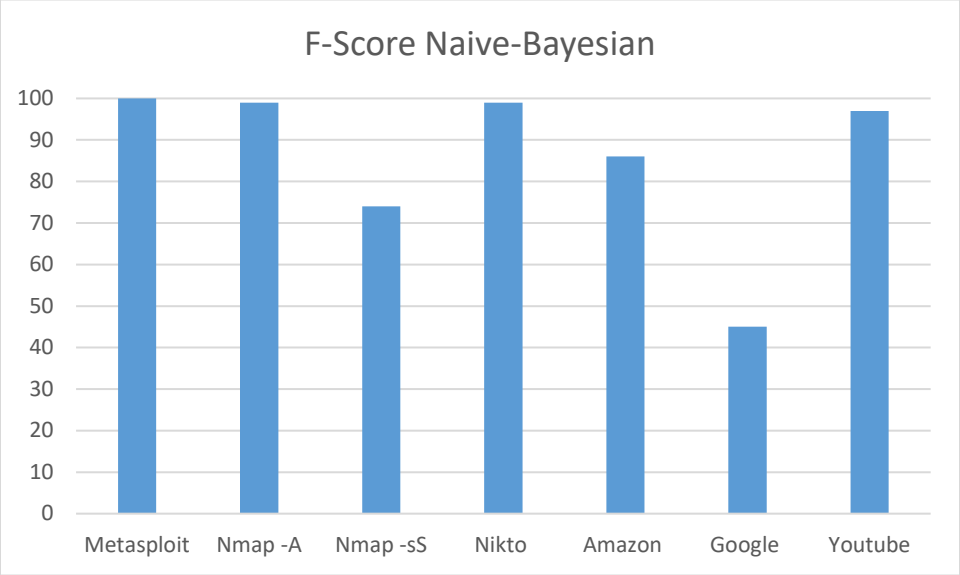


Figure 4: F-Score Naïve-Bayesian Results



4.4 Discussion

Precision and recall are two measures of performance for a machine learning model. Precision is the number of positive samples that were identified correctly, and recall is the number of samples that were correctly predicted as positive by the classifier. Precision is the percentage of traffic that was flagged as malicious that was correctly classified as malicious. Recall is the percentage of malicious traffic that was correctly identified.

One notable result of both models is label 3, which was a Nmap stealth attack. Nmap stealth attacks perform scans of a target by slowing scanning each port in a system, which makes the attack drop below the threshold of detection in a firewall or other intrusion detection system. For precision, a Decision Tree classifier will be a better choice because it has a higher detection rate for malicious traffic. For recall, a Naïve-Bayesian classifier will be a better choice, because it has a better detection rate for malicious traffic.

Chapter 5. Conclusion

This thesis proposed an end-to-end approach to automatically detect malicious traffic from non-malicious traffic. Attackers perform reconnaissance on targets, and these reconnaissance phases allow the attacker to discover vulnerabilities about a system that can later be used to exploit an organization. By collecting network traffic of certain reconnaissance attacks and extracting features from those network captures, a machine learned model can then be trained to correctly identify what legitimate traffic looks like in comparison to illegitimate traffic.

5.1 Future Work

The classification framework presented here is only a starting point for additional research and analysis on identifying malicious vs. non-malicious traffic. One avenue for future work might be to use more reconnaissance tools in the experiments. This would help to build machine learning models that are more comprehensive than the current models. Another avenue for future work might be to use deep learning models, like convolution neural networks, which would also allow for a more comprehensive model. One last avenue for future work might be to gain the permission needed to test the reconnaissance attacks in a non-virtual, live environment. This would help to provide a more comprehensive evaluation and comparison of the machine learning models.

REFERENCES

[1] S. F. Y Reich, "The Formation and Use of Abstract Concepts in Design," *Concept Formation,* pp. 323-353, 1991.

[2] X. Wu, "The Top 10 Algorithims in Data Mining," *Knowledge Information Systems,* vol. 14, no. 1, pp. 1-37, 2008.

[3] J. L. C. L. J Haung, "Comparing Naive Bayes, Decision Treesm and SVM with AUC and Accuracy," 2003.

[4] Y. L. R.D.S Raizada, "Smoothness without Smoothing: Why Gaussian naive Bayes Is Not Naive for Multi-Subject Searchlight Studies," *PLoS One,* vol. 8, no. 7, 2013.

[5] N. House, "Nmap Cheat Sheet," StationX, 1 May 2020. [Online]. Available: https://www.stationx.net/nmap-cheat-sheet/ .

[6] Sullo, "Nikto Cheat Sheet," CompariTech, [Online]. Available: https://cdn.comparitech.com/wp-content/uploads/2019/07/NIkto-Cheat-Sheet.pdf .

[7] W. Fang, X. Tan and D. Wilbur, "Application of Intrusion Detection Technology in Netowork Safety Based on Machine Learning," 2020.

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

[9] C. S. T. L. Blake Anderson, "Improving Malware Classification: Bridging the Static/Dynamic Gap," in *ACM*, 2012.

[10] J. M. Taeshik Shon, "A hybrid machine learning approach to network anomaly detection," in *Science Direct*, 2007.

[11] L. D. Vita. [Online]. Available: https://github.com/lucadivit/Pcap_Features_Extraction.

[12] O. H. Ojone, "Analysing Datasets Using Naive Bayes Classifier," 26 April 2021. [Online]. Available: https://dev.to/codinghappinessweb/analysing-dataset-using-naive-bayes-classifier-3d7o. [Accessed 2022].

[13] K. Katterjohn, "TCP SYN Port Scanner - Metasploit," [Online]. Available:

https://www.infosecmatter.com/metasploit-module-library/?mm=
auxiliary/scanner/portscan/syn. [Accessed 2022].

[14] J. Peters, "How to Use Nmap: Commands and Tutorial Guide," [Online].
Available: https://www.varonis.com/blog/nmap-commands. [Accessed 2022].

[15] M. Shivanandhan, "Nmap — A Guide To The Greatest Scanning Tool Of All Time," [Online].
Available: https://www.hardcoder.io/nmap-a-guide-to-the-greatest-scanning-tool-of-all-time
/#:~:text=Aggressive%20Scanning,to%20perform%20an%20aggressive%20scan.
&text=Aggressive%20scans%20provide%20far%20better%20information%20than
%20regular%20scans. [Accessed 2022].

[16] N. Authors, "The Phases of an Nmap Scan," [Online]. Available:
https://nmap.org/book/nmap-phases.html#:~:text=Script%20scanning.&text=
They%20commonly%20perform%20tasks%20such,services%2C%20and%20
advanced%20version%20detection. [Accessed 2022].

[17] Y. Said, "Performing Stealth Scans with Nmap," [Online]. Available:
https://linuxhint.com/stealth_scans_nmap/. [Accessed 2022].

[18] M. Bintahin, "What is Nikto Tool in Kali and how to use it?," [Online]. Available:
https://cyber-today.com/what-is-nikto-tool-in-kali-and-how-to-use-it/. [Accessed 2022].

[19] R. Saxena, "HOW THE NAIVE BAYES CLASSIFIER WORKS IN MACHINE
LEARNING," [Online]. Available: https://dataaspirant.com/naive-bayes-classifier-
machine-
learning/#:~:text=Naive%20Bayes%20is%20a%20kind,as%20the%20most%20likely
%20class. [Accessed 2022].

[20] C. Sehra, "Decision Trees Explained Easily," [Online]. Available: https://chirag-
sehra.medium.com/decision-trees-explained-easily-28f23241248. [Accessed 2022].

[21] I. C. Education, "Supervised Learning," [Online]. Available:
https://www.ibm.com/cloud/learn/supervised-learning#toc-how-superv-
A-QjXQz-.

[22] N. Authors, "TCP SYN (Stealth) Scan (-sS)," [Online]. Available:

https://
nmap.org/book/synscan.html#:~:text=SYN%20scan%20may%20be%20requested
,%2C%20%2DsS%20is%20usually%20omitted. [Accessed 2022].

VITA

RACHEL BAKALETZ

Education: M.S. Computing, East Tennessee State University, Johnson City, Tennessee, 2022

B.S. Computing, East Tennessee State University, Johnson City, Tennessee, 2020