CANShield: Signal-based Intrusion Detection for Controller Area Networks

Md Hasan Shahriar[1], Yang Xiao[1], Pablo Moriano[2⋆], Wenjing Lou[1], and Y. Thomas Hou[1]

[1] Virginia Polytechnic Institute and State University, VA, USA
{hshahriar,xiaoy,wjlou,thou}@vt.edu
[2] Oak Ridge National Laboratory, TN, USA
moriano@ornl.gov

Abstract. Modern vehicles rely on a fleet of electronic control units (ECUs) connected through controller area network (CAN) buses for critical vehicular control. However, with the expansion of advanced connectivity features in automobiles and the elevated risks of internal system exposure, the CAN bus is increasingly prone to intrusions and injection attacks. The ordinary injection attacks disrupt the typical timing properties of the CAN data stream, and the rule-based intrusion detection systems (IDS) can easily detect them. However, advanced attackers can inject false data to the time series sensory data (signal), while looking innocuous by the pattern/frequency of the CAN messages. Such attacks can bypass the rule-based IDS or any anomaly-based IDS built on binary payload data. To make the vehicles robust against such intelligent attacks, we propose CANShield, a signal-based intrusion detection framework for the CAN bus. CANShield consists of three modules: a data preprocessing module that handles the high-dimensional CAN data stream at the signal level and makes them suitable for a deep learning model; a data analyzer module consisting of multiple deep autoencoder (AE) networks, each analyzing the time-series data from a different temporal perspective; and finally an attack detection module that uses an ensemble method to make the final decision. Evaluation results on two high-fidelity signal-based CAN attack datasets show the high accuracy and responsiveness of CANShield in detecting wide-range of advanced intrusion attacks.

Keywords: Controller area networks · Intrusion detection systems · Deep learning.

1 Introduction

Modern vehicles are becoming fully computerized to ensure the driver's safety and convenience. The majority of the cars' critical functionalities involve dedicated microcontroller modules, known as electronic control units (ECUs), which

are connected by controller area network (CAN), a de facto automobile communication standard. The increased connectivity of modern vehicles nonetheless also increases the susceptibility of vehicular systems to remote attacks and message injections. The ability to hijack an ECU allows attackers to inject stealthy messages into the vehicle's internal communication systems. Researchers discovered several remote access points on connected cars and demonstrated that attackers could remotely exploit them to take control of the cars or even disable them. For instance, Miller and Valasek remotely compromised a Jeep and transmited malicious CAN messages, which led to the vehicle malfunctioning on the highway [1]. Moreover, despite the widespread implementation and high reliability, the CAN protocol also remains vulnerable to intruders due to the absence of basic security requirements, especially message authentication. Thus, due to the limited payload length, only the plaintext message is broadcast over the CAN bus, leaving no way to verify where the message comes from or its integrity. Therefore, vehicles using the current version of the CAN protocol remain insecure and attackers could, for instance, instigate sudden braking or acceleration, rendering the lives of passengers and pedestrians at risk [2].

In response, an intrusion detection system (IDS) is usually regarded as the second (and most practical) line of defense given that an attacker can hack into the vehicle's internal communication. In general, there are two types of vehicular IDSs—signature-based and anomaly-based [3]. A signature-based IDS typically formulates detection rules based on the system's behavior of the normal CAN messages and known attacks. Any violations of these rules are regarded as anomaly. In a vehicle, these rules are based on the frequency of the CAN messages, sequence of message IDs, inter-frame time differences, signal values, etc. Such IDSs are mainly effective against known attack footprints. Due to limitations in the rules, these IDSs may show a high false-negative rate in detecting advanced attacks [3, 4]. In addition, the high-dimensional CAN data structure, such as broadcasting different IDs at different frequencies, makes it difficult to extract the effective rules.

The second category of CAN IDSs analyzes anomalies in the CAN data frame. The message IDs and the binary payload data are the main source of data studied in such IDSs [5]. Despite the notable advancement in anomaly-based CAN IDS research in recent years, it is still significantly hampered by several factors [6]. For example, CAN payloads are obfuscated by the original equipment manufacturers (OEMs) for security and privacy reasons. Furthermore, a single payload may contain more than one signal, even encoded in different formats, along with some unused bits. Due to this semantic gap, the anomaly-based IDSs built directly on such obfuscated complex binary CAN message payload tend to suffer against advanced masquerade attacks at the signal level. Therefore, to achieve more robust and semantically concise defense against CAN intrusions, it is imperative to design IDS schemes at the signal level, instead only focusing on the temporal/ID patterns and binary payload. Meanwhile, there are very few number of concrete proposals for the signal-level CAN IDS.

To address this issues, we make the following contributions in this paper:

– We propose a deep learning-based intrusion detection framework, CANShield, to detect advanced and stealthy attacks from high-dimensional signal-level CAN data. It features a data processing technique (pipeline) for the high dimensional CAN signal stream by creating a temporary data queue and use the forward filling mechanism to fill the missing data. This pipeline prepare data stream suitable for the training and testing in the ML-based IDS.
– To make the multidimensional signal-level time series data suitable for the convolution neural network (CNN)-based model, we convert the two-dimensional data queues to multiple images and consider the detection as a computer vision-like problem. Multiple CNN-based autoencoder (AE) models learn the various temporal (short-term and long-term) and spatial (signal-wise) dependencies. Violations in either the temporal or spatial pattern can be detected during the reconstruction process.
– We propose a three-step analysis of the reconstruction loss of CANShield's AE models on selection of detection thresholds for the optimal accuracy, followed by an ensemble-based detector that boosts up the overall detection performance by combining the insights from all the AEs.
– We evaluate CANShield against advanced signal-level attacks using SynCAN [7] and ROAD [6] datasets and compare the results with a baseline model to show the improvements. The results show high effectiveness and responsiveness of CANShield against a wide range of fabrication, masquerade, and suspension attacks on CAN bus.

2 Preliminaries

2.1 Controller Area Network

Robert Bosch GmbH introduced CAN as an automotive communication bus with the latest version (2.0) released in 1991.

CAN Data Frame Format. Among different types of frames, data frame is the default mode for CAN data transmission, as shown in the top portion of Fig. 1. CAN data frame supports up to 8 bytes of payloads with 11 bits of arbitration ID (CAN ID), which can be extended to 29 bits. Every connected ECU broadcasts its message to the CAN bus. However, only one ECU can transmit at a time and the rest stay synchronized to receive the data correctly. The message arbitration mechanism detects and resolves collisions of messages. A message with a higher priority contains a lower binary-encoded CAN ID. Due to different priorities, different CAN IDs usually appear in the CAN bus at different frequencies.

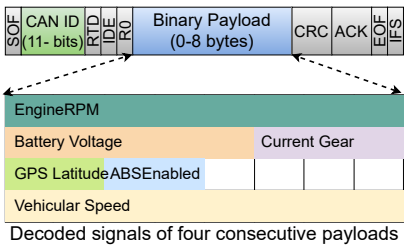


Fig. 1: (Top) CAN data frame syntax. (Bottom) Decoded signals.

Signal-level Representation of CAN Data. The binary payload can be decoded to the signal level using the specific car's database for CAN (DBC) file. The DBC file is a proprietary format,

which is quite challenging to get. However, any reverse engineering-based CAN decoder , such as the CAN-D [8], can provide an approximate DBC file. Such decoding converts the binary payloads to real-valued signals and gives a time series representation. We define the time of each message appearance as one time step, which may contain one or more associated signals along with some unused bits. The bottom part of Fig. 1 shows some samples of signal level representation of a few consecutive payloads. To prepare data input to a ML-based detector, a straightforward idea is to create a structured representation of such data stream, where the columns indicate different signals and rows show each time step. As such a data structure contains many missing entries, it cannot be directly fed to the ML-based IDS models. Thus, designing an appropriate data preprocessing pipeline to account for the missing signal entries is one of the critical challenges in building a signal-level CAN IDS, as we will address in §4.2.

2.2 Convolutional Neural Network-based Autoencoder

CNN is a class of deep neural networks mostly used to analyze image datasets. The network uses kernels or filters that slide along the input data and map the complex relationship among the features. Small filters in CNNs help to learn the local and straightforward patterns first and then combine them into more complicated patterns. Hence, CNN is an extremely powerful tool with a very low degree of connectivity and complexity. Autoencoder (AE) is an unsupervised method that consists of two parts: an encoder that maps an input to a lower-dimensional code and a decoder that reconstructs the closest form of the input from that code. Hence, a bottleneck in the middle of the network can determine the estimated states of the vehicle in a lower dimension. In intrusion detection applications, AE plays a vital role. An AE network is first trained on the normal data so that it learns how to reconstruct with minimum loss. The fundamental hypothesis is that intrusions are sufficiently anomalous with respect to the underlying distribution of the training data so that the AE will yield a high reconstruction loss, pointing to a high probability of attack.

3 System Model

The main component of CANShield is a software system that can read a vehicle's CAN messages in real time. It is loaded either on an onboard computing device connected to the OBD-II Port (e.g., laptop, Raspberry Pi) or instantiated in an existing ECU with a relative powerful processor, such as the gateway ECU. For the former case, the onboard computing device includes a CAN protocol stack, allowing monitoring and recording of the raw CAN messages. This can be achieved with open-sourced implementations (i.e., SocketCAN) or commercial CAN data loggers (i.e., CANalyzer). CANShield is pre-loaded with the vehicle's DBC file, either from OEM or CAN-D, allowing continuous decoding of the binary payloads, creating a data queue of multi-dimension time series signals, and tracking their changes in near real time.

3.1 CANShield Overview

As is shown in Fig. 2, CANShield contains three modules: *i) data preprocessing module* that creates multiple data views from signal-level CAN data, *ii) data*

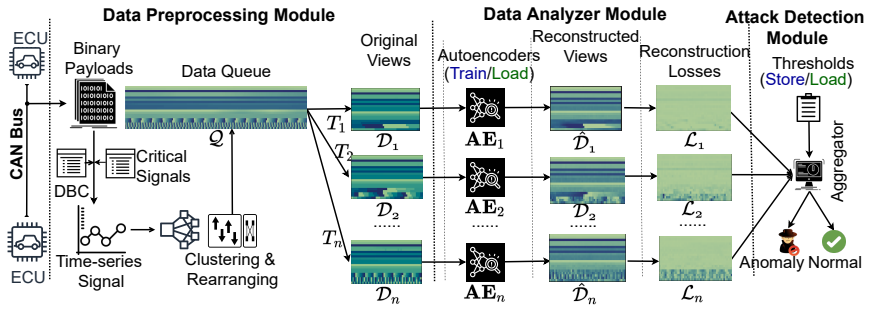


Fig. 2: CANShield workflow. The tasks with AEs and thresholds differ during the "training" and "deployment" phases.

analyzer module that employs multiple CNN-based AEs for generating anomaly scores for the data views, and *iii) attack detection module* that makes the final detection decision. CANShield has two phases of operation: *training* and *deployment*. Some of the modules play additional/slightly different roles during each of the two phases. During the training phase, the data analyzer module needs to train deep learning models. However, as the onboard devices are typically lightweight and not suitable for effective training of deep learning models, we consider two potential solutions for that. CANShield can have a secure connection to the cloud with model training capabilities or train the models on a local computer with CANShield running on that. Hence, during the training phase, the normal CAN traces are stored on the local memory first and then periodically sent to the cloud or local computer for model training. Once the model(s) are adequately trained, CANShield loads the trained model(s) into the onboard device and begins the deployment phase, which goes through the three modules in a feedforward fashion and output the detection result in near real time.

3.2 Attack Model

We assume that the intruder can access the CAN bus through an exposed interface, such as V2X, infotainment, ADAS systems, OBD-II port, etc. Moreover, we also assume that the attacker is capable of turning off any ECU [9] and/or injecting arbitrarily malicious messages. CANShield is designed to protect the vehicles from the different levels of attacks in a holistic manner. In particular, according to attacker's objective, the attacks typically fall into the following three categories:

- *Fabrication attacks*, wherein a compromised ECU injects malicious IDs and data to the CAN bus. However, all the legitimate ECUs are still active and also send their original data. This is the most prevalent and straightforward attack as the attacker does not need to hijack any ECU.
- *Suspension attacks*, wherein a legitimate ECU is turned off/incapacitated by the adversary. This attack is also called *suppress attack*, where the messages from the targeted ECU disappear for a while. To achieve this, the attacker can disconnect the ECU from the in-vehicle network to prevent it from communicating.

– *Masquerade attacks* are the most advanced, stealthiest, and destructive attacks. This is the combination of fabrication and suspension attack, where the attacker silences a legitimate ECU, spoofs it in the continuing operation while injecting malicious messages.

In evaluation, we will use a well-known CAN attack dataset and an emergent realistic CAN dataset covering specific forms of the above attacks to test the efficacy of CANShield.

3.3 Design Objectives

The design objectives of the CANShield are as follows:

– **Detecting Wide-range of Advanced Attacks.** The foremost objective of CANShield is to leverage established patterns and correlations of various ECU/signal states during normal driving and design a single IDS that can detect a variety of CAN message injection and manipulation attacks considered in the literature to date, particularly those advanced stealthy attacks that existing ID- or payload-based IDSs have shown ineffective in detecting.
– **Near real-time detection with near-zero false positives.** The IDS should respond to intrusions accurately with near-zero false-positive rate, and quickly, (at the same order of magnitude with the CAN message intervals) in order to help the vehicle avoid catastrophes.

4 CANShield Detailed Design

The CANShield IDS model relies on 2D CNN-based AE models to learn the spatio-temporal patterns of normal and attack-ridden CAN traces from the multi-dimensional time-series signals. Next we elaborate on CANShield's four constituting tasks in details.

4.1 Critical Signal Selection and Clustering

As modern vehicles have hundreds of ECUs, they contain a lot of CAN IDs and numerous associated signals. Securing all of them with IDS comes with great implementation and computation costs. On the other hand, securing only a handful of important signals from the critical sub-system of the vehicle, such as power train, engine, coolant system, etc., will reduce complexity and render feasible solutions for real-time detection. A practical challenge arises in designing an effective detection pipeline with a selected group of signals. Accordingly, we consider CANShield to keep tracks on only m pre-selected high priority signals. To find the shortlisted signals, we assume that the defender has the semantic knowledge of the critical signals. To make the detection more effective and robust CANShield adds additional signals based on the correlation coefficient, starting from the ones highly correlated with the critical signals. However, adding too many signals will lead to an expensive and ineffective system. Therefore, m is a design parameter and depends on the defender. For the rest of the paper, we will use the term "signals" to indicate only the pre-selected m signals.

The order of the signals in the created 2D input image could also impact the learning efficacy. Comparing to a random placement, placements that bring out

stronger spatial (correlations) patterns of the signals in the resulting image will enable more effective learning. To facilitate the learning of the inter-sensor correlations, CANShield calculates the Pearson correlation matrix of the time-series signal dataset. Interpreting the correlation coefficient as the distance between a pair of signals, CANShield utilizes hierarchical agglomerative clustering algorithm to find the clusters of highly correlated signals. The goal is to place the highly correlated signals together while building the 2D image so that learning the signal-to-signal correlation becomes effective for the small filters of the convolutional layers. Notably, the two tasks, signal selection and correlation-based clustering, are done only once during the initialization of the training process (i.e., off-line with recorded data) and are not parts of the deployment pipeline.

4.2 Data Prepossessing

The data preprocessing module prepares formatted 2D inputs to the AEs of the data analyzer module. It contains the following two steps.

Creating and Maintaining Data Queue First of all, the data preprocessing module continuously records the CAN traces and decodes the binary payloads containing the selected m signals. Then a first-in-first-out data queue $\mathcal{Q}$ is created with the historical time-series signal data for the last q time steps, where q is large enough for $\mathcal{Q}$ to encompass the temporal pattern of different signals.

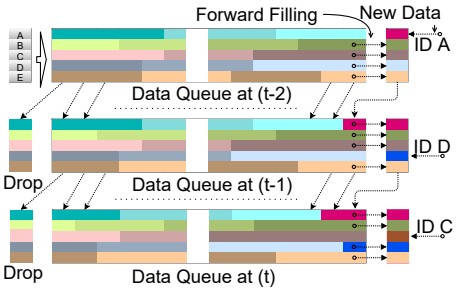


Fig. 3: Data queue generation.

Thus, every new CAN message is a new entry in $\mathcal{Q}$, where the signal values only associated with that incoming CAN ID are updated. For the rest signals, we adopt a forward filling technique, whereas, at every time step, the missing/unreported signals are copied from the previous time step. Fig. 3 shows that messages with ID A, D, and C are reported at time step $(t-2)$, $(t-1)$, and t, respectively. For the visualization, we have transposed the original image, where the signals associated with each CAN ID are presented as a single row, and the columns indicate the time steps. The color changes indicate the updates in the signal values associated with the CAN IDs. Thus, when a new message comes with a specific CAN ID, a new entry is added to the queue, dropping the last entry. Thus, as time passes, the sensor data for the latest q time steps are always stored in the data queue.

Creating Multiple Views To identify abnormality on different signals with different temporal trends, the data analyzer module trains and deploy the AE networks on different views of the data queue $\mathcal{Q}$. As different CAN IDs have different reporting periods, only the first w ($<< q$) time steps (columns) of $\mathcal{Q}$ may not be enough to represent the recognizable temporal trend for all the signals, especially for the ones with long reporting cycles. On the other hand, considering a high value for w ($\approx q$) makes the input image too large. As a

result, the AE models become more complex. This challenge boils down to *how to effectively learn the temporal patterns of all the signals, especially of the ones with long reporting periods, while still using a small time window during image generation.* We achieve these two conflicting goals by creating different views of $\mathcal{Q}$ with different sampling periods (seeing more with a less complex models). Thus, we select the first w columns from $\mathcal{Q}$ at every T_1, T_2, ..., T_n time steps, respectively. Here, $T_1, T_2, ..., T_n$ are the sampling periods to create the views $\mathcal{D}_1$, $\mathcal{D}_2$, ..., $\mathcal{D}_n$, respectively of the same $\mathcal{Q}$. With loss of generality, here we assume $T_1 < T_2 < ... < T_n$. Fig. 4 illustrates the sampling process on $\mathcal{Q}$ at time step t.

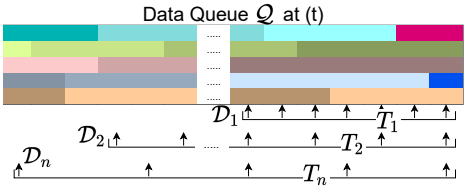


Fig. 4: Different views of data queue $\mathcal{Q}$.

Therefore, $\mathcal{D}_1$ has a more detailed view but contains a very limited historical trend, capturing short-term or fast-changing patterns. On the other hand, $\mathcal{D}_n$ has the most of the temporal trend, capturing long-term or slow-changing patterns, but with the lowest details. The multi-view design has benefit in both model accuracy and scalability. Each of these views has different primary targeted signals but collectively they cover temporal trends of variable lengths. This allows more effective and accurate detection of different advanced masquerade attacks, regardless attacking message frequency and duration. Despite having different sampling periods, the number of samples within each data view remains the same (w). As there are total m signals for the IDS, each data view will have a dimension of $m \times w$. Thus similar AE models can be used to train each type of data view.

4.3 Data Analyzing

The data analyzer module utilizes multiple AE models: $\{\mathbf{AE}_i\}_{i\in[n]}$ (where $[n] \stackrel{\text{def}}{=} \{1, 2, ..., n\}$). Each of the models is associated with each of the views of $\mathcal{Q}$ and thus learns different (and complementary) perspectives of $\mathcal{Q}$. We build the AE networks using CNN due to the observation that each view is a two-dimensional data item, and CNN is widely proven to efficiently work on 2D data with minimum complexity. The motivation of using AE is that, as there are neither explicitly defined states of the vehicle, nor any analytical model for that, we use a data-driven approach to find the states out of a small window of the historical signal data. Thus, the data in an AE's middle (bottleneck) layer represents the vehicle's state in a lower dimension. In contrast, the decoder part tries to predict the vehicle's historical signal data by looking at the state's information. If the vehicle is running in a normal state, as mostly seen in the training data, the decoder should predict accurately. Otherwise, an abnormal state will lead to an erroneous prediction, and thus, to a high reconstruction loss.

Moreover, as our considered model learns the relationship among all the signals, especially the nearby highly correlated ones, if at least one signal deviates from the regular pattern, CANShield will recognize it from the reconstruction loss. As is shown in Fig. 2, during the training phase, each $\mathbf{AE}_x$ takes a data

view $\mathcal{D}_x \in \mathbb{R}^{m \times w}$ as an input image and learns to reconstruct almost the same $\hat{\mathcal{D}}_x \in \mathbb{R}^{m \times w}$ image, $\forall x \in [n]$. Once the training is done, the deployment phase is initiated, and the trained models are loaded in CANShield. At the end of the training phase and during the deployment phase, the AEs are tested on the corresponding data stream and try to reconstruct the same image. For AE $\mathbf{AE}_x$, the difference between the original image and the reconstructed image is the reconstruction loss $\mathcal{L}_x \in \mathbb{R}^{m \times w}$. Each row contains the corresponding signal's reconstruction losses and columns for the time steps.

4.4 Thresholds Selection and Attack Detection

In this part, we discuss how to interpret a 2D reconstruction loss $\mathcal{L}_x$ into an anomaly score P_x (i.e., attack probability) for every data view $\mathcal{D}_x$ and use the results for attack detection. For a normal computer vision problem, the common practice would be evaluating the mean absolute value of the reconstruction loss matrix $\mathcal{L}$ as the anomaly score P: $P \leftarrow \frac{1}{mw} \sum_{i=1}^{m} \sum_{j=1}^{w} ||\mathcal{L}_{i,j}||$.

Table 1: Thresholds Selection and Attack Detection for AEs in CANShield.

Reconstruction loss $\mathcal{L} \in \mathbb{R}^{m \times w}$, system hyperparameters p, q, r $\mathcal{B} \leftarrow 0^{m \times w}$, $\mathcal{V}, \mathcal{S} \leftarrow 0^{m}$, Anomaly score P_x, result *attack*,

`/* Step 1 (assign` P_x `for every` $\mathbf{AE_x}$`) */`

$$\forall i \in [m]: \; R_i^{Loss} \leftarrow p^{th} \; \% \; \forall_{\in training} \; \mathcal{L}_{i,j} \quad (1)$$

$$\forall i \in [m], \forall j \in [w]: \; \mathcal{B}_{i,j} \leftarrow 1 \;\; \textbf{if} \; \mathcal{L}_{i,j} > R_i^{Loss} \quad (2)$$

$$\forall i \in [m]: \quad \mathcal{V}_i \leftarrow \sum_{j=1}^{w} \mathcal{B}_{i,j} \quad (3)$$

$$\forall i \in [m]: \quad R_i^{Time} \leftarrow q^{th} \; \% \; \forall_{\in training} \quad \mathcal{V}_i \quad (4)$$

$$\forall i \in [m]: \quad \mathcal{S}_i \leftarrow 1 \;\; \textbf{if} \;\; \mathcal{V}_i > R_i^{Time} \quad (5)$$

$$P_x \leftarrow \frac{1}{m} \sum_{i=1}^{m} \mathcal{S}_i \quad (6)$$

`/* Step 2 */`

$$P_{ens} = (P_1 + P_2 + ... + P_n)/n \quad (7)$$

$$R_{ens}^{Signal} \leftarrow r^{th} \; \% \; \forall_{\in training} \quad P_{ens} \quad (8)$$

`/* Step 3 (deployment phase only) */`

$$attack \leftarrow 1 \;\; \textbf{if} \;\; P_{ens} > R_{ens}^{Signal} \quad (9)$$

Compared to a normal computer vision problem, our input image (i.e., a data view $\mathcal{D}_x$) has a concrete structure, which gives space for systemic analysis of the detection thresholds for better accuracy. Thus, instead of taking the average value, we exploit the structural knowledge of $\mathcal{D}_x$ to interpret the P_x from $\mathcal{L}_x$. We define three types of thresholds for attack detection at each AE:

- Signal-wise reconstruction loss thresholds $R^{Loss} \in \mathbb{R}^m$
- Signal-wise time step violation thresholds $R^{Time} \in \mathbb{R}^m$
- A number of total compromised signals threshold $R^{Signal} \in \mathbb{R}$

We summarize a *three-step analysis* on $\mathcal{L}_x$ to facilitate selection of these thresholds and attack detection, as is shown in Table 1 and skip the detailed explanation for brevity. For convenience, in step 1, we have obviated the AE index x for $\mathcal{L}$, intermediate variables, and thresholds as this approach will be applied independently to each AE.

Step 1 repeats for every AEs and assigns anomaly score on each of the reconstruction losses on the data views, i.e., P_1, P_2,...., P_n and step 2 ensembles the scores in a single score P_{ens}. We configure the training phase of CANShield to run the steps 1 & 2 from Eq. (1)–(8) and stores R^{Loss} and R^{Time} for each of the AEs, and R_{ens}^{Signal} for the ensemble model, optimally tuning three system hyper-parameters p, q, r as confidence percentiles for these thresholds. During

Table 2: Description of attacks in SynCAN dataset.

Attack Name	Attack Type	Description
Flooding	Fabrication	Frequently injects high-priority messages.
Suppress	Suspension	Prevent an ECU from transmission.
Plaeau	Masquerade	Broadcasts a constant value.
Continuous		Broadcasts continuously changing values.
Playback		Broadcasts a series of recorded values.

the deployment phase, these thresholds are pre-loaded from the memory, thus, skipping (1), (4), and (8). While steps 1 & 2 are common in both training and deployment, CANShield runs one additional task (step 3) in the deployment phase to check for potential threats (Eq. 9)) and raises the alarm in the system.

5 Implementation

5.1 Datasets and Attacks

We implement CANShield on both SynCAN dataset and ROAD dataset. SynCAN dataset [7] (Synthetic CAN Bus Data) is a widely used CAN attack dataset released by ETAS (a subsidiary of Robert Bosch Gmbh) covering stealthy signal-level CAN attacks. ROAD dataset [6] was released by Oak Ridge National Laboratory and is the most realistic CAN attack dataset to date[3] Next we introduce the details of each dataset and the attacks covered.

SynCAN SynCAN dataset is built on actual CAN traces, emulating the characteristics of the real CAN traffic, with hundreds of advanced attack scenarios. It contains a total of 20 signals. There are 24 hours of logged data, of which 16.5 hours are for training and 7.5 hours are for testing with five types of advanced attacks, which resemble the three stealthy forms of attack models mentioned in §3.2. The attacks in SynCAN datasets are summarized in Table 2. A *flooding attack* creates delays the legitimate ECUs' transmission (similar as DoS attack) and a *suppress attack* turns off the corresponding ECU of the targeted signal(s). Based on time-series nature of the injected data there are three types of masquerade attacks. Whereas a *plateau attack* broadcasts the same constant value of any signal over a long period of time, the *continuous attack* and *playback attack* overwrites the signals with continuously changing values and previously recorded data, respectively, that shift naturally from the actual ones.

ROAD Dataset ROAD dataset provides the highest-fidelity CAN traces with physically verified most realistic CAN attacks. It contains a significant amount of training data covering different context of driving. We obtained raw ROAD dataset and extracted signals from the CAN messages using CAN-D. There are 3.5 hours of logged data, of which 3 hours are for training and 30 minutes are for testing with five types of advanced masquerade attacks targeting the

[3] To the best of our knowledge, SynCAN dataset is the only publicly available signal-level CAN dataset at the time of writing this paper. ROAD dataset was obfuscated and did not have signal-level interpretation in its initial release in early 2021. We obtained the raw ROAD dataset through directly contacting ORNL. Partially motivated by our work, ORNL has a plan to release signal-level ROAD dataset soon.

Table 3: Description of masquerade attacks in ROAD dataset.

Attack Name	**Description and impacts**
Correlated signals	Inject four different values for four wheel speeds that kills the car.
Max speedometer	Inject maximum (0xFF) value to display a maximum value in speedometer.
Max coolant temp	Inject maximum (0xFF) value that turns on the coolant warning light.
Reverse light on/off	Toggle the reverse light bit so that reverse lights do not reflect the gear.

engine coolant temperature, engine RPM, brake light, and wheel speeds sensors. The injected message manipulates only the specific portion of the data fields containing the targeted signals.

Whereas the attacks in the SynCAN dataset are created by post-processing (replacing original ones) on the normal driving data, the attack traces in the ROAD traces were collected from a real car under the real injection attacks. Such attack traces provide not only the injected messages but also the response from the vehicle under such attacks, which makes the ROAD dataset the most realistic one. The attacks in ROAD dataset are summarized in Table 3. In light of the model's complexity, one single IDS is not a feasible option to track all the hundreds of decoded signals within the ROAD dataset. Thus, considering individual IDS on a different critical subsystem of the vehicle is be a viable solution. In the implementation of CANShield on ROAD dataset, we consider the attacked signals in Table 3 to be of primary importance and add two highly correlated signals for each to make the IDS more robust, as detailed in §4.1.

6 Evaluation Results and Discussion

6.1 Evaluation Settings

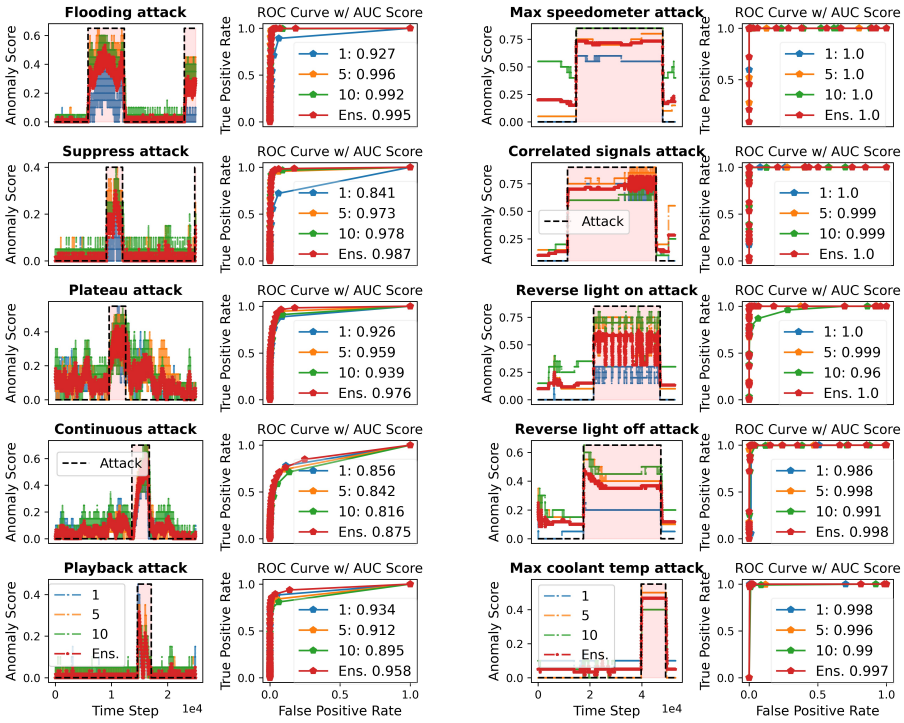
We use Python 3.7.3 with Keras 2.2.4 for training and evaluation of CANShield. We used a five-layer network, and the numbers of filters in each layer are 32, 16, 16, 32, and 1. The following section explains the impact of different parameters in attack detection and illustrates the effectiveness of CANShield. We evaluate CANShield's performance on two aspects:

Attack detection. If any view of the data queue contains one or more malicious injections, we consider the data queue as malicious. We use ROC curve and AUC score to evaluate CANShield's performance in different settings. A ROC curve plots true positive rates and false-positive rates for different thresholds of the final intrusion scores. The area under the ROC curve (AUC) indicates the robustness of the detectors. An ideal detector has an AUC score of 1.00.

Event detection latency. Depending on the type of attack, there could be a delay between the first injected message and the first correct detection in any attack event. Such delay is defined as the event detection latency.

6.2 Attack Detection

Our comprehensive evaluation shows CANShield performs the best when w is 50, there are three AEs (with sampling periods 1, 5, and 10), and R_{Loss} and R_{Time} are considered as 95-percentile and 99-percentile of the normal data. We consider these settings for the following evaluations on both the SynCAN and ROAD datasets.

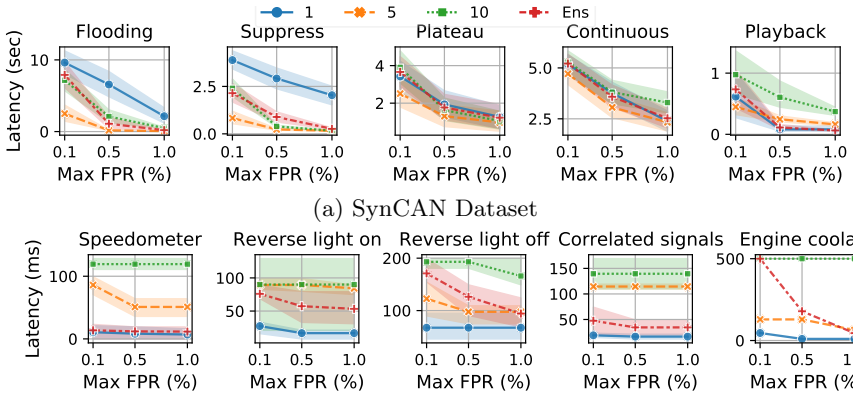


(a) Performance on SynCAN dataset. (b) Performance on ROAD dataset.

Fig. 5: Attack visualizations and ROC curves for different models and attacks.

Attack visualization and ROC Curve In this part, we visualize the anomaly scores for all the individual and ensemble detectors along with the ROC curves.

SynCAN Dataset. Fig. 5a shows the anomaly scores and the ROC curves for five different attacks on the SynCAN dataset. Different AEs show different performances on each of the attacks. However, the ensemble model yields more stable and consistent performance, leading to higher AUC scores in all the attacks than the individual AEs. For instance, higher sampling periods (i.e., 10) perform better in detecting the flooding (*fabrication*) and suppress (*suspension*) attacks, as they are more detectable looking at the long-term sequential pattern. However, the lower sampling periods (i.e., 1) offer better performance in detecting the *masquerade* attacks, where short-term views of the data queue provide a detailed look at the time-series violations. Moreover, in the case of continuous and playback attacks, the signals start to deviate gradually, which takes some time to create the recognizable deviation for the IDS. Hence, a lower AUC score is not unexpected, especially against continuous attacks. However, CANShield can detect the violations instantly for the rest of the attacks (AUC scores of $0.95 \sim 1.00$). Whereas the individual AEs are attack-specific, the ensemble model takes the best out of every model, generalizes the process, and detects most attacks with the highest AUC scores.



(a) SynCAN Dataset

(b) ROAD Dataset

Fig. 6: Event detection latency for different FPR thresholds both on dataset.

ROAD Dataset Fig. 5b shows the anomaly scores and the ROC curves for the attacks on ROAD dataset. Same as the SynCAN, the ensemble model also shows stable performance in the anomaly score. As all the attacks in the ROAD dataset are closely aligned with the plateau attack in the SynCAN dataset, both the individual and ensemble models show high performance in detecting the attacks. There are a few cases where the performance degrades a little; for example, in the reverse light on attacks, AE with a sampling period of 10 gives an AUC score of 0.960. However, the ensemble model mitigates such issues and detects all the attacks from the ROAD dataset with AUC scores of ~ 1.00.

6.3 Event Detection Latency

Fig. 6 illustrates the attack-wise event detection latency for three cases of maximally allowed FPR for different datasets. As each attack manipulates the signal at different paces, the time to observe a potential deviation varies. Hence, similar to the previous discussion, certain AEs are more responsive against certain types of attacks. For example, Fig. 6a shows all the models have higher latency in detecting *continuous* attack events as they deviate the signals gradually. However, ensembling the individual models reduces the latency in the average scenario.

Furthermore, the figures also illustrate the impact of maximum FPR on the event detection latency. Although some individual model suffers from high latency with low FPR (i.e., 0.1%), CANShield's ensemble model provides a lower event detection latency. However, allowing more false positives (max FPR of $0.5\% - 1\%$) into the system further reduces latency. Whereas in case some advanced SynCAN attacks CANShield takes up to a couple of seconds to detect, all the attack events in ROAD dataset are detected in milliseconds. Therefore, our evaluation shows that the CANShield improves detection performance, reduces overall detection latency, and makes the system more robust.

Comparison with Baseline Model. We further compare the AUC scores of the ensemble model with the baseline detector. Table 4 illustrates such comparison, which indicates drastic improvements in the detection of *flooding*, *suppress* attacks compared to the baseline model, CANet. Although CANet per-

forms slightly better on the masquerade attacks, CANShield shows decent performance as well.

Table 4: Comparison with baseline.

	Area Under the Curve (AUC)				
	Flooding	Suppress	Plateau	Continous	Playback
Ours	0.997	0.985	0.960	0.870	0.948
CANet	0.979	0.882	0.983	0.936	0.974

7 Related Work

There have been a good amount of works on CAN IDS, which can be divided into a few different categories in general. Based on the collected data during the regular operation of the vehicle, rule-based IDS creates security rules for CAN communication and ECU behaviors, e.g., message frequency, no overlapping with the original flow, no disruption of the regular sequence of CAN IDs, etc. Such properties are considered as the baseline in developing CAN IDS [5, 10, 11]. A few works utilized the physical layer attributes of the ECUs, such as clock skews [12], voltage profile [13], electrical CAN signal characteristics [14], etc. to fingerprint the transmitter ECUs.

There are a few machine learning-based IDS working on binary payloads of CAN messages. Readers are referred to [5] as a good survey for such IDSs. The payload-based models are trained on obfuscated binary data; thus, they work as a black-box and lacks explainability [6]. Most of them only look at the sequence of IDs, which will not suffice to detect advanced attacks. Additionally, these types of IDSs cannot detect attacks from an intelligent adversary who has control over the message generation in CAN bus and can launch stealthy signal-level attacks.

There are only a limited amount of works on the signal-level IDS for CAN bus [15–17]. CANet [7] is the first IDS working on such a high-dimensional structure. Indra [15] and LATTE [16] are few other attempts in the same direction. However, all of them utilizes LSTM-based networks, which are very costly to train and one LSTM for each IDS will make it impractical on the actual car with a high number of CAN IDs. Moreover, due to their architectural limitations, the existing IDSs show low detection performance on different advanced attacks and lack scalability.

8 Conclusion

As modern vehicles become more connected to external networks, we propose a CAN bus intrusion detection framework, CANShield, working at the signal level to secure the bus from the advanced attacks. Along with the capability of handling high-dimensional CAN data stream, CANShield trains multiple CNN-based autoencoder models to work on different views of the data stream across different temporal scales. With the aid of the individual models, an ensembled model is used to detect a wide range of attacks and events with very low latency and high accuracy. Evaluation on both the SynCAN and ROAD datasets shows CANShield's robustness and responsiveness against different advanced attacks.

Acknowledgements This work was supported in part by the US National Science Foundation (NSF) and the Department of Homeland Security (DHS) under NSF grant CNS-1837519, the Virginia Commonwealth Cyber Initiative (CCI), and the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC, for the U.S. Department of Energy. We are also thankful to Robert A. Bridges from ORNL for his insightful comments on the manuscript.

References

1. Andy Greenberg. Jeep hacks. `https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway`. Accessed: 2021-06-21.
2. Charlie Miller. Lessons learned from hacking a car. *IEEE Design & Test*, 2019.
3. Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, 2019.
4. Umar Khalid, Ashkan Esmaeili, Nazmul Karim, and Nazanin Rahnavard. Rodd: A self-supervised approach for robust out-of-distribution detection. *arXiv preprint arXiv:2204.02553*, 2022.
5. Siti-Farhana Lokman, Abu Othman, and Muhammad-Husaini Abu-Bakar. Intrusion detection system for automotive controller area network (can) bus system: a review. *EURASIP Journal on Wireless Communications and Networking*, 2019.
6. Miki E. Verma, Michael D. Iannacone, Robert A. Bridges, Samuel C. Hollifield, Pablo Moriano, Bill Kay, and Frank L. Combs. Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to can ids data & introduction of the road dataset, 2022.
7. Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *IEEE Access*, 8:58194–58205, 2020.
8. Miki E. Verma, Robert A. Bridges, Jordan J. Sosnowski, Samuel C. Hollifield, and Michael D. Iannacone. Can-d: A modular four-step pipeline for comprehensively decoding controller area network data. *IEEE Transactions on Vehicular Technology*, 70(10):9685–9700, 2021.
9. Kyong-Tak Cho and Kang G Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055, 2016.
10. Ulf E Larson, Dennis K Nilsson, and Erland Jonsson. An approach to specification-based attack detection for in-vehicle networks. In *2008 IEEE Intelligent Vehicles Symposium*, pages 220–225. IEEE, 2008.
11. Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709. IEEE, 2017.
12. Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th {USENIX} Security Symposium Security 16)*.
13. Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 2018.
14. Marcel Kneib and Christopher Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.
15. Vipin Kukkala, Sooryaa Vignesh Thiruloga, and Sudeep Pasricha. Indra: Intrusion detection using recurrent autoencoders in automotive embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.
16. Vipin Kumar Kukkala, Sooryaa Vignesh Thiruloga, and Sudeep Pasricha. Latte: L stm self-att ention based anomaly detection in e mbedded automotive platforms. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(5s):1–23, 2021.
17. Pablo Moriano, Robert A Bridges, and Michael D Iannacone. Detecting can masquerade attacks with signal clustering similarity. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.