Winter 2018

Applying Machine Learning to Advance Cyber Security: Network Based Intrusion Detection Systems

Hassan Hadi Latheeth AL-Maksousy
Old Dominion University

APPLYING MACHINE LEARNING TO ADVANCE CYBER SECURITY: NETWORK BASED INTRUSION DETECTION SYSTEMS

by

Hassan Hadi Latheeth AL-Maksousy
MSc. December 2011, Kent State University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2018

Approved by:

Michele C. Weigle (Director)

Cong Wang (Co-Director)

Ravi Mukkamala (Member)

Dimitrie Popescu (Member)

ABSTRACT

APPLYING MACHINE LEARNING TO ADVANCE CYBER SECURITY: NETWORK BASED INTRUSION DETECTION SYSTEMS

Hassan Hadi Latheeth AL-Maksousy
Old Dominion University, 2018
Director: Dr. Michele C. Weigle
Co-Director: Dr. Cong Wang

Many new devices, such as phones and tablets as well as traditional computer systems, rely on wireless connections to the Internet and are susceptible to attacks. Two important types of attacks are the use of malware and exploiting Internet protocol vulnerabilities in devices and network systems. These attacks form a threat on many levels and therefore any approach to dealing with these nefarious attacks will take several methods to counter. In this research, we utilize machine learning to detect and classify malware, visualize, detect and classify worms, as well as detect deauthentication attacks, a form of Denial of Service (DoS). This work also includes two prevention mechanisms for DoS attacks, namely a one-time password (OTP) and through the use of machine learning. Furthermore, we focus on an exploit of the widely used IEEE 802.11 protocol for wireless local area networks (WLANs). The work proposed here presents a threefold approach for intrusion detection to remedy the effects of malware and an Internet protocol exploit employing machine learning as a primary tool. We conclude with a comparison of dimensionality reduction methods to a deep learning classifier to demonstrate the effectiveness of these methods without compromising the accuracy of classification.

ACKNOWLEDGEMENTS

I would first like to thank my adviser Dr. Michele C. Weigle for all of her guidance, support, and advice. She has broadened my knowledge and helped me to become a better researcher. I was truly fortunate to have her as my adviser. I would like to thank my Co-Advisor Dr. Cong Wang for introducing me to deep learning. I would also like to thank the committee members Dr. Ravi Mukkamala and Dr. Dimitrie Popescu for the valuable comments and suggestions they have offered me. Their feedback has improved the quality of this dissertation.

I am so thankful for my parents and my family. I am thankful for their unconditional love and for raising me the right way, making me the person I am today. I'm also thankful for the HCED in Iraq for giving me the opportunity to study abroad and pursue my dreams. I would like to extend a special thanks to Dr. Zuhair Humadi, Dr. Abdul Hakim Al-Rawi and Mr. Safaa Alwiddy.

Finally, I would like to thank the Old Dominion University community at large and especially the department of Computer Science. They have all made me feel welcome and have surrounded me with the warmth of a family. Old Dominion University will always have a special place in my heart.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The Internet age has given rise to enormous social and economic benefits; however, it also made us face many new challenges. In a fully connected world, Internet security threats continue to evolve, always one step ahead of the most sophisticated defenses. Spam, denial of service (DoS) attacks, phishing, and worms all depend on some form of malicious software, commonly referred to as malware. Malware is a program developed with malicious intent to harm a computer or network or to steal information. The volume of malware has been growing so rapidly that researchers must quickly find new techniques to attempt to understand and eliminate these threats [85, 99]. In 2015, more than 430 million new types of malware were discovered by Symantec. This represents an increase of 36% compared to 2014 [12]. According to Trend Micro, the volume of malware has increased dramatically from 1.01 million cases in 2013 to 7.1 million in just the first half of 2015 [10, 11]. In 2015, over 6.5 million hosts were attacked, and four million unique malware objects were detected according to Kaspersky Labs [56].

The underlying mechanisms in malware can be used to generate new variants. These variants may not be detected by current antivirus programs unless their databases are updated with the new variant. The increasing amount of malware and its shocking diversity makes this problem very difficult to solve. Current anti-virus solutions are not able to detect new malware that has not previously been detected or malware that mutates or changes its binary code. The high availability of anti-detection techniques, as well as the ability to buy malware on the black market, makes it relatively easy for an attacker, even with only minimal skills, to use these tools to exploit vulnerable systems. Current studies show that more and more attacks are being carried out by "script-kiddies" or are automated [29].

1.1.1 WORM BASED ATTACKS

Computer worms are a type of malware that contain highly infectious self-duplicating code that can rapidly infect hundreds of thousands of systems and can be very difficult to

detect [42, 74, 80, 106]. The Slammer worm spread to 90% of its target systems in just 10 minutes [83]. In 2017, the NotPetya and WannaCry worms created a global panic [65]. A great deal of research based on different frameworks and numerous case studies are available for the analysis of the spread of worms across networks [42, 74, 80, 106]. Worms' detrimental influence on computer systems has been significant over the last several decades. Therefore, it has become important to develop intrusion detection systems (IDS) and prevention mechanisms to counter them.

1.1.2 MALWARE CLASSIFICATION

Malware classification approaches have been proposed as solutions to deal with the problem of detecting the increasing diversity of malware [27,78,116]. Signature-based detection is the most commonly used approach for anti-malware systems classification [63]. In computing, each object has a unique signature. Therefore, when an object is detected as malicious, its signature is added to a database of known malware. This database of signatures may contain hundreds of millions of signatures that identify malicious objects. Unfortunately, new versions of malicious code appear that are not recognized by signature-based approaches. However, signature-based detection still provides good protection from the many millions of older, but still active, threats. On the other hand, malware often utilizes small changes in its code through evasive techniques such as polymorphism, and can only be distinguished by a behavioral analysis. Behavior-based malware detection approaches monitor objects based on their intended actions to determine whether they are malicious or not. Attempts to perform actions that are unauthorized would indicate that the object is malicious, or at least suspicious.

Other research has investigated intelligent malware detection by applying data mining or machine learning techniques [78, 111]. Machine learning is a popular tool in data science that has already given excellent results in a wide range of domains such as security, image processing, traffic control, biomedicine, atmospheric study, and many more [22, 72]. These methods might achieve high detection rates at low false positive rates without the burden of human signature generation required by manual methods. In general, machine learning techniques analyze recorded attacks to learn the attack characteristics (features), so that they can be detected in the future.

1.1.3 WIFI DOS ATTACKS

In addition to malware and worm threats, the prevalence of wireless Internet (WiFi)

brings the vulnerability of targeted DoS attacks on those networks. DoS attacks are aimed at disrupting a network's legitimate traffic by sending large amounts of unwanted traffic towards the network, thereby drastically reducing its performance and preventing servers from delivering services to legitimate clients.

The IEEE 802.11 protocol provides two distinct logical wireless links: a data link and a control and management (CM) link. In this protocol, WiFi Protected Access 2 (WPA2) strengthens data protection and network access. It offers cryptographic protection of the data link traffic, but leaves the control channel unprotected. Due to the lack of security in the CM channel, DoS attacks can be triggered on the network which can compromise the confidentiality, integrity, and availability of WiFi networks.

Under normal circumstances, a WiFi network has an access point (AP) that coordinates traffic between WiFi clients. The AP is responsible for all communication to and from the clients. Clients in range of an AP are in one of three states: *client is not authenticated or associated*, *client is authenticated but not associated*, or *client is both authenticated and associated*. Once authenticated and associated, the client can communicate with the AP. When the client receives a de-authentication frame from the AP, the client must go back to the first state, *not authenticated and not associated*. The client must re-authenticate and re-associate after receiving the de-authentication frame in order to communicate with the AP.

The WiFi de-authentication DoS attack targets the communication protocol of a WiFi AP. Attackers generate a large number of de-authentication frames using a spoofing technique which can disconnect the communication between the client and AP. Since unlike data frames, de-authentication and disassociation frames are not protected by cryptographic techniques, an attacker can send de-authentication frames with a spoofed MAC address on behalf of the AP to the clients (or vice versa) and terminate data communication between clients and the AP. If clients receive spoofed de-authentication frames, it terminates communication with the AP, followed by scanning for available APs, and repeating the process of authentication and association [54]. Since it requires very little effort, an attacker can easily fabricate such spoofed frames at a high rate. Then the attacker can inject such spoofed de-authentication/dis-association frames into a WiFi network to carry out a DoS attack. Any device that uses IEEE 802.11a/g/n is affected by this attack.

One possible solution would be to use the IEEE 802.11w protocol, but this requires replacing the devices in the network and would be very expensive and time consuming. Typical solutions [8, 22, 31, 58, 88] for detecting attacks utilize modified node drivers. This

also can be time consuming or impractical in many network systems.

1.1.4 IDS PERFORMANCE

IDS that use many features to detect an attack can incur significant overhead in training and testing due to the large dataset size based on real network data. Furthermore, the amount of data largely affects resource consumption and detection stability. Although a dataset can be large, only a few features may be important for detection. Therefore, the elimination of redundant non-contributing data is essential to reducing the dataset size. Moreover, effective feature selection and data reduction are important in order to decrease the training duration and to improve the detection rate against known and unknown attacks. Thus, a good feature selection approach minimizes the information redundancy and computation complexity, while improving accuracy and detection rates. The motivation becomes a need to reduce the dimensionality without compromising accuracy of classification.

1.2 APPROACH

1.2.1 PREVENTING THE WIFI DE-AUTHENTICATION DOS ATTACK

We propose a machine learning based IDS to detect the WiFi de-authentication DoS attacks. This approach consists of two parts, a packet sniffer and a data extractor, combined with a machine learning (ML) classifier that are used to detect the attack. Our IDS does not require a reconfiguration of the nodes in the network. We also propose two mutually exclusive ways to prevent deauthentication DoS attacks in WiFi networks, but these approaches require modification of network nodes.

The first approach is targeted at prevention and makes use of one-time passwords (OTPs), which can be easily incorporated into the existing IEEE 802.11 standard. OTPs have two useful properties which makes them more effective than conventional persistent passwords [96]:

1. A password is valid for only one transaction or session.

2. An OTP is cryptographically random, which precludes it from being guessed. This method will help stop both deauthentication and disassociation attacks.

The second mechanism is a machine learning based detection approach. Once an attack is detected by this mechanism, special packets are sent out by the IDS to the protected

client nodes and AP instructing them to drop all de-authentication packets for a period of time. During this period of time, if another attack is detected, additional packets are sent, indicating the the nodes should increase the duration they should continue dropping deauthentication packets. The main goal of this de-authentication IDS is to provide a detection mechanism without needing to upgrade the nodes, and to apply a modern solution by using machine learning to solve an existing problem.

1.2.2 DETECTING MALWARE AND WORM ATTACKS

We also focus on detecting worm attacks using their network behavior and leveraging machine learning techniques. We implement a hybrid IDS that will detect worms with modified binary signatures or with similar network behavior through a machine learning approach. Then the suspicious network flows can be analyzed through a visualization tool to provide better insights into worm behavior.

Another approach that we implement is a deep neural network (DNN) to detect and classify malware based on network behavior. We show that the splitting of the system into two neural networks, detection and analysis, is the key to increasing performance. Therefore, this approach allows us to build a monitoring system with very low CPU usage. We compare our approach with other work [32, 61], and we evaluate the performance of four machine learning classifiers.

1.2.3 FEATURE SELECTION AND REDUCTION METHODS

Because the number of features used for network-based detection can be large, we developed a feature reduction method to reduce the computational cost and improve performance without compromising accuracy. The goal is to preclude the requirement for high-end hardware, thereby making it available to more end users. We accomplish this by selecting the optimal features and reducing the feature set. We propose two feature selection tools to address this problem and perform a comparison analysis of different feature reduction methods.

1.3 CONTRIBUTIONS

The overall objective of this work is to apply machine learning techniques to specific types of network attacks: WiFi de-authentication DoS and malware attacks. In the process of developing approaches to addressing these attacks, we have made the following contributions:

1. A one-time password (OTP) approach to preventing the WiFi de-authentication DoS attack. An OTP is valid for only one transaction or session and is cryptographically random, which precludes it from being guessed.

2. A machine learning based approach to detecting the WiFi de-authentication DoS attack. A trained machine learning classifier is used to detect the attack and alert clients to ignore deauthentication packets for a specified time.

3. A hybrid intrusion detection system (IDS) [24] that uses machine learning and visualization to detect and provide an in-depth analysis tool for worm attacks. Machine learning is used to classify and detect both known and unknown worms with similar network behavior. Once the suspicious activity is detected by the machine learning tool, the visualization tool is used to further analyze network traffic. The visualization tool displays the suspicious worm activity, traces the attack, and displays the affected nodes.

4. A ns-3 based worm traffic generation tool for evaluating network-based IDS. The ns-3 tool is used to simulate worm traffic generation with different topologies and multiple worm attacks. The propagation of the worm throughout the network cannot be seen when part of the data is hidden, so using a test network and a simulator allows researchers to simulate an attack and fully capture all aspects of the attack, providing a complete picture of the network behavior.

5. A deep neural network (DNN) IDS [25] to detect and classify malware based on network behavior. The DNN IDS is an efficient system used to detect and classify malware. We propose using two neural networks, one for detection, and the other for analysis, and this proves to be the key to increasing performance. This approach allows for the construction of a monitoring system with very low CPU usage. Four learning classifiers are presented, analyzed, and compared to determine which is best for accuracy and performance.

6. Two feature selection methods to select the optimal features for the malware classification and detection based on the network behavior. The accuracy estimating method uses Shannon entropy for the estimation of the importance of features. The scoring algorithm uses Shannon entropy, informational gain, and linear independence for scoring the features with further cross-correlation-based filtering. The proposed methods work with any kind of Machine Learning (ML) classifier and can be directly applied.

1.4 OUTLINE

The remainder of this work is organized as follows: Chapter 2 presents the background on de-authentication and de-association attacks, the malware and worm dataset, dimensionality reduction methods, and machine learning classifiers. Chapter 3 introduces related work. Chapter 4 presents the IDS for detecting de-authentication attacks. Chapter 5 presents the hybrid intrusion detection system for worm attacks. Chapter 6 presents the IDS for malware. Chapter 7 presents benchmarking dimensionality reduction methods for malware classification based on network behavior. Chapter 8 presents the feature selection tools. Chapter 9 provides a summary of the work and presents ideas for future work.

CHAPTER 2

BACKGROUND

To better understand the scope of this work, we will define the WiFi deauthentication DoS attack and various worm attacks. We also describe several dimensionality reduction and machine learning classification methods.

2.1 WIFI ASSOCIATION AND DEAUTHENTICATION

2.1.1 NORMAL WIFI ASSOCIATION AND DISASSOCIATION

Wireless Internet (Wi-Fi) (IEEE 802.11) networks consist of an access point (AP) and Wi-Fi clients. The AP acts as a central mediator between the Wi-Fi clients. All communication goes through the AP. When a client wants to communicate with another client, it must first authenticate and then associate with the AP.

Clients are typically in one of three states: *not authenticated or associated*, *authenticated but not associated*, or *authenticated and associated*. Once a client is authenticated and associated it can communicate with the AP. When a deauthentication packet is received from the AP, the client must go back to the first state, *not authenticated and not associated*, and must reauthenticate and reassociate to communicate with the AP.

The association process is shown in Figure 1.. The AP and clients exchange 802.11 management frames to get to the authenticated and associated state. The client starts out in the not authenticated or associated state.

- Client sends probe request to discover the 802.11 networks nearby.
- The AP receives the probe request and check to see if the client has at least one common supported data rate. The a probe response is sent.
- A client sends a low-level 802.11 authentication frame to the AP.
- The AP receives the authentication frame and sends a response.
- Once a mobile station determines which AP it would like to associate to, it will send an association request to the AP.

- If the elements in the association request match the capabilities of the AP, the AP will respond with an association response with a success message.
- The client is not associated to the AP and data transfer can begin.

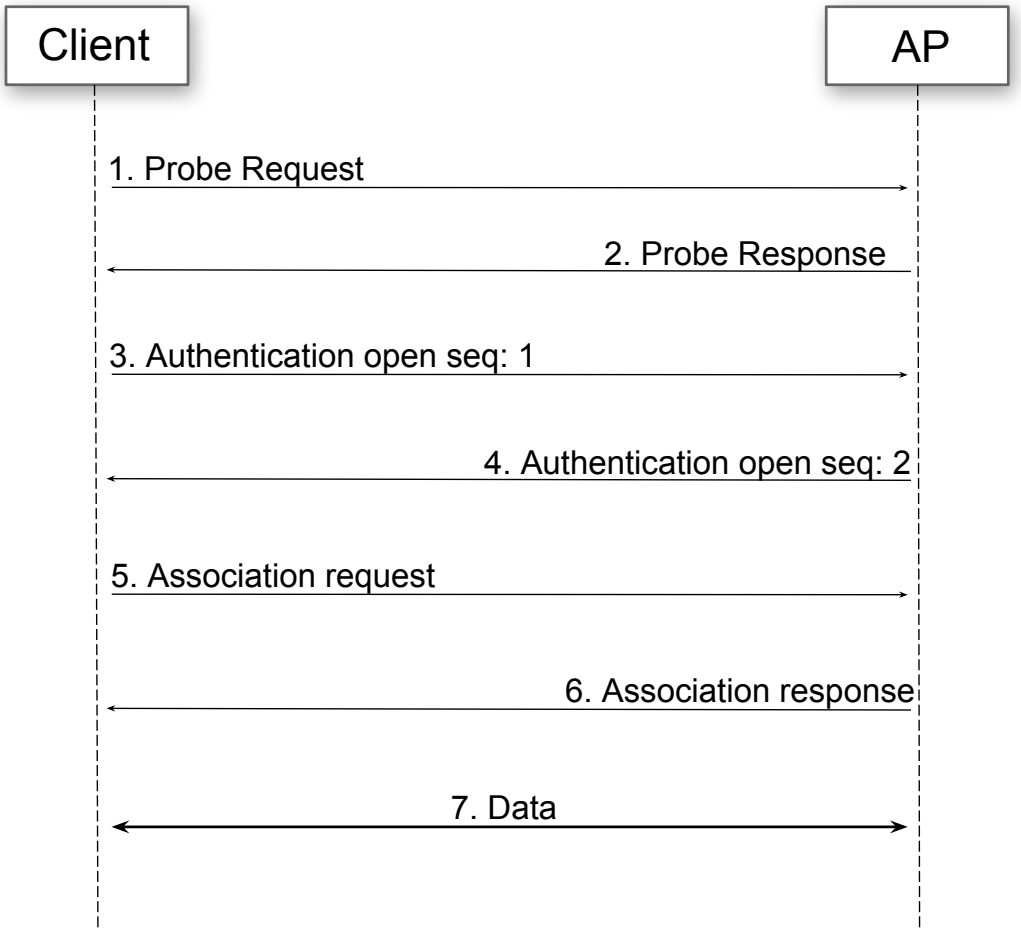

Fig. 1.: WiFi Association Process

The normal deauthentication process is shown in Figure 2.. The access point or clients can send deauthentication packets to each other (bidirectional).

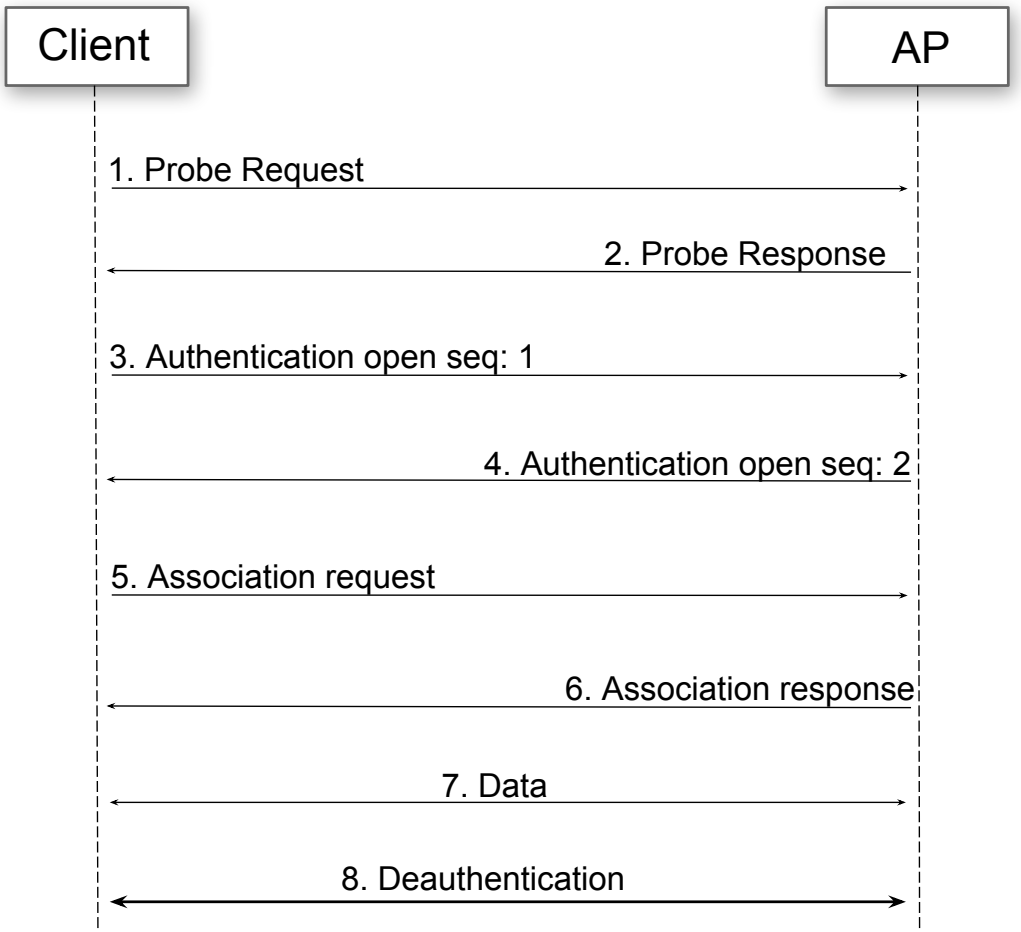

Fig. 2.: Normal Deauthentication Process

2.1.2 DE-AUTHENTICATION DOS ATTACK

Unlike data frames, deauthentication and disassociation frames are not protected by the cryptographic techniques (e.g., WPA/WPA2) thus are vulnerable to attack. Consequently, an attacker can send deauthentication frames with spoofed media access control (MAC) address on behalf of the AP to STA (or vice versa) and terminate data communication between STA and AP, as shown in Figure 3.. If STA receives spoofed deauthentication frames, it terminates communication with the AP, followed by scanning for available APs, and repeating the process of authentication and association [54]. Since it requires very little effort, an attacker can easily fabricate such spoofed frames at a high rate, and can inject such spoofed deauthentication/disassociation frames into a Wi-Fi network to affect a DoS attack.

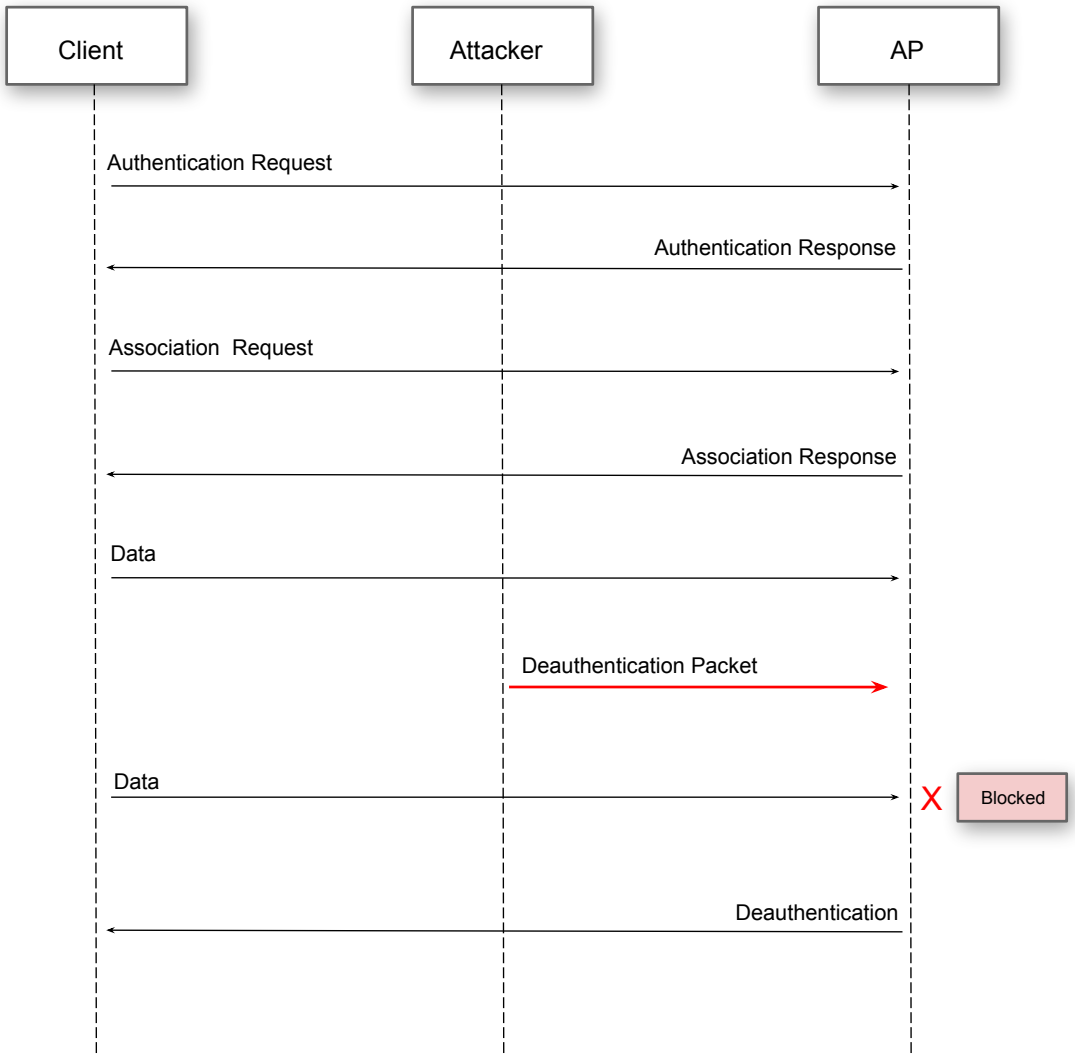

Fig. 3.: Deauthentication attack flow.

As shown in Figure 3., the attacker sends deauthentication frames with a spoofed MAC address on behalf of the AP to STA (or vice versa) and terminates data communication between STA and AP. If STA receives spoofed deauthentication frames, it terminates communication with the AP, followed by scanning for available APs, and repeating the process of authentication and association [54].

In addition to affecting a successful DoS attack, repeated authentication and re-association with the AP also provides means to steal the WPA/WPA2 passphrases of affected Wi-Fi nodes, thereby allowing it to:

1. Sniff the encrypted data frames, thereby compromising network confidentiality.

2. Inject malicious data frames to create man-in-the-middle (MITM) situation, thereby

compromising data integrity. Such an integrity compromise can lead to serious safety problems for systems that may be using WiFi networks to transporting sensor data or control commands.

Several methods can be used by an attacker who wants to launch deauthentication DoS attack. Common techniques to perform such attacks [21] are presented below:

- *Spoofed AP to client deauthentication frame:* By setting the source MAC address to the AP MAC address and the destination MAC address to the client MAC address, an attacker can create a frame that mimics a genuine frame directed from an AP to the client. This one assumes that the frame has been sent by the legitimate AP and is disconnected from the network once it performs treatment of the spoofed deauthentication frame.

- *Spoofed client to AP deauthentication frame:* The technique is the same as above. The only difference is that the source MAC address is set to the client MAC address while the destination MAC address is set to the AP MAC address.

- *Broadcast spoofed deauthentication frame:* In this case, the source MAC address is set by the attacker to the AP MAC address. The destination MAC address is FF:FF:FF:FF:FF:FF. All the clients are disconnected from the AP in this type of deauthentication DoS attack.

2.2 COMPUTER WORM ATTACKS

Computer worms are small self-duplicating code that can rapidly infect hundreds of thousands of systems. This causes a great increase in the amount of traffic and consumes computational power. The Code Red I and II worms exploited a bug in computers running Microsoft IIS web servers. The Code Red worm caused a severe service impact on several network systems by altering IP addresses [39]. In 2003, the Slammer worm spread to 90% of its target systems in just 10 minutes [83]. NotPetya and WannaCry created a global panic in 2017 [65]. A great deal of research based on different frameworks and numerous case studies are available for the analysis of the spread of worms across networks [42, 74, 80, 106]

Worms' detrimental influence on systems has been great over the last several decades. The targets are found using different methods and the worms propagate from one machine to the next. Once the worm infects the first node of a network, that node will attempt to infect the rest of the network. Therefore, it has become important to develop intrusion detection

systems and prevention mechanisms to counter them. Our main approach in providing a secure network is to detect and visualize worms by understanding the characteristics of the behavior of the worm.

The datasets analyzed in this work include the Sasser, Slammer, Petya, WannaCry, and EternalRocks works. We describe each of these.

2.2.1 SASSER WORM

The Sasser worm targets computers running Microsoft operating systems such as Windows XP and Windows 2000. It can spread rapidly over vulnerable computers via TCP port number 445 or 139 to infect other computers without human participation. This worm belongs to self-replicating worms from the W32.Sasser family [79]. The variants Sasser-A, Sasser-B, and Sasser-C were discovered within a short period of time and affected hundreds of thousands of computers worldwide. When attacking, the worm first determines the version of the remote operating system then uses the appropriate parameters to attack the host. For Windows XP and Windows 2000, the worm takes advantage of the universal exploit, lsasrv.dll and for Windows 2000 Advanced Server it uses the SP4 exploit. Operating systems such as Windows Me and NT are not impacted by this worm. The worm exploits a vulnerable LSASS.EXE [2] to infect the system, causing the attacker to gain full control of the system [20].

Figure 4. shows the state diagram of the Sasser worm attack. The attack begins with the worm performing port scanning on port 445. Upon finding an open port, the worm will attempt to exploit the unpatched LSASS.EXE file. In the third stage, the worm requests access to the shared directory X.X.X.X/ipc$, where X.X.X.X is the target's IP address. Then the worm attempts a buffer overflow by sending a file with a data length of 10 million along with its shell script. In the final stage, the worm starts an FTP service to the attacker's host port 8884 and downloads a backdoor malware, ssms.exe.

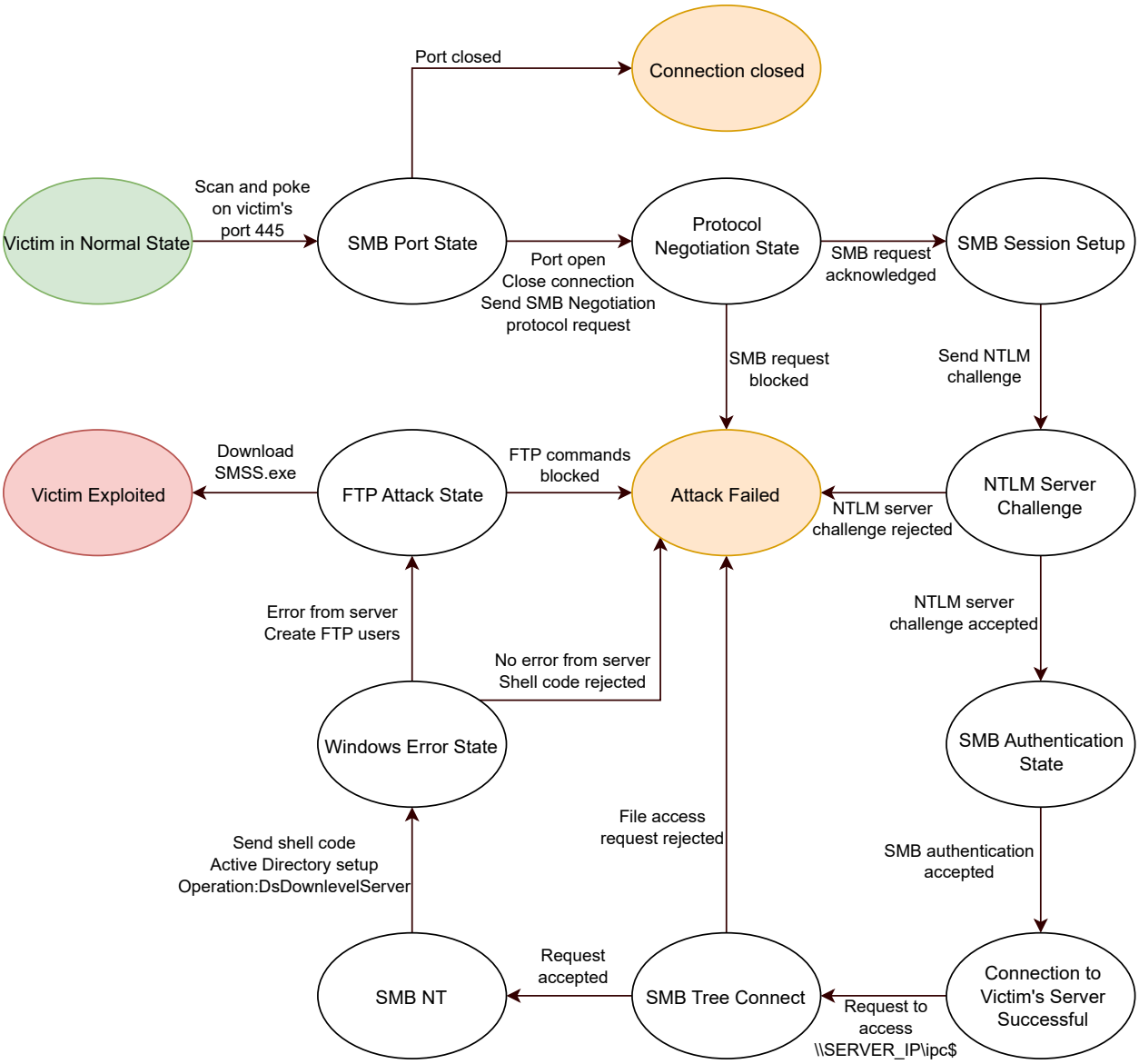

Fig. 4.: State diagram of Sasser worm attack.

2.2.2 SLAMMER WORM

The Slammer worm exploits a buffer overrun in Microsoft SQL Service 2000 [1] running on the Windows XP operating system. The worm's payload is small enough to fit in a single packet. It spreads very rapidly reaching SQL servers that are unpatched. It propagates by rapidly transmitting the same message to other SQL servers using random IP addresses. This attack can only be mitigated by taking the server offline. There is no additional malicious content included in the worm payload. However, because of the behavior and the speed with

which it attacks systems, it executes an effective DoS attack as the network's resources are drained [52]. Figure 5. shows the state diagram of the Slammer worm.

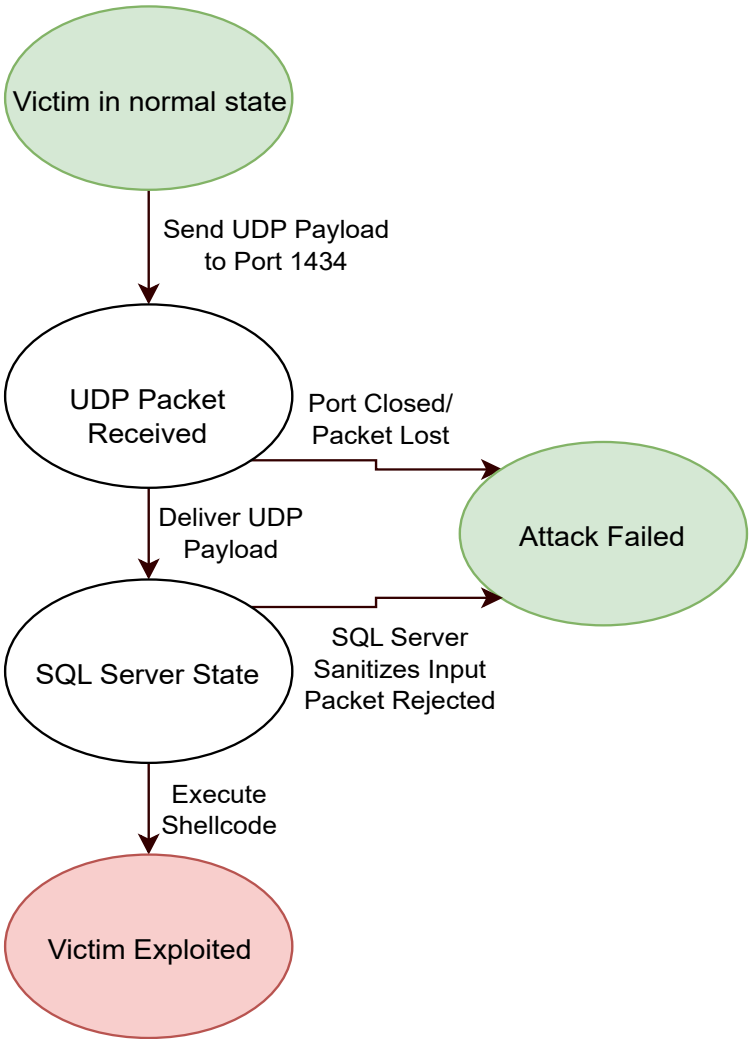

Fig. 5.: State diagram of Slammer worm attack.

2.2.3 ETERNALBLUE

The EternalBlue exploit [3] was leaked as part of the EternalRocks Shadow Brokers dataset [28]. The EternalBlue exploit targets all Windows versions since Windows 2000. The Server Message Block (SMB) protocol is an application layer network protocol, operating on TCP ports 139 and 445, and is used to provide shared file, printer, and serial port access as well as access to remote Windows services. EternalBlue exploits a flaw in the SMBv1 service's NT Transaction request handler by sending a large NT Transaction Request, making it necessary for the target to accept one or more Trans2 Secondary requests. By sending

malformed Trans2 Secondary requests, EternalBlue exploits a flaw in the protocol that allows shellcode to be delivered and executed on the target machine [65].

This exploit attack phases are shown in Figure 6.. The stages of attack start with a port scan looking for port 445 or 139 using the SMB protocol. Once the open port is discovered, the EternalBlue exploit is used to scan for the DoublePulsar backdoor. This exploit will infect systems that are unpatched. Next, the worm is delivered and starts to propagate. The Tor client, or a similar method, is launched to anonymize communications, and the worm begins encryption activity. This basic process is used by the EternalRocks, WannaCry, and Petya worms.

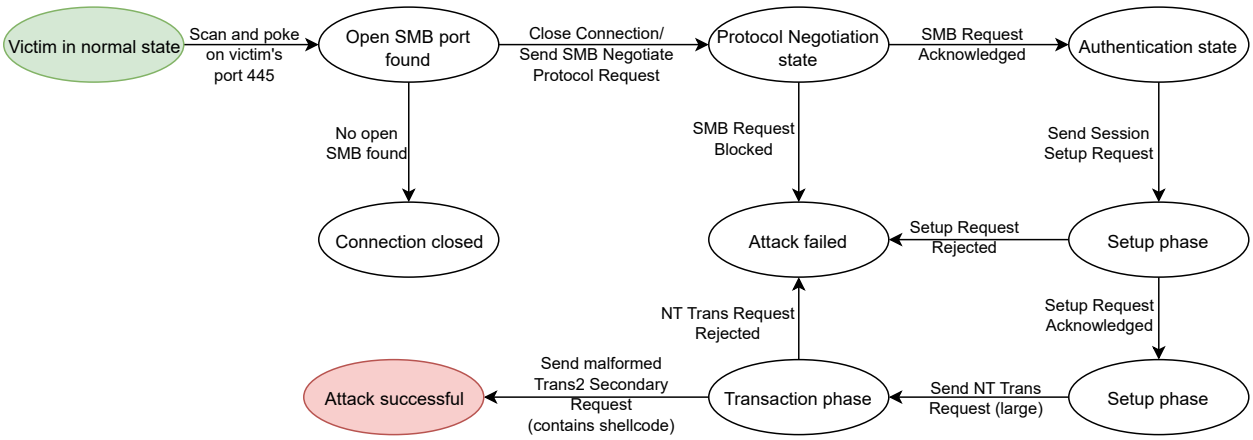

Fig. 6.: State diagram of EternalBlue exploit.

2.2.3.1 EternalRocks

In 2017, EternalRocks was an Internet worm that used EternalBlue [3] as well as six other NSA exploits leaked in the ShadowBrokers dataset. EternalRocks did not have a malicious payload and seems to have been intended only to draw users' attention to vulnerabilities on their computers.

2.2.3.2 WannaCry

Ransomware is malware which disables the functionality of a computer in some way, then displays a message demanding payment to restore the functionality. The WannaCry worm is a combination of ransomware and worm. The worm component used the EternalBlue [3] to gain access to target computers through a flaw in the SMB protocol. Once access was gained, the target computer was encrypted and a ransom demand displayed to the user [65]. WannaCry was deployed in 2017. The main attack was slowed within a few days of its

discovery due to emergency patches released by Microsoft, however unpatched computers continue to be infected.

2.2.3.3 Petya

The Petya worm is a ransomware worm similar to EternalRocks and WannaCry and was first discovered in 2016. It used the EternalBlue [3] exploit to gain access to target computers via a vulnerable SMB port. It relocates and encrypts the master boot record (MBR) of a victim computer, replacing it with a malicious MBR that shows the ransom demand [104].

An example of the Petya worm ransomware screen is shown in Figure 7..

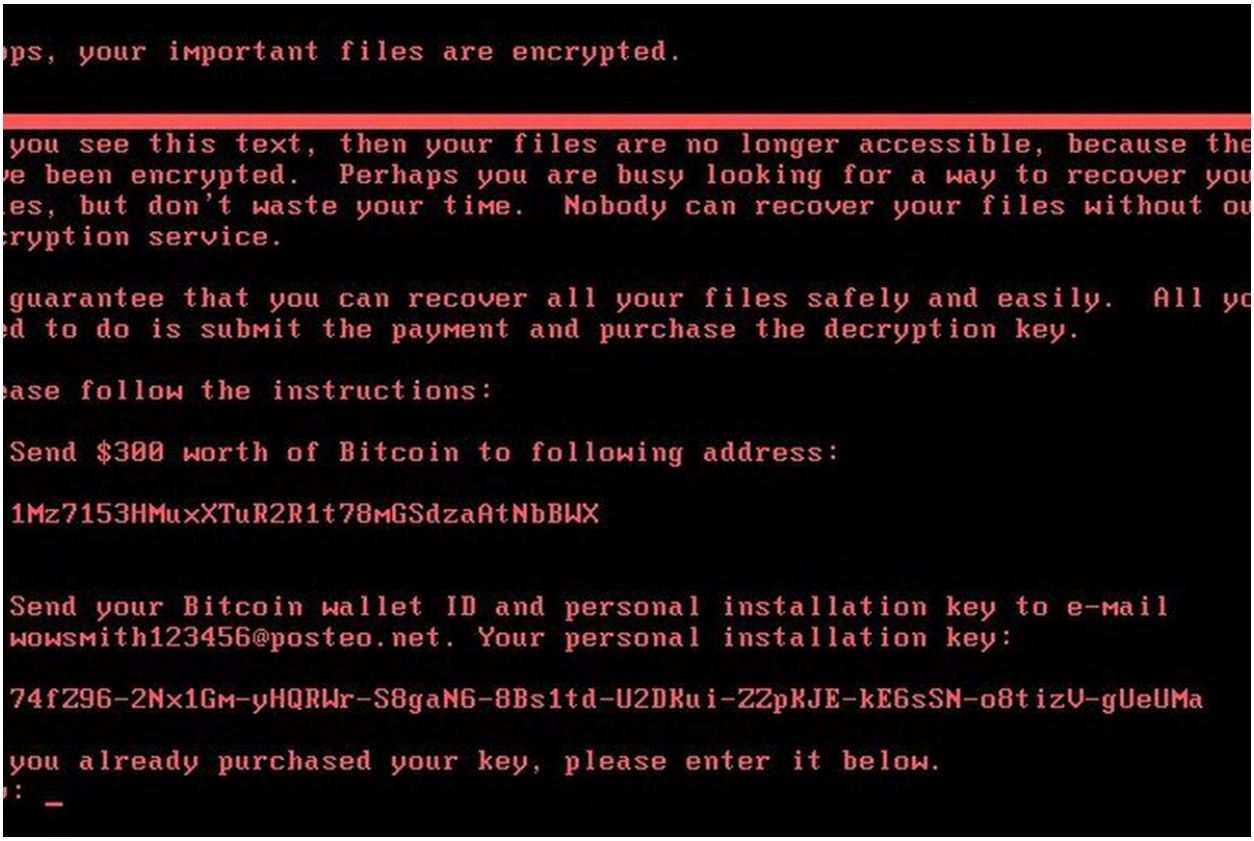

Fig. 7.: An example of the Petya worm ransomware screen.

2.3 MALWARE DATASET

2.3.1 KDD99 DATASET

In much of our evaluation, we use the KDD99 malware dataset. The KDD99 dataset is

from the 1998 Defense Advanced Research Projects agency (DARPA) Intrusion Evaluation under sponsorship of DARPA and the Air Force Research Laboratory (AFRL). The dataset is 743MB and classify malware by attack types such as: denial of service attack(DoS), buffer overflow, guess password, smurf DoS attack. MIT Lincoln Labs collected and distributed the datasets for evaluation of intrusion detection systems. This dataset has been used in many works for the malware classification problem [32, 61].

2.3.2 CTU-13 DATASET

The CTU-13 dataset is based on a large collection [13]. This dataset contains a mix of botnet, normal, and background traffic captured in the CTU University, Czech Republic, in 2011. There are a total of 13 captures called scenarios of different botnet samples, each scenario was executed with a specific malware, which used several protocols, and performed different actions. The total size of this dataset is 1.9GB.

2.3.3 OTHER MALWARE DATASET

Another malware dataset is used, malware samples [18] were selected. This dataset is 290MB in size and contains 120 malware such as sality, blackhole, and zeus.

2.4 MACHINE LEARNING CLASSIFICATION METHODS

Classification is used to identify patterns in a dataset by classifying data according to their attributes. To classify programs as malicious or benign, it is necessary to identify the properties of each program. For example, malware classification depends on several attributes like replication strategy, goal of creation, technique of propagation, etc.

Machine learning techniques are often used for classification. Machine learning makes use of both supervised and unsupervised learning. Supervised learning is the process of learning from the training dataset and making use of input variables and given output variables that you then use to learn the mapping function from the input to the output. We approximate the mapping function so that when we have new input data, we can predict the output variables for that data. Unsupervised learning models the underlying or hidden structure, also called distribution, in the data to learn more about the data. In unsupervised learning, only input data is used and no corresponding output variables are used. In this section, we provide a background for well-known supervised learning techniques.

2.4.1 NAIVE BAYES

Naive Bayes is a collection of classification algorithms based on Bayes Theorem which describes that probability of an event based on prior knowledge of conditions related to the event. It is ideal for two or more classes because of the assumption of independence between every pair of features, and it is highly scalable. The probability of each feature contained in the data is independently treated in this type of machine learning. Independent attributes are specified for each feature and classes are identified by which attributes belong to it. When a new instance arises, the attributes of this instance is compared to those of each feature and a decision is made.

2.4.2 RANDOM FOREST

Random Forest makes use of decision trees. A decision tree maps possible outcomes of a series of related choices and weighs possible actions against one another based on their costs, probabilities, and benefits and is used to map out an algorithm that predicts the best choice mathematically. Random Forest is a supervised machine learning algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression tasks. The basic concept is to build independent subsets of the dataset to be used in multiple decision trees. These collections of decision trees form a Random Forest and produce better prediction accuracy [41].

2.4.3 SUPPORT VECTOR MACHINE (SVM)

A Support vector machine (SVM) [50] is a supervised machine learning algorithm generally used for classification problems. It is very effective in high dimensional spaces. It creates a boundary by determining a two-dimensional boundary, a line, through the space and creates classes 0 and 1, those above or below this line. This line constitutes a decision boundary as follows:

- If the value calculated is greater than 0, the point is above the line, class 0.
- If the value calculated is less than 0, the point is below the line, class 1.

SVMs can use kernels in order to apply linear classification methods to non-linear data. Kernel equations may be linear, quadratic, Gaussian, etc. We can use any function that transforms the linearly non-separable data into a linearly separable one.

2.4.4 DEEP NEURAL NETWORKS

Machine Learning (ML) is a process where the association of events with consequences is performed. Deep learning refers to a class of ML techniques utilizing many layers of information processing for pattern classification. Neural networks takes input and produces a given output, achieved through the training process. A deep neural network (DNN) is meant to solve complex problems with the use of computers and have beginnings in how the human brain works. It represents one of the most efficient ways to solve problems and is classified as deep learning. A DNN processes input information and aggregates this data into key learning points. This learning approach can be done in three ways, namely, unsupervised, supervised, and semi-supervised [49]. In this work a DNN is implemented to understand malware.

2.5 DIMENSIONALITY REDUCTION METHODS

High-dimensional data containing images, videos, text, and other complex information must be reduced in dimensionality in the preprocessing step of machine learning in order for us to avoid many issues including over-fitting, high complexity, and over complicated machine learning models. In addition, we must reduce the dimensionality to increase performance while trying to maintain accuracy. We employed several methods that each can reduce the complexity of high-dimensional data before feeding it to the DNN classifier. In our tests, we evaluated four different dimensionality reduction approaches.

2.5.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (PCA) is a tool used to reduce the dimensionality of large datasets to a smaller set that still contains most of the information. PCA is a method for incorporating a new set of unconnected attributes referred to as Principal Components (PCs) from the attributes of a dataset. PCA reduces the dataset on a dimension basis, while keeping the dataset variability at a maximum.

2.5.2 SINGULAR VALUE DECOMPOSITION

The Singular-Value Decomposition (SVD) is a matrix decomposition method used to reduce a matrix to its constituent parts in order to make matrix calculations simpler. SVD is a special technique of factorization that can be applied to any real or complex valued matrix. SVD is an algorithm of dimensionality reduction from the scikit-learn library [90], which is an implementation of randomized algorithm by Halko [59].

2.5.3 NEURAL NETWORK AUTOENCODER

The neural network autoencoder is a type of DNN that can be used for dimensionality reduction of the input data [70]. Autoencoding gives successful results in a wide variety of applications such as document and image retrieval [35, 70]. It applies back-propagation, which is setting the target values to be equal to the inputs. It compresses the input into a latent-space representation, and then reconstructs the output from this representation.

2.5.4 T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING

t-Distributed Stochastic Neighbor Embedding (t-SNE) [27] is a procedure for dimensionality reduction that is perfectly adapted for embedding high-dimensional data into a space of two or three dimensions. More precisely, t-SNE maps each high-dimensional entity into a two or three dimensional point such that nearby points represent similar objects while dissimilar objects are described by distant points. This property becomes crucial when dealing with high-dimensional data lying on multiple low-dimensional manifold, which are related, e.g., images of objects coming from different classes and observed from several viewpoints. It has been used in a wide range of applications, including computer security research, music analysis, cancer research, bioinformatics, and biomedical signal processing.

2.6 SUMMARY

There are two main types of cybersecurity attacks, exploiting internet protocol vulnerabilities and the use of malicious software. In this chapter we explain the process of normal authentication and we discussed the deauthentication attack process. In addition, we showed the different datasets we used to build our three intrusion detection systems (IDS). Furthermore, we mentioned the four different machine learning classifiers that we compared in order to chose the best one for building the IDS, in terms of accuracy and performance. Finally, we defined four feature reduction methods to increase the performance of the IDS.

CHAPTER 3

RELATED WORK

This chapter reviews relevant methods suggested in literature to address deauthentication attacks, worm traffic analysis, and malware classification and detection.

3.1 INTRUSION DETECTION SYSTEM FOR DEAUTHENTICATION ATTACKS

Several methods have been proposed to mitigate the deauthentication DoS attack.

3.1.1 CRYPTOGRAPHIC METHODS

According to Nguyen et al. [88], a secret key can be established between a client and the AP for authentication of deauthentication frames using a letter-envelope protocol. The letter-envelope protocol generates two random primes, computes the product, sends the value during initial connection, and gives it again during the de-authentication request. The authors argue that this method can prevent deauthentication DoS attacks, but all client and AP firmware must be upgraded.

Bellardo and Savage [38] suggest that authentication of all management frames would encrypt the MAC address and therefore the attacker would not be able to spoof the MAC address and could not carry out a DoS attack. Clients that do not upgrade their framework will be not able to connect.

3.1.2 UPGRADING AND MODIFYING PROTOCOLS

Another method is to upgrade all nodes to the IEEE 802.11w standard [8]. IEEE 802.11w forces authentication of management frames to be mandatory. Spoofing is prevented by authentication, therefore, the deauthentication DoS attack is also prevented, but upgrading to the 802.11w standard requires a firmware upgrade. In addition, this standard has a low rate of adoption due to how recent the standard is, and therefore very few APs and clients support it.

3.1.3 NON-CRYPTOGRAPHIC METHODS

Agarwal et al. [22] carried out an experiment to prevent deauthentication DoS without using cryptographic methods. They set a threshold on a given number of deauthentication frames that were received by a client. For every client, if the number of deauthentication frames exceeded the set threshold, a deauthentication DoS attack alarm was raised.

Bellardo et al. [38] suggest eliminating the deauthentication and disassociation frames, or delaying them for a fixed interval of time (for instance, 10 seconds). If the client sends a data frame within that delay period, the AP ignores the deauthentication frame because clients are required to go back through the 4-way handshake process before sending data frames and therefore it is assumed the deauthentication frame was not valid. However, this approach may introduce problems where there is a period of time where a client associates with multiple APs at the same time. This can cause problems with the routing and handoff processses [33].

3.1.4 REVERSE ADDRESS RESOLUTION PROTOCOL (RARP)

Cardenas et al. [46] proposed using RARP (Reverse Address Resolution Protocol) in order to detect spoofed frames. RARP maps a single MAC address to a single IP address for each RARP returns multiple IP addresses for the same node, then the MAC address was possibly spoofed and should be investigated further. This method can be circumvented by attackers who spoof the IP address of the client. In addition, this method does not work for clients who have multiple IP addresses assigned to the same network interface card [76].

3.1.5 CENTRAL MANAGER SERVER

Another approach, proposed by Ding et al. [51], is to use a Central Manager (CM), which is an authentication server that manages one or more APs and instructs the AP to either process the management message or ignore it. This approach only protects the AP and not clients. In addition, the CM server adds additional delay to the network.

3.1.6 LIGHT WEIGHT AUTHENTICATION PROTOCOL

Johnson et al. [66] proposed the Light Weight Authentication Protocol in which a one-bit authentication protocol is used. Wang et al. [107] added additional improvements. This approach uses a keyed pseudo random bit stream. One bit is added for authentication on each frame from this bit stream. The idea is that only the key holders know what the bit stream is and therefore this ensures authentication. However, due to poor wireless reception or noise, this data can be corrupted and lead to frame and synchronization loss. In addition,

an attacker has a 50% chance of guessing the correct authentication bit. Increasing the number of bits used for the authentication will decrease this probability, but will increase redundancy.

3.1.7 DISASSOCIATION

More et al. [84] used a different approach, where disassociation is validated by restarting the 4-way handshake once a disassociation request is received. Normally, if the client sends a disassociation frame to the AP, the client will not respond to the 4-way handshake from the AP. If the 4-way handshake is completed successfully then this indicates the disassociation frame was spoofed. If the frame was not spoofed then the handshake fails and disassociation takes place after a timeout. This approach has a couple of problems. In many cases spoofing is used to change the assigned MAC address for many purposes such as hiding a computer or allowing it to impersonare another network device; this is done for both legitimate and illicit purposes. The delay in legitimate spoofing can cause handoff issues, and an attacker can use this periodic forcing of the 4-way handshake as a DoS attack.

Another approach proposed by Hal et al. [60] is using the transient portion of the signal when a wireless tranceiver first turns on and generating a fingerprint using this information. This fingerprint is then tied to the MAC address and therefore can be used in anomaly based intrusion detection. However, the detection system would then need to maintain a database of fingerprints and MAC addresses for each node on the wireless network, which may prove to be a difficult task in mobile networks.

Another approach is Relationship-based Detection [47]. The traffic that a particular network node generates is unique to that node. If there is an attempt to spoof, this pattern will change. This approach can be combined with the sequence number approach, described in Section 3.1.8, in an IDS, but the solution needs a separate device for monitoring the intrusion. Additionally, it can only detect the intrusion, but is not able to prevent the attack.

Another possibility is to disable disassociation by asking the AP and clients to ignore disassociation requests altogether. In this case, the association will just simply timeout and the AP and client will transition to the disassociated state at that time. The problem with this is that when a client, and in particular mobile client, wants to associate to a new AP, the old AP will still associate to the client and now the client will be associated to both APs. From the AP perspective, this may cause routing and handoff problems [33].

3.1.8 SEQUENCE NUMBER

Guo et al. [58] proposed an approach where the sequence number field in the link-layer header of IEEE 802.11 frames is leveraged. In this work there are two general approaches. The first is to modify the wireless LAN interface drivers on both the APs and clients for sequence number analysis. The other way is to use a monitoring station that is separate from the APs and clients. By analyizing the sequence number in the wireless frames, spoofing of the packets can be detected.

Xia et al. [112] proposed a different approach by using MAC sequence number tracking and observing that IEEE 802.11 frames contain a 12-bit sequence number that is expected to increment by one for each datagram being sent. These sequence numbers will remain the same though in cases of packet retransmission. Typically only hardware can set these sequence numbers because of the timing involved and software usually is not used for that purpose.

One drawback to using the sequence number approach is if the sequence numbers are assigned in a deterministic way, an attacker may be able to sniff the frames sent by the client and use this information to predict the sequence number of the next frame.

3.1.9 OUR PROPOSED APPROACH

All of these proposed methods for prevention require modification of the node drivers. No detection mechanism is available where node drivers are not modified, leaving such nodes unprotected from attacks. In addition, these unmodified nodes will not be able to connect with the modified system. In contrast, we propose a system that allows unprotected clients to stay connected. More importantly, the IDS does not require the nodes to be modified. This makes it backward-compatible and easy to roll out gradually. Finally, if the drivers are modified we propose two prevention mechanisms, one using one-time passwords and the other using machine learning.

3.2 INTRUSION DETECTION SYSTEMS FOR MALWARE AND WORMS

In the following section, we review several methods that have been proposed to detect malware attacks.

3.2.1 DEEP LEARNING APPLICATIONS ON IDS

Recently researchers have used deep learning models to detect and classify malware [105, 109, 114]. Tao et al. [103] proposed using a supervised Fisher and Deep Auto-Encoder to extract important features using the KDD99 dataset, which was described in Chapter 2. Kim et al. [71] proposed using the Long-Short-Term-Memory algorithm in conjunction with the Gradient Descent Optimization for an intrusion detection classifier and was able to achieve an accuracy of 97.54% and a recall of 98.95%. In addition, this team was able to use the Gated Recurrent Unit (GRU) and achieved a 97.06% recall, 10.01% false alarm rate, and a 97.06% accuracy rate. Furthermore, Staudemeyer et al. [100] proposed using LSTM networks on the same KDD99 IDS dataset and was able to achieve an accuracy of 93.82% but at the cost of taking longer to train. Gavrilut et al. [57] proposed a detection system using modified perception algorithms. Some of these algorithms achieved an accuracy of 69.90% - 96.18% depending on the algorithm used. However, the algorithms that had the best accuracy also resulted in the highest number of false positives. The algorithm with the most balanaced approach reducing the number of false positives had an accuracy rate of 93.01%. Singhal et al. [97] proposed a detection method that uses the modified Random Forest algorithm in combination with the Information Gain algorithm to gain a better feature representation. The dataset in this work contained only portable executable files and therefore made feature extraction generally easier. This work achieved an accuracy of 97% and 0.03% false positives.

3.2.2 BEHAVIOR-BASED DETECTION

Several recent works have analyzed network behavior for worm detection and classification. Sarnsuwan et al. [94] use network-based Internet worm detection utilizing thirteen packet features. Data mining algorithms such as Bayesian network, decision tree, and random forest are used for classification of Internet worms, normal data, or network attack data. Barhoom et al. [36] propose a model that makes use of data mining techniques by combining different classifiers with an adaptive approach for detecting known or unknown worms. Another technique proposed by Rasheed et al. [92] focuses on detecting polymorphic worms. However, only a single worm, MSblaster, was used for testing. Tang et al. [102] proposed an IDS using a Deep Neural Network (DNN) model. A visualization of the computer system state during a worm attack requiring manual intervention by the user is presented by Axelsson et al. [34]. It was observed that the worm request clusters were noticeably different from the clusters formed by normal traffic.

3.2.3 FEATURE SELECTION

Selecting the optimal features can improve the performance of a ML classifier without compromising accuracy.

Mitra et al. [81] proposed a feature selection method that selects only a subset of features to remove redundant features.

Amiri et al. [30] proposed a modified-mutual information feature selection (MMIFS) based on the forward feature selection algorithm (FFSA) and linear correlation coefficient. They performed experiments with the KDD99 dataset [5], and showed that their approach effectively detects R2L (an attempt to gain unauthorized access from a remote machine such as guessing the users password), and probe attacks, but performs less efficiently in case of DoS and U2R (an attempt to gain unauthorized access to local root privileges).

Two feature selection approaches called Random Forest-Backward Elimination Ranking (RFBER) and Random Forest Forward Selection Ranking (RFFSR) were proposed by Al-Jarrah et al. [23]. The RFBER approach eliminates lower weight features and checks the effects on detection rate, while RFFSR forms one subset with three of the most weighted features and the rest of features in other subsets. The detection rate was evaluated by adding features one by one. Similarly to previous approaches, these approaches are evaluated using the KDD99 dataset.

Battiti et al. [37] proposed a mutual information-based feature selection algorithm MIFS that selects a subset of features in order to remove redundant features. The algorithm uses both the feature-feature and the feature-class mutual information for the selection of features. In addition, it uses the greedy technique in order for the features that maximize information to be selected.

Eesa et al. [53] presented a novel feature selection technique, based on the cuttlefish optimization algorithm (CFA). In this work, the authors try to identify the optimal subset of features using CFA and used decision tree for the selection of features. Again, the KDD99 dataset is used for evaluation of detection rate (DR), false positive rate (FPR) and accuracy rate (AR). Experimental results show that the value of DR, FPR and AR improved with fewer number of features as compared to using a higher number of features.

In order to select more relevant features and eliminate irrelevant features, Karim et al. [67] proposed a hybrid selection technique, relying on symmetrical uncertainty filtering and information gain methods. In the first phase, information gain and symmetrical uncertainty filtering are used for creating two subsets of reliable features. Then, these two subsets are merged, ranked and weighted to extract the most important features. This approach achieves a better detection rate compared to other feature selection approaches.

Mukherjee et at. [86] proposed one of the first works to tackle feature reduction methods. They developed a feature vitality based reduction method (FVBRM) and compared it with the Gain Ratio (GR), Correlation-based Feature Selection (CFS), and Information Gain (IG) feature selector. The authors used WEKA 3.6 machine learning tools for experimental analysis and achieved a significant improvement in detection rate and accuracy in case of R2L attacks.

Lin et al. [75] proposed a novel feature representation technique called cluster center and nearest neighbor (CANN), which uses the k-nearest neighbor (k-NN) classifier for intrusion detection. CANN first converts the original features of a dataset into one-dimensional distance, and then the K-NN classifier uses this new dataset representation for training and testing for classification. Based on experimental results with the KDD99 dataset, they show that CANN works efficiently compared to support vector machine (SVM) and other similar k-NN classifiers. It is shown that it has higher detection and accuracy rates, and a lower false alarm rate. The pattern of two types of attacks: an attempt to gain unauthorized access from a remote machine, and an attempt to gain unauthorized access to local root privileges. These attacks are not detected by CANN effectively.

Al-Yaseen et al. [26] proposed a multi level hybrid IDS using support vector machine and extreme learning machine. This approach reduces the training dataset using k-means algorithm which reduces the training time of the classifier. Result-wise, this approach also achieved a higher accuracy rate and a lower false alarm rate. However, the classification of a totally new and unknown attack is not covered in this study.

In Sabry et al. [53], optical subset features are searched by a modified version of the cuttlefish algorithm and tested using a decision tree. It works with 20 or fewer features and gives better detection rate (DR) and accuracy rate (AR).

The review of the work mentioned above indicates that selecting an optimal number of features helps to decrease training time and improve detection rate and accuracy. However, finding the optimal features is still an open issue for various applications in a generic scenario with known and unknown attacks.

3.3 SUMMARY

Many researchers proposed methods to prevent deauthentication denial of service (DoS) attacks. All of the proposed methods require driver or hardware modification to the node, and the unmodified nodes will no longer be able to connect to modified systems.

Regarding malware, many researchers proposed signature based and behavior based detection mechanisms using machine learning. The use of machine learning introduces performance issues requiring the use of high performance CPUs. Signature based methods are not able to detect malware that is not in the database. In addition, machine learning alone is not able to trace an attack and therefore see the overall big picture.

Finally, many researchers proposed several tools to both select the optimal features and compress them. Most of these tools are general tools and can be used on any dataset.

CHAPTER 4

INTRUSION DETECTION SYSTEM FOR DEAUTHENTICATION ATTACKS

WiFi provides two distinct logical wireless links: a data link and a control and management (CM) link. The data link is protected by encryption, however the CM link is not and this leaves it vulnerable to deauthentication DoS attacks. We propose a detection mechanism using machine learning that does not require driver modification and propose two prevention mechanisms that do require driver modification.

4.1 ARCHITECTURE

The IDS consists of two main parts— a packet sniffer and a data extractor, together with an machine learning (ML) classifier to make a decision about the presence of an attack. First, packets are captured with corresponding hardware in monitor mode then are analyzed to extract necessary data features. The extracted data is used as an input to the second part of the project, a trained ML-based classifier, and its output is used for the determination of attack probability and prevention. Such an architecture allows the division of the project into a few independent modules, which can be combined to obtain a complex solution.

4.1.1 DNN ARCHITECTURE

The DNN model uses the classical fully-connected multilayer classifier structure with rectified linear activation [87]. Activation of the output layer is selected to be sigmoid, widely used for the binary classification problem. The precise structure is shown in Table 1.. The training was performed with the help of the Adam optimizer [68] with parameters set to the proposed values in the original work. The model was trained until convergence was reached, which was determined by the difference in loss between two consecutive epochs to be less than 0.0001. The logarithmic loss function, also known as binary crossentropy was utilized.

TABLE 1.: DNN architecture.

Layer	Number of units	Activation
Fully-connected	10	Relu
Fully-connected	10	Relu
Fully-connected	10	Relu
Fully-connected	10	Relu
Fully-connected	1	Sigmoid

4.1.2 DATA PREPARATION

ML-based instrumentation, first and foremost, needs large amounts of data to be trained and is very sensitive to data choice and its quality. In that case, in order to optimize the results, data was gathered *in situ* using the aircrack-ng utility run on a laptop with Kali Linux, while another laptop was working as an AP. Four clients were present on the network— two smartphones from different brands, one tablet and one laptop. All clients were on the same network segment, as shown in Figure 8..

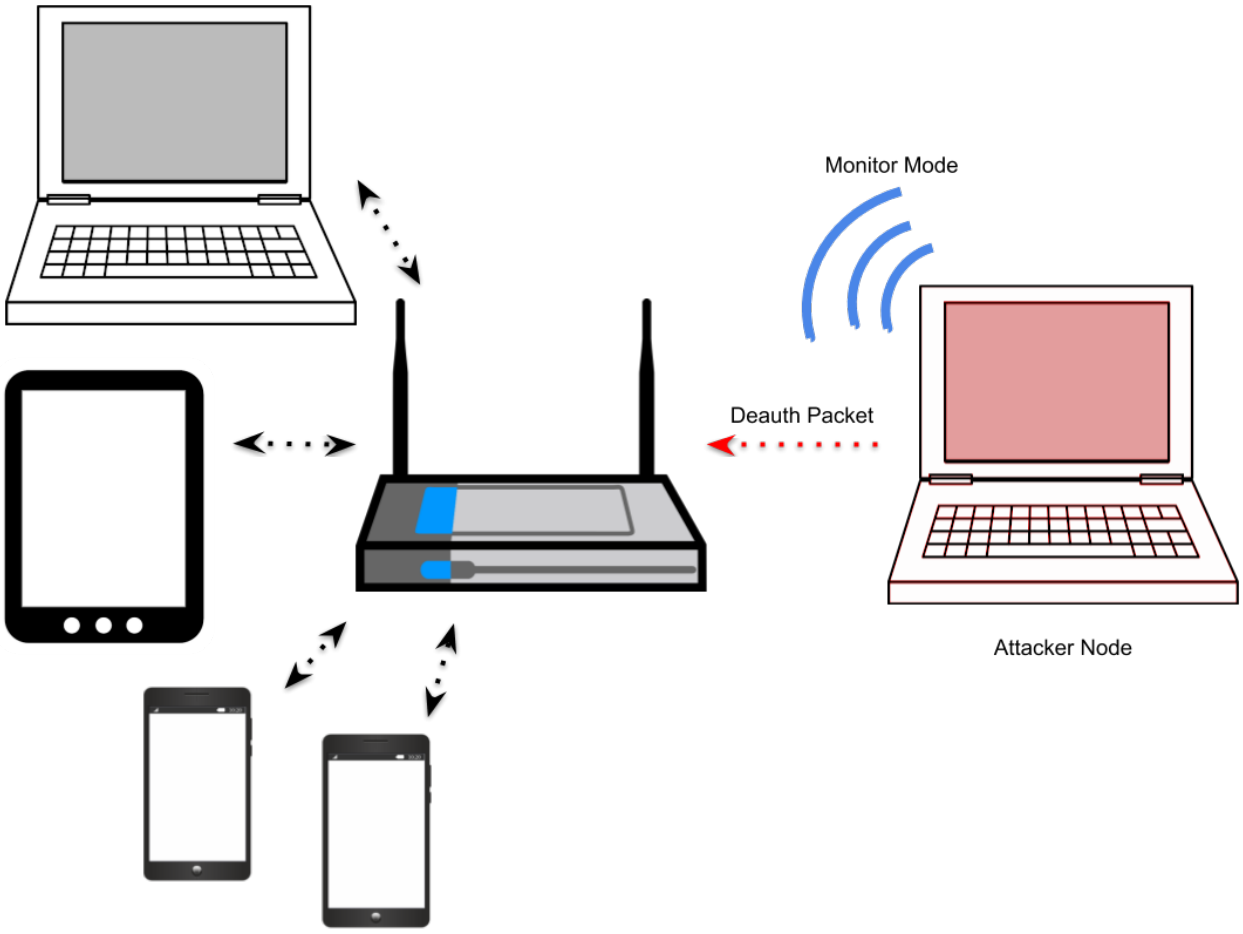

Fig. 8.: Setup to capture traffic for dataset generation.

We captured 13 hours of traffic that included normal traffic and normal deauthentication packets. An attack was initiated from another laptop with its internal network card running in monitor mode. For the attack, the aireplay-ng tool from the above-mentioned aircrack-ng utility set was used, as well as the scapy-deauth [4] script, which uses the Python scapy library to generate deauthentication packets. The attacker laptop was not associated with the network. The MAC addresses of the AP and clients were obtained using airodump-ng utility from the same utility set. The airodump-ng program was also used to capture data with its -w command line parameter.

The dataset consists of eight different traffic samples. We captured normal traffic with legitimate deauthentication packets and multiple different mixtures of normal and deauthentication attack traffic on a local network.

4.1.3 FEATURE EXTRACTION

With the help of the Wireshark tool, we analyzed the gathered data and, after comparing to recent literature [21], a few features were selected to be used as criteria for the detection of deauthentication DoS attacks. A list of these features, together with the motivations for our choice, is provided below:

1. *Number of deauthentication frames.* During normal network work, the number of deauthentication frames sent by both sides is very low, usually not more than 1 per few minutes. During an attack, to make it effective, deauthentication frames are sent in large amounts, up to thousands per second. The probability that a network is undergoing a deauth-DoS attack increases with this number.

2. *Number of association packets.* These packets are markers of trials to connect to an AP. During an attack, this number increases with respect to its value under normal conditions because disconnected devices are sending numerous association frames trying to reconnect to the AP and being disconnected.

3. *Number of authentication frames.* Similar to association packets.

4. *Number of data frames.* During an attack, the amount of effective data frames exchanged between clients and the AP decreases because the length of the session is short. This is due to frequent disconnections induced by the deauthentication DoS attack.

5. *Time between de-authentication and re-authentication.* In typical networks, this time difference varies from minutes to hours, while under attack it significantly decreases to fractions of a second. This observation comes from the fact that a genuine client will rarely reconnect immediately to an AP after it is disconnected from the same AP, while it is always the case under a deauthentication DoS attack.

4.1.4 TRAINING OF CLASSIFIERS

Two machine learning classifiers are used to train the model, a decision tree classifier and a deep neural network (DNN). The decision tree classifier used is a part of the scikit-learn project, a Python library for machine learning [90]. The classifier uses the CART algorithm [45], which is similar to the C4.5 [91] algorithm.

The DNN classifier was trained on the same dataset as the decision tree, and its output was used in the same manner to analyze network activity. We captured 13 hours of traffic

that contains both normal deauthentication packets and traffic that has deauthentication packets from an attack.

4.1.5 PREVENTION USING MACHINE LEARNING

Once an attack is detected by the system, packets are sent out to the protected client nodes telling them to drop all deauthentication packets for a period of time. If another attack is detected, additional packets are sent indicating the clients should increase the period duration they should continue dropping deauthentication packets. This method helps deter an attack and the clients will remain connected to the network.

4.1.6 PREVENTION USING ONE-TIME PASSWORD

To offer protection against other attacks as well, such as disassociation attacks, we propose a technique that uses a one-time password (OTP) within a control-management (control and management (CM)) frame which is mutually exclusive to the ML prevention mechanism. The OTP is based on two premises: password expiration and random generation. There is only one attempt to give the password within a limited time frame and since the password is pseudo-randomly generated, it cannot be guessed. If the correct OTP is not provided when requested, a legitimate deauthentication packet is sent, as shown in Figure 9..

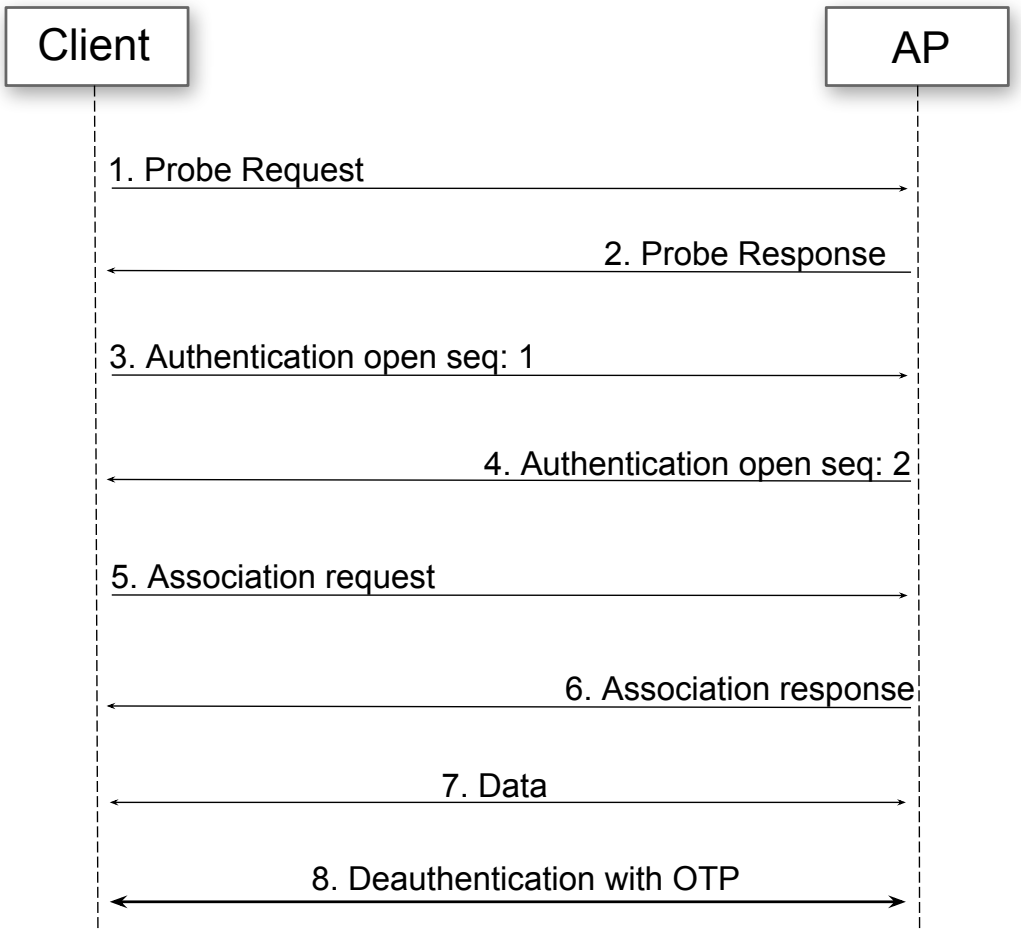

Fig. 9.: Demonstration of legitimate deauthentication

4.1.6.1 Implementation of OTP security in 802.11

The CM frame of a deauthentication packet in the 802.1X protocol includes a reason code as shown in Figure 10.. We use reason code 67, an unreserved code, to indicate that we are going to add an OTP to the packet. The packet is lengthened by 17 bytes to incorporate a 128-bit OTP, plus a null terminator. By using this special reason code as well as OTPs that are known to only the client and AP, the AP can reject the illegitimate deauthentication packets.

This proposed augmentation involves modifying the existing drivers for Wi-Fi clients as well as the software that hosts the Wi-Fi AP. We provide a C implementation of the Wi-Fi drivers for Linux, adjusted to accept these modified packets and verify them against OTP, as well as a C++ modification of hostapd, Linux software for hosting an AP.

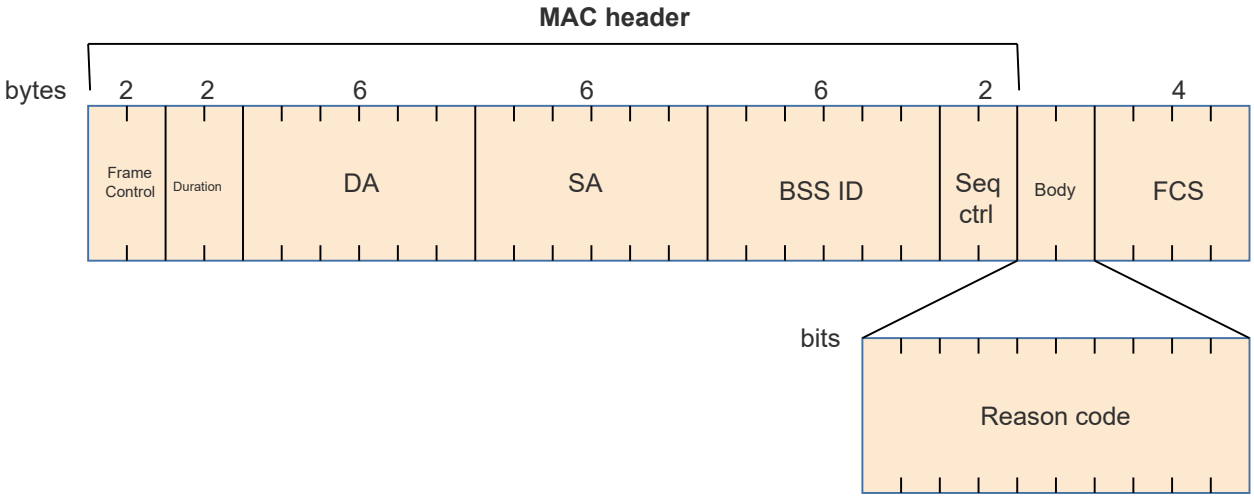

Fig. 10.: Format of deauthentication control and management frame

4.1.6.2 OTP Generation

OTP generation is done via a Sandwich-chain [64]. After two secrets are exchanged between AP and STA as shown in Figure 9., they can independently generate as many OTPs as they need. A hash chain is implemented by taking a number of hashes of the original secret. A Sandwich-chain uses a secondary chain, a mechanism by which a step-up chain can be used as a salt for the secondary chain; this offers the advantage of an efficient authentication without re-computing previous secondary chains. An adversary will not be able to predict the next OTP because a cryptographically secure hash cannot be reverse engineered. The system implements the hash chain generation. The program expects a MAC address from the client and the count number of the OTP that it should reply with for that client. The OTP of the proper count is written back to the FIFO file. A FIFO file is a special file similar to a pipe, but it has to be opened at both ends simultaneously before you can proceed to do any input or output operations on it. The AP and STA send each other the OTPs in their Wi-Fi deauthentication packet. They also use the OTP generation program to check that an OTP was the correct one that it was expecting.

There are two hash functions, one for the string and the other for integers that are used within the string hash function. The second hash function uses several hexadecimal number operations such as addition, subtraction, XOR, and bit-shift, to accurately and sufficiently shuffle the resulting numbers.

The pseudo-code for the AP's actions when receiving a deauthentication packet from the

STA is shown in Algorithm 1. The AP code keeps track of all STAs that connect and have the OTP modifications enabled during its run-time. A linked list is initialized during the configuration of basic service set (BSS). Entries in the list include the MAC address and the count number for the OTP. This new linked list data structure was added in order to persist tracking of the connecting clients, since all other client information is destroyed on disconnection of the client from the network.

Algorithm 1: Legitimate deauthentication

```
RECEIVE packet from STA
if reason_code = 67 then
    RETRIEVE OTP from packet by reading 16 bytes
    OPEN FIFO file and request OTP from C++ binary
    if binary OTP equals packet OTP then
        ACCEPT deauthentication packet
    else
        IGNORE packet
    end
end
```

4.2 EVALUATION

In this section, we evaluate our proposed intrusion detection system which detects deauthentication attacks, using machine learning (Section 4.1.4). We conducted 5 experiments each approximately 35 minutes in duration.

- Experiment 1 is broadcast client deauthentication, where packets were sent using FF:FF:FF:FF:FF:FF as the destination address, so all clients were disconnected from the AP at once.

- Experiment 2 is attack on AP, where packets, masquerading as deauthentication frames from connected clients, were sent to the AP.

- Experiment 3 is broadcast deauthentication with pause, which is the same as broadcast deauth, but attack packets were sent with a one second pause to let the client get reconnected before the next disconnect occurs.

- Experiment 4 is a normal/attack batch deauthentication with 50 normal deauthentication and 50 attack deauthentication with 30s between all the deauthentication packets.

- Experiment 5 is a normal/attack alternating deauth: the normal deauth, attack, 30s wait, and the sequence is repeated 50 times.

4.2.1 EXPERIMENT SETUP

An AP is setup on a laptop using a standard Linux hostapd utility and four clients are connected to the AP. The attack was performed using a second laptop, running Kali Linux, by executing the aireplay-ng utility and the Python script mentioned earlier to generate deauthentication packets.

4.2.2 RESULTS

Table 2. and Table 3. show the precision, recall, and F1 score for the classification task on the test dataset. The DNN classifier achieved higher precision and recall, than the DT classifier.

TABLE 2.: Precision, recall, F1-score for DNN classifier.

Class	Precision	Recall	F1-score
Normal	0.9966	1	0.9983
Attack	1	0.9951	0.9975

TABLE 3.: Precision, recall, F1-score for DT classifier.

Class	Precision	Recall	F1-score
Normal	0.9934	1	0.9967
Attack	1	0.9905	0.9952

Table 4. shows the results for the five experiments using both decision tree (DT) and DNN classifiers.

TABLE 4.: Performance of decision tree DNNs classifiers in detecting deauth attacks.

Exp	FP DNN	FP DT	TP
1	0	2	200
2	0	5	300
3	0	1	200
4	0	1	50
5	0	3	50

Although the two classifiers achieved similar accuracy with the decision tree returning 99.8% and the DNN at 99.61%, the decision tree classifier flagged a lot of false positives while the DNN did not flag any. However, both classifiers were able detect all illegitimate deauthentication packets. When we consider the false positives and accuracies together, the DNN approach achieved better results than the decision tree classifier. Moreover, a classifier with high threshold can be used to configure the sensitivity per case, which makes it a promising base for deauthentication attack prevention.

We conducted another experiment using an access point (AP), a client (C), and an attacking machine (A). We installed our intrusion detection system on the the AP. Scripts were running on C and A to send deauth packets with a probability of 20%. Thus, on average one normal deauth and one modified deauth packet were sent every 10 seconds. The script running on C sent only normal deauth packets with its own MAC address, whereas the script running on A sent modified deauth packets that contained the MAC address of the client C. The experiment was conducted for one hour, and generated 593 attack and 560 normal deauths.

The intrusion detection system captures network traffic and saves it as PCAP files. We extract the selected features from the PCAP files and after data normalization we save the data into CSV files. The CSV files were sent to two separate machine learning classifiers, one is DNN and the other is decision tree. Using the DNN classifier we detected 657 attacks, which could be justified by the fact that in some cases single attacks are detected twice. The records were filtered by the difference in timestamps between two consecutive alarms. After filtering the records, 591 attacks were observed, which corresponds to a detection rate of 99.7%. The false positive rate was observed by manual means to be zero.

Using the decision tree we detected 1137 attacks. Assuming all of the true attacks were detected by the decision tree, this gives us a false positive rate of 47.8%, which is evidence that decision tree is not suitable for solving this problem.

4.3 SUMMARY

This work proposes an intrusion detection system and two mutually exclusive methodologies which can prevent deauth DoS attacks for IEEE 802.11 networks. One mechanism is based on one time password (OTP) prevention: only deauthentication packets having the correct OTP and reason field have been recognized as valid by both the AP and STA, which is a proof of the efficiency of this implementation. In the other approach, the Deep Neural Network technique based on Machine Learning reached 99.61% accuracy in the detection of genuine connections and attacks. Therefore, the proposed mechanisms can prevent the disassociation and deauthentication attacks leading to service interruptions. They are widely applicable, and we expect that they will be productive and effective to prevent deauth attacks against both present and future IEEE 802.11 devices.

CHAPTER 5

INTRUSION DETECTION SYSTEM FOR WORM ATTACKS

Our proposed hybrid tool first employs a visualization tool to explore network traffic and detect suspicious predefined worm activity followed by a machine learning approach to detect and classify worms based on their network behavior. The visualization tool cannot detect worms which alter their network behavior. Therefore, a machine learning approach is used to further classify and detect modified known malware and unknown malware with similar behavior.

5.1 ARCHITECTURE

The machine learning approach is employed to classify and detect predefined worms and undefined worms with similar network behavior. Once the machine learning tool detects suspicious activity, then the visualization tool will be used to further analyze the network traffic. In analyzing the traffic, the visualization tool displays suspicious worm activity, traces the attack, and displays the affected nodes.

Our system architecture is shown in Figure 11.. Using a PCAP file as an input to the ipsumdump utility [7], we extract features as a CSV file. The preprocessor converts the features into a numerical array and normalizes the data. The data is passed to the deep neural network (DNN) for detection of worms. If the machine learning tool detects suspicious activity, the visualization tool is used to display the suspicious worm activity, trace the attacks, and display the affected nodes.

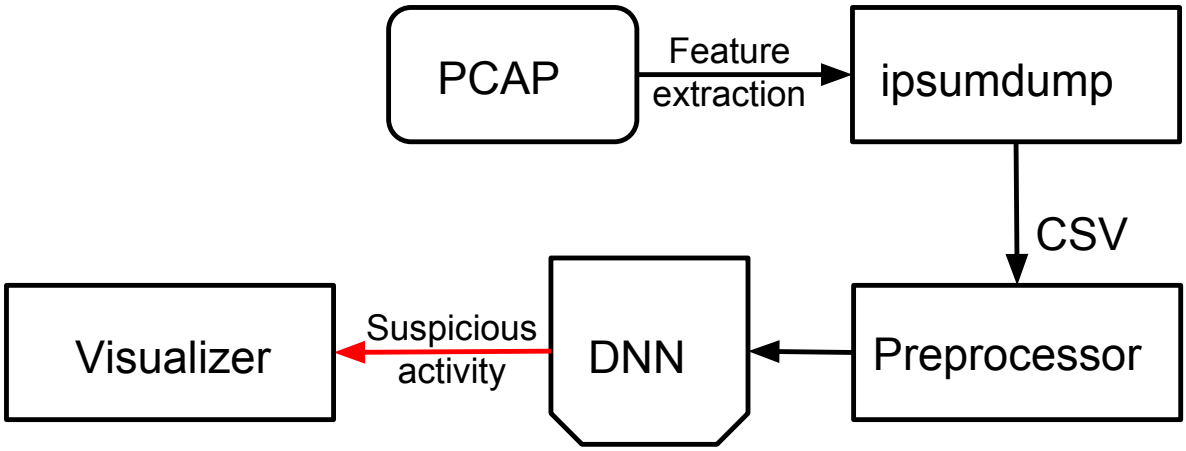

Fig. 11.: System architecture

We use the following features from ipsumdump:

- wire-length: Overall length of captured packet, including headers
- length: IP packet length
- protocol: IP protocol
- ip-id: IP-ID field
- ip-sum: IP checksum
- ip-ttl: IP time to live field
- ip-tos: IP type of service field
- ip-hl: IP header length
- capture-length: Length of IP data
- sport: Source TCP or UDP port
- dport: Destination TCP or UDP port
- payload-md5-hex: MD5 checksum of payload
- tcp-seq: TCP sequence number
- tcp-ack: TCP acknowledgement number

- tcp-window: TCP receive window
- udp-length: UDP length, reported in the header
- icmp-type: ICMP type
- icmp-code: ICMP code

5.1.1 MACHINE LEARNING

The multilayer perceptron (MLP) classifier [93] was used for solving the problem of distinguishing different patterns in malware traffic. Scaled exponential linear units (SELU) [69] were used in the hidden layer, which was found to be more efficient than classical rectified linear units (ReLU). The output layer of the classifier has the Softmax activation function [44], which outputs the probabilities of the input belonging to a specific class. This structure of the neural network is known to be most effectively trained when combined with the cross-entropy loss function [43]. In addition, dropout layers were added after each fully-connected layer to prevent overfitting [98]. The DNN classifier was trained for 50 epochs using the Adam classifier with parameters following the original paper [68].

As a proof of concept, we trained the DNN model with five real time PCAP capture files from online sources [15–17] containing captures of worm traffic. In addition to the worm traffic, we also captured normal network traffic generated by our machine. The overall size of the processed dataset for training was 11.2 MB of malicious traffic and 85.1 MB of normal traffic. The test dataset was 10% of the entire dataset.

5.1.2 WORM VISUALIZATION

One of the main contributions of this work is building the visualizer IDS with D3. D3[1] is a JavaScript library that is used to create data visualizations. The data generated by computer networks is considerably large and visualization methods such as D3 are necessary to interpret and process this overwhelming data. The visualization tool is designed to help interpret the large amount of data and display this data on the screen in a visually appealing format.

The worm visualization analysis tool presented in this paper uses a Finite State Machine (FSM) model, which defines the expected behavior of worm network traffic. When the behavior deviates from this expectation, the FSM will not display the new states, however,

[1] https://d3js.org

we will still be able to detect the worm attack using machine learning. Our packet analysis tool processes one or more captured network PCAP files and produces a JSON file. The JSON file is provided to the visualization tool to be interpreted. We use our FSM model to divide the packets into four categories: TCP, SMB, FTP, and Exploit. Packets in the Exploit category are various worm states of attack. Four colors have been assigned to these categories in order to visually determine the kind of packet. Exploit packets are red, and any red activity in the display can be associated with potential or actual worm attacks.

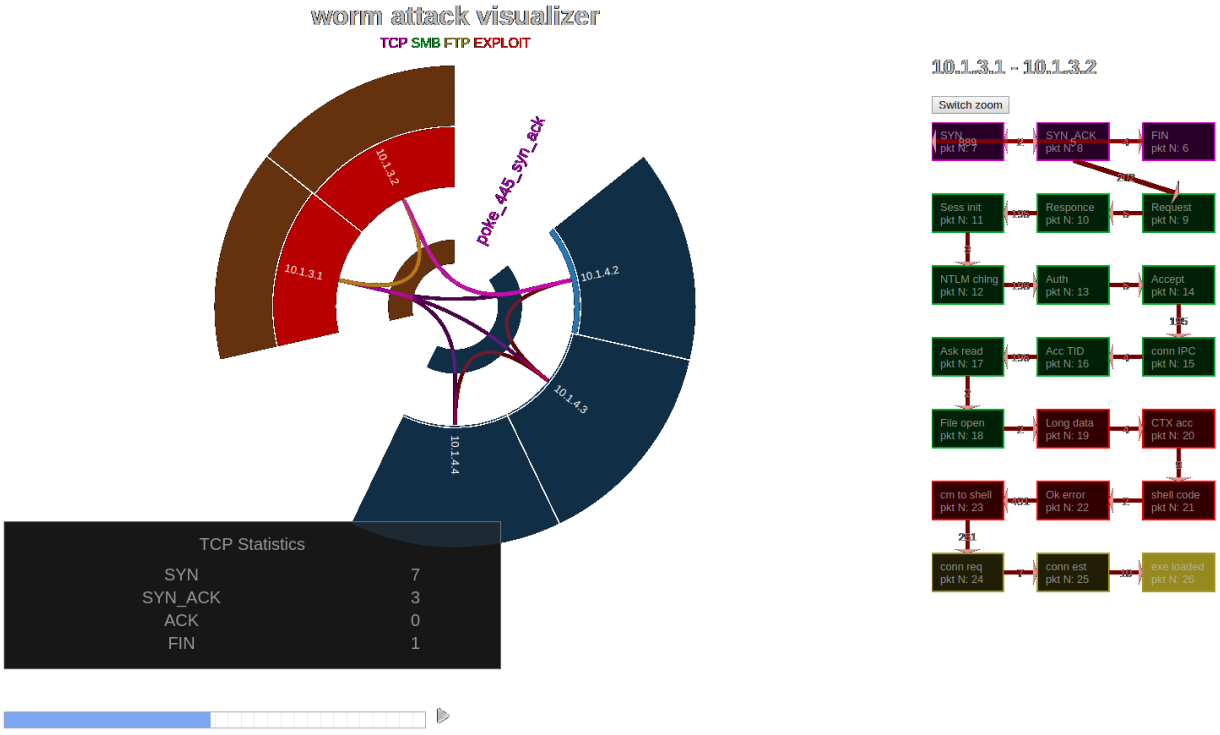

Fig. 12.: Worm visualization. The FSM of the detected worm is shown on the right.

An example of the visualization tool can be seen in Figure 12.. The visualization tool displays each node represented by a wedge shape. Nodes are grouped based on their IP address and a specific color is assigned based on the network the node belongs to. The wedge shape is then filled depending on the overall percentage of the defined packet activity. This makes it easier to observe the overall activity distribution on the nodes during an analysis session. However, if a worm attack is detected, the activity area will be displayed as red. Packet transmissions between any two nodes are depicted by an arc made in the color of the packet category. Since the visualizer shows the network traffic over time, these interconnecting arcs will change and fade color as other categorized packets are transmitted. The complete PCAP session, whether from one or multiple files, is combined into a single

timeline. We can navigate from the beginning to the end and to any point along the timeline of the session. The visualizer will update to show all activity up to the current index. Along with the pictorial representation of inter-connectivity and packet activity, the visualization tool displays packet statistics. At the center of the main graph the current packet type is always displayed. This text continuously and quickly updates based on the current timeline position. There is an overview of general TCP statistics, showing different TCP packet types and counts for each type. This is also dynamically updated with the position of the timeline. When a node is clicked, detailed statistics for that node is displayed, as seen in Figure 13.. When clicking on the connection between two nodes, we can see detailed FSM information. As with all parts of this tool, these statistics update dynamically based upon the current timeline position. This detail helps us determine in which direction the infection is occurring. From this view, we can click on a specific packet and see even more detail about its state. By changing the index on the timeline, it is easy to determine which node gets infected first by seeing the first node turn red, we can then determine the source of the attack.

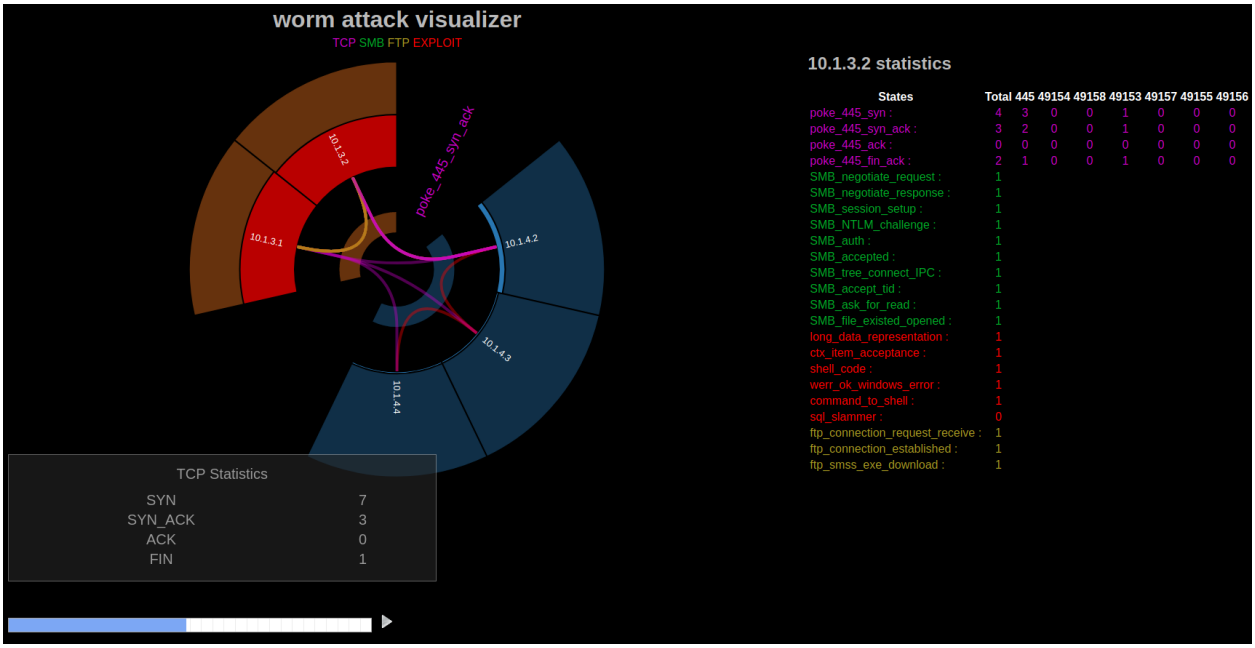

Fig. 13.: Detailed Statistics for Each Node

5.2 EVALUATION

5.2.1 WORM TRAFFIC GENERATOR USING NS3 SIMULATOR

NS-3[2] is an open-source discrete event network simulator. Using Wireshark, a packet capture and analysis tool, we gather information about a particular worm. This information is fed into the simulator to simulate an attack from the worm. We simulate an attack because it is very difficult to get all attack data from online sources as many companies hide details of the attack for security reasons. Due to the hiding of the data, we cannot see the propagation of the worm throughout the network. Using a test network and the simulator allows us to simulate an attack and provides us with complete data about the attack giving a clear picture of the network behavior of the worm. In addition, by using the traffic generator we can change the network topology and simulate multiple worm attacks, capturing the attacks to PCAP files.

The network simulator configuration consist of six nodes in total as shown in Figure 14.. Nodes 0 and 1 are in Subnet 1. Nodes 3, 4, and 5 are in Subnet 2. Node 2 is a router that exists in both subnets and routes the traffic through them.

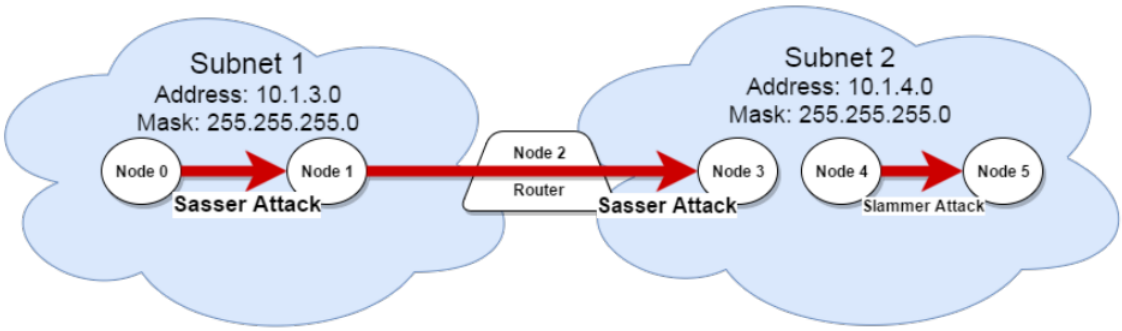

Fig. 14.: Simulated evaluation network

The generated traffic consists of normal traffic, such as SMB and FTP, and multiple packets containing the behavior of the Sasser and Slammer worms.

5.2.2 EXPERIMENT

We performed four experiments to demonstrate the ability of our method to analyze network traffic.

In the first experiment, we analyze the performance of our DNN to classify network traffic as being part of Sasser, Slammer, Eternal Rocks, Wannacry, or Petya worm attacks or as being normal traffic. Recall that our dataset, described in Section 4.1, had 11.2 MB of worm traffic and 85.1 MB of normal traffic. We used 90% of the dataset for training and 10% of the

[2]https://nsnam.org

dataset for testing. Table 5. shows the precision, recall, and F1 score for the classification task on the test dataset. Our DNN achieved high precision in most cases. The precision for Sasser was lower, likely due to its high similarities to other worms. We observed low recall for the Petya worm, likely because its sample included normal traffic packets. We note that our classification of normal traffic was at 99%, which indicates that our system would have relatively low false alarms.

TABLE 5.: Performance of the DNN on classifying worm and normal traffic with known worms.

Name	Precision	Recall	F1
Sasser	0.8387	0.8125	0.8254
Slammer	1.0000	1.0000	1.0000
EternalRocks	0.9650	0.8894	0.9257
Wannacry	0.9581	0.9970	0.9772
Petya	0.9444	0.3119	0.4690
Normal	0.9985	0.9998	0.9991

In the second experiment, we tested the system using a PCAP file containing both Sasser and Slammer worm traffic, which was generated using our NS-3 worm traffic generator tool. The PCAP file was applied as the input to the detection system and yields the results shown in Table 6.. Our system uses packet by packet analysis to detect worms. The percentage of packets classified as each type of worm (i.e., probability) are calculated as

$$P_n = \frac{\Sigma V[n]}{N}, \tag{1}$$

where $V[n]$ is the vector of percentage of packets and N is the number of vectors. In cases where the worm consists only of a single packet, the percentage of packets is low. In other cases where the worm attack consists of multiple packets, the percentage of packets is high. Despite the Slammer attack consisting of only one packet and having low percentage of packets, our system was still able to detect it.

TABLE 6.: DNN classification probability on the simulated Sasser and Slammer worms.

Name	Probability
Sasser	76.79
Normal	20.30
Slammer	00.12

In the third experiment, we wanted to test the system on a previously unseen, but similar worm. The main advantage of a DNN-based analyzer is the possibility to detect worms that may not be explicitly present in its training dataset. We obtained a sample of the NotPetya worm from CTU captures [14] and tested it with our classifier. NotPetya has similar, but not exactly the same, network behavior as the Petya worm. Table 7. shows that the DNN found 60% of packets similar to Petya due to the fact that the NotPetya trace was similar to the Petya worm, which shows that the DNN can detect worms with similar network behavior.

TABLE 7.: DNN classification probability on the unknown NotPetya sample.

Name	Probability
Petya	60.46
Normal	22.75
WannaCry	12.06

Finally to further validate our DNN, we classified a sample of normal traffic obtained from Netresec[3]. Our DNN reported that 99.996% of the traffic was normal packets, successfully classifying the normal traffic.

5.3 SUMMARY

In this work, we have presented a hybrid worm detection and analysis approach. We trained a DNN classifier on the network traffic produced by several worms and normal network traffic. On the five worms tested, our classifier had an average precision of 94.1%. We

[3]https://www.netresec.com/?page=PCAP4SICS

also successfully classified a set of normal traffic with a 99.99% probability. In addition, we used our trained classifier to detect worm traffic that was unknown to the classifier but had network traffic similar to one of the known worms. The second part in our hybrid approach was the development of a visualizer based on D3 to observe the network and infection behavior of identified worms and to trace the attacks. The proposed hybrid approach provides better insight on worm activities and better detection of worms that have similar network behavior. Finally, we were able to build a worm traffic generation tool using ns-3.

CHAPTER 6

NNIDS - NEURAL NETWORK BASED INTRUSION DETECTION SYSTEM

The goal of this work is to develop a system that uses a novel approach in real-time traffic analysis to detect and classify malware. It does not try to analyze the traffic itself, but extracts the secondary data features which contains a great deal of information about overall network state. The data used in the current system has 90 dimensions which is much more than usually used for deep neural network (DNN) classifiers, therefore allowing better accuracy in this analysis. The utilization of this large dataset of malware, used to train the DNN, allows for detection of malware, not only in the dataset, but also those which are not in the dataset but have similar network behavior. Here, two smaller dimensional DNN systems are used in this new approach to detect malware. This leads to a decrease in computational power over one larger dimensional DNN. Hence a real time monitoring system for malware can be constructed. Four machine learning classifiers are analyzed to determine which is best for accuracy and performance. In addition, a heuristic prediction evaluation is performed.

6.1 ARCHITECTURE

The overall structure is shown in Figure 15.. A data packet sniffer captures all data packets transferred in protected networks. The data obtained from the hardware sniffer is preprocessed with the analyzer, which extracts the necessary features from the capture file. The set of features saved is then analyzed using the DNN classifier. The output is the probability of the maliciousness of the data stream in the network.

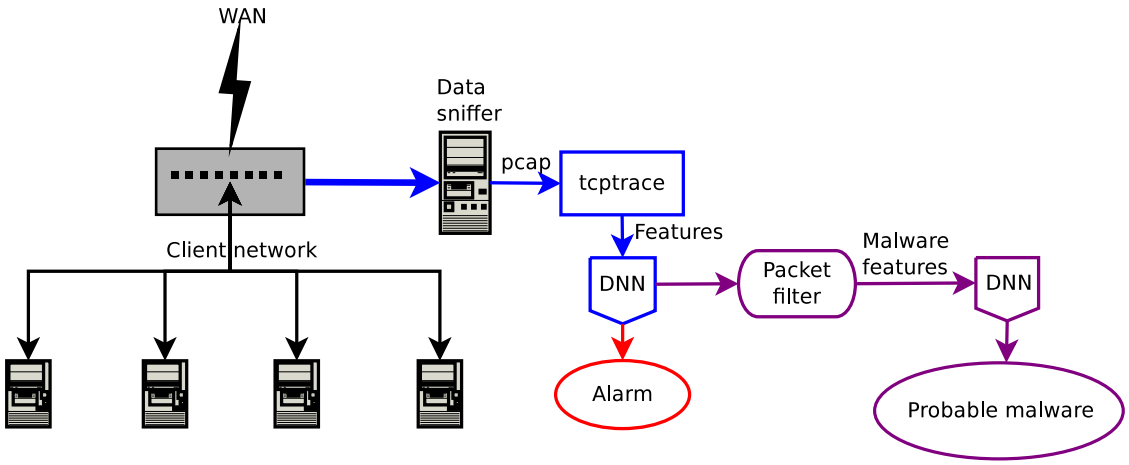

Fig. 15.: System overview

6.1.1 PCAP ANALYZER AND DATA EXTRACTOR

A data packet sniffer is a software tool that records the data flowing over network links in real time. Packets are saved in the PCAP (Packet CAPture) format. We extract features from the captured network files. This is done by using the tcptrace [89] utility, which is able to extract 90 data features from the capture files as shown in Table 8.. As a result of this step, a CSV file with the extracted data (one row for each flow in the PCAP file and 90 features per flow) is created.

TABLE 8.: TCPTRACE features.

0	conn	30	rexmt data bytes b2a	60	avg segm size b2a #
1	host a	31	zwnd probe pkts a2b	61	max win adv a2b
2	host b	32	zwnd probe pkts b2a	62	max win adv b2a
3	port a	33	zwnd probe bytes a2b	63	min win adv a2b
4	port b	34	zwnd probe bytes b2a	64	min win adv b2a
5	first packet	35	outoforder pkts a2b	65	zero win adv a2b
6	last packet	36	outoforder pkts b2a	66	zero win adv b2a
7	total packets a2b	37	pushed data pkts a2b	67	avg win adv a2b
8	total packets b2a	38	pushed data pkts b2a	68	avg win adv b2a
9	resets sent a2b	39	SYN/FIN pkts sent a2b	69	initial window bytes a2b
10	resets sent b2a	40	SYN/FIN pkts sent b2a	70	initial window bytes b2a
11	ack pkts sent a2b	41	req 1323 ws/ts a2b	71	initial window pkts a2b
12	ack pkts sent b2a	42	req 1323 ws/ts b2a	72	initial window pkts b2a
13	pure acks sent a2b	43	adv wind scale a2b	73	ttl stream length a2b
14	pure acks sent b2a	44	adv wind scale b2a	74	ttl stream length b2a
15	sack pkts sent a2b	45	req sack a2b	75	missed data a2b
16	sack pkts sent b2a	46	req sack b2a	76	missed data b2a
17	dsack pkts sent a2b	47	sacks sent a2b	77	truncated data a2b
18	dsack pkts sent b2a	48	sacks sent b2a	78	truncated data b2a
19	max sack blks/ack a2b	49	urgent data pkts a2b	79	truncated packets a2b
20	max sack blks/ack b2a	50	urgent data pkts b2a	80	truncated packets b2a
21	unique bytes sent a2b	51	urgent data bytes a2b	81	data xmit time a2b
22	unique bytes sent b2a	52	urgent data bytes b2a	82	data xmit time b2a
23	actual data pkts a2b	53	mss requested a2b	83	idletime max a2b
24	actual data pkts b2a	54	mss requested b2a	84	idletime max b2a
25	actual data bytes a2b	55	max segm size a2b	85	hardware dups a2b
26	actual data bytes b2a	56	max segm size b2a	86	hardware dups b2a
27	rexmt data pkts a2b	57	min segm size a2b	87	throughput a2b
28	rexmt data pkts b2a	58	min segm size b2a	88	throughput b2a
29	rexmt data bytes a2b	59	avg segm size a2b	89	Unnamed

6.1.2 DUAL DNN MALWARE DETECTION AND CLASSIFICATION

The 90 data features are extracted from the PCAP file and fed into the DNN classifier. The DNNs were built using the Keras framework [48]. In this work, we propose using two DNN classifiers. The first classifier is limited to only determining if the traffic is normal or suspicious. Upon detection of suspicious activity, the second DNN will further classify the network traffic based on our malware dataset. The second DNN is a multiclass classifier which tries to assign the name of known malware, from the dataset it was trained on, to the input data. Probabilities of malware traffic belonging to one of the known malware are generated, which can be helpful for heuristic detection.

6.2 EXPERIMENTS AND RESULTS

For detection we used a collection of PCAP files containing the malware dataset, which is 290MB in size and contains 120 malware such as sality, blackhole, and zeus. Malware samples [18] were selected together with ten samples of normal network traffic [6, 9, 18].

As a proof of concept, a realtime monitoring system was built using simple bash scripting. The first step is to capture traffic at specific intervals (how often we capture traffic) and for specific durations (how long is each capture) to be determined by user configurable variables. The second step is to extract the 90 features for each PCAP file and save it as a CSV file. Finally, these CSV files are fed into the first DNN classifier to make a decision to determine if this is malicious traffic. Once malicious traffic is detected, the CSV file is fed into the second DNN classifier to determine which malware family this traffic belongs to. The analysis of detection rates was done by sending malware PCAP files through a virtual machine interface and analyzing it in real-time.

6.2.1 SINGLE VS. DUAL DEEP NEURAL NETWORK CLASSIFIERS

Initially, the system only consisted of a single DNN classifier. The single DNN classifier performed malware detection and classification on all network traffic observed. The single DNN classifier is able to perform malware detection as well as classification, but we used the fact that most of the traffic going through the IDS will be normal traffic to redesign the system in order to enhance its performance. Thus, we add a small preliminary detector which will filter out the normal traffic and pass only suspicious traffic to the second classifier.

The new system uses two DNN classifiers to detect malware network traffic. This decreases data processing time by about 40%. The performance of this dual-DNN system is higher in comparison with one large DNN because the data is pre-filtered by the analyzer.

When only normal traffic is flowing through the IDS, both single and dual DNN classifiers

have the same performance. In this experiment, the same cross-validated malware dataset was analyzed using the single and DNN models separately. The results are summarized in Table 9.. As the data shows, the DNN system is about two times faster than the single DNN in prediction and three times faster in training than the original detector. During this experiment, almost identical accuracy was observed among the different methods. The single classifier's training time is more than three times the dual classifier's.

TABLE 9.: Accuracy and performance of dual and single DNN classifier.

	Training	Prediction	Accuracy
Single Classifier	214.75s	4.22s	0.998
Dual Classifier	65.03s	2.67s	0.997

6.2.2 CLASSIFICATION METHODS COMPARISON

As an alternative to the DNN classifier, other researchers have used different classification methods to build intrusion detection systems [61] and we will compare our results to three classifiers' results using the same dataset. Hassan et al. [61] used Random Forest (RF) and Support Vector Machines (SVM). We will also test against a Naive Bayes (NB) classifier.

6.2.2.1 Accuracy

The obtained results from building our DNN on the KDD'99 dataset were compared with results from previous experiments [32, 61]. These are all shown in Table 10.. In terms of accuracy, the DNN was the most accurate, while the SVM classifier comes second. The Naive Bayes comes a close third after the SVM while Random Forest has the lowest testing accuracy.

TABLE 10.: Testing accuracy.

Classifier	Testing Accuracy
Naive Bayes	0.9271
Random Forest	0.9141
SVM	0.9299
DNN	0.9997

A loss function is a method used to evaluate how well an algorithm models a dataset. If

the predictions are inaccurate, the loss function will output a higher number. We performed a 10-fold cross-validation for the DNN classifiers, and the results are shown in Table 11..

TABLE 11.: Accuracy and Loss of Dual and Single DNN Classifier.

Classifier	Accuracy	Loss
Single Classifier	0.9975	0.0095
Dual Classifier	0.9998	0.00083

6.2.2.2 Performance

The performance of the various trained models in predicting malware in actual network traffic is shown in Table 12.. In training, Naive Bayes had the fastest time followed by Random Forest. DNN had a training time double that of the Random Forest classifier, while SVM was the slowest. On the other hand, Random Forest was the fastest classifier in prediction and SVM was the slowest. The DNN was not as fast as Random Forest and Naive Bayes.

TABLE 12.: Training and testing times.

Classifier	Training Time (s)	Prediction Time (s)
Naive Bayes	5.12	21.75
Random Forest	637.2	2.93
SVM	2648.2	510.43
DNN	**1240**	**40.41**

6.2.2.3 Normal Traffic

One important aspect of building an IDS is to reduce the false positives as much as possible. We tested four classifiers against normal traffic that does not exist in our dataset. Based on the results of the experiment, the DNN classifier did not flag the normal traffic as suspicious, whereas the other classifiers flagged some of it as suspicious. The results obtained from the different classifiers are listed in Table 13..

TABLE 13.: Predicted malware given a normal traffic sample

Classifier	Percentage of Packets
DNN	0.999 Normal
Random Forest	0.943 Normal 0.056 CitadelPacked
SVM	0.784 normal_small 0.208 Ramnit
Naive Bayes	0.354 normal 0.316 Ramnit 0.145 OSX_Dockster 0.056 Tbot

When evaluating the results of the accuracy, performance, and normal traffic evaluations, the DNN emerges as the best classifier across the board. Analyzing the classification comparison, the DNN scores best since it had the highest accuracy while maintaining relatively good training and testing times. On the other hand, the second-best result is Random Forest in comparison to the Naive Bayes and SVM classifiers. Therefore, the DNN is preferable in building the IDS, which requires heuristic analysis.

6.3 SUMMARY

The developed IDS operates with a novel approach in traffic analysis. It does not try to analyze the individual packets, instead, it extracts the secondary data features which contain a large amount of data about the overall network state. The IDS extracts 90 features from captured network traffic. The usage of the DNN allows for the detection of malware which is not present in the training dataset. The performance of a dual DNN system is higher in comparison to a single DNN due to the first DNN's filtration. Therefore, the second DNN needs to process much less data. Such an approach allows us to build a IDS with very low CPU usage. New samples of normal and malware traffic can easily be used to tune and retrain the DNN. This approach yields a flexible, accurate, and reliable intrusion detection system. Furthermore, it will be able to detect new malware not present in the training dataset and will reduce false positives detected in normal traffic.

CHAPTER 7

BENCHMARKING DIMENSIONALITY REDUCTION METHODS FOR MALWARE CLASSIFICATION BASED ON NETWORK BEHAVIOR

Malware classification based on network behavior is an important step in the analysis for malware detection. The efficiency of this step is crucial for better performance of the IDS. As we have seen in Chapter 6, the deep neural network (DNN) was the best option in terms of accuracy, but not in terms of performance. We propose two ways of increasing performance by reducing the dimensionality of the datasets and selecting the optimal features.

7.1 ARCHITECTURE

The test system performs feature extraction to create a pre-processing dataset. The dataset features are compressed using several feature reduction methods and then fed into a DNN classifier. The system's three main parts, a network tool to capture network traffic and extract data, feature extraction method, and a DNN classifier, are shown in Figure 16. and discussed in more detail below.

In our experiment, feature vectors are first reduced from 90 to 45 features. Once reduced, the features are compressed to between 10 and 40 features to determine which compression method works best. This allows the DNN classifier to process features faster and with less memory consumption. We compare four compression methods to determine the best method to increase performance without compromising accuracy.

7.1.1 PCAP ANALYZER AND FEATURE EXTRACTOR

As described in Chapter 6, we extract the features from the PCAP files using tcptrace [89], which can extract information from network capture files, such as port number, total packets, throughput, etc. In our experiment, the CSV dataset had 90 features extracted from tcptrace. These features were reduced to 45 based on Shannon's entropy; features with the lowest entropy were removed. In addition, features which do not influence the output of the DNN, such as timestamps or IP addresses, were moved to the end of the feature importance list

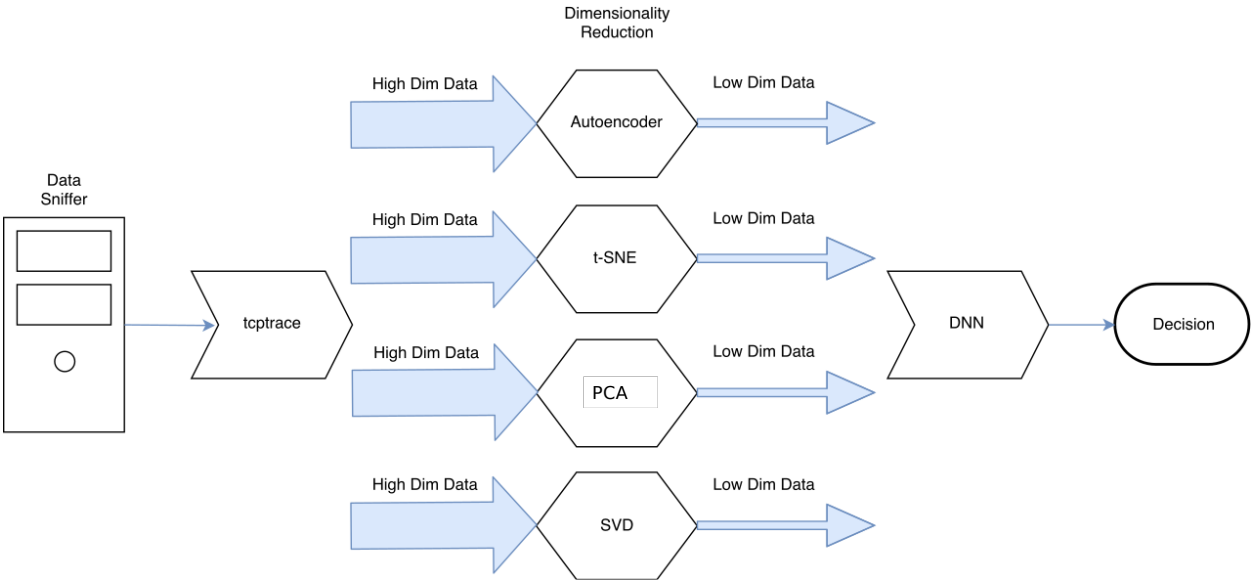

Fig. 16.: General architecture of the system.

such that they were removed first because these features will change in each attack. A detailed program output is written to a CSV file from tcptrace. In our case, the CSV file data is shown with one row for each TCP flow in the PCAP and 45 features per flow. The 45 features are shown in Table 14..

TABLE 14.: The 45 features selected based on Shannon's entropy.

Idx	Entropy	Description	Idx	Entropy	Description
0	11.001	conn_#	22	2.058	unique_bytes_sent_b2a
5	10.995	first_packet	73	2.037	ttl_stream_length_a2b
6	10.975	last_packet	59	2.028	avg_segm_size_a2b
83	8.967	idletime_max_a2b	61	2.022	max_win_adv_a2b
3	8.269	port_a	60	2.016	avg_segm_size_b2a
2	7.703	host_b	69	1.996	initial_window_bytes_a2b
4	3.802	port_b	57	1.989	min_segm_size_a2b
1	3.615	host_a	64	1.981	min_win_adv_b2a
84	2.768	idletime_max_b2a	62	1.978	max_win_adv_b2a
7	2.492	total_packets_a2b	11	1.974	ack_pkts_sent_a2b
67	2.433	avg_win_adv_a2b	55	1.973	max_segm_size_a2b
63	2.336	min_win_adv_a2b	58	1.959	min_segm_size_b2a
29	2.326	rexmt_data_bytes_a2b	13	1.951	pure_acks_sent_a2b
27	2.287	rexmt_data_pkts_a2b	56	1.947	max_segm_size_b2a
88	2.178	throughput_b2a	70	1.947	initial_window_bytes_b2a
85	2.171	hardware_dups_a2b	53	1.884	mss_requested_a2b
87	2.162	throughput_a2b	74	1.876	ttl_stream_length_b2a
68	2.141	avg_win_adv_b2a	14	1.871	pure_acks_sent_b2a
12	2.139	ack_pkts_sent_b2a	24	1.842	actual_data_pkts_b2a
8	2.133	total_packets_b2a	82	1.839	data_xmit_time_b2a
25	2.074	actual_data_bytes_a2b	38	1.819	pushed_data_pkts_b2a
26	2.068	actual_data_bytes_b2a	23	1.807	actual_data_pkts_a2b
21	2.067	unique_bytes_sent_a2b			

7.2 IMPLEMENTATION AND EXPERIMENTAL RESULTS

7.2.1 DATASET

The CTU-13 dataset is based on a large collection [13] created by the Czech Technical University in 2011. It is a large capture of real botnet traffic such as Neris, Rbot, Virur, Menti, and Sogou, mixed with other normal and background traffic. Overall, the CTU-13 dataset consists in 13 captures of different botnet samples and each scenarios was executed with a specific malware, which used several protocols, and performed different actions. The total size of this dataset is 1.9GB.

As a proof of concept, we will be using only eight PCAP files from the CTU-13 dataset. The capture files were truncated such that only malware traffic is present in the PCAP files. One quarter of the dataset was used for training. In each scenario we executed a specific malware, which used several protocols and performed different actions.

7.2.2 PROCEDURE

All of the prepared files are processed with tcptrace and given probable malware names. Next, all CSV files are processed with one of the above-mentioned feature reduction algorithms, then, reduced from 45 to 40, 30, 20, and 10 features using four different compression algorithms. The DNN classifier is trained on this data for 50 epochs. Finally, the data is fed into a fully connected DNN classifier, which assigned each test sample to one of the malware in the dataset. The accuracy and training/compression time for all the datasets for the 50 fitting epochs was then measured.

7.2.3 DNN ARCHITECTURE

The classifier was built based on the multilayer perceptron classifier architecture [93]. As we can see in Table 15., the number of hidden units in each layer vary on the input data dimensionality (number of features). Activation of the output layer was chosen to be the Softmax function, which outputs the probability of the input belonging to each of the classes [44]. The model was trained for 50 epochs by the Adam optimizer with default parameters as it is specified in the original paper of Kingma et al. [68].

TABLE 15.: DNN architecture.

Layer	Number of units	Activation
Fully-connected	dim*2	Relu
Fully-connected	dim*3	Relu
Fully-connected	dim*3	Relu
Fully-connected	dim*2	Relu
Fully-connected	dim	Softmax

7.2.4 RESULTS

We conducted a comparison between no compression and four dimensionality reduction techniques: autoencoder, PCA, SVD, and t-SNE.

7.2.4.1 Accuracy

The obtained accuracy is shown in Table 16. and Fig. 17.. Fig. 17. shows that the SVD and PCA methods are the most accurate, with PCA being slightly more accurate than SVD when we reduce the dimensionality even more. Both methods are slightly more accurate than no compression when the feature dimensionality is 40. Finally, t-SNE has the worst accuracy across all compression efforts. Note that t-SNE appears to fluctuate in accuracy. It is by definition a stochastic algorithm which tries to compress the data as much as possible, but without any guarantees. Stochastic methods are a kind of lottery, so once in a while t-SNE will compress well and return somewhat accurate results, while other times it will not be so accurate.

TABLE 16.: Accuracy at various dimensionality.

Compression Method	Dimensionality 40 - 30 - 20 - 10			
Original	0.9963 (45 Dim)			
SVD	0.9975	0.9960	0.9946	0.9939
PCA	0.9973	0.9972	0.9971	0.9966
Autoencoder	0.9041	0.9037	0.8998	0.8679
t-SNE	0.6105	0.6569	0.5965	0.6705

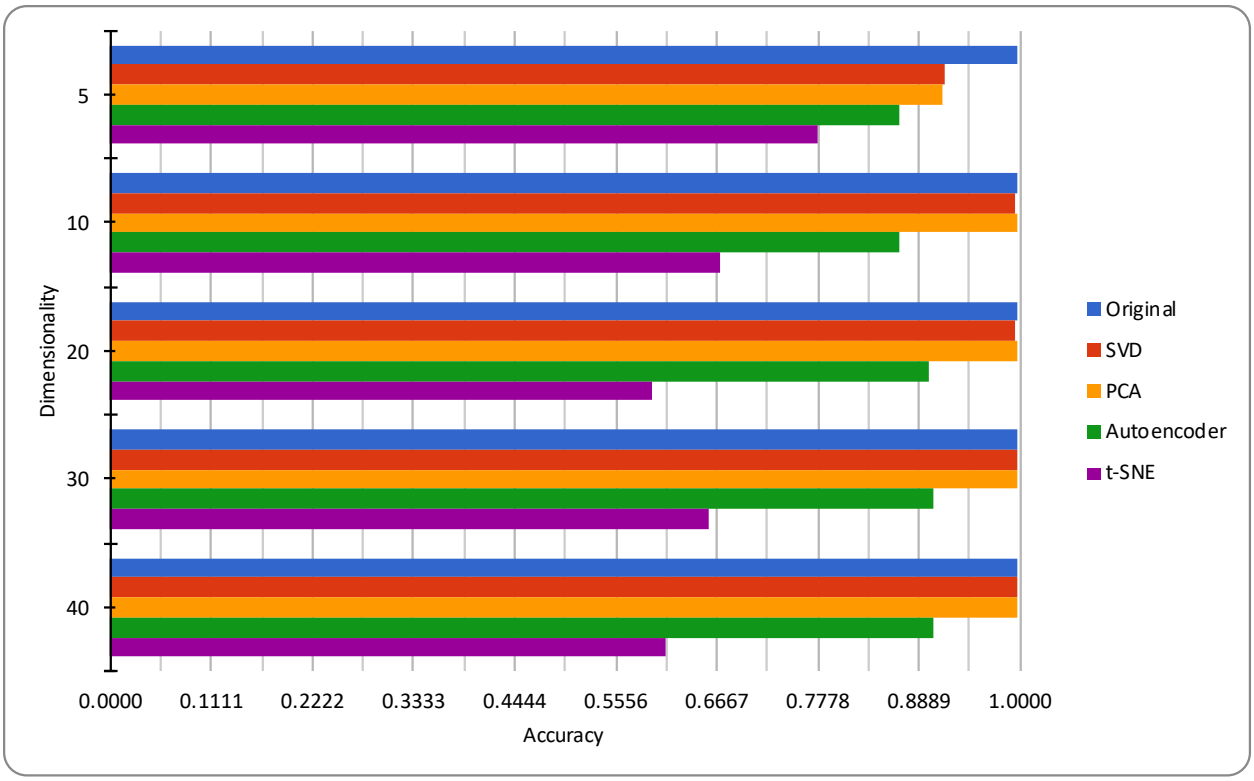

Fig. 17.: Comparison of accuracy over various dimensionality.

However, in this experiment t-SNE is shown to always be less accurate than the other methods because of an almost complete overlap of signatures, as shown in Fig. 18.. As a result, t-SNE with compressed data showed worse classification accuracy than all others. Reducing dimensionality with the DNN Autoencoder is less efficient compared to the SVD and PCA methods, and it is likely that some data features are lost, resulting in lower accuracy.

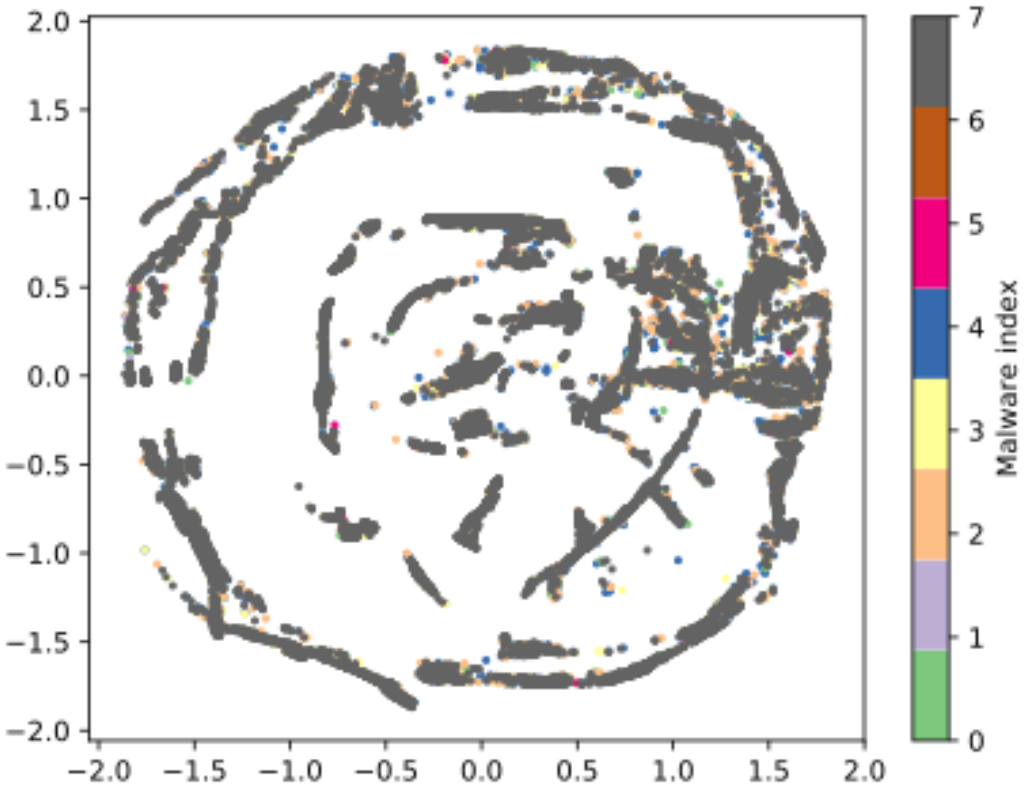

Fig. 18.: t-SNE: The data was compressed into two dimensions and visualized using different colors to show points from different datasets. However, t-SNE ultimately did not distinguish between the various malware signatures, and we see nearly a complete overlap on the graph.

7.2.4.2 Performance

In terms of performance, SVD and Autoencoder perform better than with no compression, especially as the dimensionality is reduced. SVD was the best in terms of performance by almost 45% compared to no compression at 10 dimensionality. PCA was 50% slower compared to no compression. However, t-SNE performs many times slower than all of the other methods combined. The processing time for t-SNE is about 2 hours for the whole dataset, while all other methods require less than two minutes as seen in Table 17.. In Fig. 19. we show that this takes about half a minute to complete the classification (training and compression) time.

TABLE 17.: Performance (runtime in seconds) at various dimensionality

Compression Method	Dimensionality 40 - 30 - 20 - 10			
Original	43.6s (45 Dim)			
SVD	41.8	33.5	26.8	19.6
PCA	109.8	107.2	103.4	92.7
Autoencoder	42.5	34.0	27.6	21.2
t-SNE	5,126.4	2,825.2	1,857.1	4,785.0

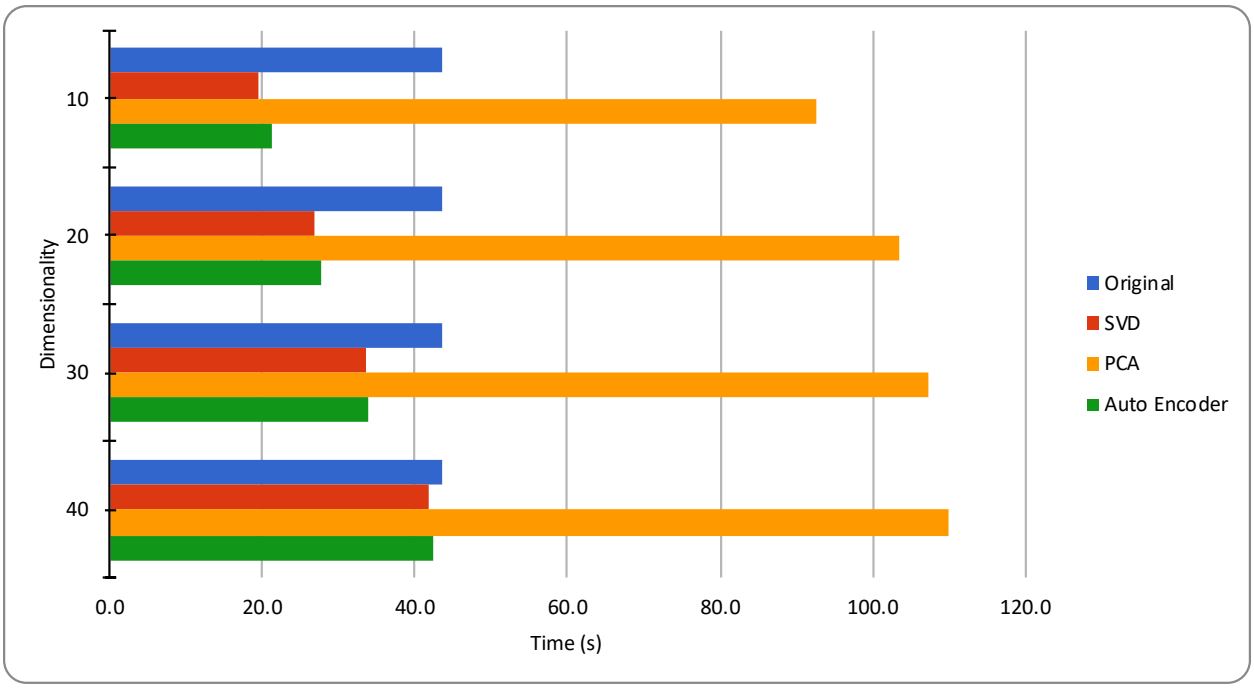

Fig. 19.: Comparison of performance over various dimensionality.

By using these compression methods, we were able to increase performance while maintaining high accuracy over a relatively short computational time. We achieved a 99.3% classification accuracy through SVD compression and gain about 45% of performance increase over no feature reduction.

7.3 SUMMARY

We introduced a comparison analysis between different feature reduction methods through

compression for DNNs to classify malware based on network behavior. In the results, SVD performed better than other techniques as it maintains high accuracy and better performance. In our comparisons, t-SNE performed the worst for DNN classifying of malware signatures.

CHAPTER 8

FEATURE SELECTION TOOLS

8.1 INTRODUCTION

Building an intrusion detection system based on network behavior consumes a lot of system resources. The Deep Neural Network (DNN) classifier requires more resources in comparison to other machine learning classifiers. Our proposal is to increase performance of the DNN classifier by reducing the number of features required for the classification of malware based on network behavior. Using more features can incur significant overhead in training and testing. Furthermore, the more features we have will largely affect resource consumption and detection stability. Therefore it is necessary to reduce the number of features by eliminating features with low importance, without compromising accuracy.

Although the set of features can be large, only a few features are important for detection. Therefore, the elimination of redundant non-contributing features is essential to reducing the dataset size. Moreover, effective feature selection and data reduction are important in order to decrease the training duration and to improve the detection rate. Thus, a good feature selection approach minimizes the information redundancy, computation complexity, while maintaining accuracy and detection rates.

There are two main feature selection techniques: *Filter* and *Wrapper* techniques [40]. In the filter technique, selection of important features and elimination of other features are based on some predefined criteria, while the wrapper technique evaluates features based on a training dataset. This work proposes two new feature selection techniques: accuracy estimation and scoring.

8.2 PROPOSED APPROACH

We propose two novel feature selection algorithms: the accuracy estimation algorithm and the scoring algorithm. The accuracy estimation algorithm uses Shannon entropy for the estimation of feature importance. The features are then sorted by importance. The second algorithm consist of two parts, scoring and correlation. The scoring algorithm utilizes Shannon entropy, informational gain, and linear independence (expressed as residual sum of

squares) for feature scoring with further cross-correlation-based filtering. Utilizing the three statistical features which are used for scoring and the additional filtering by means of cross-correlation allows the selection of the optimal features that bear most of the information in the dataset. The proposed methods work with any kind of Machine Learning (ML) classifier and can be directly applied.

The CTU-13 [19] dataset was used for the classification. The data features were extracted using the `tcptrace` utility [89] which takes a PCAP file as an input, extracts 90 features, and saves the output as a CSV file. Normalization of the CSV data is required to convert string values into numbers and to scale the data. We normalized the data to zero mean and unit variance and used this data as input to the DNN classifier, described in Chapter 5.

8.2.1 ACCURACY ESTIMATION ALGORITHM

We propose the accuracy estimation algorithm described in Algorithm 2 to estimate the influence of a specific feature on the accuracy of the classification.

Algorithm 2: Algorithm for estimation of real influence of specific features on classification accuracy.

```
Define feature_list containing feature names from dataset
repeat
    foreach feature in feature_list do
        Delete feature from dataset
        Train DNN classifier on modified dataset
        Evaluate classification accuracy
        Store accuracy
        Restore feature
    end
    Delete feature with minimal accuracy change
until dataset dimensionality > 0
```

We remove one feature, train the classifier, measure the accuracy of the entire system, store this value, restore the deleted feature and select another feature to be deleted. We continue the process until all features have been processed. We do this to measure the effect of feature removal on the overall classification accuracy.

We used Shannon entropy to provide weights for each feature indicating its importance in the dataset. Shannon entropy characterizes the overall amount of information that is

contained in a given datavector. The vector consists of values between 0 and 1 indicating how much information is present. Zero indicates no information is present and 1 indicates information is present. Other values in different ratios indicate entropy is high, meaning it *may* contain lots of information (or just noise). Shannon entropy is defined as

$$H = -\sum_{i=1}^{n} p_i * log_2(p_i) \tag{2}$$

where p_i is the probability of the specific value i.

For the initial estimation of feature importance, the entropies of all features were calculated. The feature key table and entropy values are shown in Table 18.. In order to determine which feature to be remove, we compare the entropy values and remove the features with the least entropy value.

TABLE 18.: Feature codes and entropies.

Feature	Feature name	Entropy	Feature	Feature name	Entropy	Feature	Feature name	Entropy
0	conn #	11	30	rexmt data bytes b2a	1.67	60	avg segm size b2a	2.02
1	host a	3.61	31	zwnd probe pkts a2b	1.03	61	max win adv a2b	2.02
2	host b	7.7	32	zwnd_probe_pkts_b2a	0.5	62	max_win_adv_b2a	1.98
3	port_a	8.27	33	zwnd_probe_bytes_a2b	1.03	63	min_win_adv_a2b	2.34
4	port_b	3.8	34	zwnd_probe_bytes_b2a	0	64	min_win_adv_b2a	1.98
5	first_packet	10.99	35	outoforder_pkts_a2b	0.05	65	zero_win_adv_a2b	1.47
6	last_packet	10.97	36	outoforder_pkts_b2a	1.47	66	zero_win_adv_b2a	1.29
7	total_packets_a2b	2.49	37	pushed_data_pkts_a2b	1.79	67	avg_win_adv_a2b	2.43
8	total_packets_b2a	2.13	38	pushed_data_pkts_b2a	1.82	68	avg_win_adv_b2a	2.14
9	resets_sent_a2b	1.79	39	SYN/FIN_pkts_sent_a2b	0	69	initial_window_bytes_a2b	2
10	resets_sent_b2a	1.79	40	SYN/FIN_pkts_sent_b2a	0	70	initial_window_bytes_b2a	1.95
11	ack_pkts_sent_a2b	1.97	41	req_1323_ws/ts_a2b	0	71	initial_window_pkts_a2b	1.74
12	ack_pkts_sent_b2a	2.14	42	req_1323_ws/ts_b2a	0	72	initial_window_pkts_b2a	1.75
13	pure_acks_sent_a2b	1.95	43	adv_wind_scale_a2b	0.64	73	ttl_stream_length_a2b	2.04
14	pure_acks_sent_b2a	1.87	44	adv_wind_scale_b2a	0	74	ttl_stream_length_b2a	1.88
15	sack_pkts_sent_a2b	1.48	45	req_sack_a2b	0	75	missed_data_a2b	1.11
16	sack_pkts_sent_b2a	1.24	46	req_sack_b2a	0	76	missed_data_b2a	0.57
17	dsack_pkts_sent_a2b	0.62	47	sacks_sent_a2b	1.48	77	truncated_data_a2b	0
18	dsack_pkts_sent_b2a	1.05	48	sacks_sent_b2a	1.24	78	truncated_data_b2a	0
19	max_sack_blks/ack_a2b	1.48	49	urgent_data_pkts_a2b	0	79	truncated_packets_a2b	0
20	max_sack_blks/ack_b2a	1.24	50	urgent_data_pkts_b2a	0	80	truncated_packets_b2a	0
21	unique_bytes_sent_a2b	2.07	51	urgent_data_bytes_a2b	0	81	data_xmit_time_a2b	1.8
22	unique_bytes_sent_b2a	2.06	52	urgent_data_bytes_b2a	0	82	data_xmit_time_b2a	1.84
23	actual_data_pkts_a2b	1.81	53	mss_requested_a2b	1.88	83	idletime_max_a2b	8.97
24	actual_data_pkts_b2a	1.84	54	mss_requested_b2a	1.79	84	idletime_max_b2a	2.77
25	actual_data_bytes_a2b	2.07	55	max_segm_size_a2b	1.97	85	hardware_dups_a2b	2.17
26	actual_data_bytes_b2a	2.07	56	max_segm_size_b2a	1.95	86	hardware_dups_b2a	1.59
27	rexmt_data_pkts_a2b	2.29	57	min_segm_size_a2b	1.99	87	throughput_a2b	2.16
28	rexmt_data_pkts_b2a	1.6	58	min_segm_size_b2a	1.96	88	throughput_b2a	2.18
29	rexmt_data_bytes_a2b	2.33	59	avg_segm_size_a2b	2.03	89	Unnamed	0

TABLE 19.: Feature importance order.

Feature deleted	Accuracy	Δ accuracy	Feature deleted	Accuracy	Δ accuracy
84	0.9978		38	0.9886	0.0018
12	0.9967	0.0010	3	0.9869	0.0017
62	0.9974	-0.0006	61	0.9888	-0.0019
60	0.9971	0.0003	26	0.9861	0.0027
74	0.9962	0.0009	58	0.9851	0.0010
53	0.9955	0.0007	2	0.9819	0.0033
87	0.9955	0.0000	13	0.9845	-0.0027
11	0.9966	-0.0012	59	0.9781	0.0065
6	0.9950	0.0016	8	0.9745	0.0036
14	0.9949	0.0001	57	0.9776	-0.0031
55	0.9943	0.0006	68	0.9711	0.0064
69	0.9940	0.0003	25	0.9426	0.0285
4	0.9945	-0.0006	5	0.9336	0.0090
24	0.9936	0.0010	67	0.9211	0.0124
27	0.9935	0.0001	63	0.8924	0.0288
88	0.9934	0.0001	7	0.8835	0.0088
85	0.9932	0.0001	23	0.8613	0.0222
81	0.9930	0.0003	64	0.8147	0.0467
56	0.9931	-0.0001	82	0.7269	0.0878
70	0.9919	0.0012	21	0.6449	0.0820
1	0.9913	0.0006	83	0.5468	0.0981
22	0.9904	0.0008			

From Table 18. it can be seen that some of the features have zero entropy, these features will be removed because they have little or no impact on the accuracy of malware classification and detection. Features with higher entropy are the features that are selected because they may have an impact on the accuracy. We exclude the timestamp (first packet, feature 5) feature because it does not have an impact on accuracy because the timestamp changes from one PCAP file to another. Afterwards, a series of training cycles were performed, where we deleted one feature with minimal entropy on every cycle.

The DNN classifier was trained for 50 epochs to make analysis time reasonable. The process continues until all the features are sorted in the order of highest to lowest importance. The results are tabulated in Table 19.. The influence of each feature was estimated as a difference in accuracy before and after feature deletion.

We arranged the features in order from highest to lowest entropy and then removed 17 features, listed in Table 20., that had 0 entropy. Figure 20. shows the accuracy as features are removed in increasing order of entropy. Note that the maximum dimensionality is 73 because of the 17 that were removed. Table 22. shows the 14 features that had the most

effect on accuracy.

TABLE 20.: 17 features with zero entropy.

Feature Number	Feature Name
34	zwnd-probe-bytes-b2a
39	SYN/FIN-pkts-sent-a2b
40	SYN/FIN-pkts-sent-b2a
41	req-1323-ws/ts-a2b
42	req-1323-ws/ts-b2a
44	adv-wind-scale-b2a
45	req-sack-a2b
46	req-sack-b2a
49	urgent-data-pkts-a2b
50	urgent-data-pkts-b2a
51	urgent-data-bytes-a2b
52	urgent-data-bytes-b2a
77	truncated-data-a2b
78	truncated-data-b2a
79	truncated-packets-a2b
80	truncated-packets-b2a
89	Unnamed

Figure 20. shows the accuracy as more features (in increasing order of entropy) are removed. The 14 features, as shown in Table 22., were selected and have better performance without compromising accuracy.

Figure 21. shows the change in accuracy as features are removed. For example, if we have three features named 1, 2, 3 and the initial accuracy is 95%, and then remove feature 1 and the accuracy is 94.6%, then the delta when removing feature 1 is 0.4%. If we then remove feature 2 (so both features 1 and 2 are removed) and obtain an accuracy of 94.9%, then the delta is -0.3%, demonstrating that we can have negative values in Figure 21.. The change of accuracy as features are removed is shown in Figure 21..

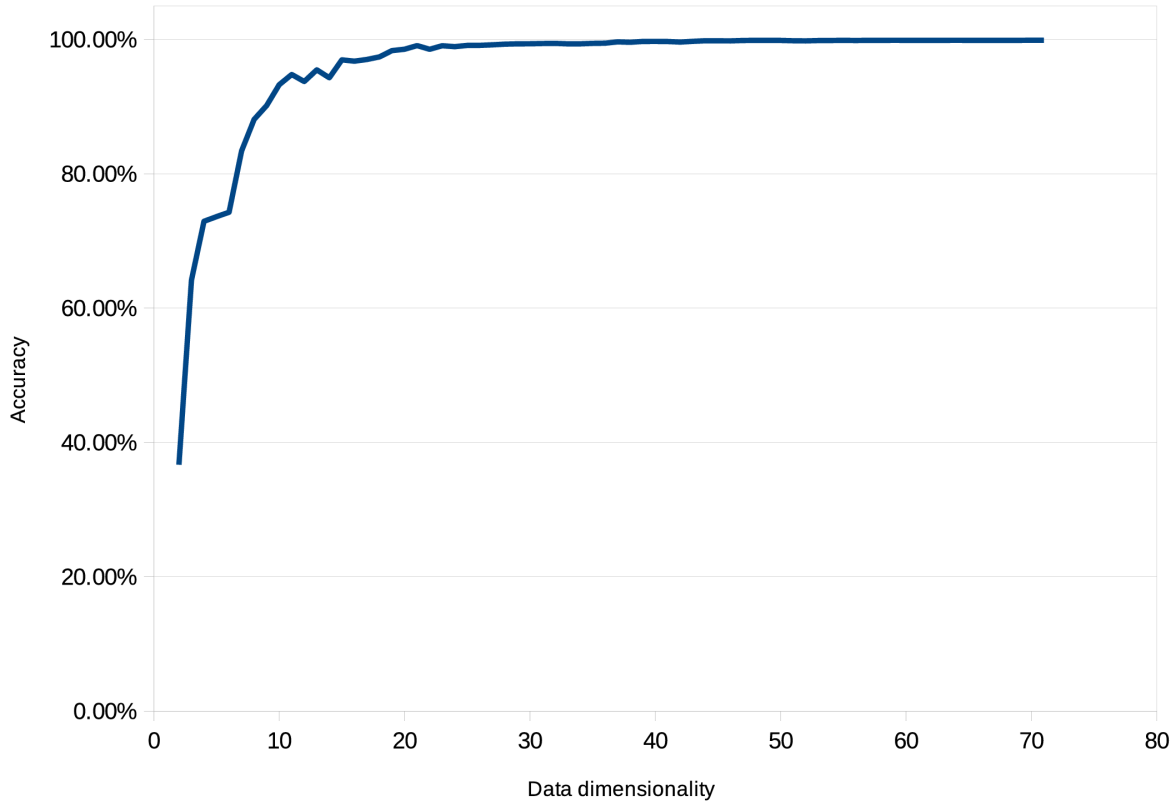

Fig. 20.: The accuracy of the training model based on removing the features with the lowest entropies.

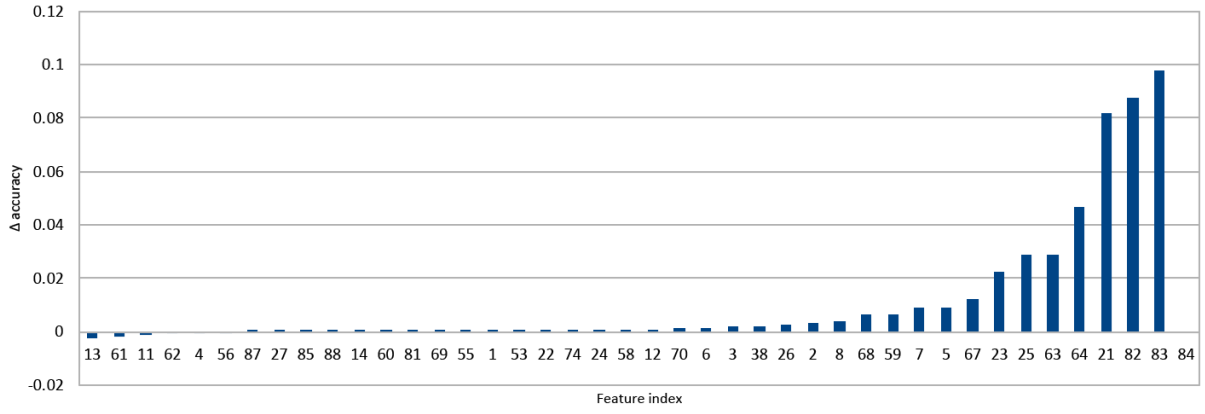

Fig. 21.: Change of delta accuracy when removing features

8.2.2 THE SCORING ALGORITHM

Although the accuracy of the estimation algorithm demonstrates the real influences of features, it takes a long time to process larger datasets. Therefore we propose an automated score based algorithm.

Algorithm 3: Scoring algorithm

```
var scores[n_features]
foreach column in data do
    if linear_nonslimilarity < threshold then
        score=0
    else
        score = entropy(column)*K_entr+ linear_nonsimilarity(column, data)*K_lin+
         informational_gain(column,data)*K_infgain
        scores.push(score)
    end
end
scored_data = data[where score > score_percentile(scores)]
var corellations ;
foreach column1, column2 in combinations(data) do
    corellations.push(correlate(column1,column2));
end
filtered_data = scored_data[where correlation < corr_percentile(correlations)];
```

The algorithm consists of two parts: scoring and correlation. Increasing performance without compromising high accuracy is important. Removing some of the features that are not important will allow us to improve performance without compromising accuracy. We use scoring to measure the importance of each feature and assign a value of importance marking the feature with either high or low importance. The features that are high importance have a large affect on the accuracy if removed, whereas the features with low importance have a minimal impact to the accuracy. Three measures of data are used to score and delete the least important features.

The first measure is Shannon entropy. The second measure is informational gain, which characterizes the amount of information added to the system by a given data vector and is defined as the following:

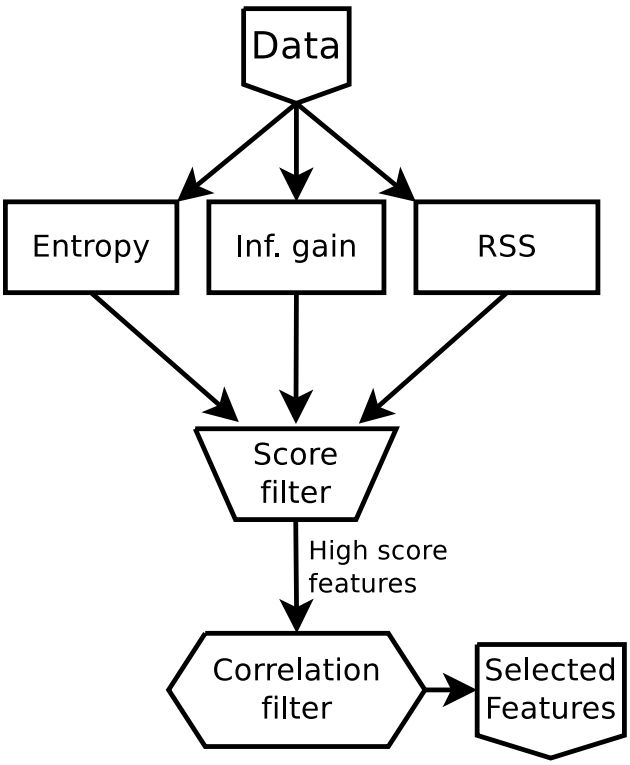

Fig. 22.: The scoring algorithm.

$$G(\vec{v}, A) = H(\vec{v}) - H(\vec{v}|A)$$

where $H(\vec{v})$ is the Shannon entropy of a given data vector $\vec{v}$ and $H(\vec{v}|A)$ is the conditional entropy of $\vec{v}$ with respect to the data matrix A [82].

The third measure is linear nonsimilarity, which describes how one feature is dependent on other features. By definition, the vector $\vec{a}$ is linearly dependent on vectors $\vec{b_1}, \vec{b}_2, \vec{b_n}$ when there exists a nontrivial solution of an equation $\sum \beta_i \vec{b_i} = \vec{a}$ [95]. This can be rewritten in the form of a matrix equation, namely $B\vec{\beta} = \vec{a}$, where B is a matrix composed from vectors $\vec{b_1}, \vec{b}_2, \vec{b_n}$. By finding the inverse B^{-1} the linear dependence coefficients could be found from equation $\vec{\beta} = B^{-1}\vec{a}$. As inverse matrices are defined only for square matrices, Moore–Penrose pseudoinverse can be used. In that manner, the error of linear combination can be estimated as $\sum(B\vec{\beta} - \vec{a})^2$, which is used as a measure for linear nonsimilarity in that case. If the vector can be expressed as a linear combination of the existing (error is close to zero), it is removed.

All three values $(B, \vec{\beta}, \vec{a})$ characterize information contained in vector $\vec{a}$, which are summed up (with coefficients to have similar ranges) to get the score, and only features with the highest score are selected. The final score is adding the three measures (entropy, information gain, and linear nonsimilarity).

Scoring will result in features that have low entropy, low informational gain, and low linear independence being assigned a near zero score. If the feature has a high entropy, low informational gain, and low linear independence, the score will be low. In cases where there is high entropy, high informational gain, and high linear independence, the feature will get a maximal score.

In the second step we use cross-correlation to remove the features with the lowest importance as calculated in the first step. Cross-correlation is basically a dotwise multiplication with further addition. If the data is similar, this operation will lead to large numbers. If not, it will lead to small numbers. From the high scored data, selected in the previous step, the features with the highest cross-correlation are deleted.

The cross-correlation between possible pairs of vectors is calculated from the definition [110]

$$C(\vec{a}, \vec{b})[k] = \sum_{n=0}^{n} a_n^* v_{n+k}$$

Displacement k is selected to be zero as vectors are not temporal series in that case but independent data sequences. The pairs of data-vectors with a cross-correlation higher than the defined percentile are found and filtered, forming the final truncated dataset.

For a given implementation, constants K_{entr}, K_{lin} and $K_{infgain}$ were selected to be 100, 0.1 and 100 respectively to meet the ranges of each other. The threshold for linear nonsimilarity was selected to be equal to 0.01.

The scoring percentile determines the minimum score under which data-vectors are removed. The correlation percentile determines the maximum cross-correlation each pair could have, one vector of the pair with the higher correlation is removed. A percentile of 80 means that we will take only 20 percent of the highest scoring features, thereby removing 80 percent of the total features. To examine the general behavior of the feature selection algorithm together with influences of both parameters, two sets of experiments were conducted. In the first step, the features were selected from the 90 feature dataset varying one of the parameters in each set. In the second step the DNN classifier was trained on the obtained reduced dataset and results were analyzed.

The above-mentioned 90 feature dataset generated from the CTU-13 dataset was processed with the suggested algorithm. The algorithm were adjusted based on the scoring and correlation percentiles.

The loss is calculated on training and its interperation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage, it is a summation of the errors made.

8.3 ALTERNATIVE FEATURE SELECTION TOOLS

This section provides an overview of alternative feature selection tools, including using a decision tree, correlation, the Adaboost classifier, ICAP, and mutual information.

8.3.1 DECISION TREE (RANDOM FOREST)

Sugumaran et al. [101] and Fleuret [55] have suggested the use of decision tree classifiers for evaluation of feature importance. For comparison, this experiment was conducted using the Extremely Randomized Forest algorithm. The implementation from Scikit-Learn library [90], namely ExtraTreesClassifier was used. The decision tree forest was built on the same CTU-13 dataset and was used for feature importance evaluation. The value of the obtained feature importance from highest to lowest for our 90 features is shown in Figure 23..

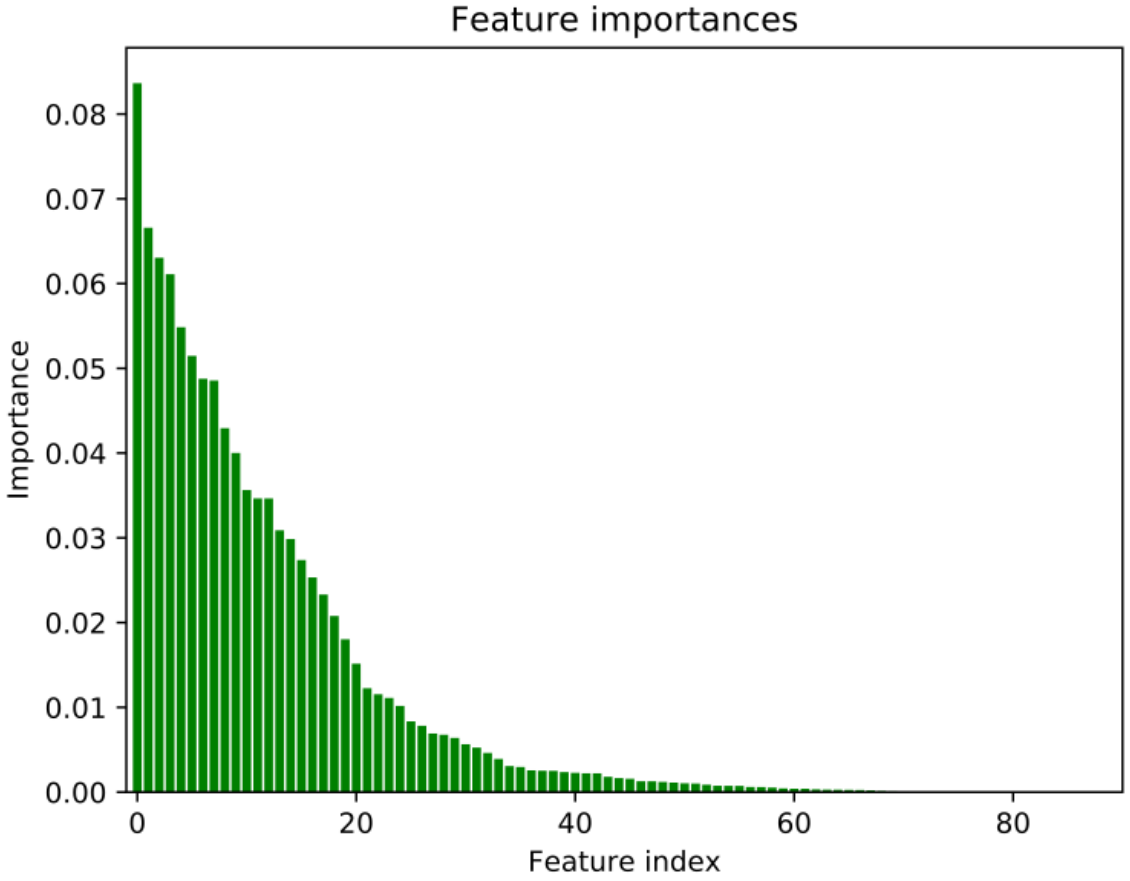

Fig. 23.: Obtained importances distribution.

8.3.2 FAST CORRELATION BASED FILTER

Fast Correlation Based Filter (FCBF) is a statistical measure describing the association between random variables and quantifies how they are dependent on each other. In general a feature is good if it is relevant to the class but not redundant to any other relevant features. In another words, if the correlation between a feature and the class is high enough to make it relevant to the class and the correlation between it and any other relevant features does not reach a level so that it can be predicted by any of the other relevant features, it will be regarded as a good feature [115]. The Fast Correlation Based Filter algorithm from Yu and Liu [115] was used for feature selection in our case.

8.3.3 ADABOOST CLASSIFIER

The AdaBoost algorithm was suggested to estimate feature importance by Fleuret [55] and Wang [108]. One of the main ideas of the AdaBoost algorithm is to maintain a distribution or set of weights over the training set. The AdaBoost algorithm creates a set of poor learners by maintaining a collection of weights over the training data and adjusts these weights after each weak learning cycle adaptively. In our case the implementation of the AdaBoost Classifier from Scikit-Learn library was used, which is based on ideas from the work of Zou et al. [62]. Firstly, the DNN classifier was trained on the whole dataset, and feature importances were extracted directly from the classifier model. Only nine of them were classified as important, so the number of features was less than in our case (12).

8.3.4 ICAP

Interaction Capping (ICAP) is a supervised feature selection algorithm developed by Aleks Jakulin. It was successfully used by different researchers in the world, showing relatively good results in comparison with other similar algorithms [77]. The implementation from scikit-feature, Python library for feature selection was used [73].

8.3.5 MUTUAL INFORMATION

Mutual Information Feature Selection (MIFS), proposed by Battini [37], uses a measure of importance. Mutual information is the change of uncertainty in the classification problem after the addition of a new feature vector to the dataset [37] and is defined as:

$$I(C, F) = H(C) - H(C|F) \tag{3}$$

where I stands for mutual information, $H(C)$ for classification uncertainty measured as entropy of class probabilities, while $H(C|F)$ represents conditional entropy, i.e. classification

TABLE 21.: Selected features for alternative tools.

Correlation	AdaBoost	ICAP	MIFS	Acc Est	Scoring
36	31	83	83	7	0
65	75	58	36	8	1
86	15	3	18	21	2
28	30	10	34	23	3
20	47	86	39	25	4
16	17	56	40	59	10
48	76	29	41	63	65
9	19	60	42	64	83
74	36	43	44	67	84
53	–	59	45	68	–
18	–	15	46	82	–
43	–	47	49	83	–

uncertainty after addition of feature F. Here, the implementation of MIFS from Li et al. [73] was used.

8.4 RESULTS

8.4.1 SELECTED FEATURES FOR ALTERNATIVE TOOLS

Each feature selection tool was configured to select 12 features with the exception of Adaboost which automatically selected 9 features. Random Forest and Adaboost provided an importance value for each feature.

We the trained the DNN classifier with those features and measured accuracy. The results are shown in Table 21.. The maximum accuracy obtained in the DNN Classifier was 99.83%.

8.4.2 ACCURACY ESTIMATION ALGORITHM

To investigate the entropy-importance connection in more detail, the change in accuracy was visualized together with the entropy of a specific feature in Figure 24..

It shows that entropy can be used for initial feature selection but with some precautions. For example, the highest entropy observed in the dataset contained the timestamp and IP address features of both communicating sides, which definitely should not influence the classification.

We used informational entropy to determine feature importance estimation.

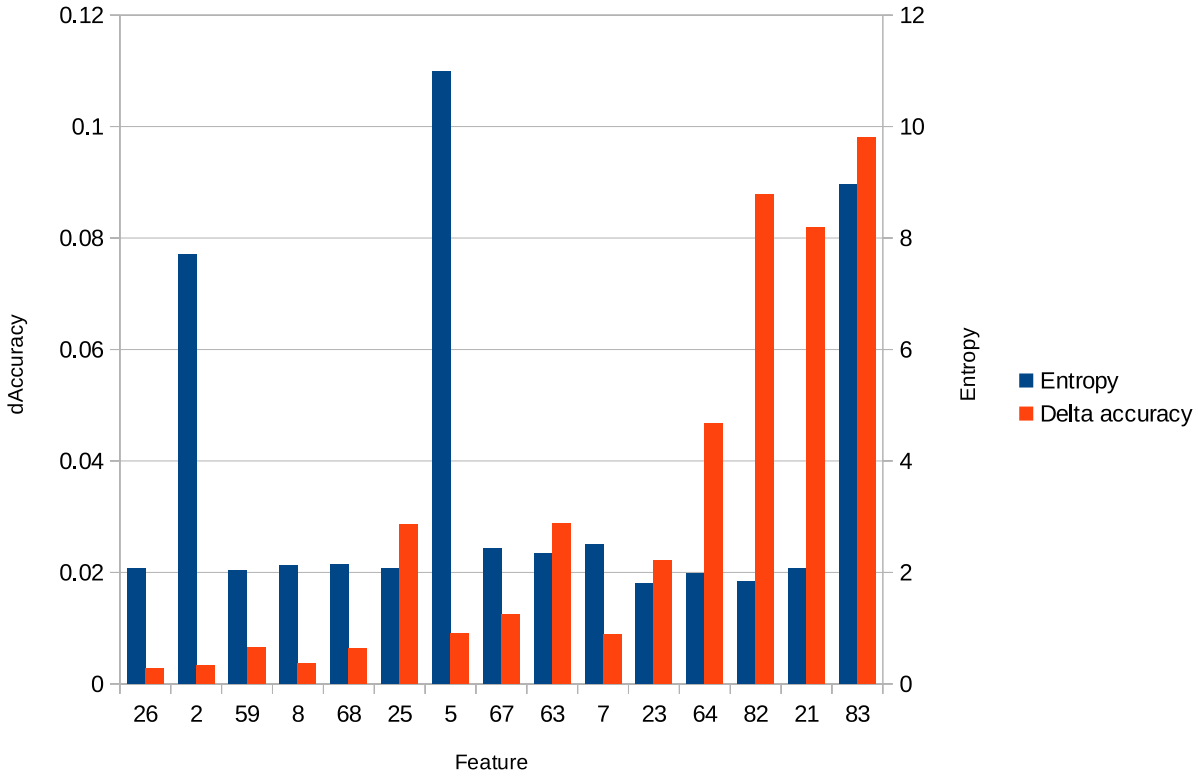

Fig. 24.: Delta accuracy for different features.

In our experiment the most important features are presented in Table 22..

TABLE 22.: The most important features obtained through experimentation.

Feature #	Name	Comments
2	host_b	IP address of B
59	avg_segm_size_a2b	Average size of segment from A to B
8	total_packets_b2a	Total number of packets in stream from B to A
68	avg_win_adv_b2a	The average window advertisement seen from B to A
25	actual_data_bytes_a2b	The number of actual data sent from A to B
5	first_packet	Timestamp of first packet receiving
67	avg_win_adv_a2b	The average window advertisement seen from A to B
63	min_win_adv_a2b	The minimal window advertisement seen from A to B
7	total_packets_a2b	Total number of packets send from A to B
23	actual_data_pkts_a2b	The count of all the packets with at least a byte of TCP data payload.
64	min_win_adv_b2a	The minimal window advertisement seen from B to A
82	data_xmit_time_b2a	Total data transmit time from B to A
21	unique_bytes_sent_a2b	The number of unique bytes sent from A to B
83	idletime_max_a2b	Maximum idle time between consecutive packets

Most of the high importance data are statistics from network streams, which changes for different malware types. Two of them (2 and 5), which are the IP address and timestamp, are misclassified as important, because the IP address and the timestamp will differ from one attack to another. Finally, we increased training iterations to 200 and obtained 99.83% accuracy.

8.4.3 THE SCORING ALGORITHM

The scoring algorithm demonstrated high classification accuracy with a minimal number of features. The algorithm was able to find the set of five data features, which lead to accuracy as high as 99.86%. Any further reduction of features led to a noticeable decrease of accuracy. Stronger data filtering in the scoring step (increase of scoring percentile) was found to be more advantageous over stronger filtering in the correlation step (decrease of correlation percentile) for a given dataset. Generally, the algorithm provided an eighteen-fold decrease, reducing the features from 90 to 5, keeping the accuracy to be close to one. A significant feature reduction lead to decrease of the DNN classifier size and consequently considerably influences the performance of the training time.

Overall, only five similar features were selected by these different methods as it is shown in Table 23. with the last row containing accuracy achieved on that subset. RandomForest

and AdaBoost gave similar results, being similar tree-based methods. Fast Correlation Based Filter (FCBF) and our Accuracy Estimation algorithm (aliased as Acc Est) both gave quite different results with different features selected as important ones. The Scoring algorithm (aliased as Scoring) found features having similarities to almost all other selectors, probably due to its internal utilization of few statistical measures. As ICAP and MIFS uses very different approaches, their result differ strongly from other selectors. Although all of these approaches share five features and similar classification accuracy was achieved, being more than 99.5%, MIFS only achieved about 95% accuracy.

TABLE 23.: Accuracies of the feature selection tools.

Name	Accuracy
Accuracy Estimation	99.83
Scoring	99.81
ICAP	99.77
Randomforest	99.71
Adaboost	99.65
FCBF	99.65
MIFS	94.74

Another interesting set of statistics which can be extracted from the conducted experiments is selected features overlap. The number of feature observations in the results from all of the selectors were counted and are presented in Table 24.. Features which occur less than two times were not included in the table.

TABLE 24.: Feature overlap from the 7 selection algorithms.

Feature Number	Feature Name	Count
36	outoforder-pkts-b2a	4
83	idletime-max-a2b	4
15	sack-pkts-sent-a2b	3
47	sacks-sent-a2b	3
65	zero-win-adv-a2b	3
10	resets-sent-b2a	2
17	dsack-pkts-sent-a2b	2
18	dsack-pkts-sent-b2a	2
19	max-sack-blks/ack-a2b	2
30	rexmt-data-bytes-b2a	2
31	zwnd-probe-pkts-a2b	2
43	adv-wind-scale-a2b	2
59	avg-segm-size-a2b	2
75	min-segm-size-a2b	2
82	data-xmit-time-b2a	2
84	idletime-max-b2a	2
86	hardware-dups-b2a	2

8.4.4 PERFORMANCE ANALYSIS

We compared the scoring and accuracy estimating algorithms to five other feature selection tools. As shown in Table 25. the accuracy prediction algorithm is the most accurate, but it was the slowest because it requires removing one feature at a time and measuring the accuracy to find the importance of each feature. This is done for each of the 90 features. Random Forest was the fastest tool, however accuracy was lower compared to our proposed algorithms. The scoring algorithm and Adaboost both selected only nine features requiring less processing time for classification. Although Adaboost was faster, the scoring algorithm achieved higher accuracy.

TABLE 25.: Performance of feature selection tools.

Name	Time(s)
Randomforest	5.60
Adaboost	23.75
Scoring	112.69
MIFS	134.78
ICAP	143.07
FCBF	172.35
Accuracy Estimation	3932.59

8.4.5 FURTHER COMPARISON

We compared our scoring algorithm to the feature selection algorithm proposed by Yin et al. [113]. Their algorithm is based on deviation and correlation, they were able to reduce the KDD99 dataset from 41 to 21 features. The KDD99 dataset was preprocessed by encoding the categorical features with integers and normalizing the data to the same scale. The dataset was split into 80% for training and 20% for testing, which were then fed into the DNN classifier. We selected 3 types of network traffic: normal traffic, Denial of Service (DoS), and Probe traffic. DoS is an attempt by an attacker to prevent legitimate users from using a service. Probe traffic is a type of attack where the attacker tries to gain information about the target host.

Table 26. shows the results of precision, recall, and F1 score using all 41 features.

TABLE 26.: Original KDD99 dataset with 41 features.

Class	Precision	Recall	F1
DoS	0.9999	1.0000	0.9999
Normal	0.9987	0.9986	0.9986
Probe	0.9914	0.9911	0.9912

We conducted three experiments, the first experiment used Yin et al. selected features and we obtained an accuracy of 0.9991. The precision, recall, and FL values are shown in Table 27..

TABLE 27.: Truncated KDD99 dataset with 21 features from Yin et al.

Class	Precision	Recall	F1
DOS	0.9998	0.9997	0.9997
Normal	0.9981	0.9973	0.9977
Probe	0.9868	0.9880	0.9874

The second experiment used our scoring algorithm which reduced the set of features to 18 and obtained the same accuracy of 0.9991. We were able to reduce the number of features by 7.3% in comparison to to Yin et al. while maintaining the same accuracy. The precision recall, and F1 values are shown in Table 28.

TABLE 28.: Truncated KDD99 dataset with 18 features using the Scoring algorithm.

Class	Precision	Recall	F1
DOS	0.9998	0.9995	0.9996
Normal	0.9976	0.9978	0.9977
Probe	0.9853	0.9710	0.9781

In the third experiment, we were able to further reduce the number of features to 13

while achieving an accuracy of 0.9954. We were able to reduce the number of features by 68.3% which is 19.5% fewer features than the compared feature selection algorithm, while maintaining almost similar accuracy. Reducing the features by 68.3% increases the performance of the DNN malware classification. The precision, recall, and F1 values are shown in Table 29..

TABLE 29.: Truncated KDD99 dataset with 13 features using the Scoring algorithm.

Class	Precision	Recall	F1
DoS	0.9989	0.9993	0.9991
Normal	0.9856	0.9942	0.9899
Probe	0.9224	0.9129	0.9176

8.5 SUMMARY

We proposed two feature selection algorithms utilizing feature reduction without compromising accuracy. The correlation between Shannon entropy and feature importance proves to be a very straightforward and fast method of feature selection. The first algorithm allowed us to reduce features from 90 to 12, that is 7.5 times fewer features. Furthermore, the scoring algorithm allowed us to achieve even higher feature reduction without compromising accuracy, while reducing the features from 90 to 5, that is 18 times fewer features. Both methods could be used for any machine learning application, especially having quadratic or higher complexities. The further reduction of features is possible by using some of the existing feature compression methods such as PCA, SVD, and Autoencoder.

CHAPTER 9

CONCLUSION

As devices continue to rely more and more upon wireless networks, these connections need to be secured and protected faster each day. This research proposes three intrusion detection systems (IDS) using machine learning. The first IDS is able to detect deauthentication attacks, the second detects worms based on their attack phases and via the use of machine learning, while the last detects malware in general based on its network behavior. We evaluate our approach by comparing our results to three different classifiers which were built using the same dataset in terms of accuracy and performance. We propose two methods for improving the performance of these IDS, selecting the optimal features and reducing the features through compression.

1. **Deauth attack prevention using one-time password and Deep Neural Network machine learning approach**

   First, we build an intrusion detection system using machine learning and we propose two mutually exclusive methodologies to prevent deauthentication DoS attacks for IEEE 802.11 networks. One mechanism is based on one time password (OTP). The access point will only accept deauthentication packets having the correct OTP. The other approach is utilizing machine learning. When the detection tool detects the invalid deauthentication packet, it sends a packet to the AP and all nodes indicating they should drop all deauthentication packets for a certain amount of time. This system allows unmodified clients to still get connected. More importantly, the intrusion detection system does not require the nodes to be modified. This makes it backward-compatible and easy to roll out gradually. The Deep Neural Network classifier is shown to be best for the problem and achieved an accuracy of 99.61%.

   For future work, a comparison analysis between other machine learning classifiers is needed and exploring methods to increase performance.

2. **Deep Neural network approach for Worm Traffic Visualization and Analysis**

   Then, we propose a hybrid worm detection and analysis approach. We trained a DNN classifier on the network traffic produced by five worms and normal network traffic.

The average precision of the classifier was 94.1%. We were able to detect a worm attack, which does not exist in our dataset, that has similar network behavior. The hybrid system consist of the visualizer tool based on D3 that visualizes network activities, identifies worm attack phases, and traces the attack. The proposed network intrusion detection system provides better insight on worm activities and better detection of worms that have similar network behavior. In addition, we built a worm traffic generation tool using ns-3.

For future work, we need to expand the dataset to include more worms and perform a comparison between other machine learning classifiers and explore methods to increase performance.

3. **NNIDS - Neural network based intrusion detection system**

   Further, we built an intrusion detection system and a traffic analysis tool that uses 90 features and permits a better accuracy. We use a dual DNN system which is has better performance than a single DNN when carrying out malware detection. Therefore, an intrusion detection system with very low CPU usage is developed, which is flexible and sensitive to malware threats based on their network behavior. We evaluate our method by comparing four different classification methods using the same dataset in terms of accuracy and performance.

   For future work, we need to expand the dataset to include more malware and select the optimal features to increase performance.

4. **Benchmarking Dimensionality Reduction Methods for Malware Classification Based on Network Behavior**

   Then, we perform a comparison analysis of four feature reduction methods through compression for DNNs to classify malware based on their network behavior. We show that SVD performed better than other techniques as it maintains high accuracy and better performance. In addition, t-SNE performed the worst for DNN classifying of malware signatures.

   For future work, we need to select the optimal features before reducing the features.

5. **Feature selection algorithm for the malware classification and detection based on network behavior**

   Finally, we propose two novel feature selection algorithms using feature reduction without compromising accuracy. The first algorithm, accuracy estimation, uses Shannon

entropy to estimate the feature importance The second algorithm, scoring, consists of two parts, scoring and correlation. The first algorithm reduced the features from 90 to 12, which is 7.5 times fewer features. In addition, the scoring algorithm allowed us to achieve a higher feature reduction from 90 to 5, which is 18 times fewer features. Both of these methods can be used for any machine learning application.

For future work, exploring the effects of compressing the data after selecting the optimal features to see if we can increase the performance even more without compromising accuracy.

REFERENCES

[1] Cve-2002-0649. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0649.

[2] Cve-2003-0533. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0533.

[3] Cve-2017-0143. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143.

[4] Scapy-deauth. https://github.com/catalyst256/MyJunk/blob/master/scapy-deauth.py.

[5] KDD dataset. http://kdd.ics.uci.edu/databases/kddcup99, 1999.

[6] Wireshark sample captures. https://wiki.wireshark.org/SampleCaptures, 2000.

[7] IPSumDump. http://read.seas.harvard.edu/~kohler/ipsumdump, 2004.

[8] IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 4: Protected management frames. *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)*, pages 1–111, Sept 2009.

[9] Netresec. http://www.netresec.com/?page=PcapFiles, 2010.

[10] The mobile landscape roundup: 1h 2014. http://www.trendmicro.com/vinfo/us/security/news/mobile-safety/themobile-landscape-roundup-1h-2014, 2014.

[11] Mind the (security) gaps: The 1h 2015 mobile threat landscape. http://www.trendmicro.com/vinfo/us/security/news/mobile-safety/mindthe-security-gaps-1h-2015-mobile-threat-landscape, 2015.

[12] Symantec corp. internet security threat report. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, 2016.

[13] CTU-13 Dataset. https://mcfp.felk.cvut.cz/publicDatasets/CTU-13-Dataset/, 2017.

[14] CTU Malware Capture Botnet. https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-288-1/, 2017.

[15] EternalRocks. https://github.com/stamparm/EternalRocks/blob/master/misc/exploitation.pcap, 2017.

[16] Sasser. https://wiki.wireshark.org/SampleCaptures, 2017.

[17] Wannacry. https://precisionsec.com/wannacry-pcap-smb-445/, 2017.

[18] ContagioDump. http://contagiodump.blogspot.com, 2018.

[19] CTU University Dataset. https://mcfp.felk.cvut.cz/publicDatasets/CTU-13-Dataset/, 2018.

[20] Travis Abrams. Microsoft LSASS buffer overflow from exploit to worm. SANS Network Security, 2004.

[21] Mayank Agarwal, Santosh Biswas, and Sukumar Nandi. Detection of de-authentication DoS attacks in Wi-Fi networks: A machine learning approach. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetrics (SMC)*, pages 246–251. IEEE, 2015.

[22] Mayank Agarwal, Dileep Pasumarthi, Santosh Biswas, and Sukumar Nandi. Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization. *International Journal of Machine Learning and Cybernetics*, 7(6):1035–1051, 2016.

[23] OY Al-Jarrah, A Siddiqui, M Elsalamouny, Paul D Yoo, Sami Muhaidat, and Kwangjo Kim. Machine-learning-based feature selection techniques for large-scale network intrusion detection. In *Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on*, pages 177–181. IEEE, 2014.

[24] Hassan Al-Maksousy and Michele C. Weigle. Hybrid intrusion detection system for worm attacks based on their network behavior. In *Proceedings of the EAI International Conference on Digital Forensics and Cyber Crime (ICDF2C)*, New Orleans, LA, September 2018.

[25] Hassan Al-Maksousy, Michele C. Weigle, and Cong Wang. Nnids: Neural network based intrusion detection system. In *Proceedings of the IEEE International Symposium on Technologies for Homeland Security*, Woburn, MA, October 2018.

[26] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications*, 67:296–303, 2017.

[27] Ali Alatabbi, Moudhi Aljamea, and Costas S. Iliopoulos. Malware detection using computational biology tools. *International Journal of Engineering and Technology*, page 315, 2013.

[28] Winny Thomas Ali Islam, Nicole Oppenheim. SMB Exploited: WannaCry Use of "EternalBlue" . https://www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html, 2017.

[29] Vusal Aliyev. *Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network Department*. PhD thesis, Thesis memory of Computer Science and Engineering Division of Computer Security Chalmers University of technology Göteborg, Sweden, 2010.

[30] F. Amiri. Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4):1184 – 1199, 2011.

[31] Farooq Anjum, Subir Das, Praveen Gopalakrishnan, Latha Kant, and Byungsuk Kim. Security in an insecure WLAN network. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 1, pages 292–297. IEEE, 2005.

[32] Iknoor Singh Arora, Gurpriya Kaur Bhatia, and Amrit Pal Singh. Comparative analysis of classification algorithms on KDD'99 data set. *International Journal of Computer Network and Information Security*, 8(9):34, 2016.

[33] Baber Aslam, M Hasan Islam, and Shoab A Khan. 802.11 disassociation DoS attack and its solutions: A survey. In *Mobile Computing and Wireless Communication International Conference, 2006. MCWC 2006. Proceedings of the First*, pages 221–226. IEEE, 2006.

[34] Stefan Axelsson. Visualization for intrusion detection-hooking the worm. In *8th European Symposium on Research in Computer Security ESORICS 2003*, 2003.

[35] Maroš Barabas, Ivan Homoliak, Michal Drozd, and Petr Hanáček. Automated malware detection based on novel network behavioral signatures. *International Journal of Engineering and Technology*, 5(2):249–253, 2013.

[36] Tawfeeq S Barhoom and Hanaa A Qeshta. Adaptive worm detection model based on multi classifiers. In *Information and Communication Technology (PICICT), 2013 Palestinian International Conference on*, pages 57–65. IEEE, 2013.

[37] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[38] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *USENIX security symposium*, volume 12, pages 2–2. Washington DC, 2003.

[39] Hal Berghel. The code red worm. *Commun. ACM*, 44(12):15–19, December 2001.

[40] Pablo Bermejo, Luis de la Ossa, José A Gámez, and José M Puerta. Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking. *Knowledge-Based Systems*, 25(1):35–44, 2012.

[41] Gèrard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, pages 1063–1095, 2013.

[42] Bimal Kumar Mishra and Navnit Jha. SEIQRS model for the transmission of malicious objects in computer network. *Applied Mathematical Modelling*, 2010.

[43] CM Bishop. Neural networks for pattern recognition.(oxford: University press1995). *Google Scholar*, 1995.

[44] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[45] Jason Brownlee. Classification and regression trees for machine learning - machine learning mastery. `https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/`, 2017.

[46] Edgar D Cardenas. Mac spoofing–an introduction. *GIAC Security Essentials Certification (GSEC)*, 2003.

[47] Yingying Chen, Wenyuan Xu, Wade Trappe, and Yanyong Zhang. Relationship-based detection of spoofing-related anomalous traffic. In *Securing Emerging Wireless Systems*, pages 1–31. Springer, 2009.

[48] Francois Chollet. Keras. `https://keras.io/`, 2015.

[49] Krzysztof Cios. *Deep Neural Networks - A Brief History.* 2017.

[50] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[51] Ping Q Ding, JN Holliday, and Aslihan Celik. Improving the security of wireless lans by managing 802.1 x disassociation. In *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, pages 53–58. IEEE, 2004.

[52] Thomas Dübendorfer, Arno Wagner, Theus Hossmann, and Bernhard Plattner. Flow-level traffic analysis of the blaster and sobig worm outbreaks in an internet backbone. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 103–122, 2005.

[53] Adel Sabry Eesa, Zeynep Orman, and Adnan Mohsin Abdulazeez Brifcani. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications*, 42(5):2670–2679, 2015.

[54] Daniel B Faria and David R Cheriton. DoS and authentication in wireless public access networks. In *Proceedings of the 1st ACM workshop on Wireless security*, pages 47–56. ACM, 2002.

[55] François Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, December 2004.

[56] Maria Garnaeva, Jornt van der Wiel, Denis Makrushin, Anton Ivanov, and Yury Namestnikov. Kaspersky security bulletin 2015. overall statistics for 2015. https://securelist.com/kaspersky-security-bulletin-2015-overall-statistics-for-2015/73038/.

[57] Dragoş Gavriluţ, Mihai Cimpoeşu, Dan Anton, and Liviu Ciortuz. Malware detection using machine learning. In *Computer Science and Information Technology, 2009. IMCSIT'09. International Multiconference on*, pages 735–741. IEEE, 2009.

[58] Fanglu Guo and Tzi-cker Chiueh. Sequence number-based mac address spoof detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 309–329. Springer, 2005.

[59] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[60] J Hall, M Barbeau, and E Kranakis. Using transceiverprints for anomaly based intrusion detection. In *CIIT: The third IASTED International Conference on Communications, Internet, and Information Technology*, pages 22–24, 2004.

[61] Md Al Mehedi Hasan, Mohammed Nasser, Biprodip Pal, and Shamim Ahmad. Support vector machine and random forest modeling for intrusion detection system (IDS). *Journal of Intelligent Learning Systems and Applications*, 6(01):45, 2014.

[62] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[63] Hannes Holm. Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 4895–4904. IEEE, 2014.

[64] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. Efficient constructions for one-way hash chains. *Applied Cryptography and Network Security*, pages 423–441, 2005.

[65] A. Islam, N. Oppenheim, and W. Thomas. EternalBlue SMB protocol exploit. `https://www.fireeye.com/blog/threatresearch/2017/05/smb-exploited-wannacry-useof-eternalblue.html`, 2017.

[66] Henric Johnson, Arne Nilsson, Judy Fu, Shyhtsun Felix Wu, Albert Chen, and He Huang. Sola: A one-bit identity authentication protocol for access control in ieee 802.11. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 768–772. IEEE, 2002.

[67] Zahra Karimi, Mohammad Mansour Riahi Kashani, and Ali Harounabadi. Feature ranking in intrusion detection dataset using combination of filtering methods. *International Journal of Computer Applications*, 78(4), 2013.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, pages 1097–1105, 2012.

[71] T. Le, J. Kim, and H. Kim. An effective intrusion detection classifier using long short-term memory with gradient descent optimization. In *Platform Technology and Service (PlatCon), 2017 International Conference on*, pages 1–6. IEEE, 2017.

[72] Min-Ki Lee, Seung-Hyun Moon, Yong-Hyuk Kim, and Byung-Ro Moon. Correcting abnormalities in meteorological data by machine learning. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 888–893. IEEE, 2014.

[73] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6):94:1–94:45, December 2017.

[74] Pele Li, Mehdi Salour, and Xiao Su. A survey of internet worm detection and containment. *IEEE Communications Surveys & Tutorials*, 10:20–35, 2008.

[75] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21, 2015.

[76] Chibiao Liu. 802.11 disassociation denial of service (dos) attacks. *School of CTI DePaul University*, 2005.

[77] Chuan Liu, Wenyong Wang, Qiang Zhao, Xiaoming Shen, and Martin Konan. A new feature selection method based on a validity index of feature subset. *Pattern Recognition Letters*, 92:1 – 8, 2017.

[78] D. Mahawer and A. Nagaraju. Security and communication networks. *Metamorphic malware detection using base malware identification approach*, 7(11), 2013.

[79] S. Malaspinas. Getting Sassy with Microsoft - An in depth analysis of the LSASRV.dll vulnerability. *Global Information Assurance Certification Paper*, pages pp. 1–56, 2004.

[80] Mishra, Bimal and Pandey, Samir. Fuzzy epidemic model for the transmission of worms in computer network. *Nonlinear Analysis Real World Applications*, 10 2010.

[81] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[82] Andrew Moore. Information Gain. `https://www.autonlab.org/_media/tutorials/infogain11.pdf`, 2003.

[83] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security Privacy*, 1(4):33–39, July 2003.

[84] T Moore. Validating 802.11 disassociation and deauthentication messages. *Submission to IEEE P802.11 TGi*, 802, 2002.

[85] Jose Andre Morales, Peter J. Clarke, and Yi Deng. Identification of file infecting viruses through detection of self-reference replication. *Journal in Computer Virology*, 6:161–180, 2008.

[86] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4:119–128, 2012.

[87] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[88] Thuc D Nguyen, Duc HM Nguyen, Bao N Tran, Hai Trong Vu, and Neeraj Mittal. A lightweight solution for defending against deauthentication/disassociation attacks on 802.11 networks. In *ICCCN*, pages 185–190, 2008.

[89] Shawn Ostermann. Tcptrace, `http://tcptrace.org`, 2005.

[90] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[91] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[92] M. M. Rasheed, S. Badrawi, M. K. Faaeq, and A. K. Faieq. Detecting and optimizing internet worm traffic signature. In *2017 8th International Conference on Information Technology (ICIT)*, pages 870–874, May 2017.

[93] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[94] N. Sarnsuwan, C. Charnsripinyo, and N. Wattanapongsakorn. A new approach for internet worm detection and classification. In *INC2010: 6th International Conference on Networked Computing*, pages 1–4, May 2010.

[95] Georgy Evgen'evich SHILOV and Richard A SILVERMAN. *Linear Algebra. Revised English Ed. Trs. RA Silverman*. Prentice-Hall, 1971.

[96] Rajeev Singh and Teek Parval Sharma. Detecting and reducing the denial of service attacks in wlans. In *Information and Communication Technologies (WICT), 2011 World Congress on*, pages 968–973. IEEE, 2011.

[97] Priyank Singhal and Nataasha Raul. Malware detection module using machine learning algorithms to assist in centralized security in enterprise networks. *arXiv preprint arXiv:1205.3062*, 2012.

[98] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[99] N. Stakhanova, M. Couture, and A. A. Ghorbani. Exploring network-based malware classification. *2011 6th International Conference on Malicious and Unwanted Software*, pages 14–20, Oct 2011.

[100] Ralf C Staudemeyer and Christian W Omlin. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 218–224. ACM, 2013.

[101] V. Sugumaran, V. Muralidharan, and K.I. Ramachandran. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical Systems and Signal Processing*, 21(2):930 – 942, 2007.

[102] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 258–263, Oct 2016.

[103] Xiaoling Tao, Deyan Kong, Yi Wei, and Yong Wang. A big network traffic data fusion approach based on fisher and deep auto-encoder. *Information*, 7(2):20, 2016.

[104] Iain Thomson. Everything you need to know about the petya, er, notpetya nasty trashing pcs worldwide. *The Register*, 2017.

[105] Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, and Takeshi Yagi. Malware detection with deep neural network using process behavior. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 2, pages 577–582. IEEE, 2016.

[106] Ossama A Toutonji, Seong-Moo Yoo, and Moongyu Park. Stability analysis of VEISV propagation modeling for network worm attack. *Applied Mathematical Modelling*, 36(6):2751–2761, 2012.

[107] Haoli Wang and Aravind Velayutham. An enhanced one-bit identity authentication protocol for access control in ieee 802.11. In *Military Communications Conference, 2003. MILCOM'03. 2003 IEEE*, volume 2, pages 839–843. IEEE, 2003.

[108] Ruihu Wang. Adaboost for feature selection, classification and its relation with SVM, a review. *Physics Procedia*, 25:800–807, 2012.

[109] Yao Wang, Wan-dong Cai, and Peng-cheng Wei. A deep learning approach for detecting malicious javascript code. *Security and Communication Networks*, 9(11):1520–1534, 2016.

[110] Eric W Weisstein. Cross-correlation. from mathworld–a wolfram web resource http://mathworld.wolfram.com/correlation.html, 2009.

[111] Tobias Wuechner, Aleksander Cislak, Martin Ochoa, and Alexander Pretschner. Leveraging compression-based graph mining for behavior-based malware detection. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[112] Haidong Xia and José Brustoloni. Detecting and blocking unauthorized access in Wi-Fi networks. In *International Conference on Research in Networking*, pages 795–806. Springer, 2004.

[113] Lizhong Xiao and Yunxiang Liu. A two-step feature selection algorithm adapting to intrusion detection. In *Artificial Intelligence, 2009. JCAI'09. International Joint Conference on*, pages 618–622. IEEE, 2009.

[114] Lifan Xu, Dongping Zhang, Nuwan Jayasena, and John Cavazos. Hadm: Hybrid analysis for detection of malware. In *Proceedings of SAI Intelligent Systems Conference*, pages 702–724. Springer, 2016.

[115] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[116] Mohaddeseh Zakeri, Fatemeh Faraji Daneshgar, and Maghsoud Abbaspour. A static heuristic approach to detecting malware targets. *Security and Communication Networks*, 8(17):3015–3027, 2015.

VITA

Hassan Hadi Latheeth AL-Maksousy
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

E-mail: halma002@odu.edu

Education

- Ph.D., Computer Science, Old Dominion University, Norfolk, VA, USA -Dec. 2018.

  Dissertation: Applying Machine Learning to Advance Cyber Security: Network Behavior Intrusion Detection Systems

- MSc., Computer Science, Kent State University, Kent, Ohio, USA -Dec. 2012.

  Thesis: Performance Analysis Of Multi-Hop Wireless Networks.

Selected Publications

- Hassan Al-Maksousy and Michele C. Weigle. Hybrid intrusion detection system for worm attacks based on their network behavior. In Proceedings of the EAI International Conference on Digital Forensics and Cyber Crime (ICDF2C), New Orleans, LA, September 2018.

- Hassan Al-Maksousy, Michele C. Weigle and Cong Wang. NNIDS: Neural network based intrusion detection system. In Proceedings of the IEEE International Symposium on Technologies for Homeland Security, Woburn, MA, October 2018.

Posters

Benchmarking Dimensionality Reduction Methods for Malware Classification Based on Network Behavior. In Proceedings of the EAI International Conference on Digital Forensics and Cyber Crime (ICDF2C), New Orleans, LA, September 2018.

Workshops

I developed a parallel program for the ClearSpeed CSX600 accelerator for Professor Baker to use as a demonstration program for SIGCSE 2012 Workshop: Parallelism and Concurrency for Data-Structures and Algorithms Courses.

Typeset using LaTeX.