Spring 5-20-2019

Masquerade Detection in Automotive Security

Ashraf Saber
San Jose State University

Masquerade Detection in Automotive Security

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ashraf Saber

May 2019

The Designated Project Committee Approves the Project Titled

Masquerade Detection in Automotive Security

by

Ashraf Saber

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2019

Dr. Mark Stamp	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Fabio Di Troia	Department of Computer Science

ABSTRACT

Masquerade Detection in Automotive Security

by Ashraf Saber

In this paper, we consider intrusion detection systems (IDS) in the context of a controller area network (CAN), which is also known as the CAN bus. We provide a discussion of various IDS topics, including masquerade detection, and we include a selective survey of previous research involving IDS in a CAN network. We also discuss background topics and relevant practical issues, such as data collection on the CAN bus. Finally, we present experimental results where we have applied a variety of machine learning techniques to CAN data. We use both actual and simulated data in order to detect the status of a vehicle from its network packets as well as detect masquerade behavior on a vehicle network.

ACKNOWLEDGMENTS

I would like to thank Dr. Stamp who provided me with guidance and support throughout the Master's project. His mentorship helped me stay on the right path to deliver on the project's requirements.

I would also like to thank my committee members Dr. Thomas Austin and Fabio Di Troia for their valuable time and support for the project.

I would also like to thank my family and friends for their constant support.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Research in automotive security is of increasing importance due to cars being more networked and interconnected than ever before. Providing security to consumers and maintaining their safety requires a considerable focus on automotive security as well as awareness from users of security related issues [1]. In recent years, hackers and security researchers have demonstrated the ability to remotely breach vehicle security systems and gain unauthorized access. In one costly example, the successful hacking of a Jeep Cherokee led to the recall of 1.4 million vehicles in 2015 [1].

Intrusion detection systems (IDS) have been widely studied in the information security research literature. Such systems also have potentially great relevance to vehicle security as IDS would enable us to identify malicious users or activities on a CAN network. Yet, in comparison to IDS in a more general setting, relatively little research has been conducted on IDS in CAN networks.

In this research, we will be applying machine learning techniques to CAN traffic data for classification and masquerade behavior detection. The paper is organized as follows. In Chapter 2, we discuss relevant background topics, including a brief introduction to CAN networks and a similarly brief discussion of IDS in a general setting. We provide a selective survey of IDS in CAN networks and we discuss a few related topics in Chapter 3. Then in Chapter 4 we outline research in the area of masquerade detection and discuss why this is likely a fertile area of research for CAN networks. In Chapter 5 we consider collection issues related to CAN data and, finally, Chapter 6 gives our conclusion and points to directions for future work.

CHAPTER 2

Background

2.1 CAN Bus

Automotive vehicles manufactured in the US after 2008 have a standard internal controller area network (CAN), which is commonly referred to as the CAN bus. Each vehicle has several small electronic control units (ECUs) that are responsible for controlling different car components. These ECUs communicate over the CAN bus, sending and receiving packets during the vehicle's operation. The CAN bus serves to replace a complex wiring harness in vehicles.

CAN is a message broadcast system---a node broadcasts its packet, the receiving node takes the packet, and other nodes should drop the packet. While CAN is conceptually similar to Ethernet, CAN is slower but offers reliable service, in the sense that high priority data will be transmitted. This makes CAN suitable for the challenging and safety-critical environment found in an automobile.

Another important component of vehicle networks is the on-board diagnostic (OBD-II) port. In the majority of vehicles, this port is to the left and below the steering wheel, and in some cars it is visible to the driver, while in others it is hidden. The OBD-II port enables users to check various engine conditions and to sniff traffic on the CAN bus [2].

Automotive security researchers have primarily focused on two aspects of the CAN bus. First, the possibility of breaching the vehicle network has been widely considered, and second, attacks based on packet injection have been studied.

Next, we provide a high level discussion of IDS. Then we turn our attention to a more detailed discussion of IDS, with the emphasis on CAN networks.

2.2 Basics of Intrusion Detection

Intrusion detection systems (IDS) are a fundamental tool in the field of information security. The purpose of IDS is to notify users when their systems are compromised. IDS is typically considered to be distinct from an intrusion prevention system (IPS). As the names indicate, IPS is designed to prevent attacks, whereas IDS is designed to detect attacks once they have occurred---an IDS would be needed when, for example, an IPS fails to prevent an attack.

IDS can operate at the host level or the network level, or some combination thereof. Whether at the host or network level, there are many approaches to detecting a breach. From a high level perspective, anomaly detection and signature detection are the main techniques used by IDS [3]. IDS methods analogous to those used in general networks can be applied to automotive vehicle systems to detect malicious behavior.

2.3 Host Based IDS

A host-based IDS attempts to detect intrusions using information available at the host, without taking network behavior into consideration. That is, host based IDS monitors behavior on a specific host or set of hosts to detect malicious behavior [3]. This method of intrusion detection relies on data stored in logs, audit trails, checksum values, characteristics of user behavior, and so on. One potential advantage of host based IDS is that it may be able to detect the individuals behind the malicious behavior, since logs can reflect the actions of each user [4].

Host based IDS has some disadvantages, depending on the specific implementation. For example, host based IDS might require large storage to maintain the necessary data that the IDS relies on [5]. And typically, multiple hosts need to each have their own host based IDS, which makes setup and configuration challenging [4].

2.4 Network Based IDS

Network based IDS attempts to detect intrusions at the network level. Such an IDS monitors network traffic and typically inspects the header and possibly other aspects of packets passing through the network [4]. An advantage of network based IDS is that it can detect scans (e.g., Nmap scans), DoS attacks, and other network based attacks [3]. This type of IDS is also easy to deploy and install, as compared to a host based system.

One drawback to a network based IDS is that it does not have a clear view of host behavior. In practice, host based and network based IDS are typically both used, at least to some extent. This provides layered security and allows for defense in depth.

2.5 Anomaly vs Signature Detection

Anomaly detection and signature detection can be considered as two broad categories for classifying IDS systems---whether host based or signature based. In anomaly detection, we attempt to model characteristics of the system and when the behavior of the system diverges sufficiently from the model, we flag it as a possible attack. Although challenging, anomaly detection can potentially enable us to detect zero-day attacks [6]. In contrast, signature detection is a form of pattern matching. In such an approach, we extract a pattern or signature from a known attack, then when this pattern is detected, the corresponding attack may have occurred [7]. While relatively accurate and precise, signature scanning can only detect known attacks for which a signature has been previously extracted.

CHAPTER 3

Intrusion Detection

3.1 Intrusion Detection

In this section, our primary focus is to survey selected work on intrusion detection, primarily within the context of CAN networks. We have organized the material among several non-disjoint subtopics.

3.2 Anomaly Detection

As previously mentioned, a strength of anomaly detection is that it holds out the possibility of detecting new attacks based on zero-day vulnerabilities [7]. Generically, in anomaly detection, we train a model on normal behavior and significant deviation from the norm is considered a potential attack [8].

A major issue with anomaly detection is the inherent challenge in trying to model normal benign activities [6]. It is possible for benign activity to occur that was not modeled during the training phase, and it is likely that normal behavior will change over time. These and other similar issues can lead to an excessive number false positives. However, even with the drawback of false alarms, anomaly detection is popular in security research because of the potential to detect zero-day attacks. Detection of zero-day attacks can be viewed as the holy grail of security research [9].

The work of Ye et al. [7] relies on using anomaly detection methods for cyber-attack identification. The authors of [7] discuss the differences between several anomaly detection techniques. First, they consider a process that they refer to as specification-based anomaly detection. In this approach, the benign network events are well defined and properly described. The ordering of network events is also important to establishing a benign condition.

A second technique discussed by Ye et al. [7], is statistical-based anomaly-detection. In this approach, the ordering of events is not important. The model

learns benign behavior from historical data and, thus, the presence of historical data is essential. These authors argue that including the ordering of events guarantees a reduction in false alarms. In their work, they use network data and audit-trail data to train a Markov-chain model for the purpose of detecting intrusions. The model is tested for robustness and accuracy by altering the test data and checking the percentage of false alarms. This approach was shown to be reliable throughout the conducted tests.

Similar to Ye et al. [7], the work of Feng et al. [8] as well as that of Shon and Moon [6] relies on anomaly detection. However, these authors do not use Markov-chain models. They instead rely on the well-known support vector machine (SVM) algorithm. In both of these papers, the authors considered SVM as their starting point and make several modifications to the algorithm in an effort to improve their results.

Shon and Moon [6] use an enhanced SVM algorithm that is applicable to both supervised and unsupervised learning. As for Feng et al. [8], their research combines SVM and clustering, based on a self-organized ant colony network (SOACN) algorithm [8]. Both models were tested for accuracy and yielded low rates of false alarms. These papers indicate that customizing standard machine learning algorithms for the specific task at hand can yield better results than simply using the baseline algorithm.

The work by Tsai et al. [9] considers the usage of different machine learning algorithms in network intrusion detection. The authors of this paper consider 55 research papers on machine learning and intrusion detection in the period between 2000 and 2007. They categorize these papers according to the machine learning algorithm used, as well as their effectiveness in detecting intrusions. On average, hybrid methods where baseline algorithms are modified yielded the top results, both in terms of effectiveness and popularity among researchers [9].

The work of Javaid et al. [10] applies methods of deep learning in anomaly detection. These authors use self-taught learning (STL) on the NSL-KDD dataset, a dataset that was provided by the Canadian Institute for Cybersecurity. These researchers use both unlabeled and labeled data in their models. The first phase of their work is referred to as unsupervised feature learning (UFL), and in this phase unlabeled data is used to train a model based on a sparse autoencoder. In the second phase, the authors consider learning based on labeled data with a softmax function used for classification.

It is worth mentioning that the applications of machine learning in intrusion detection go well beyond computer networks. That is, the same machine learning and deep learning techniques used to detect intrusions in a computer network can be applied to other networks, such as CAN bus networks in vehicles. The type of data used for training will differ, but the mathematical model will remain the same. For example, Naduri and Sherry [11] propose an anomaly detection model for aircraft that relies on recurrent neural networks (RNNs). In their paper, the authors use data from X-Plane simulation software and the X-Plane Software Development Kit (XSDK). Later in this section, we will discuss research that uses RNNs for anomaly detection in CAN networks. The point here is that machine learning algorithms have a wide variety of applications and are easily adapted to different datasets.

In all of the CAN bus IDS papers we have studied, researchers focus on a certain feature to use in training their models. That is, researches extract different features based on what they believe will provide the best representation of ‘‘normal behavior.’’ As we will see, different approaches tend to prove more effective against specific types of attacks.

There are several limitations that researchers face when studying CAN networks. For one, high quality data can be challenging to obtain---this is a topic that we discuss

in more detail in Chapter 5. Another factor is that it may be challenging to obtain a complete understanding of the meaning of all CAN packets. This is because car manufacturers tend to keep such information confidential. In many cases, researchers would need to sniff traffic or simulate traffic and observe the behavior of the system. This analysis might include replaying a packet in question repeatedly to observe its effect on the vehicle.

Anomalous behavior is usually detected by observing the data values in CAN messages, the sequence of the messages, the IDs associated with each packet, the timing of packets, and the frequency of the packets. Below, when we discuss various CAN based IDS systems, the relevance of these various attributes should become clear.

3.3 Sequence Anomalies

In this section, we discuss the specific topic of CAN network anomaly detection research that relies on sequences of data and packet IDs. Wang et al. [12] propose a live anomaly detection system for CAN networks that is based on hierarchical temporal memory (HTM). The goal of this model is to alert the user during an attack. These authors focus on specific data sequences to detect anomalies. Since each data packet is associated to a particular ID, the model processes the data section of each ID. Then, the model predicts the next data packet---the actual packet is compared to the predicted packet to generate a score. If the score is below a certain threshold, then the systems identifies it as an anomaly.

To collect their dataset, Wang et al. [12] sniffed 20 hours of data packets from an Impreza vehicle. The data gathered was divided into sections, with the first 70% used for training and 10% for validation. Finally, the remaining 20% of the data was split into normal data and anomalies. The anomalies consisted of altered normal data,

based on the authors' expectations of what anomalous CAN packets might look like. The authors argue that anomalies can be discerned from the frequency of a specific packet. That is, a packet that appears at a rate different than what is expected can provide a strong indication of anomalous behavior. In addition, anomalies can be present in the data fields of the packets, either in the form of extra data or truncated data. Wang et al. [12] compare their HTM model to a hidden Markov model (HMM) and a recurrent neural network (RNN). Their HTM model is shown to be superior to both other models.

The work of Marchetti and Stabili [13] presents an anomaly detection algorithm that is based on the analysis of ID sequences. These researchers focus primarily on the ID section of a CAN message instead of the data section. In their work, they collected more than 10 hours of CAN bus data. Then they analyzed the possible ID transitions throughout the collected CAN packets and created a transition matrix. The transition matrix includes a boolean value to indicate whether a transition is possible from ID i to ID j, for each possible i and j. This transition matrix can be viewed as a representation of the normal behavior of the CAN network. Thus, the authors are able to detect possible anomalies based on anomalous ID transitions.

An advantage of the system proposed by Marchetti and Stabili in [13] is that the intrusion detection algorithm can operate in either a centralized or distributed mode. The authors emphasize that their algorithm could be implemented inside any of the gateway ECUs that have full visibility of the CAN network, and they highlight that their algorithm can operate in a distributed mode by implementing the algorithm in one ECU in each subnetwork.

To test their algorithm, Marchetti and Stabili [13] conduct two main sets of tests. In the first test, they inject realistic CAN traffic with IDs that have different frequencies. That is, they analyze the frequency of the IDs assigned to each CAN

message. Injected packets tend to alter these expected frequencies. A single message injection was detected with a high rate when the injected packet was from the pool of IDs corresponding to low frequency packets. In contrast, when IDs corresponding to high frequency packets are introduced, they usually go undetected. In other words, high frequency IDs have a low rate of detection, which is intuitive. These results improve when the number of packets used in the attack are increased. A strength of these experiments is that no false positives were generated.

As a second set of experiments, Marchetti and Stabili [13] conducted replay, bad injection, and mixed injection attacks. In the replay attack a set of previous CAN messages is repeated, while for bad injection, a new set of CAN messages is introduced, that is, the injected packets are new to the CAN network under consideration. Finally, the mixed injection attack consists of injecting several random messages. In all cases, these messages are crafted by the attacker without taking transitions into consideration---only the function of the messages is considered. The detection of replay attacks was sporadic and did not follow a clear trend. However, for the bad injections, the detection rate was 100%, and for the mixed injections, the detection percentage increases with the number of messages. For a one message mixed injection, detection was 40%; when this number increased to two CAN messages, the detection rate improved dramatically, reaching virtually 100%.

Malhotra et al. [14] propose an anomaly detection technique based on long short term memory (LSTM) neural networks. The datasets used in this research consist of long patterns of data of variable length. LSTM was chosen due to the long term memory capabilities it provides, which enables such a model to take advantage of long sequences of data. Four datasets were used in the experiments. The first was electrocardiogram (ECG) data, which contained only one anomaly. The second was space shuttle valve data, which contains three anomalous regions. The third was

a power demand dataset, containing actual power consumption data. The fourth and final dataset was multi-sensor engine dataset. This latter dataset includes the behavior of 12 engine sensors.

Malhotra et al. [14] used stacked LSTMs in their models. They showed that no prior exposure or knowledge of pattern duration is required when such an LSTM is used. In these experiments, LSTMs yielded results that were either better than or equivalent to those obtained with standard RNNs.

Taylor et al. [15] also consider an LSTM based anomaly detection method. These authors focused on detecting anomalies in CAN data sequences. Their work demonstrated that LSTM does not necessarily need to understand the target protocol. However, they highlight that LSTM networks have some drawbacks, including the fact that LSTMs deal with each CAN message ID sequence independently. These authors conjecture that if all IDs were considered at the same time---and hence the relationships between ID sequences could be taken into account---then the model would yield significantly higher accuracies. But, such an approach would likely be computationally intensive.

3.4 Physical Anomalies

Integrating the physical environment into an IDS can yield better detection of security breaches. For example, Wasicek et al. [16] proposes a proof of concept anomaly based IDS that they call context aware intrusion detection (CAID). In this paper, the authors build a model that can detect alterations to the physical systems. They rely on sensors to detect changes in the physical medium. The data they measure includes speed, rpm, fuel rate, pedal position, temperature, and fuel-to-air ratio. For anomaly detection, the CAID system relies on an artificial neural network (ANN). The CAID framework consists of three main modules. The first modules are monitors

that are used to collect raw data from the vehicle network. The second modules are known as detectors, which are responsible for analysis. The third and final modules are reporters, which, not surprisingly, communicate the detector result to the user. The CAID framework was tested on a 2015 vehicle---to test the model, a modified chip (used for vehicle tuning) with enhanced parameters was introduced to the vehicle. The CAID framework detected the deviations in behavior.

3.5 Time Windows

Malicious activity on CAN networks would typically impact the timing of packets, as well as the frequency with which certain packets appear. As a result, several papers focus on detecting anomalies in packet timing. In this section, we discuss some examples of this type of research.

Taylor et al. [17] proposed to detect CAN attacks based on packet timing. The authors claim that packets usually arrive at a certain frequency and timing, and thus they consider an anomaly detection system based on historical timing behavior. In their algorithm, the authors measured inter-packet timings over a sliding window. An anomalous behavior is detected whenever a sufficient deviation from historical behavior occurs. Note that the time sequence of CAN messages is essential for this analysis.

To collect their data, Taylor et al. [17] logged the CAN packets of a 2011 Ford Explorer. They made five trips, each lasting five minutes. During those trips the driver did not operate any user controls and they maintained a low speed and came to a complete stop. The first three trips were used in training and the last two trips were used for attack simulation. Attack simulation was conducted by inserting new packets at different timings. The result showed that inter-packet timing yielded strong detection results.

Tomlinson et al. [18] also investigated anomalies in time windows and proposed an IDS that detects intrusions based on deviations in these timings. They utilized three statistical methods, namely, autoregressive integrated moving average (ARIMA), the well known Z score, and a supervised threshold. In their work, they consider non-overlapping windows. They preprocess the data in each window to reduce the necessity of recalculating their various metrics. Each metric was used to classify all broadcasts within the same window.

To collect their data, Tomlinson et al. [18] logged 127 minutes of driving data from an unspecified target vehicle. Then they analyzed the frequency of broadcast packets. They found that packets with a higher priority and low IDs (priority and ID numbers are inversely related) had the least variation in timing, and these were also broadcast with the highest frequency. The analysis of the data showed that the majority of ECUs would broadcast at a consistent rate of at least 100 times per second, while other ECUs broadcast at least 10 times per second. After this analysis was complete, the authors simulated several attacks on the CAN network. To do so, they altered the sniffed CAN packet data to create two simulated malicious datasets. In the first set, they dropped several packets from a normal broadcast, while for the second set, they injected additional packets to existing broadcasts. For testing, the authors applied their three detection methods on a sample consisting of the five highest priority IDs. They compared the broadcast interval against the mean of the normal window, which acted as a supervised threshold against which the Z score and ARIMA results were compared. Overall, the supervised threshold attained the best results, followed by the Z score, then ARIMA.

3.6 Entropy Based Anomalies

According to Marchetti et al. [19] entropy based anomaly detection models can be effective only against cyber attacks that cause a high rate of chaos or randomness. In other words, when the rate of malicious activity increases, entropy based IDS is more likely to be effective. These authors show that entropy based approaches enable the identification of malicious behavior without the necessity of disclosing manufacturer proprietary material related to the meaning of various CAN messages. From the manufacturer's point of view, this could be seen as a significant advantage.

The work of Müter and Asaj [20] also considers an entropy based IDS for CAN networks. These authors measure entropy in the context of coincidence in a given dataset---entropy and the proposed coincidence measure are directly proportional. Müter and Asaj make the point that in CAN networks, there is a low rate of coincidence between packets. Thus, when an attack occurs, their measure of coincidence (i.e., entropy) should increase.

To simulate attacks, Müter and Adaj [20] follow three approaches. In the first case, they slightly increase the frequency of a certain message, and in the second, they flood the network with a specific message. Finally, in their third approach, they consider the ''plausibility of interrelated events,'' where the goal is to understand the correlation between certain events. The specific example they mention is that of a driver in the city who would reach a speed of 60km/h then stop completely due to a stop sign. This higher level of understanding enables their model to detect sporadic single message injection.

3.7 Signature Detection

Signature or pattern matching can be used to detect an attacks. Each attack type, or related sequence of instructions, has a specific pattern and patterns collected

from several known attacks can used to train a model. After training, such a model essentially acts as a meta-signature that is able to detect any of the patterns on which it was trained, and possibly other similar attacks. A disadvantage of such an approach is that it requires constant updating to the model as new attack patterns become available. Another disadvantage is that such a model is unlikely to be effective in defending against zero-day vulnerabilities [7].

3.8 Language Theory Based Detection

Studina et al. [21] proposed an intrusion detection method that relies on formal language theory. They derive attack signatures from different ECUs. Their method then detects malicious message sequences based on attack signatures, which depend on the fact that ECUs operate with consistent rules. Thus, the authors are able to leverage predictable ECU behavior to generate a language that characterizes certain types of attacks.

CHAPTER 4

Masquerade Detection

The masquerade detection problem has been extensively studied in the literature. Specifically, masquerade detection based on UNIX commands has received considerable attention. The seminal work in this field is the Schonlau, et al, paper [22], which was published in 2001. There continues to be considerable interest in the topic, as evidenced by recent papers such as [23, 24, 25, 26, 27, 28, 29, 30, 31].

The author is unaware of any masquerade detection research that has been specifically applied to CAN networks. Hence, the brief survey of masquerade detection that we provide in this chapter does not cite research that is directly related to vehicle networks. However, we believe that masquerade detection is highly relevant to the field of CAN networks, and in our experiments conclusion sections, we return to this topic.

The survey article [32] cites approximately 40 relevant papers published prior to 2009, most of which use the Schonlau dataset. In [32], the authors identify the following general approaches to masquerade detection.

- Information-theoretic
- Text mining
- Hidden Markov model (HMM)
- Naïve Bayes
- Sequences and bioinformatics
- Support vector machine (SVM)
- Other approaches

In the remainder of this chapter, we summarize some of the relevant work in each of the categories listed above, and we discuss a few examples of recent work.

4.1 Information-Theoretic

The original work by Schonlau, et al, in [22] included analysis of a compression technique, based on the fact that commands issued by the same user tend to compress more than those involving an intruder. By the standards of subsequent work, the results are not particularly strong. More recently, related techniques have been pursued in [33, 34, 35], but the results have not improved dramatically.

4.2 Text Mining

In [26], a data mining approach is used to extract repetitive sequences of commands from training data. These sequences are then used for scoring. Other data mining approaches have been studied, including principle component analysis (PCA); for example, in [36], good results are obtained using PCA, although the computational cost is relatively high during training. Another example of a data mining technique being applied to this problem can be found in [37].

4.3 Hidden Markov Model

Hidden Markov model (HMM) techniques are considered in [38] and [39], for example. To date, HMMs have achieved some of the best detection results, and HMMs are often used as a baseline for measuring the effectiveness of proposed techniques.

4.4 Naïve Bayes

A naïve Bayes classifier can be viewed as a static form of an HMM, in the sense that naïve Bayes relies on frequencies, without using sequential information. Such an approach is applied to the masquerade detection problem in [40] and [41]. Although simple, naïve Bayes performs well. Additional relevant work can be found in [42, 43], for example.

4.5 Sequences and Bioinformatics

Sequence-based and bioinformatics-like approaches are, in some sense, at the opposite extreme of naïve Bayes. Recall that naïve Bayes does not account for sequential information, while bioinformatics is focused on extracting sequence-related information.

In the Schonlau, et al, paper [22], a sequence-based analysis is considered. However, the only previous work on masquerade detection involving standard bioinformatics techniques appears to be [23], where the authors use the Smith-Waterman algorithm [44] to create local alignments of sequences. This alignment technique is analogous to a profile hidden Markov model (PHMM), as discussed, for example, in [45]. However, in [23], the resulting alignments are used directly for classification, whereas in a standard PHMM, we use these alignments to generate a model, which is then used for classification. Consequently, a PHMM based detection algorithm is considerably more efficient, while the training is no more costly.

4.6 Support Vector Machine

Support vector machines (SVM) are a class of machine learning algorithms that separate data points using a hyperplanes. The points in the original input space are typically mapped to a higher dimensional feature space, where separation is likely to be much easier. SVMs maximize the margin (i.e., the minimum separation between the sets of points), while keeping the computational cost low [46].

For example, in [47], an SVM-based masquerade detection system achieves results comparable to naïve Bayes. Additional masquerade detection work involving SVMs can be found in [27, 48, 49], where the focus is primarily on improved efficiency, as compared to [47].

4.7 Other Approaches

Several other approaches that do not easily fit into any of the categories above have been considered. However, most of these other approaches have produced relatively poor results. For example, in [50] low frequency (i.e., not commonly used) commands form the basis for detection. In contrast, the paper [29] shows that relying on high frequency commands can yield comparable results.

Among other non-standard techniques, a ‘‘hybrid Bayes one step Markov’’ approach and a ‘‘hybrid multistep Markov’’ method (i.e., a Markov process of order greater than one) are considere in the paper [22]. Neither of these achieve impressive results.

A non-negative matrix factorization (NMF) technique is developed and analyzed in [51]. These NMF results are improved upon in [52], where this approach is shown to achieve reasonable detection results.

4.8 Discussion

The masquerade detection research discussed in this section highlights some important points that are relevant to IDS in CAN networks. First, the topic of masquerade detection seems particularly relevant to CAN networks. That is, an attacker that is aware that an IDS is in use, will likely try to masquerade as a normal user. Thus, a high degree of sensitivity will likely be needed to detect such attacks. Another important point to glean from the discussion above is that a standard dataset in invaluable in such research. The Schonlau dataset is far from perfect, but it has enabled researchers to directly compare their results, and hence the problem of masquerade detection based on UNIX commands has been thoroughly analyzed. A widely available standard dataset for masquerade detection on CAN networks would be an invaluable asset and would help focus research in this area.

CHAPTER 5

Data Collection

Regardless of the IDS technique under consideration, researchers need access to data. In the case of CAN networks, according to Rajbahadur et al. [53] most researchers use simulated datasets, as opposed to data collected for real vehicles. Rajbahadur et al. studied 65 papers dealing with intrusion in vehicle networks and discovered that only 19 of these papers used real datasets. Here, we briefly discuss data related issues. In descending order from the most expensive to the least expensive, we consider the following three methods of obtaining data: real vehicles, ECU testbeds, and simulations.

5.1 Real Vehicles

The most expensive method for obtaining CAN data is using an actual vehicle. Note that a specific make and model would likely be needed to ensure consistency and so that the results could be easily reproduced. The advantage of such data is that it provides access to all in-vehicle systems, including infotainment, air conditioning, GPS, door locks, etc. However, due to the cost, this option is not feasible for most research.

5.2 ECU Testbeds

Another option for data collection is an ECU testbed that includes actual vehicle components. Smith [54] explains how ECUs can be extracted from a vehicle and connected together to enable this type of data collection, and for automotive research in general. ECU benches could be relatively simple, including only a single ECU, or they could be very complex, including most of the components found in an actual vehicle. Such components would include the body control module, an engine control module, an instrument cluster, and so on.

Miller and Valasek [55] explain how ECUs can be connected together to build a test bench. They also show how data can be sniffed from an ECU and details needed to wire test bench components together.

Another option is the portable automotive security testbed with adaptability (PASTA), as developed by Toyota [56]. PASTA is a portable testbed that is the size of a suitcase, and it significantly reduces the barrier to entry for researchers. This testbed contains several ECUs and displays simulated vehicle behavior on a screen. Thus, the data is generated from actual ECUs, and the result of the generated data is displayed to the user through the simulated vehicle behavior.

5.3 Simulation

The least expensive option is to simulate the data in its entirety. Several tools are available to simulate CAN traffic. One of the best known simulation tools is ICSim [57], which is a vehicle simulator that runs on Ubuntu. ICSim allows users to generate CAN traffic and operate a virtual vehicle. It also includes sniffing, replay and data injection capabilities.

CHAPTER 6

Experimental Results

In this section, we discuss a variety of experiments that we have conducted that are relevant to the problem of IDS on CAN networks. We conducted two sets of experiments. The first set deals with classification. Our goal in these experiments was to identify the behavior of a vehicle based on the network packets. In this stage of experimentation, we relied on both real and simulated data. As for the second set of experiments, our goal was to detect masquerade behavior. In this stage, we relied on simulated data. Throughout all experiments, we employ a variety of machine learning techniques.

6.1 Datasets

For the classification experiments, we consider two sources of data. We use a dataset collected from a 2010 Ford Escape [58] and we also consider a simulated dataset that was generated using ICSim [57]. ICSim enables CAN simulation, sniffing and injecting messages. ICSim user interface is displayed in Figure 1. We refer to the Ford Escape data as our ‘‘real’’ dataset, which the ICSim data is our ‘‘simulated’’ dataset.

Each CAN packet consists of eight bytes, where these byte can range over all possible values, that is, from `0x00` to `0xFF`. The real dataset contains CAN messages representing three different states, namely, idle, drive, and park. In contrast, the simulated dataset only contains CAN packets representing the idle and drive states.

For the Masquerade detection experiments, we generated 2 simulated data sets. A summary of the data sets is captured in Table 1. The first set represented the behavior of an authenticated user and that of masquerading users. One trip is the authenticated user and the remaining 10 trips are the masquerading users. Each trip constitutes of several actions. The 10 masquerading trips include 5 with high deviations from

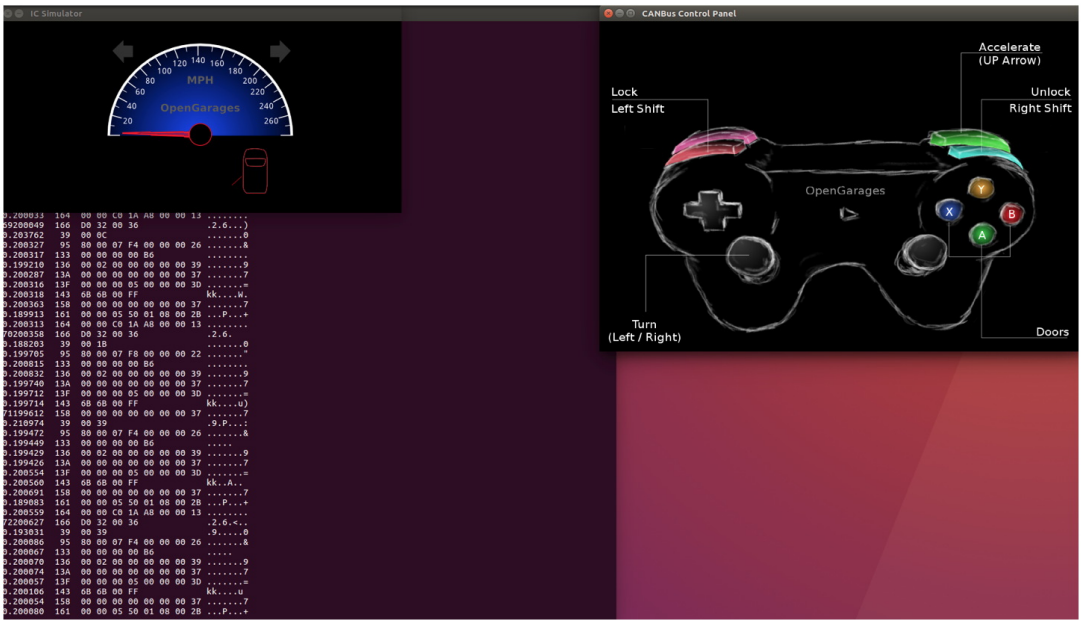


Figure 1: ICSim Display

the authenticated trip as well as 5 with small deviations. The second set consists of 7 simulations. Each simulation completely contains the packets necessary for a single action that the simulator allows. These actions are:

- 20 mph
- 40 mph
- 60 mph
- Left Turn
- Right Turn
- Driver Door Open Close
- Right Passenger Door Open Close

6.2 Feature Extraction

For our initial experiment, each CAN message is converted to its decimal equivalent. We then use these decimal numbers as observations and train models based on sequences of numbers. We also apply a word embedding technique, Word2Vec, to

Table 1: Datasets

Experiments	Classification		Masquerade Detection	
Dataset	1	2	3	4
Source	real	simulated	simulated	simulated
Data	idle/drive/park	idle/drive	11 users	7 actions

CAN packets (as numbers). We compare results obtained with and without Word2Vec conversions. For other experiments, we only used Word2Vec conversion.

Word2Vec is based on a shallow 2-layer neural network and is commonly used to find the context of words in the natural language processing (NLP) domain [59]. In effect, Word2Vec groups common words together, in a form that is suitable as input to other machine learning techniques. For our Word2Vec embedding, we consider CAN messages of length five, based on overlapping sliding windows. We train the Word2Vec model on these words, with the output vectors serving as a feature set in some of the experiments discussed below.

6.3 Experiments

We apply various machine learning models to the CAN packets (treated as numbers), and also experiment with Word2Vec features. Specifically, we consider the following machine learning techniques: k-nearest neighbor (k-NN), hidden Markov models (HMM), a long short-term memory (LSTM) model, deep neural network (DNN), support vector machines (SVM) and Naïve Bayes.

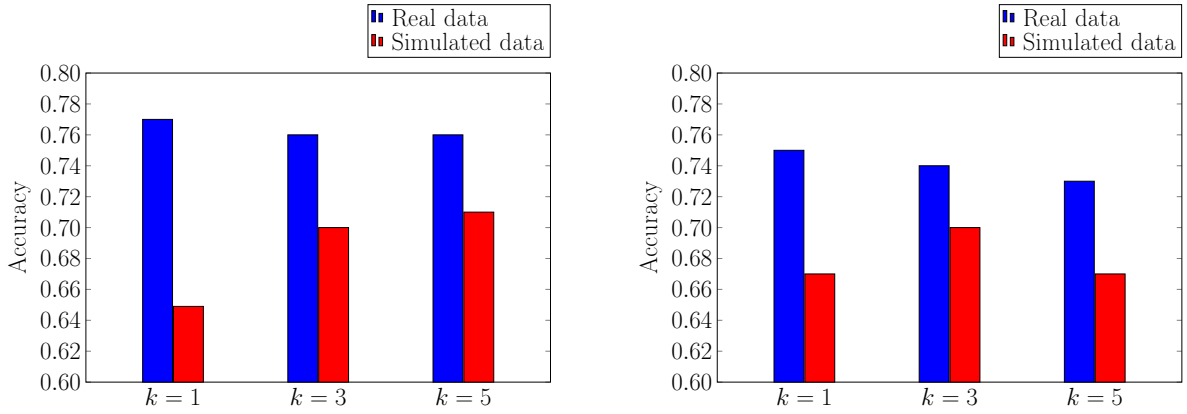
6.4 Classification

In this section, we go over the results of the classification experiments. In these experiments we trained on data sets 1 and 2. The scope was to identify the status of a vehicle from its network packets without having a visual.

6.4.1 k-NN

We first consider k-NN, which we apply to both the numerical CAN packets and the Word2Vec features, with the real data and simulated data being treated as separate experiments. In these experiments, we measure how well we can distinguish "idle" CAN packets from "drive" packets.

Our k-NN results without Word2Vec are summarized in Figure 2, while results for k-NN experiments based on the Word2Vec features are given in Figure 3. From these results, we see that the Word2Vec features are far more informative, yielding much higher accuracies.



(a) No Data Splitting (b) Data Splitting

Figure 2: k-NN Results (CAN Packets)

6.4.2 HMM

A hidden Markov model (HMM) includes an underlying Markov process that is "hidden" in the sense that it is not directly observable. But, we do have access to an observation sequence that is probabilistically related to the hidden Markov process. In the standard terminology, as found in [60], for example, the A matrix drives the underlying (hidden) Markov process, while the B matrix relates the observations to the hidden states, and the π matrix contains the initial state distribution. All three of these matrices are row stochastic.

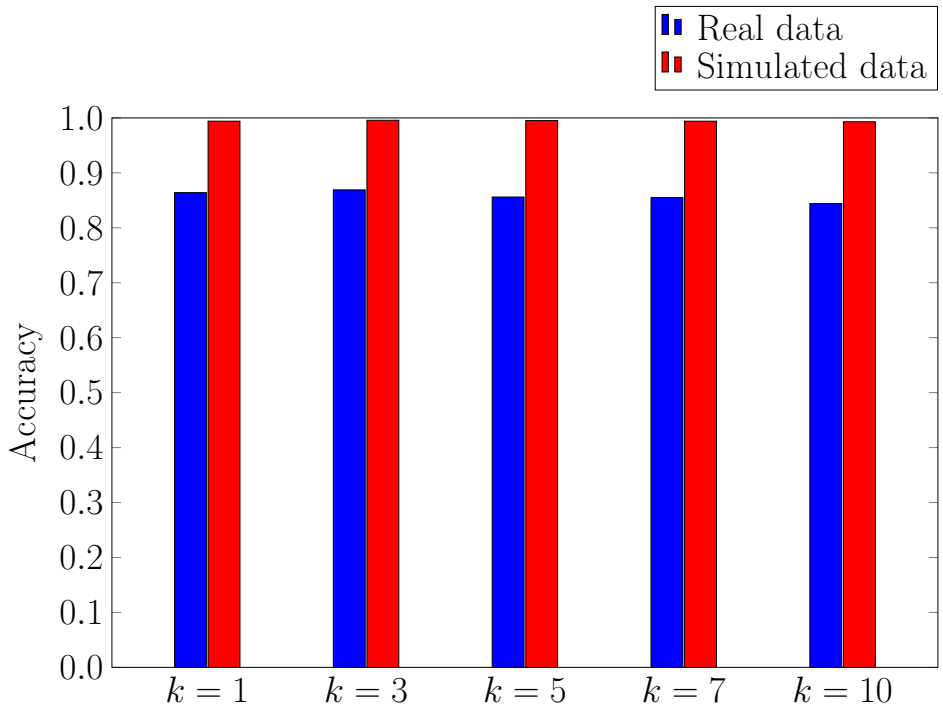
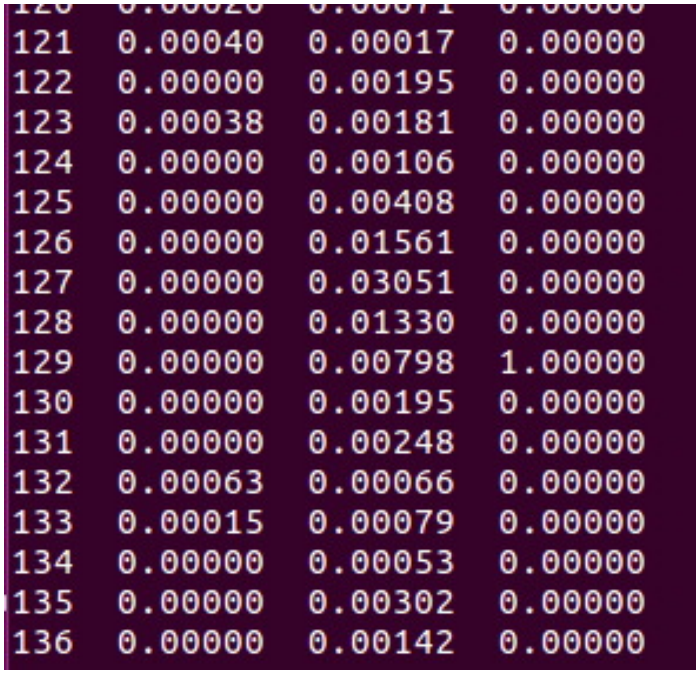


Figure 3: k-NN with Word2Vec

Note that for the experiments discussed in this section, we are operating in a data exploration mode. That is, we are training HMMs and we will then examine the models to see what they tell us about the data.

For our first HMM experiments, we train models treating the available CAN messages as observations, where we considered each byte as one observation. Following the notation in [60], we have $M = 256$ distinct observations, and we train for 500 iterations of the Baum-Welch re-estimation algorithm. In addition, we have $T = 16800$ observations, and we consider a model with $N = 3$ hidden states. The idea here is that these three hidden states should correspond to idle, drive, and park.

A snippet of the final converged B matrix---which relates the hidden states to the observations---is given in Figure 4. It is not immediately clear which column of this B matrix corresponds to which hidden state (idle, drive, park).



121	0.00040	0.00017	0.00000
122	0.00000	0.00195	0.00000
123	0.00038	0.00181	0.00000
124	0.00000	0.00106	0.00000
125	0.00000	0.00408	0.00000
126	0.00000	0.01561	0.00000
127	0.00000	0.03051	0.00000
128	0.00000	0.01330	0.00000
129	0.00000	0.00798	1.00000
130	0.00000	0.00195	0.00000
131	0.00000	0.00248	0.00000
132	0.00063	0.00066	0.00000
133	0.00015	0.00079	0.00000
134	0.00000	0.00053	0.00000
135	0.00000	0.00302	0.00000
136	0.00000	0.00142	0.00000

Figure 4: Converged B Matrix (Pairs of Characters as Observations)

From Figure 4, we see that observation 129 (hex representation of `0x81`), has probability 1.0 in the third state, while all other probabilities for the third state are, of course, zero. This signifies that a message with hex value of `0x81` is in the third state. However, we still do not know what this state actually represents.

To take this further, we trained another HMM with entire messages as observations. In this case, $M = 2830$, $T = 16800$, and we again choose $N = 3$ hidden states. A snippet of the final B matrix for this model is displayed in Figure 5.

Similar results were observed for the third state in this case, with an ID of 0 having the probability of 1. During the preprocessing of the data, each of the $T = 2830$ messages was assigned a unique ID. The ID 0 was assigned to the specific message `81 08 80 00 00 00 00 00`. This packet was from the autopark file, which shows that the HMM was able to correctly identify the packet that indicates when the car is in the park state. This was also in line with the results obtained from the first HMM

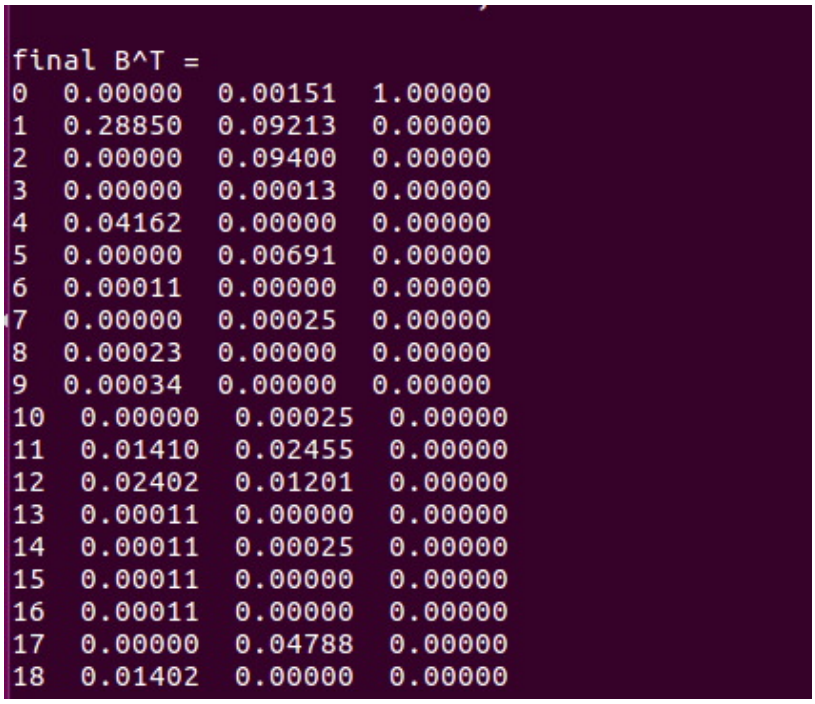


Figure 5: Converged B Matrix (Packets as Observations)

model since we have `0x81` in the data section of this packet, and `0x81` does not appear elsewhere in the data. Again, this packet was only found in the autopark file and was absent from both the drive and idle files. Again, the HMM has associated the third state with the "park" state. This illustrates the strength of an HMM (and machine learning in general) for this data analysis problem.

The A matrix obtained when the HMM was trained on CAN messages is displayed in Figure 6. This matrix gives the transition probabilities between hidden states.

Based on the results discussed above, we observe that the third state is the park state. Further analysis shows that the first state is the idle state while the second is the drive state. As we can see from the A matrix in Figure 6, from the park state, there is only one possible transition and that is to the idle state. This is entirely consistent with the way that a car is actually driven---a car cannot move directly from the park state to the drive state, as it must first pass through the idle state.

```
2827  0.00000  0.00000  0.00000
2828  0.00000  0.00000  0.00000
sum[0] = 1.000000 sum[1] = 1.000000 sum[2] = 1.000000
completed iteration = 499, log [P(observation | lambda)] = -81137.504826

T = 16800, N = 3, M = 2829, iterations = 500

final pi =
 0.00000  0.00000  1.00000 ,  sum = 1.000000

final A =
 0.10455  0.89545  0.00000 , sum = 1.000000
 0.99503  0.00497 -0.00000 , sum = 1.000000
 1.00000  0.00000  0.00000 , sum = 1.000000
```

Figure 6: Converged A and π Matrices

Another key observation is the fact that from the first two rows of the A matrix, we can see that there are frequent transitions from the idle to the drive state and vice versa. These results generated by our HMM nicely illustrate the ‘‘learning’’ aspect of machine learning models, since we never explicitly told the model anything about the states or about driving, yet the model was able to discern this information, which is completely consistent with real-life driving situations.

6.4.3 LSTM

Long short-term memory (LSTM) model is a type of recurrent neural network (RNN) that can be used to predict new information based the previous known information. Unlike other types of RNNs, LSTMs not only take into account recent past information, but also considers a much larger context to predict new information. The intuition behind using LSTMs in the CAN network context is that they are known to work well with time series and sequence data. We experimented with different number of LSTM layers and found that the best results were obtained with five LSTM layers, in which case we were able to obtain an accuracy of 100% for the problem of distinguishing idle and drive CAN packets.

6.4.4 DNN and SVM

Deep neural networks (DNN) are a type of complex artificial neural network (ANN) which consists of multiple layers between the input and output layer. We implemented a DNN, where each layer contains 128 neurons. The model was trained on word embeddings and we tested various numbers of CAN messages. We found that models based on five CAN messages gave us the best results, yielding an accuracy of 99.46%. Again, these results are for the problem of distinguishing between idle and drive CAN packets.

Support vector machines (SVM) was also applied to the data. Using SVMs, we obtained good results on both the real and simulated datasets. Our DNN and SVM results are summarized in Figure 7.

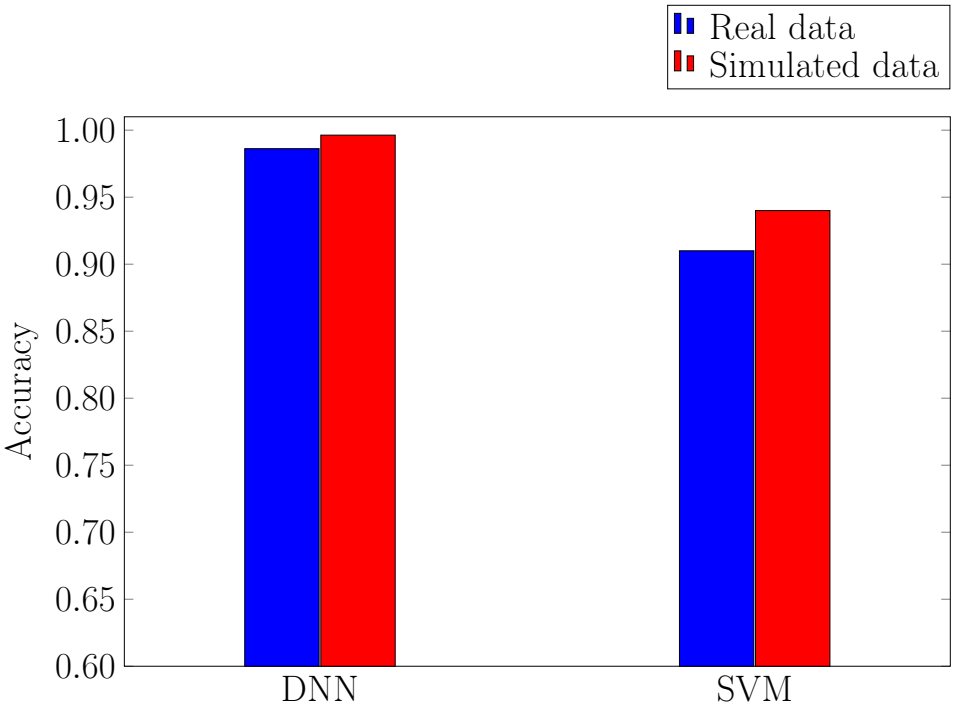


Figure 7: DNN and SVM with Word2Vec

6.5 Masquerade Detection

In this section we go over the results of the masquerade detection experiments. These experiments rely on data sets 3 and 4. These experiments deal with the different simulated users as well as the actions of their simulated trips . All data is simulated in this section and is converted to Word2Vec.

6.5.1 k-NN Speed Detection

The first steps we followed to tackle a masquerade detection problem was using both k-NN and DNN. First, we generated two simulated files, each at a different speed. For one file, the speed ranged from 20 to 40 mph and for the other, the speed ranged from 40 to 60. Then, the data from each file was converted to a vector with Word2Vec and labeled. Afterwards, we applied the two machine learning methods under consideration, attaining high accuracy in both cases. These k-NN and DNN results are summarized in Figure 8. The use case of these tests is to detect changed behavior of a user. For instance if a vehicle owner maintains an average speed that is suddenly altered then this could be an indicator of theft.

6.5.2 k-NN User and State Detection

In this section, we explore the results of the k-NN experiments conducted against the 11 simulated trips and the 7 actions mentioned in section 6.1. First, we trained a KNN model on the simulated CAN traffic for the 11 trips (authenticated user + 10 trips). Afterwards, we split the data into 2 sets: 70% for training and 30% for testing. All trip data was labeled and accuracy calculations relied on whether the prediction meets the initial label or not. The results for this experiment are displayed in Figure 9.

We also applied k-NN to differentiate between the 7 possible trip actions. The scope of this experiment is to check if we can differentiate between these actions with

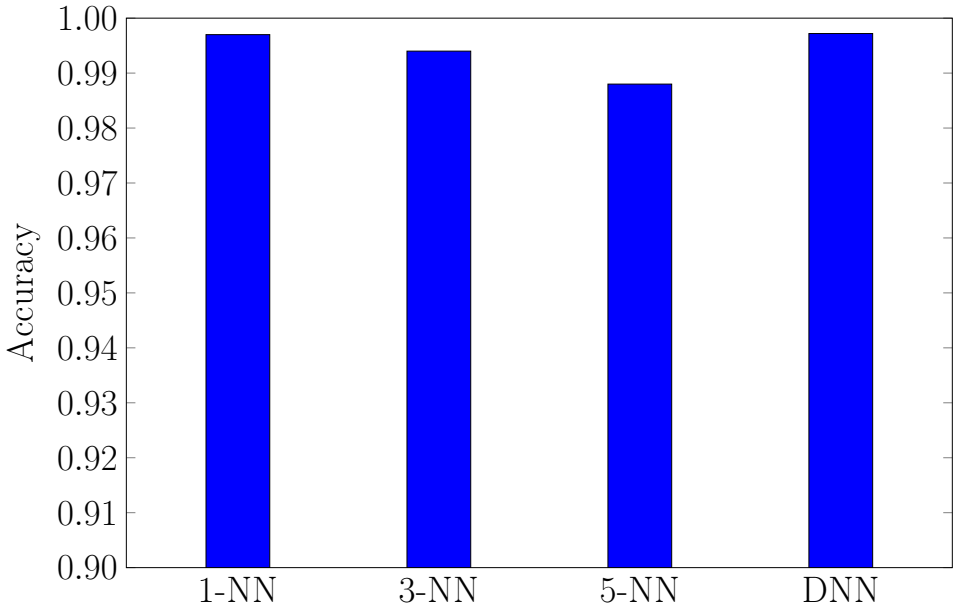


Figure 8: k-NN and DNN Speed Detection

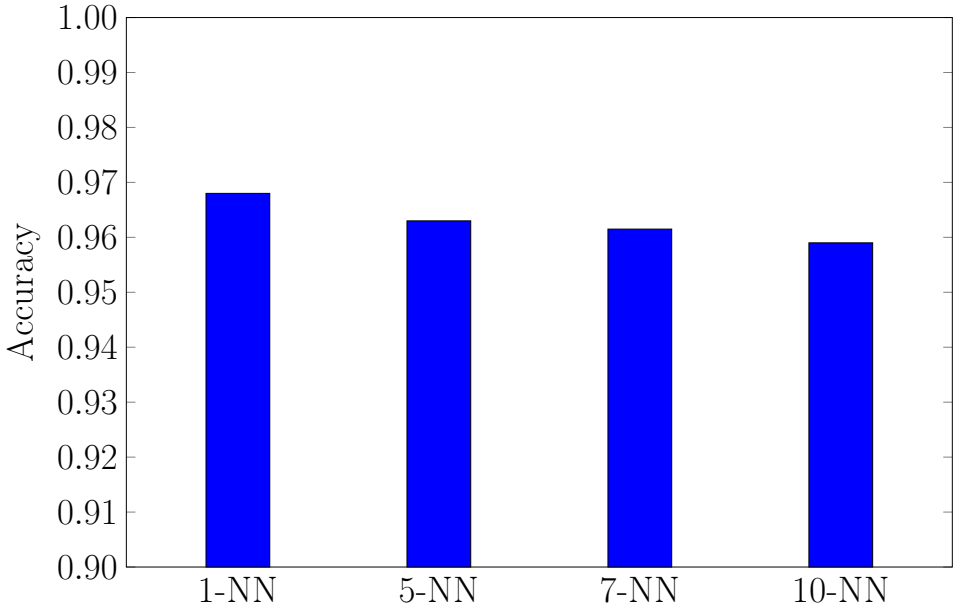


Figure 9: k-NN User Detection Results

high accuracy in order to prepare for the Naïve Bayes tests in the next section. Data was also split into 70% for training and 30% for testing. The results are displayed in Figure 10.

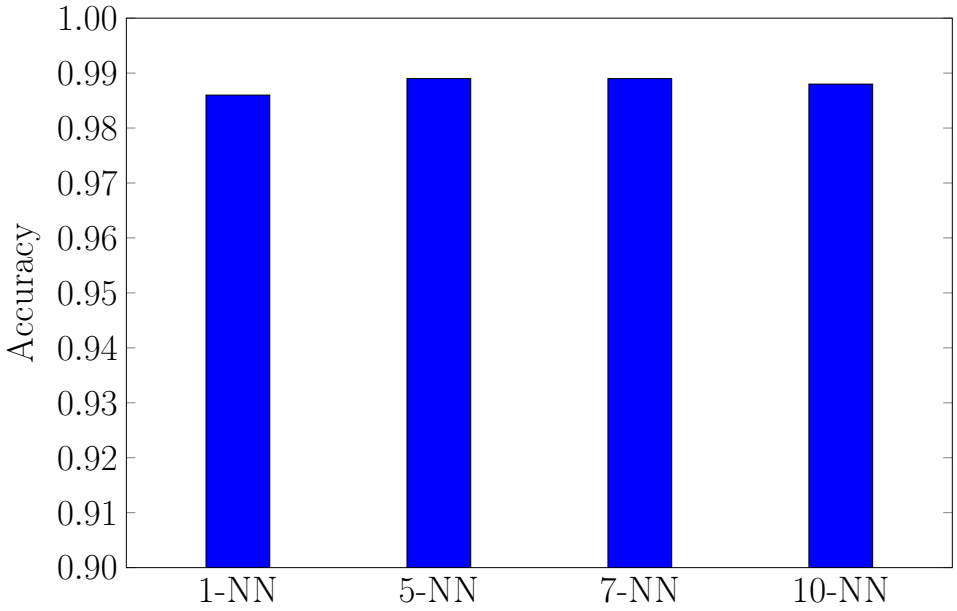


Figure 10: k-NN State Detection Results

6.5.3 Naïve Bayes

In this section, we discuss the results achieved after applying Naïve Bayes to the data. We applied Naïve Bayes in 2 different sets of experiments.

In the first set of experiments, we trained a Naïve Bayes model on the simulated CAN traffic for the 7 trips (each representing a single action or state). The data is split into 2 sets: 70% for training and 30% for testing. Afterwards, we trained a Naïve Bayes model on the simulated CAN traffic for the 11 trips (authenticated user + 10 trips). We also split data into 2 sets: 70% for training and 30% for testing. The trips are labeled and we calculate accuracy based on whether the prediction meets the initial label or not. Results are displayed in Figure 11. In the graph, the User label represents the accuracies attained when we trained on the 11 trips and the States label represents the accuracies of the 7 trips.

The second set of experiments was more complex. We trained a Naïve Bayes model on the simulated data for the 7 possible actions. Then we predicted what each

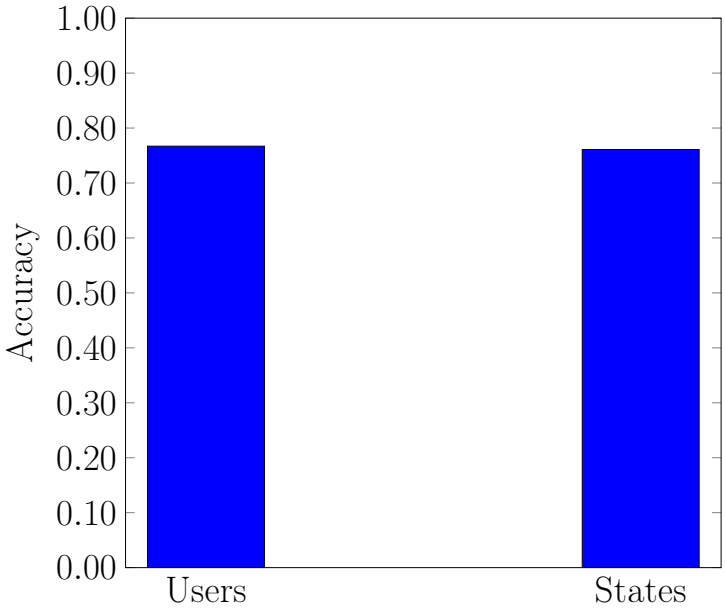


Figure 11: Naïve Bayes Detect User and State

packet indicated for all 11 user trips. Afterwards, we cross referenced the prediction with our initial simulation to identify the accuracy. Thus, we attained 11 accuracies displayed in Figure 12. We can attribute the low accuracies to the fact that the Naïve Bayes Model was not trained on any of the users' trips but rather on the 7 trips representing possible actions.

6.6 Summary

The graph in Figure 13 summarizes the accuracy of the various models considered for the problem of distinguishing the idle state from the drive state. From these results, we seen that LSTM was the clear winner, particularly on the more challenging real dataset.

In Figure 14, we can see the accuracies of both the Naïve Bayes and k-NN implementations for the Masquerade detection problem. We included the accuracies of the user detection experiments only to narrow down the comparison. We can tell that k-NN provides better results than Naïve Bayes in this case.

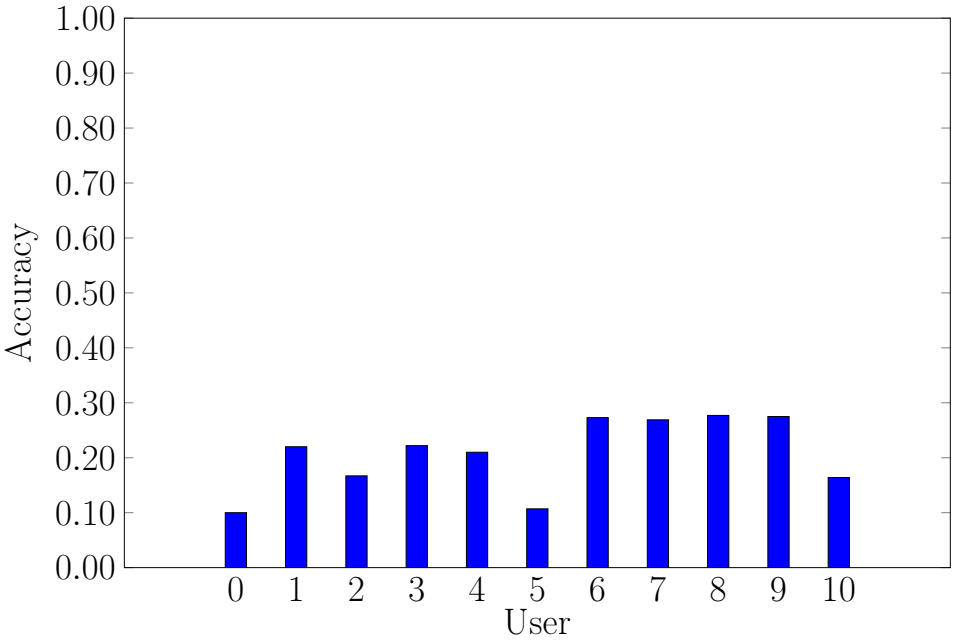


Figure 12: Naïve Bayes Train on 7 Actions

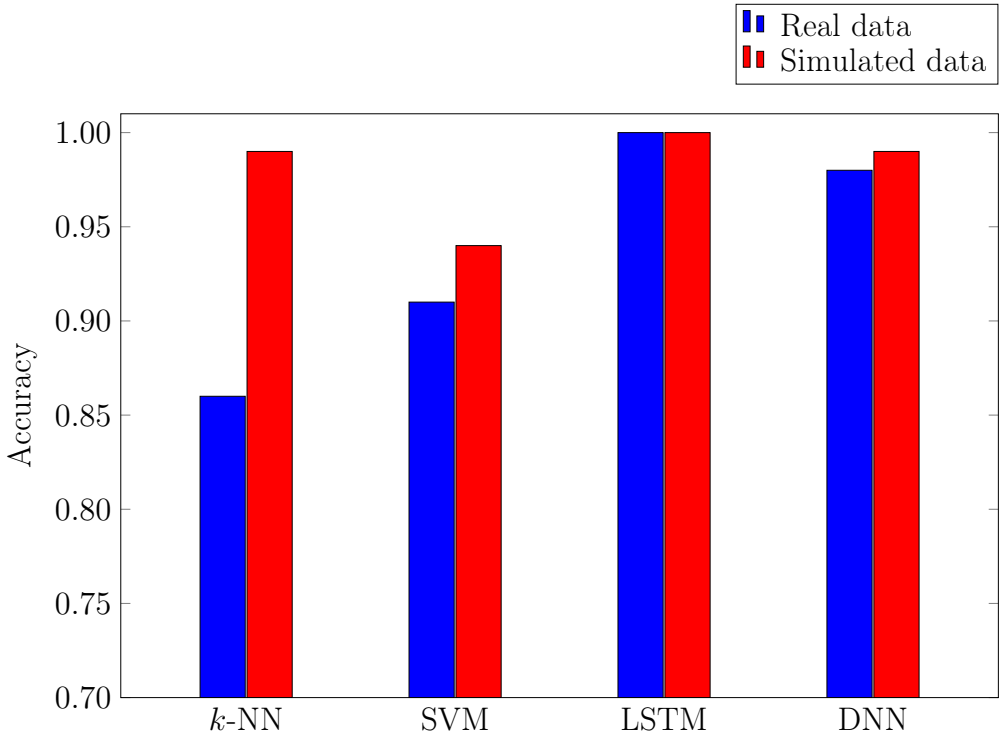


Figure 13: Summary of CAN Traffic Analysis Results

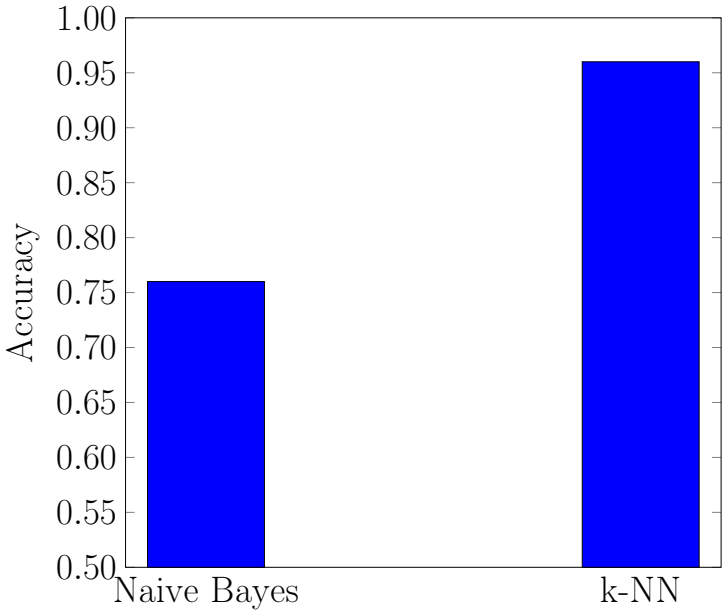


Figure 14: Naïve Bayes vs. k-NN Results for Masquerade Detection

CHAPTER 7

Conclusion and Future Work

In this report, we considered the problem of intrusion detection in CAN networks. We provided a selective survey of research in the field, organized around several major approaches. We also discussed masquerade detection research and suggested that this would be a productive path to follow for future research in CAN network security.

We then did 2 sets of experiments. In the 1st set of experiments, our scope was to identify the status of a vehicle from its network packets without having a visual on the car. In order to do so, we analyzed CAN data using various machine learning models. We applied k-NN, LSTM, HMM, DNN and SVM on 2 data sets. One data set consisted of simulated data and the other consisted of actual data. From the results of the HMM model, we were able to identify important characteristic of CAN packets. From the other models, we were able to properly classify the vehicle status from the packets. All of the models produced good results on at least one of the data sets, with LSTM giving us the best result of 100% accuracy on both data sets under consideration.

In the 2nd set of experiments, our scope was to identify the user driving the vehicle as well as the action in the trip being studied. For this set of experiments we used 2 simulated data sets. The first set consisted of 11 simulated trips representing an authenticating user and 10 masquerading users. In that set, we applied both K-NN and Naïve Bayes. We applied K-NN for different values and got the best accuracy at 96.28%. We got a lower accuracy for Naïve Bayes at 76%. The econd data set we used consisted of 7 files, each has the packets representing only 1 action --from the 7 actions that constituted the 11 user trips. Our goal was to train on the 7 actions and try to predict the sequence of each trip. We applied Naïve Bayes and the accuracies for these tests ranged from 10 % - 27 %.

For future work, we can expand the scope of masquerade detection to include a combination of GPS location paired with CAN traffic, Speed, Infotainment System usage and daily times of operation. This would require the availability of more data.

A hollistic masquerade detection model could rely on several steps. Initially, authenticating the user using a biometric feature--or a mix of several features. Then monitoring CAN traffic paired with GPS as well as infotainment behavior. In addition, we can use sentiment analysis to analyze the tone of voice and facial features.

LIST OF REFERENCES

[1] C. Miller and C. Valasek, ''Remote exploitation of an unaltered passenger vehicle,'' https://www.ioactive.com/pdfs/IOActive_Remote_Car_Hacking.pdf, presented at DEFCON 23, Las Vegas, Nevada, August 6--9, 2015.

[2] C. Smith, *The Car Hacker's Handbook: A Guide for the Penetration Tester*. San Francisco, CA: No Starch Press, 2016, ch. Bus Protocols, pp. 15--35.

[3] M. Stamp, *Information Security: Principles and Practice*, 2nd ed. Wiley, 2011.

[4] SANS Institute, ''Host- vs. network-based intrusion detection systems,'' https://cyber-defense.sans.org/resources/papers/gsec/host-vs-network-based-intrusion-detection-systems-102574, 2000.

[5] M. Reilly and M. Stillman, ''Open infrastructure for scalable intrusion detection,'' in *1998 IEEE Information Technology Conference, Information Environment for the Future*, 1998, pp. 129--133.

[6] T. Shon and J. Moon, ''A hybrid machine learning approach to network anomaly detection,'' *Information Sciences*, vol. 177, no. 18, pp. 3799--3821, 2007.

[7] N. Ye, Y. Zhang, and C. M. Borror, ''Robustness of the Markov-chain model for cyber-attack detection,'' *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 116--123, 2004.

[8] W. Feng, Q. Zhang, G. Hu, and X. Huang, ''Mining network data for intrusion detection through combining SVMs with ant colony networks,'' *Future Generation Computer Systems*, vol. 37, pp. 127--140, 2014.

[9] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, ''Intrusion detection by machine learning: A review,'' *Expert Systems With Applications*, vol. 36, pp. 11 994--12 000, 2009.

[10] A. Y. Javaid, Q. Niyaz, W. Sun, and M. Alam, ''A deep learning approach for network intrusion detection system,'' *ICST Transactions on Security Safety*, vol. 3, pp. 1--6, 2015.

[11] A. Nanduri and L. Sherry, ''Anomaly detection in aircraft data using recurrent neural networks (rnn),'' in *Proceedings of 2016 Integrated Communications Navigation and Surveillance*, ser. ICNS 2016, 2016, pp. 5C2--1--5C2--8.

[12] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu, and X. Cheng, ''A distributed anomaly detection system for in-vehicle network using htm,'' *IEEE Access*, vol. 6, pp. 9091--9098, 2018.

[13] M. Marchetti and D. Stabili, ''Anomaly detection of CAN bus messages through analysis of ID sequences,'' in *IEEE Intelligent Vehicles Symposium*, ser. IV 2017, 2017, pp. 1577--1583.

[14] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, ''Long short term memory networks for anomaly detection in time series,'' in *ESANN*, 2015.

[15] A. Taylor, S. Leblanc, and N. Japkowicz, ''Anomaly detection in automobile control network data with long short-term memory networks,'' in *2016 IEEE International Conference on Data Science and Advanced Analytics*, ser. DSAA 2016. IEEE, 2016, pp. 130--139.

[16] A. R. Wasicek, M. D. Pesé, A. Weimerskirch, Y. Burakova, and K. Singh, ''Context-aware intrusion detection system for autonomous cars,'' https://web.eecs.umich.edu/~mpese/papers/Escar17_CAID_Paper.pdf, 2017.

[17] A. Taylor, N. Japkowicz, and S. Leblanc, ''Frequency-based anomaly detection for the automotive CAN bus,'' in *Proceedings of 2015 World Congress on Industrial Control Systems Security*, ser. WCICSS 2015, 2015, pp. 45--49.

[18] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, ''Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows,'' in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, ser. DSN-W 2018, 2018, pp. 231--238.

[19] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, ''Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms,'' in *2nd International Forum on Research and Technologies for Society and Industry*, ser. RTSI 2016. IEEE, 2016, pp. 1--6.

[20] M. Müter and N. Asaj, ''Entropy-based anomaly detection for in-vehicle networks,'' in *IEEE Intelligent Vehicles Symposium*, ser. IV 2011, 2011, pp. 1110--1115.

[21] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, ''A language-based intrusion detection approach for automotive embedded networks,'' in *The 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015)*, Zhangjiajie, China, Nov. 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01419020

[22] T. Jakobsen, ''A fast method for the cryptanalysis of substitution ciphers,'' *Cryptologia*, vol. 19, pp. 265--274, 1995.

[23] S. E. Coull and B. K. Szymanski, ‘‘Sequence alignment for masquerade detection,’’ *Computational Statistics and Data Analysis*, vol. 52, no. 8, pp. 4116--4131, 2008, https://www.cs.rpi.edu/~szymansk/theses/coull.ms.05.pdf.

[24] S. K. Dash, K. S. Reddy, and A. K. Pujari, ‘‘Adaptive naïve Bayes method for masquerade detection,’’ *Security and Communication Networks*, vol. 4, no. 4, pp. 410--417, 2011.

[25] D. Geng, T. Odaka, J. Kuroiwa, and H. Ogura, ‘‘An n-gram and STF-IDF model for masquerade detection in a UNIX environment,’’ *Journal in Computer Virology*, vol. 7, no. 2, pp. 133--142, 2011, https://link.springer.com/article/10.1007/s11416-010-0143-3.

[26] M. Latendresse, ‘‘Masquerade detection via customized grammars,’’ in *Proceedings of Second International Conference on Intrusion and Malware Detection and Vulnerability Assessment. Lecture Notes in Computer Science (LNCS 3548)*, ser. DIMVA 2005. Vienna, Austria: Springer, July 2005.

[27] Z. Li, Z. Li, and B. Liu, ‘‘Masquerade detection system based on correlation eigen matrix and support vector machine,’’ in *Proceedings 2006 International Conference on Computational Intelligence and Security*, ser. ICCIAS 2006, 2006, pp. 625--628.

[28] U. Premaratne, A. Nait-Abdallah, J. Samarabandu, and T. Sidhu, ‘‘A formal model for masquerade detection software based upon natural mimicry,’’ in *Proceedings of the 2010 5th International Conference on Information and Automation for Sustainability*, ser. ICIAfS 2010, 2010, pp. 14--19.

[29] M. D. Wan, H. Wu, Y. Kuo, J. Marshall, and S. S. Huang, ‘‘Detecting masqueraders using high frequency commands as signatures,’’ in *Proceedings of 22nd International Conference on Advanced Information Networking and Applications*, ser. AINA, 2008, pp. 596--601.

[30] H.-C. Wu and S.-H. S. Huang, ‘‘User behavior analysis in masquerade detection using principal component analysis,’’ *Proceedings 8th International Conference on Intelligent Systems Design and Applications*, pp. 201--206, 2008.

[31] H.-C. Wu and S.-H. S. Huang, ‘‘Masquerade detection using command prediction and association rules mining,’’ in *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications*, ser. AINA ’09, 2009, pp. 552--559.

[32] M. Bertacchini and P. I. Fierens, ‘‘A survey on masquerader detection approaches,’’ in *Proceedings of CIBSI 2009*, 2009, http://www.criptored.upm.es/cibsi/cibsi2009/docs/Papers/CIBSI-Dia2-Sesion5(2).pdf.

[33] M. Bertacchini and P. I. Fierens, ''Preliminary results on masquerader detection using compression based similarity metrics,'' *Electronic Journal of SADIO*, vol. 7, no. 1, pp. 31--42, 2007.

[34] M. Bertacchini and C. Benitez, ''NCD based masquerader detection using enriched command lines,'' in *Proceedings of the IV Congreso Iberoamericano de Seguridad Informatica*, ser. CIBSI 07, 2007, pp. 329--338.

[35] S. Evans, E. Eiland, S. Markham, J. Impson, and A. Laczo, ''MDLcompress for intrusion detection: Signature inference and masquerade attack,'' in *Proceedings of 2007 IEEE Military Communications Conference*, ser. MILCOM 2007, 2007, pp. 1--7.

[36] M. Oka, Y. Oyama, H. Abe, and K. Kato, ''Anomaly detection using layered networks based on eigen cooccurrence matrix,'' in *Proceedings of RAID 2004, Lecture Notes in Computer Science (LNCS 3224*. Springer, 2004, pp. 223--237.

[37] L. Chen and G. Dong, ''Masquerader detection using OCLEP: One-class classification using length statistics of emerging patterns,'' in *Proceedings of the Seventh International Conference on Web-Age Information Management Workshops*, ser. WAIM '06, 2006, p. 5.

[38] T. Okamoto and Y. Ishida, ''Framework of an immunity-based anomaly detection system for user behavior,'' in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R. J. Howlett, and L. Jain, Eds. Springer, 2007, pp. 821--829.

[39] R. Posadas, J. C. Mex-Perera, R. Monroy, and J. A. Nolazco-Flores, ''Hybrid method for detecting masqueraders using session folding and hidden Markov models,'' in *MICAI*, ser. Lecture Notes in Computer Science, vol. 4293. Springer, 2006, pp. 622--631.

[40] R. A. Maxion and T. N. Townsend, ''Masquerade detection using truncated command lines,'' in *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, ser. DSN 2002. IEEE Computer Society, 2002, pp. 219--228.

[41] R. A. Maxion and T. N. Townsend, ''Masquerade detection augmented with error analysis,'' *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 124--147, 2004.

[42] K. H. Yung, ''Using feedback to improve masquerade detection,'' in *Applied Cryptography and Network Security*, ser. ACNS 2003, J. Zhou, M. Yung, and Y. Han, Eds. Springer, 2003, pp. 48--62.

[43] K. H. Yung, ''Using self-consistent naïve-Bayes to detect masquerades,'' in *Advances in Knowledge Discovery and Data Mining*, ser. PAKDD 2004, H. Dai, R. Srikant, and C. Zhang, Eds. Springer, 2004, pp. 329--340.

[44] R. A. Wagner and M. J. Fischer, ''The string-to-string correction problem,'' *Journal of the ACM*, vol. 21, no. 1, pp. 168--173, 1974.

[45] M. Stamp, *Introduction to Machine Learning with Applications in Information Security*. Boca Raton: Chapman and Hall/CRC, 2017.

[46] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. New York, NY, USA: Cambridge University Press, 2000.

[47] K. Wang and S. J. Stolfo, ''One-class training for masquerade detection,'' in *3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security*, ser. DMCS 2003, 2003.

[48] L. Chen and M. Aritsugi, ''An SVM-based masquerade detection method with online update using co-occurrence matrix,'' in *Proceedings of the Third International Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, ser. DIMVA'06. Berlin, Heidelberg: Springer, 2006, pp. 37--53.

[49] H.-S. Kim and S.-D. Cha, ''Empirical evaluation of SVM-based masquerade detection using unix commands,'' *Computers & Security*, vol. 24, no. 2, pp. 160--168, 2005.

[50] M. Schonlau and M. Theus, ''Detecting masquerades in intrusion detection based on unpopular commands,'' *Information Processing Letters*, vol. 76, no. 1--2, pp. 33--38, 2000.

[51] W. Wang, X. Guan, and X. Zhang, ''Profiling program and user behaviors for anomaly intrusion detection based on non-negative matrix factorization,'' in *43rd IEEE Conference on Decision and Control*, ser. CDC 2004, vol. 1, 2004, pp. 99--104 Vol. 1.

[52] C. Mex-Perera, R. Posadas, J. A. Nolazco, R. Monroy, A. Soberanes, and L. A. Trejo, ''An improved non-negative matrix factorization method for masquerade detection,'' in *Proceedings of the 1st Mexican International Conference on Informatics Security*, ser. MCIS 2006, 2006.

[53] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, ''A survey of anomaly detection for connected vehicle cybersecurity and safety,'' in *2018 IEEE Intelligent Vehicles Symposium*, ser. IV 2018, 2018, pp. 421--426.

[54] C. Smith, *The Car Hacker's Handbook: A Guide for the Penetration Tester*, 1st ed. San Francisco, CA, USA: No Starch Press, 2016.

[55] C. Miller and C. Valasek, ''Car hacking: For poories,'' http://illmatics.com/car_hacking_poories.pdf, 2018.

[56] T. Toyama, T. Yoshida, H. Oguma, and T. Matsumoto, ''Pasta: Portable automotive security testbed with adaptability,'' https://i.blackhat.com/eu-18/Wed-Dec-5/eu-18-Toyama-PASTA-Portable-Automotive-Security-Testbed-with-Adaptability[1].pdf, 2018, presented at BlackHat Europe.

[57] ''zombieCraig,'' https://github.com/zombieCraig/ICSim.

[58] C. Miller and C. Valasek, ''Car hacking,'' http://illmatics.com/carhacking.html, 2018.

[59] ''A beginner's guide to Word2Vec and neural word embeddings,'' https://skymind.ai/wiki/word2vec, 2018.

[60] M. Stamp, ''A revealing introduction to hidden Markov models,'' https://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf, 2004.