ABSTRACT
The Controller Area Network (CAN) protocol is the most widely used standard for in-vehicle networks. However, the CAN protocol lacks essential security features (e.g., encryption) which makes it vulnerable to exploits by an adversary. In this paper, we develop CANLP, a Natural Language Processing (NLP)-based intrusion detection system to find whether a transmitted message originated from a legitimate ECU or an adversary. CANLP uses the Term Frequency-Inverse Document Frequency (TF-IDF) to discern complex features associated with CAN data and trains ML models with these features to identify fuzzing, spoofing, and masquerade attacks. When an attack is detected, CANLP identifies the specific ECU on which the attack was mounted. Through extensive experiments on four publicly available vehicle network datasets, we show that CANLP performs attack classification with high F1-score of 0.9974. We also demonstrate using a testbed that CANLP can be deployed for attack detection on resource-constrained hardware a latency of < 0.05 ms.

KEYWORDS
Controller Area Network, Natural Language Processing, TF-IDF

1 INTRODUCTION
The Controller Area Network (CAN) protocol is the most widely used standard for in-vehicle messaging standards lacks critical security features and can be exploited to compromise Electronic Control Units (ECUs), potentially affecting driver and passenger safety. Intrusion Detection Systems (IDSs), particularly based on Machine Learning (ML), offer a scalable solution to these security challenges.

However, deploying ML-based IDSs in real-time can be challenging since: (i) Stealthy attacks mimic the frequency of legitimate CAN messages, making them undetectable by timing-based IDSs; (ii) IDSs analyzing a window of messages are unable to identify the specific ECU and frame of attack; (iii) CAN transmission speeds (up to 1 Mbps) require low latency attack prediction which hinders the deployment of large ML-based IDSs using limited hardware.

In this paper, we develop CANLP, a NLP-Based Intrusion Detection System for CAN, which uses the TF-IDF technique [1] to group binary CAN data into bits of different lengths and extract fine-grained features for real-time attack detection. We exploit the structure of CAN messages to construct a human-readable format according to the CAN Database (DBC) file and interpret this using features obtained via Natural Language Processing (NLP) based techniques. CANLP uses the features to train a Deep Neural Network (DNN) to distinguish between legitimate transmissions and adversarial behavior to perform multi-class attack classification. Our feature extraction demonstrates variability in features among different attacks, thus allowing us to use a small DNN model, which is further reduced in size using quantization [3]. This enables CANLP to perform attack detection in real-time with low overhead.

We evaluate the effectiveness of CANLP through extensive experiments on four publicly available vehicle network datasets. Our IDS deployment on a CAN testbed show that CANLP detects fuzzing, spoofing, or masquerade attacks with a high F1-score of 0.9974 even after model compression. We implement CANLP on a Raspberry Pi 4 Model B running at 1.5GHz with 1GB of RAM achieving classification latency as low as 0.049ms.
2 THREAT AND DEFENSE MODELS

Table 1: Types of attacks that an adversary can carry out on CAN Bus and importance of attributes (ID, Data, and Timestamp) of malicious CAN packets for attack identification. Symbols $\bullet$, $\bigcirc$, and $\bigotimes$ represent full, partial, or no dependency on the particular attribute. CANLP detects Timing Opaque attacks using feature extraction techniques on ID and Data.

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Attack</th>
<th>ID</th>
<th>Data</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing Translucent</td>
<td>Replay</td>
<td>$\bigcirc$</td>
<td>$\bigcirc$</td>
<td>$\bullet$</td>
</tr>
<tr>
<td></td>
<td>DoS</td>
<td>$\bigcirc$</td>
<td>$\bigcirc$</td>
<td>$\bigotimes$</td>
</tr>
<tr>
<td>Timing Opaque</td>
<td>Spoofing</td>
<td>$\bullet$</td>
<td>$\bigcirc$</td>
<td>$\bigotimes$</td>
</tr>
<tr>
<td></td>
<td>Fuzzing</td>
<td>$\bullet$</td>
<td>$\bigcirc$</td>
<td>$\bigotimes$</td>
</tr>
<tr>
<td></td>
<td>Masquerade</td>
<td>$\bullet$</td>
<td>$\bigcirc$</td>
<td>$\bigotimes$</td>
</tr>
</tbody>
</table>

2.1 Adversary Assumptions

We assume that the adversary can intercept all CAN traffic and has ability to inject messages of their own into the bus. The adversary performs attacks through the bus by introducing a new ECU to the bus through the vehicle’s OBD-II port or taking control of a compromised ECU [2]. We consider different levels of compromised ECUs based on the control that the adversary exerts [13].

2.2 Attack Scenarios

The objective of the adversary is to compromise the safety of the vehicle by injecting anomalous messages into the CAN bus by carrying out different classes of CAN attacks shown in Table 1. Attacks can broadly be classified into timing transparent (TT) and timing opaque (TO) attacks. TT (TO) attacks are detectable (not detectable) using a frequency-based method [13].

Due to the nature of TO attacks, specific methods such as a payload-based detector that uses data field, or a side-channel method monitoring the physical layer must be deployed [13]. In this paper, we design CANLP to detect fuzzing, spoofing, and masquerade TO attacks using feature extraction techniques on the CAN payload. The adversary is assumed to perform multiple attacks in a sporadic manner with no observable pattern in attack frequency, while targeting any component of the vehicle to cause abnormal behavior.

2.3 Defense Model

The defense has limited real-time computational resources and uses ML to train a model or re-train publicly available models to achieve high F1-score for attack classification. We assume that the defense is aware that the adversary performs different types of attacks on the CAN Bus and has knowledge of these attacks, but has no knowledge of the order or frequency in which these attacks are performed. The defense has access to a set of CAN packets whose labels are known for model training. Our goal is to develop an IDS that detects fuzzing, spoofing and masquerade attacks on the CAN Bus in real-time and can be deployed inside a vehicle.

3 DATASETS AND CANLP IMPLEMENTATION

We use four publicly available CAN Datasets to evaluate CANLP: (i) HCRL Car Hacking: Attack and Defense Challenge (AD) [7], (ii) HCRL Car Hacking (CH) [11], (iii) HCRL Survival Analysis Dataset for Automobile IDS (SA) [4], and (iv) Real ORNL (Oak Ridge National Laboratory) Automotive Dynamometer (ROAD) CAN Intrusion Dataset [13]. We carry out necessary preprocessing to remove duplicate CAN frames and to balance the dataset in terms of number of samples considered from each class.

Table 2: Number of features and number of trainable DNN parameters for 1-gram, (1,2)-gram and (1,2,3)-gram Character (Char) level TF-IDF features.

<table>
<thead>
<tr>
<th>Granularity of Features</th>
<th>No of TF-IDF Features</th>
<th>Trainable Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char 1-gram</td>
<td>16</td>
<td>10575</td>
</tr>
<tr>
<td>Char (1,2)-gram</td>
<td>272</td>
<td>43523</td>
</tr>
<tr>
<td>Char (1,2,3)-gram</td>
<td>4368</td>
<td>567811</td>
</tr>
</tbody>
</table>

A raw CAN data frame is transmitted in a hexadecimal format which can be decoded using the CAN DBC in a bitwise manner to interpret signal definition as physical parameters [6]. The data in a CAN frame can be segmented into blocks of 8 one-byte values, 64 one-bit values, 1 sixty-four bits value, or any combination of these [6]. Therefore, we choose a character as a term while using N-gram TF-IDF for extracting features. Our proposed character-level CAN data features include all (1,2)-gram byte patterns.

Adversary-manipulated CAN data related to TO attacks might exhibit different bit orderings compared to the patterns found in normal CAN messages. We observe that such consecutive bit patterns of length N can be captured through N-gram units [10]. The total number of required TF-IDF features to represent each CAN frame and number of trainable DNN parameters are shown in Table 2.

The features extracted from CAN frames are used to train five ML models for classification, including Support Vector Machine (SVM), Logistic Regression (LR), Gaussian Naive Bayes (GNB), Decision Tree (DT), Random Forest (RF) [5]. We also train a DNN to effectively extract features at different levels of abstraction. The DNN learns complex patterns compared to traditional ML algorithms [9] allowing easy deployment on resource-constrained hardware [12].

4 EXPERIMENTS AND TESTBED EVALUATION

We evaluated CANLP on four datasets using five different ML algorithms- SVM, LR, GNB, DT, and RF - and the DNN and computed the F1-score. Our results are shown in Table 3. Since DNN achieves best performance compared to other models (highest F1-score for three datasets and close to best F1-score for the other), we choose DNNs for deploying CANLP on the testbed.

To prove feasibility of deploying CANLP in a real scenario, we implemented it on a CAN testbed supporting bandwidths up to 1Mbps, shown in Fig. 1. To simulate bus traffic from different ECUs, we used a Raspberry Pi 3 Model B equipped with a PiCAN2 Duo hat and program it to replay one dataset at a time. A comprehensive analysis of CANLP and for other SOTA hardware-deployed models is presented in Table 4 when tested for F1-score and latency.
Table 4: This table shows the F1-Score and mean and standard deviation (Std) of latency for CANLP and other SOTA hardware-deployed models when tested on the CH dataset. Here, NR refers to Not Reported. The F1-score and latency of the models which perform binary classification (MTH-IDS and MA-QCNN) have been averaged and summed up respectively across all the attacks for fair comparison against the multi-class classification models presented in ACGAN and CANLP (Our Work). While the F1-scores cannot be directly compared due to the incorporation of synthesized data into our dataset, the achieved F1-score remains comparable to those achieved by other SOTA models. Our findings highlight that CANLP outperforms SOTA models in terms of latency and maintains a comparable F1-score across all attacks. Consequently, these results underscore CANLP as the optimal choice for real-time deployment.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hardware Platform</th>
<th>CPU</th>
<th>F1-Score</th>
<th>Latency</th>
<th>Mean</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTH-IDS [14]</td>
<td>Raspberry Pi 3 Model B</td>
<td>1.2 GHz Broadcom BCM2837</td>
<td>0.999</td>
<td>1.722 ms</td>
<td>NR</td>
<td></td>
</tr>
<tr>
<td>MA-QCNN [8]</td>
<td>Zynq FPGA</td>
<td>1.3 GHz Arm Cortex-A53 MPCore</td>
<td>0.996</td>
<td>1.290 ms</td>
<td>NR</td>
<td></td>
</tr>
<tr>
<td>ACGAN [15]</td>
<td>Raspberry Pi 4 Model B</td>
<td>1.8 GHz ARM Cortex-A72</td>
<td>0.992</td>
<td>0.538 ms</td>
<td>0.030 ms</td>
<td></td>
</tr>
<tr>
<td>CANLP</td>
<td>Raspberry Pi 3 Model B</td>
<td>1.2 GHz Broadcom BCM2837</td>
<td>0.997</td>
<td>0.213 ms</td>
<td>0.030 ms</td>
<td></td>
</tr>
<tr>
<td>CANLP</td>
<td>Raspberry Pi 4 Model B</td>
<td>1.8 GHz ARM Cortex-A72</td>
<td>0.997</td>
<td>0.049 ms</td>
<td>0.011 ms</td>
<td></td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation, the Office of Naval Research, European Commission under the Horizon Europe Program, as part of the project LAZARUS, via grants CNS-2153136, N00014-20-1-2636, N00014-23-1-2386, and 10170303 respectively. We thank Omitech S.r.l. for supporting the research of Denis Donadel. The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

REFERENCES