CANnolo: An Anomaly Detection System based on LSTM Autoencoders for Controller Area Network

Stefano Longari*, Daniel Humberto Nova Valcarcel*, Mattia Zago†, Michele Carminati*, Stefano Zanero*
*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan (Italy)
{stefano.longari, michele.carminati, stefano.zanero}@polimi.it
danielhumberto.nova@mail.polimi.it
†Department of Information Engineering and Communications, University of Murcia, Murcia (Spain)
mattia.zago@um.es

***Abstract*—Automotive security has gained significant traction in the last decade thanks to the development of new connectivity features that have brought the vehicle from an isolated environment to an externally facing domain. Researchers have shown that modern vehicles are vulnerable to multiple types of attacks leveraging remote, direct and indirect physical access, which allow attackers to gain control and affect safety-critical systems. Conversely, Intrusion Detection Systems (IDSs) have been proposed by both industry and academia to identify attacks and anomalous behaviours.**

In this paper, we propose CANnolo, an IDS based on Long Short-Term Memory (LSTM)-autoencoders to identify anomalies in Controller Area Networks (CANs). During a training phase, CANnolo automatically analyzes the CAN streams and builds a model of the legitimate data sequences. Then, it detects anomalies by computing the difference between the reconstructed and the respective real sequences. We experimentally evaluated CANnolo on a set of simulated attacks applied over a real-world dataset. We show that our approach outperforms the state-of-the-art model by improving the detection rate and precision.

***Index Terms*—Network Security, Intrusion Detection System, Controller Area Network, Deep Learning, Unsupervised Learning**

I. Introduction

The automotive field has witnessed, over the past 30 years, rapid adoption of electronics throughout all vehicle systems that control most of the vehicle's functionalities: from automatic transmission to (adaptive) cruise control, up to all of the types of assisted and autonomous driving technologies.

The majority of these systems consist of Electronic Control Units (ECUs) interconnected to onboard vehicular networks, mostly based on the CAN bus [1]–[3]. CAN is considered as the de-facto standard in automotive onboard networks [4] and is also commonly found in other industrial applications. Also, vehicles are now extensively connected to the outside world, through local (such as Bluetooth or USB ports) and remote connection (mostly through data-enabled mobile networks). The development and growth in the relevance of onboard networked systems, and their interconnection with the external world, has created a large attack surface (in a pattern that has been witnessed already in other types of cyber-physical systems, such as industrial control systems and IoT devices [5]).

In the last decade, researchers have shown that it is possible to gain control of vehicles from remote and be able to affect the safety of people inside and around the vehicle consistently [6]–[8]. Koscher et al. and Checkoway et al. published the two first analyses of vulnerabilities in vehicles [9], [10]. Their works prove both the inherent security challenges of the CAN protocol both the feasibility of accessing the vehicle onboard networks. At the same time, they made it clear that some attacks on vehicles can be disruptive and dangerous for the safety of people on and around the vehicle. Since Miller and Valasek's demonstration on a Jeep Cherokee in 2014 [11], both industry – automotive manufacturers (denoted as OEMs in the automotive world) and ECU vendors – and academia have started to express concerns regarding vehicle security [7], [8], [12].

Many countermeasures and Intrusion Detection Systems (IDSs) have been proposed in the automotive domain based on the consolidated knowledge acquired in the cybersecurity field. Intrusion Detection Systems (IDSs) monitor the events in a computer system or network for signs of intrusions. Generally speaking in the automotive context countermeasures and IDSs can be applied onboard of the vehicle or off-board to the whole VANET or fleet (e.g., [13]–[16]). In the paper at hand, we define an intrusion as a malicious agent that has already gained access to the CAN bus. Therefore our context is that of onboard detection, intending to prevent the attacker from implementing attacks abusing of their CAN bus access.

In this paper, we propose CANnolo, a reconstruction-based unsupervised IDS that exploits the power of LSTM autoencoders for detecting anomalies in CANs. During a training phase, CANnolo automatically analyzes the streams of data and builds a latent representation of CAN traffic data sequences using only legitimate data. Different from existing approaches, CANnolo does not require knowledge about data semantics. At runtime, CANnolo exploits the trained autoencoder to reconstruct CAN traffic and to predict the subsequent sequences. Then, it detects anomalies in terms of the reconstruction error, which measures the difference between the forecasted and the real sequences.

We experimentally evaluated CANnolo on a publicly available real-world dataset. We show that our approach improves the state-of-the-art solution by outperforming it in detecting the categories of attacks for which these systems are designed.

In summary, our contributions are the following:

1) An unsupervised Intrusion Detection System (IDS) based

on LSTM autoencoders for CAN traffic that automatically analyze streams of data – without prior knowledge of its semantics – and detect anomalies in terms of the reconstruction error between the real and the forecasted data.

2) An extensive evaluation on a real-world dataset that corroborate the effectiveness of our Anomaly Detection System (ADS).

The paper is structured as follows:

i Section II reports the necessary CAN properties and the threat model required to comprehend the reasoning behind CANnolo;

ii Section III presents the related works;

iii in Section IV the CANnolo approach is presented, including a formal definition of the methodology regarding the LSTM-base autoencoder and the anomaly detector;

iv Section V presents the results, alongside those of the state of the art model used to evaluate CANnolo's performances;

v Section VI discusses the conclusions.

II. Background and Motivation

Since CANnolo's approach directly relies on LSTM and Autoencoders, a brief primer on the topics is necessary, including a short excursus on attacks and security measures. Moreover, this section will introduce some essential aspects of CAN, precisely how data frames are structured, transmitted, and how the arbitration mechanism works. For additional details, we refer to the official CAN specification [3].

A. Primer on LSTM and Autoencoders

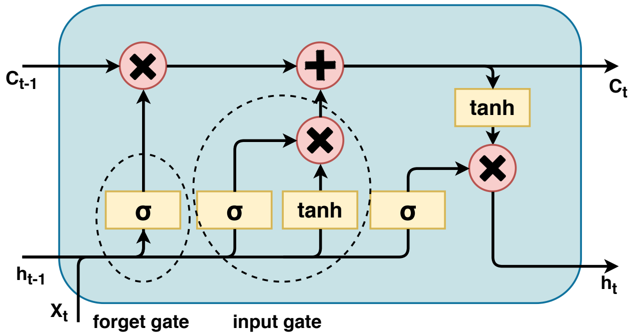


Fig. 1: LSTM module layout. Rectangles represent layers, circles represent pointwise operations. σ and tanh symbols represent respectively sigmoid and activation functions. X_i, C_i, and h_i are respectively ith input, cell-state, and output.

In the paper at hand, CANnolo makes use of LSTM autoencoders, namely an implementation of autoencoders that uses LSTM as learning layers.

On the subject of temporal analysis, Recurrent Neural Networks (RNNs) are a category of neural networks that features loops to feed process information to the network itself in such a way that enables data persistence. Nevertheless, this compelling point is also its major drawback, i.e., a limited amount of time steps that can be stored. As pointed out by Hochreiter [17], classical RNNs' back-propagation algorithm do suffer from the vanishing gradient problem, that is the progressive elimination of weights that are multiplied by small numbers at each interaction.

One of the solutions to this issue is to deploy LSTM units. As Figure 1 presents, a LSTM unit is a structure that permits to memorize the cell state for further usages. To be precise, the forget gate is used to decide how much of the previous cell state should be retained or discarded, while the input gate is used to decide how much of the new information should be stored in the cell state. The LSTM structure, reported in Figure 1, enables the update of the cell states with new and relevant information. Finally, the last section of the whole module filters the output. We refer the reader to Hochreiter et al's work for further details [18].

With regards to the autoencoders, they are unsupervised neural networks that are capable of learning how to encode (usually compress) and then decode data. The strength of autoencoders often stems from the compression; in other words, the compression itself removes part of the data that do not provide enough information. The training process forces the autoencoder to remove only the noise and not relevant information.

B. Primer on CAN

CAN is a bus-based network designed in the '80 by Bosch to supply to the lack of communication between on-board vehicle systems. Its main strengths and benefits are mainly cheapness, real-time capabilities (up to a certain degree), and resistance to electromagnetic noise. However, it lacks security and privacy features.

Data frames carry the data payloads, while remote frames are rarer and used to request information on demand. Error frames are specific small sequences of bits that overwrite the data frame they are referring to, thus notifying all nodes of the error. Finally, overload frames are kept for backward compatibility; their uncommon usage is that of forcing a delay in the transmission of data in the case a node is not able to process the received information fast enough.

The paper at hand focuses on data frames as they are the primary carriers of attacks. Although several other attacks have been documented (such as error frames alongside the arbitration protocol), they are not the focus of this work due to the intrinsic differences in implementation and detection of such attacks.

Data frame structure. Data frames are composed of several fields, the majority of which are out of the scope of this paper. We will focus on the ID and data fields. The ID field, which is 11 or 29 bits long, distinguishes data packets, and it is positioned at the beginning of the packet. The ID does not represent the sender node but the contents of the packet itself. Each node can send messages with multiple IDs, but each ID can be sent only by one node, which is an essential requirement due to the arbitration mechanism of CAN, explained below. Nodes can read any packet from the bus since packets are sent in broadcast, and the protocol does not provide any form of encryption. The data field, which can be up to 64 bits long, contains the data transmitted. CAN packets are mostly used to send statuses, sensor values, and commands.

Therefore, the CAN data field is structured such that specific ranges of bits are assigned to a specific value, allowing the receiving node to analyze them quickly. For example, ID 0x555 may be set up so that bits from 0 to 15 report the speed of the vehicle, bits 16 to 19 are always zeroes, bits 20 to 23 describe the current gear, and bits 24 to 39 report the current steering wheel angle. Every time ID 0x555 is transmitted, the node that requires the information contained in the packet can retrieve it and directly search for those bits.

C. Threat Model

Different attacks and threat models have been proven feasible on CAN by many different researchers [7], [19]. We present them according to the attacker's goal.

- **Sniffing** CAN is a broadcast protocol; any ECU connected to the network can read all traffic passing through the bus. Sniffing can be used to either forward data to a malicious agent, learn the behavior of a target ECU, or for other attacks. Generally speaking, this is an attack not considered in defense mechanisms for CAN due to the costs and often even feasibility of countermeasures.
- **Forging** Although not meant to do so, each ECU can potentially write on the bus using any ID and, hence, it can impersonate any other ECU. Forging can be further divided into Masquerade, Replay, and Fuzzing.

  Masquerade. Means that a compromised component with access to the network can inject data in the bus impersonating a target ECU. It is the more general of the three categories, where we can assume the attacker is creating packets with a crafted data payload. The strength of this category is that of giving the attacker the possibility of sending any command.

  Replay. Sniff previously transmitted messages to collect data and retransmit it on the bus later. The purpose of a replay attack is to reenact a previously seen state by using the sniffed data. The strength of these attacks is that they are harder to detect since they are intrinsically compliant with the accepted messages.

  Fuzzing. Consists of injecting random or partially random messages. Although it is often not as efficient as other techniques, it can be used when reverse-engineering is not possible. Fuzzing has proven to be a useful tool to explore unexpected protocol behavior and can cause significant damage in the vehicle [9].
- **Denial of Service** The network as a whole, or its components can be targeted to compromise their availability. A Denial of Service (DoS) attack can be implemented either on a network scale through flooding the bus with high priority messages, hence always winning arbitration, or on ECU scale, as demonstrated by Palanca et al. [20], shutting an ECU off the network. Network-level DoS is trivial to detect, while targeted DoS is harder but feasible to detect, as proven in [21]. DoS by itself is also hardly a final goal for an attacker since vehicles are usually programmed to not carry safety risks even in case the bus goes offline. DoS is usually implemented as a first step to implement a forging attack.

D. Motivation

Machine Learning (ML) solutions, and precisely Deep Learning (DL) solutions have proven effective intrusion detection systems in the automotive field [22]–[24]. Autoencoders are an intuitive approach in scenarios where the model needs to learn to represent a time-series, in this specific case, the CAN traffic ones. Previous work in this direction [23], [25], [26] supports such intuition, and, as further explained in Section III, time-series-based intrusion detection in CAN has brought promising results through the use of LSTM. Furthermore, reconstruction based detection has been proved [23] potentially more efficient than prediction based detection when the underlying data is inherently unpredictable.

Hence, the driver of this research is to build a model capable of replicating sequences of non-malicious CAN data so that when given as input CAN traffic, failing in reconstructing it can indicate that an attack is taking place.

Autoencoders are, in theory, capable of learning what constitutes authentic traffic by removing noise during the encoding process, without the need for attack examples. Hence, once it learns the non-malicious patterns of the data, anything that would differentiate from it would be flagged as an anomaly. Ideally, as long as the input is non-malicious, the ideal model would always output the same sequence that was used as input.

In other words, the reconstructed traffic originated from malicious sequences would present a high-degree of reconstruction error, thus indicating an anomaly in the source sequence. A post-processing module is thus required to process this information and to generate a configurable anomaly score that might indicate an attack.

III. Related Work

Generally speaking, Intrusion Detection Systems (IDSs) can be divided into two main categories [27], [28]: signature-based and anomaly-based detection. Signature-based detection uses supervised learning and assumes previous knowledge of the attacks (i.e., a database of known attack signatures). It examines ongoing traffic looking for messages that match these signatures. However, in the automotive context, this information is rarely available [29]. The database would be manufacturer-dependent, and additionally, real attack examples on CAN are not publicly available. Therefore, signature-based detection is often a non-viable solution. Anomaly-based detection, instead, uses an unsupervised learning approach, and it does not require attack signatures examples to work. It creates a baseline using exclusively legitimate traffic and evaluates anomalies in terms of deviations from this norm.

Regarding CAN intrusion detection, multiple different methods have been proposed over the years. The most common one are time/frequency based detection [30]–[34], which focuses on incongruences in the periodicity of messages and is usually efficient in detecting attackers trying to overwrite periodic messages. Frequency based detection is extremely effective against attacks that increase the number of messages on the bus, reaching detection rates close to 100%, but has a number of downsides: Some CAN messages are not periodic, making it intrinsically harder for these systems to detect attacks. Besides, assuming that the attacker has the capability of silencing the victim, which is hard but not impossible [20], [35], detecting the attack is harder since the attacker becomes the only one writing on the bus. Finally, Sagong et al. [36] prove that it is also possible for attackers to simulate the exact arrival time of the silenced victim, making it close to impossible to detect it. Specification based detection [21], [37], [38], which analyze and ensure that some specific rules are not being violated, and usually does not defend from the majority of attacks but from a specific kind. Evaluating the detection rate of these IDSs is harder, since any attack aside the specific one that they defend from is usually undetectable. Nonetheless they usually reach extremely high detection rates on the specific attacks they defend from. Signal based detection [39]–[41] exploits the signal characteristics of CAN transceivers, in terms of fingerprints, to identify the valid sender (ECU) of a packet. Although in some cases the detection rate of these IDSs is high, they are extremely dependent on the specific implementation of the network they're being used in, making it hard to use them in a broad scale. Finally payload based detection [24], [42]–[44] works by detecting anomalies in the payload of the packet, and is usually effective against masquerade attacks, where the attacker tries to change the payload with one that does not respect some fixed or dynamic rules. The IDS usually recognizes patterns inside the packet that, if not respected, trigger the detection of the attack. For this reason these IDSs are not particularly effective against replay attacks, which intrinsically respect said patterns, or against well forged attacks that remains inside the boundaries of what is considered normal by the IDS.

One of the newest approaches that have been implemented to try to overcome the issues of the previously presented IDSs is temporal analysis. In fact, CAN traffic payload can be seen as a time-series, with its own natural temporal ordering, meaning that analyzing anomalies in CAN traffic is a time-series anomaly detection task. One of the main advantages of this approach is that it aims to detect abnormalities in messages that can be considered normal as individual observations, but that are actually anomalous in the *context* of surrounding messages, overcoming previous limitations such as the low results of payload based IDSs in detecting replay attacks. In fact, it can detect values that are outside a norm model (outliers) and also when they are out of context; as a simplified example sudden unrealistic changes in speed but otherwise normal absolute speed values, e.g., 1km/h to 100km/h in a second. Another advantage of this approach is that it can be used regardless of the periodicity of the CAN ID to be monitored. Malhotra et al. [23] in particular prove the possibility of using LSTM encoder-decoder architectures as reconstructors (instead of the more intuitive predictors) to detect anomalies in multi-sensor contexts.

Regarding time-series anomaly detection on automotive networks, Narayanan et al. [45], and Levi et al. [46], propose an ADS by training a Hidden Markov model (HMM) to learn the vehicle's normal behavior and classify anomalies. The intuition behind this approach is that a vehicle's behavior is considered as a sequence of finite events that are dependent on the previous state, similar to a Markov process. However the number of symbols that CAN IDs can produce is not necessarily restricted to a fixed dictionary, rendering Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) inadequate in most cases. Another interesting approach to temporal analysis is the one of Stabili and Marchetti [47] and of Groza and Murvay [48]. They implement two machine learning algorithms that focus on the hamming distance between two successive packets to detect abnormalities. Although in some specific contexts the IDSs have low performances, such as in detecting replay attacks, the results in detecting masquerade and fuzzing attacks are extremely promising.

Finally, RNNs have been shown to be effective modelers in time-series analysis. They are considered to be a natural fit when dealing with multivariate sequences, with no natural restrictions on the symbols' dictionary size, and even allowing the use of arbitrary sequence lengths when using specialized units. Taylor et al. [22], [49] proposed a system that trains models as predictors of CAN traffic data, then it classifies unexpected predictions as potential anomalies. This model works by applying LSTM neural networks to learn the normal sequences in the data. These models are trained to predict data symbols given a set of normal sequences. An anomaly score is computed based on the difference between the real and predicted data frame (i.e., the prediction error and it is used as the anomaly detection metric. The authors also compared the predictor with a multi-step multivariate HMM, which is adapted to handle high dimensional data. However, the detector struggled with several ID with low word-variability, as well as, short-lasting anomalies. The affinity between Taylor et al.'s work and CANnolo makes it perfect as a comparison to show CANnolo's strengths.

IV. Methodology

In the paper at hand, CANnolo consists of two main components, namely an LSTM-based autoencoder, a trainable model to learn the behaviour of normal signals, and an anomaly detector, to analyze the reconstruction errors. Figure 2 shows an overview of the autoencoder-based Anomaly Detection System (ADS) inhere proposed. The idea behind this schema is to create a reconstructed time series of CAN packets for each CAN ID that minimizes the reconstruction error so that a greater error rate would flag any potential anomaly.

The anomaly detector works by computing the statistical characteristics of reconstruction errors over a separate validation dataset; then, it assigns a distance score that indicates how far a given reconstruction error is from the expected normal distribution. Differently from the detector proposed by Taylor et al. [22] autoencoders do not use any validation data with anomalous traffic to tune the anomaly signal. Instead, the inhere proposed CANnolo is a completely unsupervised approach to anomaly detection.

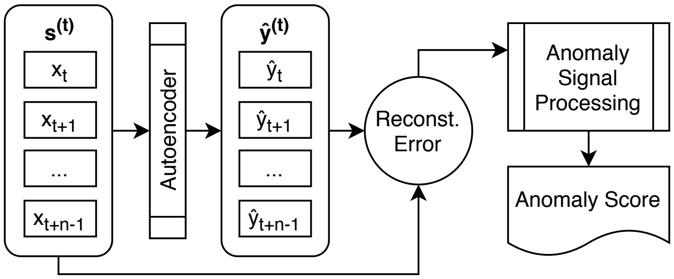


Fig. 2: Overview of the autoencoder-based ADS.

A. Model Architecture

The starting point for this research has been provided by Malhotra et al. [23]. In their publication, the authors proposed an autoencoder architecture with one single recurrent LSTM layer for the encoder and one identical for the decoder. Moreover, since the context of this research is of temporal nature, and following the suggestions of Taylor et al. [22] , the input data features a rolling window of n CAN packets, as will be discussed later in the section, the n hyperparameter is configurable. Experiments with $n = 20, ..., 40$ have been carried out, leading to the decision to use 40 packets as the window size. Each packet consists in at most $k = 64$ bits, where this hyperparameter changes according to each CAN ID. To be precise, for each ID, the condensed notation is used, i.e., the constant bits are removed [REF].

Encoder and Decoder hidden layers.

Designing a DL network is a tedious and significant time-consuming task, hence instead of starting from scratch, we picked the base models presented by Malhotra et al. [23]. In their model, the authors deployed two single recurrent layers, one for the encoding and one for the decoding. During the search for a better network layout, we tested several combinations of data timesteps, layers, numbers of LSTM and GRU units (from 64 to 256), and anti-overfitting techniques (such as different activation functions and weights, dropout layers [50] and L1/L2 regularizers [51] among others). This search has been performed, for each CAN ID, following the random hyperparameter search methodology, i.e., test a casual combination of parameters to be tested against a validation set to find the network with the lowest reconstruction error. The chosen CANnolo architecture is the network with the on-average lowest reconstruction error across all IDs. Figure 3 presents the architecture for the autoencoder.

Hence, CANnolo architecture is defined as follows:

- An input layer receiving a matrix of $n = 40$ CAN packets $\times$ k bit for each ID. Notice that the size of k varies for each ID since the compact notation removes the constant bits.
- A dense layer, consisting of $C = 256$ units with hyperbolic tangent activation functions.
- A 20% dropout layer.
- Two recurrent LSTM layer with $L = 128$ units each.

The final layer provides two outputs, namely the encoder's output and the hidden states h (technically, this also contains the cell states c).

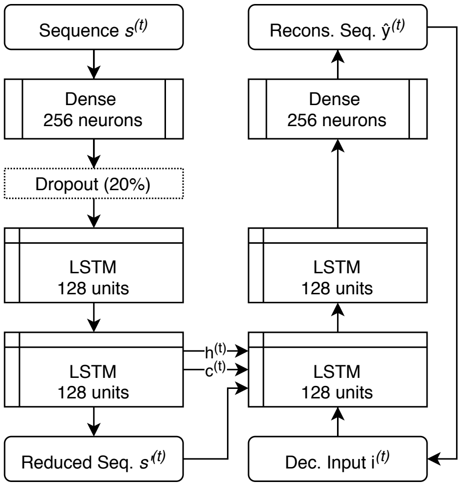


Fig. 3: Autoencoder architecture.

Symmetrically, the decoder consists of:

- Two recurrent LSTM layers with $L = 128$ units each.
- A dense layer, consisting of $C = 256$ units with sigmoid activation functions (to scale the results to the [0,1] interval).

Autoencoder postprocessing. The post-processing procedure for the autoencoder is straightforward, and it does not require evaluating parameter combinations. Following what proposed by Malhotra et al. [23], [52], for each CAN ID, the reconstruction error distribution is calculated using only legitimate data from the validation set. The resulting mean μ and covariance Σ matrixes indicate the multivariate Gaussian distribution of the non-anomalous data. A Mahalanobis distance between this distribution and any given reconstruction error distribution from a test sequence will suffice to identify an anomaly.

B. Training

For the autoencoder, the training phase consists of reconstructing normal traffic data by minimizing the reconstruction error between a given source sequence $s^{(t)}$ and a target sequence $y^{(t)}$. The model is optimized with a 128 Adam batch [53] and a binary cross-entropy loss function. Early stop checks are in place in case the reconstruction error fails to improve for 10 consecutive epochs.

For one thing, one may consider using the same source sequence as the target. However, Sutskever et al. [54] demonstrated that sequence reconstruction performance improves when reversing the symbols in the target sequence. Experimental evidence demonstrates that there is no difference in reversing the source or the target sequence.

For another thing, as demonstrated by Malhotra et al. [23], the previous outputs from the decoders can be used as input for the first layer of the decoder LSTM. Thus, during the training phase, the reconstructed sequence $\hat{y}^{(t)}$, alongside with the hidden states $h^{(t)}$ and cell states $c^{(t)}$, are stored and reinjected in the next iteration $t + 1$. On the contrary, during the testing phase, the decoder is initialized with a vector of zeros.

Note that the source s and target y sequences are constructed by shifting a $n = 40$ time window. Thus, each instant t, will be included in at most n sequences (partially empty sequences are discarded).

To formalize the notation, for any give time t, let:

- the source sequence $s^{(t)} \in X^n$ be a vector $x_t, \dots, x_{t+n-1}$ of n symbols;
- the target sequence $y^{(t)} \in X^n$ be a vector consisting in the same symbols as $s^{(t)}$, but in a *reversed* order, i.e., $y^{(t)} = x_{t+n-1}, \dots, x_1$;
- the encoder outputs be:
  - $s'^{(t)}$ for the reduced sequence,
  - $h^{(t)}$ for the hidden states, and
  - $c^{(t)}$ for the cell states;
- the final reconstructed sequence (i.e., the decoder's output) be $\hat{y}^{(t)} = \hat{x}_t, \dots, \hat{x}_{t+n-1}$.

Hence, the decoder input will be composed by the aforementioned $s'^{(t)}$, $h^{(t)}$, $c^{(t)}$, and a vector of n symbols $i^{(t)} \in X^n$. This last piece is defined as:

- a vector of zeros if $t = 0$ (i.e., $i^{(t=0)} = 0_0, \dots, 0_{n-1}$), and,
- as the decoder's output of the previous stage if $t > 0$ (i.e., $i^{(t>0)} = \hat{y}^{(t-1)}$).

C. Anomaly signal processing

The anomaly detection procedure consists in calculating the Mahalanobis distance between two sequences. For a single reconstruction, unlike Malhotra's approach [23], a binary cross-entropy function is employed. Early tests suggested that this function is to be preferred to the absolute error function used by the authors [23]. Formally:

$$-(\hat{b}_k \log(b_k + \epsilon) + (1 - \hat{b}_k) \log(1 - b_k) + \epsilon)$$

where $\hat{b}_k$ is the k-th bit in the reconstructed sequence $\hat{y}^{(t)}$ and b_k is the k-th bit of the source sequence $s^{(t)}$.

Normal reconstruction error distribution. A dataset consisting only of legitimate data sequences has been used as the baseline to establish the 'normal' behaviour for any given CAN ID. The reconstruction error distribution has been extracted from the join list of reconstruction errors as described above. The distribution is thus fitted to a multivariate Gaussian distribution, providing the mean μ (of $1 \times k$ cardinality) and covariance matrix Σ (of $k \times t$ cardinality).

Anomaly score. The anomaly score indicates the likelihood of the test sequence to be anomalous. Similarly to [23], [52], an anomaly score a is derived from the Mahalanobis distance measuring the distance between a reconstruction error and the distribution of normal/legitimate errors, and it is computed as follows:

$$a = (e - \mu)^T \Sigma^{-1} (e - \mu)$$

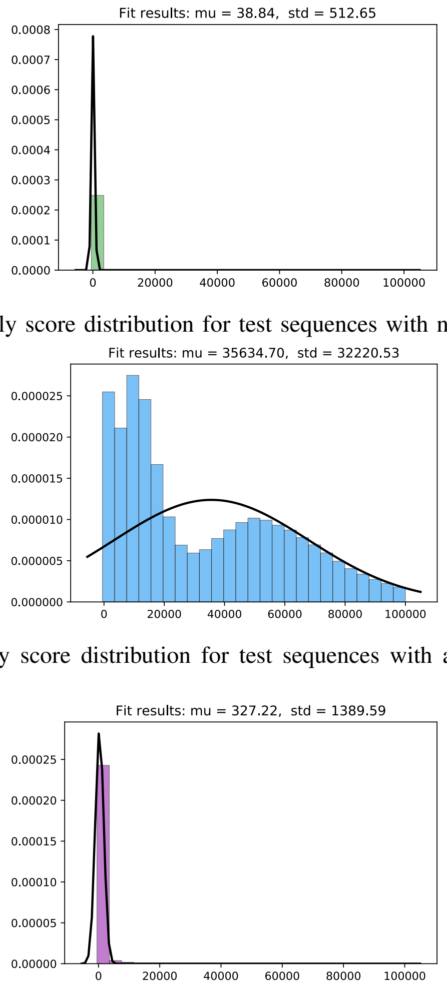
Where e is the reshaped reconstruction error vector.

Figure 4 presents three sample distributions.

Firstly, in Figure 4a, the baseline reconstruction error is presented showing that the distance between the source and the reconstructed sequences are generally low.

Secondly, in Figure 4b, an interleave anomaly is presented.

Fig. 4: Anomaly score distribution for various test sequences.



(a) Anomaly score distribution for test sequences with normal data.

(b) Anomaly score distribution for test sequences with an interleave anomaly.

(c) Anomaly score distribution for test sequences with a replay attack targeting a high variability field.

Finally, Figure 4c presents a replay anomaly. By comparison with Figure 4a, it is clear that the sequences are recognized as legitimate but temporally misplaced (low reconstruction errors interleaved with high reconstruction errors on anomaly's boundaries).

V. Experimental Evaluation

In this section we describe the steps required to implement the methodology presented above, alongside the analysis and tests done to compare our solution to the current state of the art. Specifically, we directly compare our results to those of Taylor et al. [22] (see Appendix A for a brief overview), since the challenge that the two IDSs tackle is the same. After, we proceed to make a short analysis of how the results change if we test our system against real world attacks. We finally present an analysis of the system requirements to implement our methodology.

A. Dataset Overview

The dataset we used to test CANnolo is a 10 million packets long real-world dataset retrieved from an Alfa Romeo Giulia

Veloce. The dataset has been generated while driving for an hour in city and highway contexts multiple times, both at high and slow speeds, with the goal to showcase the majority of the possible values obtainable by each ID. For more information we refer the reader to Zago et al. [55].

Following a hold-out validation approach, but keeping into consideration the time dependence, we split our dataset into three parts. We used the first 60% of the dataset for training CANnolo. We used the following 20% for validation. In particular, we perform hyper-parameter tuning, and we fit the distribution of reconstruction errors and the anomaly scores of the legitimate CAN data. During this phase, we applied early stopping to avoid overfitting. Finally, we used the lasts 20% solely for testing: we injected the crafted anomalies into the real-world dataset and evaluated CANnolo's performance.

We consider only CAN IDs compatible with temporal-based analysis [22]: with high frequency, high symbol variability, and non-trivial symbol complexity (i.e., the percentage of unique symbols is higher than 1%). The ID in our dataset that match these characteristics are 12, specifically: `0DE, 0EE, 0FB, 0FC, 0FE, 0FF, 1F7, 1FB, 11C, 100, 104, 116`

Data preprocessing. CAN data traffic, as directly logged from the bus, is not suitable for training a machine learning model. For these reasons, we first pre-process the dataset. To do so, we apply the field classification algorithm presented by Marchetti and Stabili in READ [56], with the improvements and adjustments presented in [55]. The algorithm follows this general procedure: it uses as an input the 64 bits of the data frame; then, it calculates the rate at which each bit changes value, and, based on this information, it divides the packet in blocks that are considered to be a single piece of data. The output includes the leftmost-bit (i.e., index), length, type (i.e., multi-value, sensor, constant), category of the field (i.e., low/medium/high variability), and the number of unique symbols. The category of the field as in Taylor et al. work [22] is defined as in Table I.

B. Anomaly generation

To test CANnolo, we artificially crafted anomalies by modifying the CAN data messages. The injected anomalies allow recreating the same data flows that would be present in the attacks presented in II-C.

We consider mainly three different types of anomaly, which are *interleave*, *discontinuity* and *data field* anomalies. *Interleave* anomalies represent situations in which the attacker starts sending messages on the bus without first silencing the victim ECU. This would result in two parallel streams of competing messages on the same ID, for example, messages detecting the vehicle speed being both at both 50km/h and 120km/h at the same time. To implement this attack, we take a sequence from a different time instance and we insert it alternated with the correct one.

TABLE I: Classification of IDs according to their unique symbol count in the CAN dataset

Classification	N° of unique symbols
Low variability	< 100
Medium variability	$[100, 500]$
High variability	> 500

TABLE II: Non data-related anomaly Area Under the Curve (AUC) for both IDSs.

	Interleave		**Discontinuity**	
	Predictor	CANnolo	Predictor	CANnolo
0DE	1	1	0.9996	1
0EE	1	0.9846	0.9939	0.9437
0FB	1	1	0.9938	1
0FC	1	1	0.9921	1
0FE	1	1	0.9998	1
0FF	1	1	0.9911	1
1F7	0.9935	1	0.9563	1
1FB	0.9894	1	0.9961	1
11C	0.9982	1	1	1
100	1	1	0.9957	1
104	0.9965	1	1	1
116	1	0.9997	0.9974	0.7892
avg	0.9981	0.9986	0.9930	0.9777

Discontinuity anomalies represent the case where an attacker silences an ECU, hence creating a discontinuity in the stream of data if IDs sent by said ECU. This attack is implemented by removing small sequences of messages from an authentic stream of data.

Both *discontinuity* and *interleave* anomalies would, although, cause an abnormality at the rate in which messages appear in the CAN bus, assuming these are periodic. Therefore, they could be efficiently detected by frequency-based methods. However, we consider them for completeness as they alter the normal flow of a data sequence, and, hence, they can be easily detected by CANnolo.

Our threat model (see II-C) also includes cases where data frame contents are directly modified. To test them, since we do not have knowledge of the effective meaning of each bit of all the tested IDs, we employ a fuzzing-like approach, following a similar procedure of Taylor et al. [22]. As Taylor et al. did, we define five different modifiers functions: set the field of the packet to its maximum value, minimum, a constant, a random value, or replaying a previous value found in the dataset. We refer to these anomaly tests as *data field* anomalies. In general, it is hard to recreate attacks without being capable to know the actual meaning of the attack itself. This is a common problem while testing automotive security measures, since often there's no documentation available to describe the function of a field of a packet. We claim that using this five different modifier functions we still cover the majority of realistic cases, since the attacker's goal may lead him to force a specific value, potentially close to the boundaries of the accepted ones but without crossing said boundaries (minimum, maximum and constant value functions), or to replay previous sequences to lower detection rates (replay) or to try a fuzzing attack to trigger unexpected behaviors (random).

An anomaly we purposefully did not consider is the alteration of bits that are meant to be fixed in the packet. Although the alteration of such bits may indicate an exploitation attempt, we consider the task of detecting such anomalies trivial through rule-based and not machine learning-based IDSs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3038991, IEEE Transactions on Network and Service Management

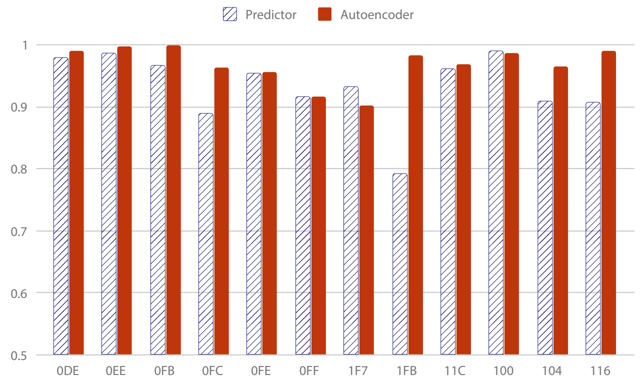


Fig. 5: Average AUC per ID for all data field anomalies. In blue Taylor et al.'s Predictor, in red CANnolo's.

C. Experimental Settings

We perform the experiments using continuous and non-overlapping data sequences. Each sequence is 3 seconds long and contains 300 observations since all the CAN IDs we considered have a message frequency of 10 milliseconds.

For each anomaly type, we create a new set of test sequences. For the data field anomalies we keep 60% of the validation dataset as legitimate sequences; the remaining 40% is altered using the anomaly functions, which are different functions that we used to generate the data anomalies, specifically: set to maximum, set to minimum, set to a constant, set to random value, replay field.

Four duration values for the anomalies are evaluated: 0.2, 0.5, 1, and 1.5 seconds. The anomaly starting time is chosen at random, but not earlier than one-third of the total length of the test sequence and early enough to accommodate the entire anomaly duration. This allows us to evaluate the impact of the duration time for each anomaly.

Note that Taylor et al. in their experiments implemented field anomalies in a way that only targets sensor fields. In this work we removed this restriction so we can better evaluate the detector performance using a greater range of fields.

D. Experimental Results

We proceed to present the results and comparisons between CANnolo and Taylor et al. [22]. As explained above, the first two anomalies (interleave, discontinuity) are intrinsically different from the last one (data field anomaly). Therefore, we evaluated them separately as *sequence-based* and *data field-based* anomalies. We use the AUC as the performance metric to compare CANnolo with Taylor et al.'s detector.

1) Sequence-based anomalies: **Interleave anomalies.** The AUC for every ID is presented in Table II as well as the average performance for both detectors. These anomalies were the easiest to detect because of the noticeable impact they have in the data stream. Half of the symbols in the sequences are drawn from a pool of different points in time. Hence, each anomalous sequence has half of its symbols from a completely different context. Both detectors score very high AUC values.

TABLE III: AUC difference between CANnolo and Taylor et al's detector for all data field anomalies, followed by the overall AUC for both.

ID	AUC difference
0DE	0.0090
0EE	0.0093
0FB	0.0327
0FC	0.0729
0FE	0.008
0FF	0.006
1F7	-0.0305
1FB	0.1913
11C	0.0067
100	-0.0049
104	0.0551
116	0.0822
Overall	
CANnolo	**Taylor et al**
0.967711	0.932265

Discontinuity anomalies. Both detectors yield significantly good results overall, as visible in Table II. However, on average, Taylor et al. obtained a higher AUC, albeit slightly. The main factor contributing to this result come from a specific ID (`116`), for which CANnolo performed poorly with an AUC of 0.789 while the predictor achieved 0.997. Through a visual and statistical analysis of the data we can claim with a good level of certainty that ID `116` is composed almost if not completely of counters. Although we could not find another ID with similar properties, what we assume is that given the repetitiveness of said fields, CANnolo remains fairly capable of reconstructing the time-series even when there is a discontinuity in the middle of the packet, hence not increasing the reconstruction error enough to easily distinguish between attacks and valid data sequences. Nonetheless we are satisfied with the results since, as explained above in Section V-B these anomalies are mainly evaluated for completeness.

2) Data field-based anomalies: In Figure 5, we illustrate a direct comparison between the detectors for each ID, we observe that in 7 out of 12 cases performance was very similar as there's an AUC difference less than 0.02. However, for 4 IDs the difference was more significant, with AUC improvements ranging from 0.05 to 0.19 as also shown in Table III alongside the average AUC results for all IDs over all

TABLE IV: Overall AUC of the detectors grouped by anomaly function.

Anomaly function	Taylor et al.'s Predictor	CANnolo
Max	0.9341	0.9440
Min	0.8986	0.9264
Constant	0.9508	0.9514
Random	0.9908	0.9944
Replay	0.8087	0.9521

TABLE V: Overall data field AUC anomaly results divided by field variability.

Field variability	Taylor et al.'s Predictor	CANnolo
Low	0.9239	0.9741
Medium	0.8837	0.9106
High	0.9594	0.9754

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3038991, IEEE Transactions on Network and Service Management

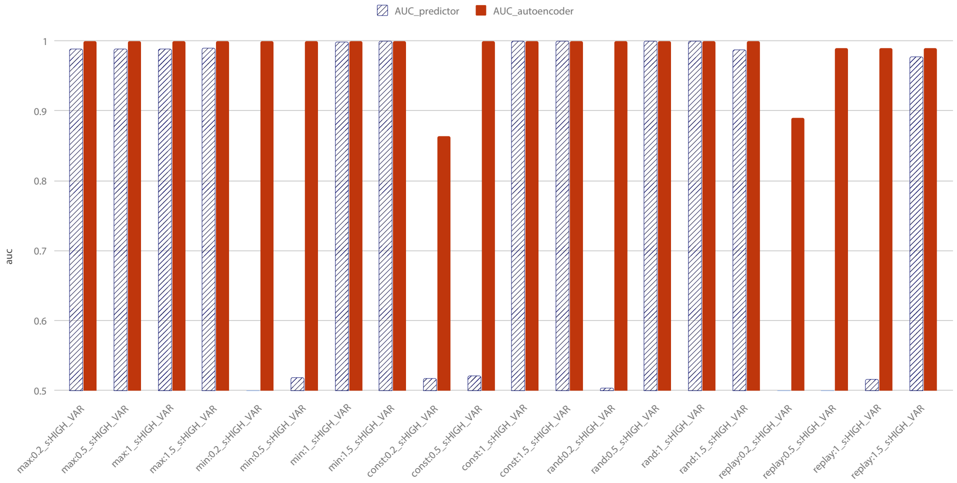


Fig. 6: AUC results for ID 1FB. The x-axis indicates the parameters for each test. In blue Taylor et al.'s Predictor, in red CANnolo's.

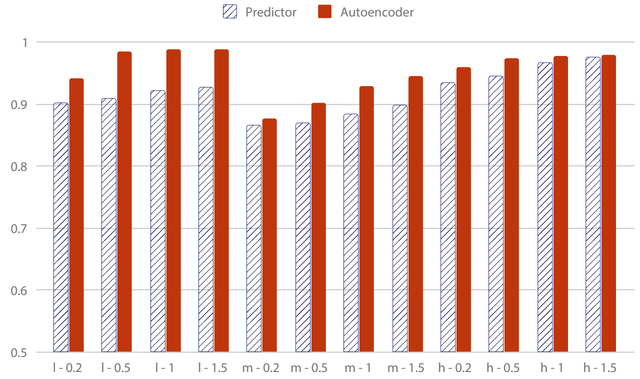


Fig. 7: Overall detection AUC, grouped by variability and duration. The first letter stands for the target field variability (Low, Medium, and High), and it is followed by the anomaly duration in seconds. In blue Taylor et al.'s Predictor, in red CANnolo's.

TABLE VI: Overall detector AUC anomaly grouped by attack duration.

Anomaly duration (s)	Taylor at al.'s Predictor	CANnolo
0.2	0.9003	0.9260
0.5	0.9084	0.9535
1	0.9241	0.9648
1.5	0.9336	0.9704

data field anomalies, showing that CANnolo higly outperforms Taylor et al's predictor.

Analyzing the results for each ID allows us to discern trends that are otherwise obfuscated by overall results, particularly when the detectors have similar performance across the parameter space. To better illustrate the performance of the detectors under analysis, we take ID `1FB` as an example, which is illustrated in Figure 6. We chose this ID as it presents the most significant improvement of CANnolo with respect to Taylor et al.'s detector. We observe that the type of anomaly influences the performance of the detector. In particular, for ID `1FB`, the CANnolo performs significantly better for every duration when testing for *field replay* anomalies. The detectors share good results in several scenarios. For ID `1FB`, we can observe that Taylor et al., eventually catches up with the autoencoder with longer-lasting anomalies. However, the performance gap at shorter durations is quite significant, even yielding an AUC difference of 0.49. In this particular case, Taylor et al.'s predictor performs slightly better than just making a random choice on whether or not the sequences are anomalous.

Influence of the anomaly function. We evaluate if there is a relation between the anomaly function and the performance of the detector. We combine the AUC of all IDs and group them by the anomaly function, as presented in Table IV. We can observe that performance is relatively similar between the detectors, yet CANnolo always resulted as the highest performer overall. However, we can point out a clear pattern, CANnolo is significantly better at detecting *replay* anomalies than Taylor et al.'s predictor. The difference between their respective AUCs is 0.15.

Influence of field variability. In Table V, we see the average AUC results for all IDs, grouped by the field target variability. We see that in this framing, CANnolo on average performs better. One of the most remarkable results is the improvement of attacks targeting low variability fields. Taylor et al. highlighted that their detector performed worse using fields of low variability, and on high variability cases, it performs best. Our results partially confirm this conclusion.

Influence of duration. The duration of an anomaly has a clear impact on the performance of the detector, as can be seen in Table VI. For both detectors, longer anomalies result in higher AUC values. By grouping variability and duration, as illustrated in Figure 7 and in Table VII, we can confirm CANnolo performs significantly better on average for all short-lasting anomalies that target low variability fields.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3038991, IEEE Transactions on Network and Service Management

TABLE VII: AUC results grouped by anomaly duration and target field category. Blank cells indicate that a relevant field does not exist. P stands for the predictor results, and AE. stands for the autoencoder results.

		Low var.				Mid var.				High var.			
ID		0.2(s)	0.5(s)	1(s)	1.5(s)	0.2(s)	0.5(s)	1(s)	1.5(s)	0.2(s)	0.5(s)	1	1.5(s)
0DE	P.	0.9721	0.9725	0.9772	0.9795					0.9520	0.9764	0.9879	0.9914
	AE.	0.9819	0.9944	0.9996	0.9997					0.9560	0.9832	0.9897	0.9910
0EE	P.									0.9870	0.9749	0.9878	0.9888
	AE.									0.9911	0.9957	1.0000	0.9994
0FB	P.					0.9561	0.9553	0.9013	0.9640	0.9903	0.9905	0.9920	0.9940
	AE.					0.9989	1.0000	0.9983	1.0000	1.0000	1.0000	1.0000	1.0000
0FC	P.	0.9376	0.9419	0.9456	0.9475	0.7119	0.7104	0.7396	0.7566	0.9855	0.9924	0.9933	0.9941
	AE.	0.9942	0.9976	0.9951	1.0000	0.8490	0.8974	0.8837	0.9203	1.0000	1.0000	1.0000	1.0000
0FE	P.	0.9252	0.9377	0.9432	0.9504	0.8965	0.9019	0.9047	0.9107	0.9966	0.9998	0.9991	0.9997
	AE.	0.9044	1.0000	1.0000	1.0000	0.8289	0.9063	0.9913	0.9924	1.0000	1.0000	1.0000	1.0000
0FF	P.	0.9502	0.9603	0.9635	0.9667	0.7677	0.7515	0.8253	0.8245	0.9573	0.9737	0.9803	0.9827
	AE.	0.9754	0.9997	1.0000	0.9993	0.7768	0.7519	0.8136	0.8352	0.9304	0.9893	0.9951	0.9981
1F7	P.	0.9326	0.9367	0.9439	0.9521	0.9081	0.9425	0.9734	0.9744	0.8734	0.8760	0.8796	0.8778
	AE.	0.9039	0.9157	0.9198	0.9133	0.8442	0.8793	0.9148	0.9397	0.8306	0.8475	0.8565	0.8706
1FB	P.									0.5970	0.7031	0.9005	0.9909
	AE.									0.9507	0.9980	0.9979	0.9979
11C	P.	0.9168	0.9149	0.9191	0.9212					0.9879	0.9894	0.9918	0.9934
	AE.	0.8272	0.9841	0.9978	0.9998					0.9918	0.9990	0.9999	1.0000
100	P.									0.9822	0.9882	0.9906	0.9954
	AE.									0.9745	0.9778	0.9873	0.9964
104	P.	0.8179	0.8174	0.8205	0.8218	0.9559	0.9579	0.9609	0.9607	0.8939	0.8881	0.9001	0.9026
	AE.	0.9884	0.9995	1.0000	1.0000	0.9670	0.9759	0.9719	0.9802	0.8941	0.9014	0.9020	0.9020
116	P.	0.7599	0.7954	0.8579	0.8748					0.9973	0.9976	0.9978	0.9993
	AE.	0.9508	0.9854	0.9938	0.9869					0.9982	0.9988	0.9995	0.9977
Avg	P.	0.9016	0.9096	0.9214	0.9268	0.8660	0.8699	0.8842	0.8985	0.9334	0.9458	0.9667	0.9758
	AE.	0.9408	0.9846	0.9883	0.9874	0.8775	0.9018	0.9289	0.9446	0.9598	0.9742	0.9773	0.9794

TABLE VIII: Overall AUC anomaly results for speed-related ID targeted attacks.

Attack	Precision	Recall	F1score	AUC
Single replay	0.9017	1	0.9483	0.9853
Forced changing data	0.9017	1	0.9483	0.9901
Complete overwrite	0.9017	1	0.9483	0.9901

E. Real-world attacks results

Since all the attacks proposed above have been simulated using a set of given patterns, one may claim that they do not closely represent the behavior of an actual attacker. Although we do consider them a good representation, proving that more targeted attacks can be detected anyway just strengthens CANnolo's position. For this reason we created three different attacks that are meant to simulate a knowledgeable attacker on the ID that carries the speed signal of our vehicle (as explained in [55]). The first attack simulates a replay attack of a single packet, repeated frequently as an attacker would do to force the vehicle to consider his value over the non malicious one passing on the bus. The second attack, slightly more complex, simulates an attacker sending a stream of data on the bus with a value changing in a realistic way, while the non malicious value still passes on the bus. Finally, the last and more complex attack has the attacker completely taking control of the bus in a random moment in the middle of the time-series, and then sending a sequence of coherent data (replaying a previous sequence). Since our attacks had to be realistic, we took inspiration from Seo et al.'s [57] dataset, available here [58]. As visible in Table VIII, the results in all three cases are extremely close to the ones obtained above.

F. Computation and timing analysis

The memory requirements for CANnolo depend on the specific loaded model, but generally float between 300 and 800 Mb. After loading the model, the actual memory usage for a single time-series reconstruction is lower than 10 Mb, making it negligible in relation to the overall requirements. We tested CANnolo's performance on a 2014 3.5ghz single core processor. It is hard to retrieve valuable information on the actual types of processors used in new vehicles' infotainment systems and highest-end ECUs, but we claim that given the nowadays low cost of computation and the new devices presented by automotive chip manufacturers like Tesla and Nvidia [59] with multi-core high frequency architectures, our testbed is realistic. Especially considering that CANnolo should be installed on one single ECU. The average computation overhead for a time-series is 0.65 seconds, with worst cases taking up to 0.95 seconds if no bits can be removed from the packet. This makes CANnolo react fast, but potentially not fast enough for strong real-time requirements. This said, we have to consider that ML based IDSs should not be used for safety critical reactions in vehicular environments, due to the fact that it is extremely hard to prove their complete lack of false positives, hence ensuring that the IDS cannot become a threat to the safety of the passengers. Overall, we consider CANnolo an extremely valuable asset for non strong real-time intrusion detection in vehicular networks.

VI. Conclusions

In this paper, we presented CANnolo, a novel IDS based on LSTM autoencoders designed for automotive on-board networks. CANnolo works in completely unsupervised learning fashion, where no attack data was used to either train or fine-tune the detector.

Overall, CANnolo showed better performance than the state-of-the-art model by improving the detection rate and covering its main weaknesses. To evaluate our approach, we compared it to a state-of-the-art anomaly detection system designed for automotive data, on a set of simulated attacks applied over a real-world dataset. Through our experiments, we observed that CANnolo is a suitable modeling technique for CAN traffic data, as it can successfully model time-series, with no assumptions on time-series characteristics. These include whether the signal is predictable in some way, or more generally, its stationarity.

CANnolo's biggest limitation has to be its relatively slow computation. In future works, the first aim should be that of obtaining the same results while creating an overall lighter system. It is also of extreme interest the study of correlation between different IDs, since correlating IDs that carry non-independent data has already been proven useful in intrusion detection. Nonetheless, correlation between different data over time does increase the complexity of the system, potentially increasing an hypothetical reaction time. A future work would focus on studying the most lightweight system capable of correlating data from different IDs.

Acknowledgment

This project has been partially supported by BVTech SpA under project grant UCSA and by a predoctoral grant from the Spanish National Cybersecurity Institute (INCIBE) within the program "Grants for the Excellence of Advanced Cybersecurity Research Teams", with code INCIBEI-2015-27353.

References

[1] T. Nolte, H. Hansson, and L. L. Bello, "Automotive communications-past, current and future," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, Sept 2005, pp. 8 pp.–992.

[2] National Instruments. Controller area network (CAN) overview. [Online]. Available: http://www.ni.com/white-paper/2732/en/

[3] R. B. GmbH, "Can specification, version 2.0," Robert Bosch GmbH, Stuttgart, Germany, Standard, 1991.

[4] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication "can-bus" security and vulnerabilities," *International Journal of Computer Science and Network*, vol. 6, no. 6, p. 720–727, Dec 2017.

[5] S. Zanero, "When cyber got real: Challenges in securing cyber-physical systems," in *IEEE SENSORS 2018*. New Delhi, India: IEEE, 2018.

[6] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks—practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11—25, 2011.

[7] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *DEF CON*, vol. 21, pp. 260–264, 2013.

[8] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[9] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 447–462.

[10] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[11] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *black hat USA*, vol. 2014, 2014.

[12] A. Greenberg. (2015, Jul) Hackers remotely kill a jeep on the highway - with me in it. Wired, Online. Last visited on 2019-12-12. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[13] S. Garg, K. Kaur, G. Kaddoum, F. Gagnon, N. Kumar, and Z. Han, "Sec-iov: A multi-stage anomaly detection scheme for internet of vehicles," in *Proceedings of the ACM MobiHoc Workshop on Pervasive Systems in the IoT Era*, 2019, pp. 37–42.

[14] S. Garg, K. Kaur, S. Batra, G. Kaddoum, N. Kumar, and A. Boukerche, "A multi-stage anomaly detection scheme for augmenting the security in iot-enabled applications," *Future Generation Computer Systems*, vol. 104, pp. 105–118, 2020.

[15] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[16] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin, and M. Wen, "Mbid: Micro-blockchain-based geographical dynamic intrusion detection for v2x," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 77–83, 2019.

[17] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] C. Miller and C. Valasek, "Advanced can message injection," June 2016. [Online]. Available: http://illmatics.com/can%20message%20injection.pdf

[20] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017, pp. 185–206.

[21] S. Longari, M. Penco, M. Carminati, and S. Zanero, "Copycan: An error-handling protocol based intrusion detection system for controller area network," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019, pp. 39–50.

[22] A. Taylor, "Anomaly-based detection of malicious activity in in-vehicle networks," Ph.D. dissertation, Université d'Ottawa/University of Ottawa, 2017.

[23] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[24] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, pp. 1–5.

[25] M. S. alDosari, "Unsupervised anomaly detection in sequences using long short term memory recurrent neural networks," Ph.D. dissertation, 2016.

[26] J. P. Assendorp, "Deep learning for anomaly detection in multivariate time series data," Ph.D. dissertation, Hochschule für Angewandte Wissenschaften Hamburg, 2017.

[27] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[28] J. Andress, *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Syngress, 2014.

[29] C. Smith, *The car hackers handbook: a guide for the penetration tester*. No Starch Press, 2016.

[30] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*. IEEE, 2016, pp. 1–6.

[31] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *Industrial Control Systems Security (WCICSS), 2015 World Congress on*. IEEE, 2015, pp. 45–49.

[32] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 63–68.

[33] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, 2017, pp. 1–4.

[34] H. Lee, S. H. Jeong, and H. K. 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). IEEE, 2017, pp. 57–5709.

[35] K.-T. Cho and K. G. Shin, “Error handling of in-vehicle networks makes them vulnerable,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1044–1055.

[36] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, “Cloaking the clock: Emulating clock skew in controller area networks,” *arXiv preprint arXiv:1710.02692*, 2017.

[37] M. Müter, A. Groll, and F. C. Freiling, “A structured approach to anomaly detection for in-vehicle networks,” in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. IEEE, 2010, pp. 92–98.

[38] T. Dagan and A. Wool, “Parrot, a software-only anti-spoofing defense system for the can bus,” *ESCAR EUROPE*, 2016.

[39] P.-S. Murvay and B. Groza, “Source identification using signal characteristics in controller area networks,” *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 395–399, 2014.

[40] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, “Identifying ecus using inimitable characteristics of signals in controller area networks,” *arXiv preprint arXiv:1607.00497*, 2016.

[41] M. W. Spicer, “Intrusion detection system for electronic communication buses: A new approach,” Master’s thesis, Virginia Tech, 2018.

[42] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, “Car hacking identification through fuzzy logic algorithms,” in *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

[43] M.-J. Kang and J.-W. Kang, “Intrusion detection system using deep neural network for in-vehicle network security,” *PloS one*, vol. 11, no. 6, p. e0155781, 2016.

[44] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, “A language-based intrusion detection approach for automotive embedded networks,” *International Journal of Embedded Systems*, vol. 10, no. 1, pp. 1–12, 2018.

[45] S. N. Narayanan, S. Mittal, and A. Joshi, “Obd_securealert: An anomaly detection system for vehicles,” in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[46] M. Levi, Y. Allouche, and A. Kontorovich, “Advanced analytics for connected cars cyber security,” *arXiv preprint arXiv:1711.01939*, 2017.

[47] D. Stabili, M. Marchetti, and M. Colajanni, “Detecting attacks to internal vehicle networks through hamming distance,” in *2017 AEIT International Annual Conference*. IEEE, 2017, pp. 1–6.

[48] B. Groza and P.-S. Murvay, “Efficient intrusion detection with bloom filtering in controller area networks,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, 2018.

[49] 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 (DSAA)*. IEEE, 2016, pp. 130–139.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[51] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[52] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*. Presses universitaires de Louvain, 2015, p. 89.

[53] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[54] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” pp. 3104–3112, 2014.

[55] M. Zago, S. Longari, A. Tricarico, M. Gil Pérez, M. Carminati, G. Martínez Pérez, and S. Zanero, “Recan - dataset for reverse engineering of controller area networks,” *Data in Brief*, vol. 29, p. 105149, 2020.

[56] M. Marchetti and D. Stabili, “Read: Reverse engineering of automotive data frames,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[57] E. Seo, H. M. Song, and H. K. Kim, “Gids: Gan based intrusion detection system for in-vehicle network,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, pp. 1–6.

[58] H. Lab. Car-hacking dataset for the intrusion detection. [Online]. Available: https://sites.google.com/a/hksecurity.net/ocslab/Datasets/CAN-intrusion-dataset

[59] Nvidia. Jetson agx xavier developer kit. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

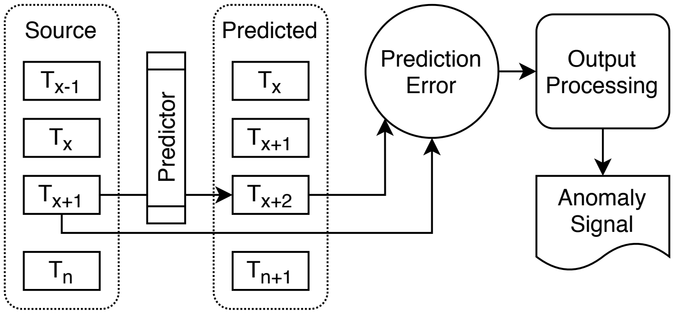


Fig. 8: Overview of the predictor algorithm.

Appendix

Since we confront with Taylor et al.’s work [22] we deem that a brief overview of their system is needed. As presented in Section III it is an LSTM based RNN which act as a predictor of the next sequence, as shown in Figure 8. Their network is composed of two hidden layers each with 128 units, and with two LSTM layers each composed of 512 units. To calculate the final anomaly score they tested different techniques depending on which one is the most efficient for the specific ID.

TABLE IX: Symbol score and combination results for all IDs

ID	Symbol output	Symbol combination

`0DE,OEE,0FB,0FC,`
`OFF,11C,100,104`
	Max bit loss	Rolling window

`0FE,1F7,1FB`	Max bit loss	Maximum symbol loss
`116`	Average bit loss	Average symbol loss

While reimplementing the work done by Taylor et al. [22] we made some minor modifications due to performance in the postprocessing of the input data. Although we do not consider these modifications to significantly change the performances, we still deem necessary to notify the reader about them:

While choosing the best match of symbol scores and combination methods for all the IDs we analyzed, we discovered that there is no single best match. We used anomalous traffic samples from the validation dataset to choose the combination of symbol scores and combinations that maximises the AUC. In Taylor et al’s works [22], [49] the maximum loss symbol score was considered to perform best for all CAN IDs, however, when we replicated the experiment with our own data the result were more varied. For each ID we computed the AUC for every anomaly type using all possible output processing methods, the combination that performed best overall was chosen. The final results included a combination of the symbol scores: maximum and average, with combination scores: rolling window, log sum, maximum, and average.

The best combination results can be seen in Table IX. These are the anomaly signal outputs which indicate an anomaly score for a given test sequence.