Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Anomaly Detection in Vehicular CAN Bus Using Message Identifier Sequences

TAHSİN C. M. DÖNMEZ[1]

[1]Department of Computing, University of Turku, Turku, FI-20014, Finland (e-mail: tcmdon@utu.fi)

This work was supported by the PRYSTINE Project, funded by Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's H2020 Framework Programme and National Authorities, under grant agreement no. 783190.

ABSTRACT As the automotive industry moves forward, security of vehicular networks becomes increasingly important. Controller area network (CAN bus) remains as one of the most widely-used protocols for in-vehicle communication. In this work, we study an intrusion detection system (IDS) which detects anomalies in vehicular CAN bus traffic by analyzing message identifier sequences. We collected CAN bus data from a heavy-duty truck over a period of several months. First, we identify the properties of CAN bus traffic which enable the described approach, and demonstrate that they hold in different datasets collected from different vehicles. Then, we perform an experimental study of the IDS, using the collected CAN bus data and procedurally generated attacks. We analyze the performance of the IDS, considering various attack types and hyperparameter values. The analysis yields promising sensitivity and specificity values, as well as very fast decision times and acceptable memory footprint.

INDEX TERMS anomaly detection, intrusion detection, network security, controller area network, CAN bus, vehicular network, in-vehicle network

I. INTRODUCTION

Automotive industry is moving in a direction where connectedness and autonomy become essential features of vehicles. Connected vehicles and vehicles with autonomous driving capabilities are expected to bring benefits such as decreased number of traffic accidents, decreased traffic congestion, decreased fuel consumption and emissions, and increased mobility for people who cannot operate vehicles. Furthermore, realization of connected and autonomous vehicles might lead to widespread adoption of shared vehicles concept, which, in turn, could significantly reduce the total number of vehicles and the amount of parking space needed. Unfortunately, connectivity and autonomous driving capabilities come at a cost. Vehicles become more complex due to the inclusion of new hardware, software, and communication protocols. From a security point of view, this translates to the attack surface becoming larger. Moreover, it is not hard to imagine possible motivations for launching cyber-attacks on vehicles. These facts lead to legitimate concerns about security, and are the motivation behind numerous publications on the topic. Cui et al. [1], Thing et al. [2], and Yağdereli et al. [3] provide an overview of cyber-security aspect of connected and autonomous vehicles, including classification of applicable cyber-attacks and corresponding countermeasures. Parkinson et al. [4] also present an overview of the problem, and compile an extensive list of challenges to be addressed. Dominic et al. [5] investigate possible motivations for attacks, and assess risks.

Focus of this work is on the security of in-vehicle networks, which is only one aspect of this multifaceted problem. Communication capabilities in vehicles are often classified into two major groups: vehicle-to-everything (V2X) and in-vehicle. V2X communications take place between a vehicle and external entities such as other vehicles or infrastructure, whereas in-vehicle communications take place between on-board components such as sensors, actuators, electronic control units (ECU). Unlike V2X communication which has to rely on wireless technologies, in-vehicle communication takes place almost exclusively over wired networks using technologies such as Controller Area Network (CAN bus), FlexRay, Local Interconnect Network (LIN), Automotive Ethernet, Media Oriented Systems Transport (MOST), Byteflight, and Low-Voltage Differential Signaling (LVDS). CAN bus is one of the oldest and most widely adopted technologies for in-vehicle communications, and the security solution studied in this work is specifically targeted at this technology.

Contributions of this work are as follows:

- Previous work suggested intrusion detection in vehicular CAN bus via whitelisting message identifier sequences. We build upon previous work by treating the length of message identifier sequences as a hyper-parameter, and propose a variant which is less prone to false alerts. We build a prototype implementation, and perform an experimental study of the developed intrusion detection system using data collected from a vehicle.
- We identify the domain-specific property of vehicular CAN bus data which makes whitelisting of message identifier sequences a viable approach, and demonstrate that it holds for different vehicles.
- We introduce an open CAN bus dataset collected from a heavy-duty truck.

The rest of the paper is organized as follows. Section II is aimed at providing background information about Controller Area Network and its security, as well as about intrusion detection systems. Section III presents related work. The studied intrusion detection system is described in Section IV. Materials and methods are presented in Section V. Results of the experiment are presented in Section VI. In Section VII, we summarize and discuss our results. Section VIII concludes the paper.

II. BACKGROUND

In this section we provide background information about CAN bus and security aspects of its usage in in-vehicle communication, as well as intrusion detection systems, and their application to vehicular CAN bus.

A. CONTROLLER AREA NETWORK

Controller Area Network (CAN bus) was originally developed for interconnecting electronic control units (ECU) within automobiles, and the first version of the protocol was published in 1986 [6]. Today, it is a widely adopted technology for in-vehicle communication, and it also has applications in other contexts. Second version of the protocol CAN 2.0 was standardized in ISO 11898-1 (Part 1: Data link layer and physical signalling) [7] in 1993.

CAN bus is known for providing robustness, flexibility, speed at relatively low cost. Even very cleverly designed systems can in time turn into insecure systems, simply due to the changes in their environments and invalidation of design-time assumptions. As vehicles become connected and autonomous, total lack of security mechanisms in CAN bus becomes a cause for concern. No claims can be made on the confidentiality, integrity, authenticity, and freshness of messages in the presence of a malicious attacker. CAN bus features a cyclic redundancy check (CRC) against non-malicious bit errors.

CAN 2.0 defines four types of frames: DATA, REMOTE, ERROR, and OVERLOAD. Data (e.g. steering angle, wheel speeds) are carried in DATA frames. An ECU can request data from another ECU by sending a REMOTE frame. Fig. 1 shows the structure of DATA frames in CAN 2.0A.

Arbitration field holds the CAN identifier and Remote Transmission Request (RTR) bit. Arbitration field is the part of a frame which is used for deciding priority when two or more nodes attempt to transmit at the same time. By convention, lower value has higher priority. A security related consequence of this mechanism is that, persistent injection of messages with low values in arbitration field can lead to denial of service. Due to the lack of security features, an attacker who gains access to the network is free to replay previous messages or inject crafted messages. Combined with the priority mechanism, carefully timed injections can effectively turn into deletion or overwriting of messages. A CAN 2.0A identifier is a 11-bit value, whereas CAN 2.0B allows 29-bit identifiers. DATA and REMOTE frames are distinguished from each other via the RTR bit. A REMOTE frame is a request for data associated with the CAN identifier which is the CAN identifier of the REMOTE frame. In other words, a request (REMOTE frame) and corresponding reply (DATA frame) have the same CAN identifier. In REMOTE frames, data field is empty.

We make a small digression here to define the *message identifier* concept, which we use throughout this paper. In the presence of REMOTE frames, we define message identifier as the concatenation of CAN identifier and RTR bit:

$$msg\_identifier_{remote} = CAN\_identifier \,||\, RTR. \quad (1)$$

If there are no REMOTE frames in the network, we define message identifier to be the same as CAN identifier. A message identifier uniquely defines a message type (e.g. DATA frame carrying wheel speed data) in the network.

CAN bus protocol defines the structure of a DATA frame, but not the representation of data within the data field, which is decided by vehicle manufacturers. The information necessary for interpreting the data field (e.g. starting bit, length, byte order, offset, resolution) may be kept private by the vehicle manufacturers.

In vehicular networks there usually is a predefined relationship between an ECU, and the message types it will send or process. CAN messages are broadcast over the network, and receivers themselves decide whether to process or discard a message, based on the CAN identifier.

J1939 [8] is a higher-layer protocol built on physical and data link layers of CAN 2.0, and as such it inherits the vulnerabilities of CAN 2.0 [9]. In J1939, the 29-bit identifier of CAN 2.0B is given internal structure to support additional functionality. There are no remote frames in J1939 (i.e. RTR bit is always zero).

B. INTRUSION DETECTION SYSTEM

An intrusion detection system (IDS) monitors a system or network for malicious activity. Detection methodology, technology type, and time of detection are important describing

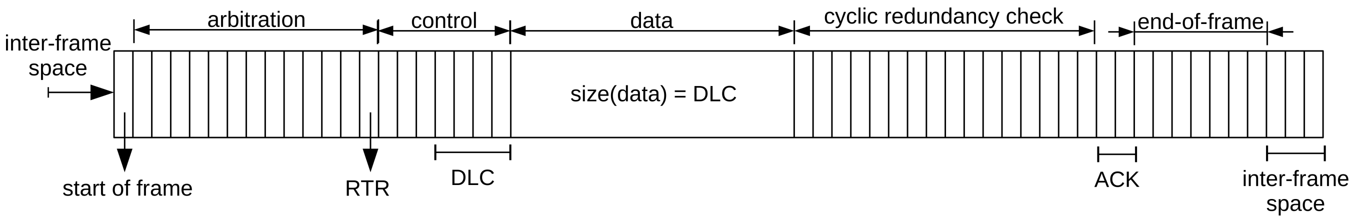

FIGURE 1. Structure of a DATA frame (CAN 2.0A). Each small segment represents a single bit in the frame. Number of bytes in the data field is specified via the four DLC bits, and can take any value in the range 0–8 (inclusive).

features for an IDS [10]. We will briefly touch each of these features, and bring them into the context of an IDS for vehicular CAN bus.

Based on their detection methodology, IDSs can be classified into three major categories: Signature-based Detection, Anomaly-based Detection, and Stateful Protocol Analysis [10]. Anomaly-based Detection is the most relevant category for vehicular CAN bus intrusion detection, and almost all the existing work we encountered so far falls into this category. Signature-based Detection is in principle applicable to CAN bus intrusion detection; however, it would require a comprehensive list of attack message sequences to be compiled, and then converted into attack signatures. Stateful Protocol Analysis is rarely used for intrusion detection in vehicular CAN bus. Most messages in a vehicular CAN bus are periodically broadcast, and this does not leave too much room for performing detections by tracing protocol state, for example, by trying to match requests (remote frames) with replies (corresponding data frames).

IDSs can also be categorized based on technology type. Host-based and Network-based are the two most well-known categories. In case of CAN bus, the distinction between the two categories is not as meaningful as it is for the general case. A CAN bus IDS running on an individual ECU would fall into the host-based category. But because CAN bus messages are broadcast, it would also be able to generate alerts for the whole network. However, a more natural place for a network-based IDS is outside of any of the ECUs on the CAN bus.

Based on time of detection, IDSs can be categorized into two: on-line IDS and off-line IDS. An on-line IDS detects intrusions in real time, allowing the possibility of immediate actions such as dropping a frame before it reaches its intended destination. Such actions are not possible with an off-line IDS, because detections do not necessarily occur before the admittance of the frame causing the detection into the protected host or network, often due to the fact that processing is carried out on replicated data. Time of detection is strongly related to the placement of the IDS within the network topology. In the context of vehicular CAN bus, there are two factors at play which favor off-line IDS behaviour over on-line. First factor is latency. An off-line IDS can be introduced into a CAN bus without introducing any significant additional latency. Second factor is the safety and legal consequences of using an on-line IDS which discards suspicious frames. Totally eliminating false-positives is very difficult if not impossible. Dropped frames may affect a vehicle's operation, possibly compromising its safety. This possibility could prevent such an IDS from being certified for use in traffic.

C. VEHICULAR CAN BUS SECURITY

Attacks can be carried out on vehicular CAN bus via compromised gateway ECUs, or via the addition of a new device to the network (e.g installing an attacker controlled device during maintenance). Nie et al. [11], Miller and Valasek [12], and Woo et al. [13] demonstrate remote attacks where the attackers gain access to the vehicular CAN bus.

There are two main approaches to securing vehicular CAN bus communication. These are explained in the following subsections.

1) Security Features

In this approach, one seeks to increase the security of the network by adding security features into the communication protocol. These security features may be intended for CAN bus itself, or they may be part of a higher-level protocol built on top of it. One important challenge associated with this approach is the computational overhead due to the use of cryptographic methods, which leads to higher latencies and/or increased hardware costs. Also, as is generally the case for security solutions which involve changes at the protocol level, transition costs and backward compatibility are limiting factors.

2) Security Tools

In this approach, one seeks to increase the security of the network by introducing a security tool, such as an intrusion detection system, into the network. Most, if not all, existing vehicular CAN bus IDSs do anomaly-based detection, but they seek anomalies in a variety of different places including:

- Message payload (interpreted)
- Message payload (not interpreted)
- Frequency of messages
- Ordering of messages

If the message payload can be interpreted by the IDS, abrupt changes in values and inconsistencies between data contained in temporally close messages of different types can be used as the basis for anomaly detection. If the representation of data within the data field is not published by the manufacturer and is unknown, interpretation of data is not possible; however, it is still possible to find anomalies in message payload (i.e. data field of the CAN frame) by inspecting certain properties such as entropy, or Hamming distances between consecutive payloads. When either periodicity or ordering is present in CAN bus traffic as emergent properties (e.g. due to majority of CAN bus message types being periodic), then injection (resp. deletion) of messages into (resp. from) normal traffic leads to detectable anomalies in frequencies and ordering of messages. Anomaly detection based on (1) frequencies of messages, and (2) ordering of messages, are related approaches, but the latter works with reduced and qualitative information, rather than full quantitative information about time distances between messages, as is the case with the former. Since request messages and reply messages are always ordered by causality, it makes sense to treat them separately, especially when anomalies are sought in ordering of messages. This is the reason behind the inclusion of RTR bit in the definition of *message identifier* in Equation 1.

Whatever the nature of the anomaly looked for by the CAN bus IDS is, there is often some machine learning method involved in the process. The purpose of involving machine learning is to give the IDS the capability of generalization, so that it can make correct decisions when fed with previously unseen data. Need for generalization is often inevitable in other domains; however, in this work, we claim that CAN bus traffic data has a domain-specific property (Section VI-A) which allows whitelisting, an approach based on pure memorization, to be practical. While it is the cryptographic algorithms which create the performance bottleneck for the previously mentioned "Security Features" approach, for anomaly-based detection, usually the machine learning method is the bottleneck. Decision time of an online IDS must be fast enough to keep up with the 250-500 kbit/s bit rate of CAN bus. And while achieving that decision time, hardware requirements of the IDS in terms of processing speed and memory must remain reasonable, or the practicality of the solution will be questioned.

A shortcoming of relying on an IDS for securing vehicular CAN bus is the limitations regarding available actions in case of a detection. Automated preventive actions may not be possible due to the reasons mentioned at the end of Section II-B. A human specialist will not be available to respond to the alert raised by the IDS, unless the vehicle is served by an Information Security Operations Center (ISOC). Even when specialist response is available, human intervention may not be fast enough to prevent the attack from succeeding. In case of a detected intrusion, one reasonable response is to alert the driver and recommend a safe-stop, or perform an automated safe-stop. However, the vehicle has to have fail-safe or fail-operational capabilities to allow the safe-stop despite the ongoing attack.

III. RELATED WORK

Kleberger et al. [14] and Bozdal et al. [15] both give an overview of in-vehicle network security. The latter focuses exclusively on CAN bus.

Wang and Sawhney [16] propose a security framework which provides message authentication for CAN bus. Halabi and Artail [17] use symmetric-key encryption to build security mechanisms on top of CAN bus. These works serve as examples for the "Security Features" approach (Section II-C1).

IDS proposed by Lee et al. [18] can fit in both Anomaly-based Detection and Stateful Protocol Analysis categories in terms of its detection methodology, as it detects anomalies in timings associated with replies to remote frames.

Marchetti and Stabili [19] propose an IDS which bases its decisions on the Hamming distances between consecutive payloads with same identifiers. Müter and Asaj [20] detect anomalies in entropy of identifier and data fields. Both works detect anomalies in message payloads without interpreting them.

As stated in Section II-C2, several works make use of machine learning methods for anomaly-based intrusion detection. Song et al. [21] use a deep convolutional neural network for their IDS, which takes as input individual bits of CAN identifiers. They also present a comparison of their results with the results obtained using other machine learning algorithms. IDS proposed by Casillo et al. [22] uses Bayesian Networks. IDS proposed by Tian et al. [23] uses Gradient Boosting Decision Trees. Taylor et al. [24] use a one-class support vector machine for their IDS. In this same work [24], they consider a set of features and their combinations, concluding that mean time between packets is the most effective among the considered features.

Gmiden et al. [25] and Moore et al. [26] both propose IDSs which detect anomalies in inter-message time intervals.

Marchetti and Stabili [27] propose an IDS which detects anomalies in the ordering of message identifiers. In this work, we follow the same approach.

IV. DESCRIPTION OF THE STUDIED IDS

The intrusion detection system we study in this paper is a generalization of the intrusion detection system proposed in [27]. Marchetti and Stabili [27] consider sequences of length 2, whereas this work treats sequence length k as a hyperparameter. We also propose and study a *stateful* version of this IDS.

We define the term k-*sequence* as a length k sequence of message identifiers (Section II-A). We describe the IDS by describing the two different phases it operates in: training phase and testing phase. In both phases IDS takes in as input and processes CAN bus messages one at a time. Every k-sequence encountered during the training phase is memorized and stored in an internal data structure we will refer to as the $whitelist$. Contents of $whitelist$ at the end of the training phase correspond to the learned model. Learning is limited to memorization and the IDS does not have the capability for generalization. Algorithm 1 describes the behaviour of the IDS in the training phase. In testing phase, the IDS generates a Boolean output $decision$ for each CAN bus message it processes. The message being processed forms a new k-sequence together with the messages which were processed before it. If this k-sequence is not in $whitelist$, then an intrusion is detected and $decision$ is set to $False$. Algorithm 2 describes the behaviour of the IDS in the testing phase.

Algorithm 1 Processing of a message in training phase.

```
 1: IDS Internal State
 2:   $whitelist$                        ▷ A set of $k$-sequences
 3:   $history$       ▷ A FIFO queue with maximum length $k$
 4: procedure PROCESS_TR($msg$)
 5:   $history.enqueue(msg.ID)$ ▷ $ID$: message identifier
 6:   if $history.length() < k$ then
 7:     return
 8:   $k\_seq \leftarrow history.as\_kseq()$         ▷ A $k$-sequence
 9:   $whitelist.add(k\_seq)$
10:   return
```

Algorithm 2 Processing of a message in testing phase.

```
 1: IDS Internal State
 2:   $whitelist$                        ▷ A set of $k$-sequences
 3:   $history$       ▷ A FIFO queue with maximum length $k$
 4: procedure PROCESS_TE($msg$)
 5:   $decision \leftarrow True$      ▷ $True$ indicates no detection
 6:   $history.enqueue(msg.ID)$ ▷ $ID$: message identifier
 7:   if $history.length() < k$ then
 8:     return $decision$
 9:   $k\_seq \leftarrow history.as\_kseq()$         ▷ A $k$-sequence
10:   $decision \leftarrow whitelist.contains(k\_seq)$
11:   return $decision$
```

A $decision_i$ is output as soon as the processing of an input message M_i is complete, and before the processing of next message begins. However, M_i will continue to appear in k-sequences formed during the processing of subsequent messages M_j, where $i < j < i + k$ and k is the sequence length hyperparameter. In the rest of the paper, we will refer to this process as *tainting*. Because M_i taints its neighbouring k-sequences, the actual decision on anomaly detection for message M_i is given by

$$decision = decision_i \wedge decision_{i+1} \wedge ... \wedge decision_{i+k-1}. \tag{2}$$

Based on this observation, higher k values should lead to higher sensitivity (i.e. higher true positive rate) values for the IDS. Our results from Section VI-C support this hypothesis.

Stateful version of the IDS introduces an additional parameter t_{alert}, and maintains two additional state variables $status\_alert$ and a timer $timer\_alert$. A detection is reported only if the IDS is already in alerted state; otherwise, IDS enters alerted state but does not report a detection. Training of the stateful IDS is the same as that of the original IDS. Algorithm 3 describes the behaviour of the stateful version in the testing phase.

Algorithm 3 Processing of a message in testing phase.

```
 1: IDS Internal State
 2:   $whitelist$                        ▷ A set of $k$-sequences
 3:   $history$       ▷ A FIFO queue with maximum length $k$
 4:   $status\_alert$                               ▷ A Boolean
 5:   $timer\_alert$                               ▷ An integer
 6: procedure PROCESS_TE($msg$)
 7:   $decision \leftarrow True$      ▷ $True$ indicates no detection
 8:   $history.enqueue(msg.ID)$ ▷ $ID$: message identifier
 9:   if $history.length() < k$ then
10:     $decision \leftarrow True$
11:   else
12:     $k\_seq \leftarrow history.as\_kseq()$       ▷ A $k$-sequence
13:     if $whitelist.contains(k\_seq)$ then
14:       $decision \leftarrow True$
15:     else
16:       if $status\_alert$ then
17:         if $timer\_alert > k + 1$ then
18:           $decision \leftarrow False$
19:         else▷ $history$ is tainted by initial cause of alert
20:           $decision \leftarrow True$
21:       else
22:         $decision \leftarrow True$
23:         $status\_alert \leftarrow True$
24:   if $status\_alert$ then
25:     $timer\_alert \leftarrow timer\_alert + 1$
26:   if $(timer\_alert > t_{alert})$ or $\neg$ $decision$ then
27:     $status\_alert \leftarrow False$
28:     $timer\_alert \leftarrow 0$
29:   return $decision$
```

1) Attacker Capabilities and IDS Limitations

We assume that the attacker has total control over one device which is on the CAN bus. Using the device under its control, the attacker is able to read from the CAN bus, and is also able to craft and send messages into the CAN bus. Furthermore, we assume that the attacker has the capability to stop any device on the CAN bus from sending a message of the attacker's choosing. With the addition of this capability, the attacker is able to not only insert but also delete a single message, without causing any additional changes to message ordering. We define a *basic attack* as the insertion or deletion of a single message, and a *complex attack* as any combination of basic attacks immediately following one another within the observed CAN bus traffic.

The most significant limitation of the studied IDS is that it is by design oblivious to attacks which do not affect the ordering of messages. Restating this in terms of attacker capabilities, we say that the attacker is allowed to perform any complex attack, as long as the attack affects message ordering. An example of a complex attack which preserves message ordering is a special case of an overwrite: a deletion of message type ID_i, followed by an insertion of *same* message type ID_i. Such an overwrite attack can be used by the attacker to effectively alter the payload of a message. If ID_i is the message identifier of a periodic message, this attack can be carried out as follows. By observing the time intervals between subsequent messages of type ID_i, the attacker learns the period associated with ID_i, and calculates the timing of next message of type ID_i as t_i. The device controlled by the attacker starts sending a message of type ID_i (setting the data field to any value of attacker's choice) just before $t = t_i$, effectively overwriting the legitimate message, while leaving the ordering of messages intact. This particular attack could be excluded by restricting the attacker capability as follows: the attacker cannot carry out attacks with precise timing (but can enforce the ordering of all basic attacks within a complex attack). However, we accept the stronger restriction instead: the attacker is not allowed to perform attacks which preserve ordering of messages.

The benefit we seek with the *stateful* version of the IDS is the reduction of false positive rates, and the sacrifice we make is the relaxation of the threat model via the introduction of the following additional assumption. We assume that for an attack to pose a threat to the targeted system, at least two *basic attacks* have to be performed within a certain time frame determined by t_{alert}.

V. MATERIALS AND METHODS

A. DATASETS

Training and testing the IDS is carried out using UTU dataset [28] which was prepared specifically for this work. UTU dataset contains over 180 hours of attack-free CAN bus traffic collected from a Renault Euro VI heavy-duty truck (Renault T520 6X2) over several driving sessions (with different drivers) in varying traffic conditions (e.g. urban, rural). Data was collected from the Fleet Management Systems (FMS) Interface [29] of the truck via a PEAK-System PCAN-USB adapter. Frames in UTU dataset conform to the J1939 standard. 29-bit identifiers of frames are directly used as message identifiers.

We make use of two additional datasets in Section VI-A. These are CAN Dataset for intrusion detection (OTIDS) [18], [30] and Automotive CAN Bus Intrusion Dataset v2 [31], which we will refer to as ACID in the rest of this paper. OTIDS is collected from a Kia Soul SUV, and ACID includes data collected from both an Opel Astra and also a Renault Clio. We use only attack-free data from both datasets. Since remote frames are present in the data, message identifiers are calculated as in Equation 1. In case of OTIDS, the 45 distinct CAN identifiers lead to 54 distinct message identifiers.

All data used in this work is collected from certified vehicles while they are operated on a road. Details of collection can be found in metadata associated with each dataset. All datasets used in this paper are publicly available. Table 1 presents summary information about the datasets.

TABLE 1. Summary information about datasets used in this paper.

Dataset	Protocol	Number of messages	Number of distinct message identifiers
UTU	J1939 / CAN 2.0B	530,810,616	70
OTIDS	CAN 2.0A	2,369,398	54
ACID (Opel)	CAN 2.0A	2,690,069	85
ACID (Renault)	CAN 2.0A	386,567	55

B. CODE

Implementation of the IDS, and other code and data used for producing the results in this paper (e.g. serialized trained IDS instances, partitioning of dataset) are available upon request. Implementation is in Python 3.

C. METHODS

1) Preparation of data for training and testing

For training and testing we used UTU dataset which contains 11191 files each containing 50001 CAN bus messages.[1] For a message which is at the end of one file, the message following it is guaranteed to be the first message in the next file. However, for the sake of convenience we treated files as independent units and ignored all message sequences which are split over multiple files. While partitioning the dataset into training and test data, files were treated as the smallest assignable unit, i.e. all messages in a file ended up in the same partition. Using non-exhaustive 3-fold cross-validation, we obtained three $(trainingdata, testdata)$ pairs, one for each iteration. Attacks used in the testing phase are generated procedurally. The same attacks (i.e. same payloads at same attack locations) are generated for each iteration.

2) Description of attacks used in evaluation

The following attack types were considered in our experiments:

- *Type 1* – Delete single message
- *Type 2* – Insert single message
- *Type 3* – Insert random sequence
- *Type 4* – Insert observed sequence
- *Type 5* – Overwrite with observed sequence

Type 1 attacks delete a single message. A *Type 1* attack taints the least number of $decision$ variables (See Equation 2), and hence yields the worst case scenario for IDS sensitivity. Table 2 compares the number of taints between deletion and insertion for $k = 2$ and $k = 3$. *Type 2* attacks insert a single message such that the message identifier is randomly selected among known (i.e. observed in training phase) message identifiers. *Type 1* and *Type 2* attacks are basic attacks. *Type 3* and *Type 4* attacks both insert a single k-sequence. For a *Type 4* attack the k-sequence is randomly

[1]We observed an anomaly in one of the files. We ignored this file and its two neighbours in our analysis.

TABLE 2. Comparison of tainted sequences for deletion and insertion. k is the sequence length.

	delete C $A\ B\ C\ D\ E \rightarrow A\ B\ D\ E$	insert C $A\ B\ D\ E \rightarrow A\ B\ C\ D\ E$
$k = 2$	$\{BD\}$	$\{BC, CD\}$
$k = 3$	$\{ABD, BDE\}$	$\{ABC, BCD, CDE\}$

TABLE 3. Sparsity values associated with data collected from four different vehicles, for different choices of sequence length k.

Vehicle	Dataset	k = 2	k = 3	k = 4	k = 5
Renault T520 6X2	UTU	0.8531	0.9946	0.9998	0.9999
Kia Soul SUV	OTIDS	0.3700	0.8865	0.9922	0.9996
Opel Astra	ACID	0.4768	0.9528	0.9988	0.9999
Renault Clio	ACID	0.4863	0.9654	0.9922	0.9999

chosen among all k-sequences observed in training phase, whereas for a *Type 3* attack the k-sequence is built by randomly choosing k elements (with replacement) among known message identifiers. For a *Type 5* attack the payload is generated the same way as for a *Type 4* attack, but prior to its insertion, k messages are deleted starting from the attack location. It is possible to associate these attack types with the more familiar categories of realistic attacks. A denial-of-service attack may result from repeated insertions (*Type 2*) with a low value in CAN identifier. *Type 3* and *Type 4* attacks correspond to fuzzing and replay, respectively.

Each iteration involves 74600 instances of each attack type. In addition to these, we used 7460 pairs of attacks of *Type 2* for evaluation of the stateful version of the IDS. These pairs were generated such that the second insertion does not occur immediately after the first, but the separation between them is smaller than t_{alert}.

VI. RESULTS

A. SPARSITY OF VEHICULAR CAN BUS TRAFFIC

We define sparsity as the ratio of number of k-sequences which are not in the dataset to the number of all possible k-sequences:

$$sparsity = 1 - \frac{obs_k}{N^k}, \tag{3}$$

where obs_k is the (distinct) number of observed k-sequences, N is the number of distinct message identifiers, and k is the sequence length.

In order to demonstrate that high sparsity property is not specific to the UTU dataset which is used to train and test the studied IDS (and to the vehicle which generated its data), we calculated sparsity values also for data from different datasets and vehicles. Results are presented in Table 3.

High sparsity property of CAN bus traffic is crucial for the viability of the evaluated IDS. Any action that affects the ordering of messages produces a set of tainted k-sequences, each of which has a higher probability of being a whitelisted sequence if sparsity is low. Hence, low sparsity adversely affects the ability of the IDS to detect anomalies, reducing its sensitivity. Low sparsity also leads to larger memory footprint.

Abundance of periodic messages, and the arbitration mechanism based on CAN identifiers might be behind the emergence of this property. Studied IDS is a domain-specific solution for vehicular CAN bus networks.

B. LEARNING CURVES

It is informative to look at how the training process progresses with time, as the high sparsity property does not guarantee that the training process described by Algorithm 1 will stabilize and yield a usable model. For this purpose we plot the learning curves for $k = 2$ and $k = 3$ in Fig. 2.

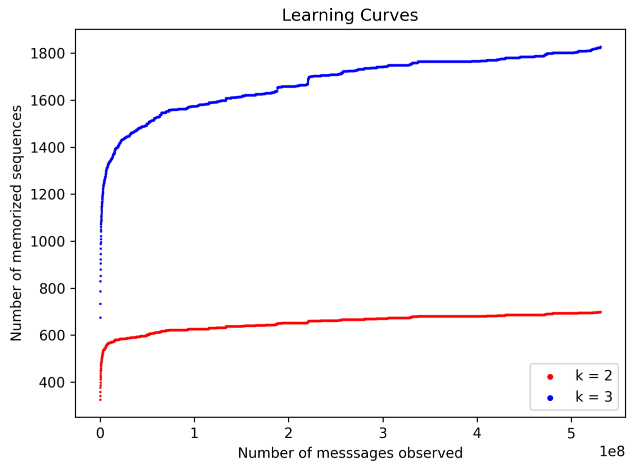

FIGURE 2. Learning curves for $k = 2$ and $k = 3$. At points where the slope of the curve is relatively small, the model has matured.

In Fig. 2, the range for the y-axis is determined by the sparsity of the dataset. The x-axis can be alternatively interpreted as time. Number of memorized k-sequences is polled every time a file is completely processed, and the files are fed to the IDS in the same order that they were created. Every file contains roughly 1 minute of recorded traffic, so the *imagined* time axis extends from 0 to 180 hours. The slope of the curve corresponds to the rate at which previously unknown k-sequences are encountered. The higher rate for $k = 3$ compared to $k = 2$ suggests that higher k values may result in lower specificity (i.e. higher false positive rate) values when both IDS receives the same amount of training. Our results from Section VI-C agree with this prediction. Alternatively, it can be stated that when $k = 3$ the IDS needs to be trained with more data compared to when $k = 2$, for the model to achieve the same level of maturity. Inspection of learning curves revealed that the models have matured by the time one third of the data is used for training. This observation guided our decision on how to partition the dataset into training and test data, and the choice of 3 as the number of iterations for cross-validation.

TABLE 4. True positive rates for different choices of sequence length k and different attack types. Values shown are the average of values obtained from the three iterations.

	$k=2$	$k=3$	$k=4$	$k=5$
Attack Type 1	0.638	1	1	1
Attack Type 2	0.906	1	1	1
Attack Type 3	0.985	1	1	1
Attack Type 4	0.898	1	1	1
Attack Type 5	0.887	1	1	1

TABLE 5. False positive rates for different choices of sequence length k. Values shown are the average of values obtained from the three iterations.

$k=2$	$k=3$	$k=4$	$k=5$
7.184×10^{-7}	2.129×10^{-6}	4.199×10^{-6}	6.836×10^{-6}

C. SENSITIVITY AND SPECIFICITY OF IDS

Table 4 and Table 5 present a summary of our results with respect to sensitivity and specificity, respectively. Table 4 indicates that sequence length $k = 3$ leads to significantly better sensitivity compared to $k = 2$, and increasing k beyond 3 does not yield additional benefits as maximum sensitivity of 1 is achieved for $k = 3$ even for the worst case scenario, i.e. for *Type 1* attacks. Table 5 indicates that false positive rates increase as k increases, but remain small for the tested k values.

Fig. 3 depicts true positive rates and false positive rates for basic attacks.

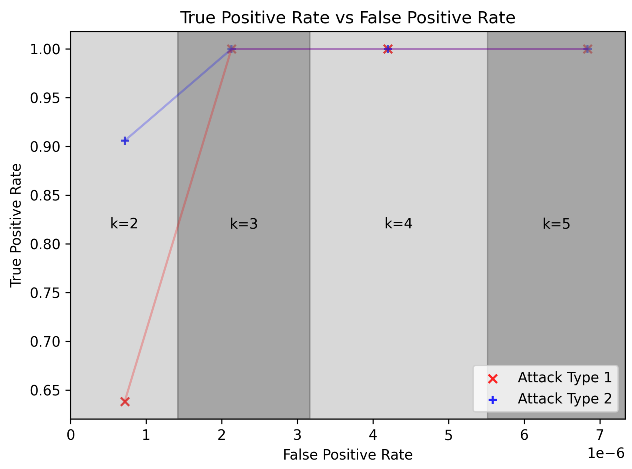

FIGURE 3. True positive rates and false positive rates for *Type 1* and *Type 2* attacks.

Table 6 and Table 7 present the sensitivity and specificity values for both the original and the stateful version of the IDS. Stateful version achieved higher specificity for all tested k values, and there was no difference in sensitivity between the two versions. Parameter t_{alert} was set to 25000 messages which correspond to roughly 30 seconds at normal traffic rates.

TABLE 6. A comparison of true positive rates between original and stateful versions of the IDS. k represents the sequence length hyperparameter. Values shown are the average of values obtained from the three iterations.

	$k=2$	$k=3$	$k=4$	$k=5$
original	0.911	1	1	1
stateful	0.911	1	1	1

TABLE 7. A comparison of false positive rates between original and stateful versions of the IDS. k represents the sequence length hyperparameter. Values shown are the average of values obtained from the three iterations.

	$k=2$	$k=3$	$k=4$	$k=5$
original	7.184×10^{-7}	2.129×10^{-6}	4.199×10^{-6}	6.836×10^{-6}
stateful	2.278×10^{-7}	6.630×10^{-7}	9.042×10^{-7}	1.129×10^{-6}

D. MEMORY FOOTPRINT

Memory footprint of an IDS is an important factor to consider when deciding its feasibility and practical value due to increased costs associated with high memory requirements. Choice of sequence length k affects the memory requirements of the IDS both directly, and indirectly through its influence on the number of k-sequences that are memorized. Fig. 4 shows the memory footprint of our prototype implementation for different k values. The difference in required memory size between the cases $k = 2$ and $k = 3$ is approximately 4 kilobytes. The only difference between the stateful version and the original IDS in terms of required memory is that the former holds two additional state variables whose contribution to memory footprint is negligible.

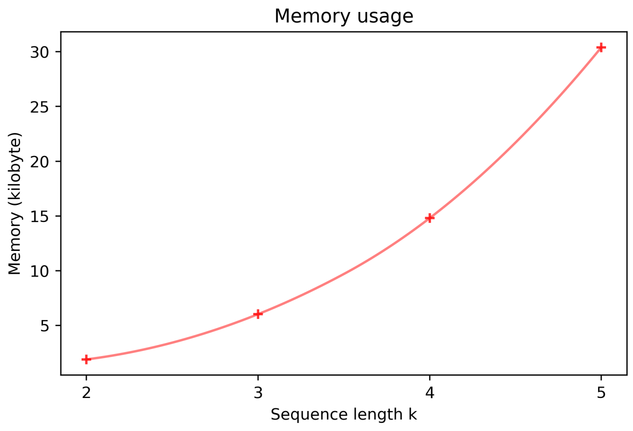

FIGURE 4. Memory requirement of the IDS for different k values.

E. DECISION TIME

We define decision time as the time required to process a message, where the required processing is described in Algorithm 2 for the original IDS, and in Algorithm 3 for the stateful version.

If the IDS cannot complete the processing of a message before the arrival of the next message, if becomes necessary

TABLE 8. Mean and standard deviation (SD) for the decision time measurements. Unit of measurement is microsecond. k is the sequence length.

	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Mean	4.3	4.6	6.1	6.8
SD	1.3	1.2	1.5	1.2

to buffer relevant parts of incoming messages. Buffering increases the complexity and cost of the IDS. Latency is another reason to seek low decision times, especially for an on-line IDS.

For the CAN frames in UTU dataset, up to 64 bits of data follow the identifier. Most frames in UTU dataset carry 64 bits of data, and average size for the data field is 63 bits. The CAN bus which generated UTU dataset operates at 250 kbit/s. Assuming the worst case scenario where a node starts sending its message as soon as the sending of the previous message is completed, the time available to the IDS for completing the processing of the first message before the arrival of the second one is equal to the number of bits in the data field divided by bit rate. With a bitrate of 250 kbit/s, and average data size of 63 bits, average available time $\bar{t}_{available}$ is 252 microseconds.

We measured decision times for the IDS trained in iteration 1, on a single CPU core running roughly at 1500 MHz. Table 8 shows the measurement results. Mean decision times are significantly lower than $\bar{t}_{available}$. We observed no meaningful difference between the original and stateful version in terms of decision times.

VII. DISCUSSION

In this section, we summarize and discuss our results. Our results from Section VI showed promising sensitivity and specificity values, as well as very fast decision times and acceptable memory footprint. Our main results are as follows:

- Sequence length is a useful hyperparameter for the considered IDS.
- Increasing sequence length from $k = 2$ to $k = 3$ improves IDS sensitivity, while only slightly increasing false positive rate and memory footprint, and having no impact on decision time.
- False positive rate for the stateful version for $k = 3$ was lower than the false positive rate for the original IDS for $k = 2$.
- Our results support the main findings of Marchetti and Stabili [27], and show that whitelisting identifier sequences is a viable approach for anomaly detection in vehicular CAN bus.

Marchetti and Stabili [27] report that their experiment did not generate any false positives. While our experiment did yield small false positive rates, they were not zero even for the $k = 2$ case, and it can be seen in Fig. 2 that new sequences kept appearing even after several billions of observed messages. This apparent conflict might be due to some differences between the datasets; however, because the dataset used in their study was not available to us, we were not able to investigate the cause.

VIII. CONCLUSION

We studied an intrusion detection system (IDS) which detects anomalies in vehicular CAN bus traffic by analyzing message identifier sequences. We implemented the IDS, and evaluated its performance using CAN bus data collected from a heavy-duty truck.

The IDS we studied in this work is a lightweight solution in terms of its hardware requirements, and as such it may be suitable for being used together with other anomaly detection approaches, complementing them. It may also serve as a baseline for evaluating other approaches which are more demanding in terms of time and memory.

ACKNOWLEDGMENT

We thank our project partners at TTS Työtehoseura for their help in collection of CAN bus data.

REFERENCES

[1] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, “A review on safety failures, security attacks, and available countermeasures for autonomous vehicles,” *Ad Hoc Networks*, vol. 90, p. 101823, 2019, recent advances on security and privacy in Intelligent Transportation Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870518309260

[2] V. L. Thing and J. Wu, “Autonomous vehicle security: A taxonomy of attacks and defences,” in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 164–170.

[3] E. Yağdereli, C. Gemci, and A. Z. Aktaş, “A study on cyber-security of autonomous and unmanned vehicles,” *The Journal of Defense Modeling and Simulation*, vol. 12, no. 4, pp. 369–381, 2015. [Online]. Available: https://doi.org/10.1177/1548512915575803

[4] S. Parkinson, P. Ward, K. Wilson, and J. Miller, “Cyber threats facing autonomous and connected vehicles: Future challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[5] D. Dominic, S. Chhawri, R. M. Eustice, D. Ma, and A. Weimerskirch, “Risk assessment for cooperative automated driving,” in *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, ser. CPS-SPC ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 47–58. [Online]. Available: https://doi.org/10.1145/2994487.2994499

[6] U. Kiencke, S. Dais, and M. Litschel, “Automotive serial controller area network,” *SAE Transactions*, vol. 95, pp. 823–828, 1986. [Online]. Available: http://www.jstor.org/stable/44722673

[7] “Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling,” International Organization for Standardization, Geneva, CH, Standard, Dec. 2015.

[8] “Recommended practice for a serial control and communications vehicle network,” SAE International, Standard.

[9] P.-S. Murvay and B. Groza, “Security shortcomings and countermeasures for the sae j1939 commercial vehicle bus protocol,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4325–4339, 2018.

[10] H.-J. Liao, C.-H. Richard 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804512001944

[11] S. Nie, L. Liu, and Y. Du, “Free-fall: Hacking tesla from wireless to can bus,” *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[12] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” *Black Hat USA*, vol. 2015, no. S 91, 2015.

[13] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on intelligent transportation systems*, vol. 16, no. 2, pp. 993–1006, 2014.
[14] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 528–533.
[15] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of can bus security challenges," *Sensors*, vol. 20, no. 8, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/8/2364
[16] Q. Wang and S. Sawhney, "Vecure: A practical security framework to protect the can bus of vehicles," in *2014 International Conference on the Internet of Things (IOT)*, 2014, pp. 13–18.
[17] J. Halabi and H. Artail, "A lightweight synchronous cryptographic hash chain solution to securing the vehicle can bus," in *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, 2018, pp. 1–6.
[18] 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)*, vol. 00, Aug 2017, pp. 57–5709. [Online]. Available: doi.ieeecomputersociety.org/10.1109/PST.2017.00017
[19] D. Stabili, M. Marchetti, and M. Colajanni, "Detecting attacks to internal vehicle networks through hamming distance," in *2017 AEIT International Annual Conference*, 2017, pp. 1–6.
[20] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 1110–1115.
[21] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209619302451
[22] M. Casillo, S. Coppola, M. De Santo, F. Pascale, and E. Santonicola, "Embedded intrusion detection system for detecting attacks over can-bus," in *2019 4th International Conference on System Reliability and Safety (ICSRS)*, 2019, pp. 136–141.
[23] D. Tian, Y. Li, Y. Wang, X. Duan, C. Wang, W. Wang, R. Hui, and P. Guo, "An intrusion detection system based on machine learning for can-bus," in *Industrial Networks and Intelligent Systems*, Y. Chen and T. Q. Duong, Eds. Cham: Springer International Publishing, 2018, pp. 285–294.
[24] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, 2015, pp. 45–49.
[25] M. Gmiden, M. H. Gmiden, and H. Trabelsi, "An intrusion detection method for securing in-vehicle can bus," in *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2016, pp. 176–180.
[26] 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*, ser. CISRC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3064814.3064816
[27] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1577–1583.
[28] University of Turku, "Can bus dataset collected from a heavy-duty truck," https://doi.org/10.23729/3160254e-85e9-4268-a636-5b3e54091706, 2021, accessed: 2021-09-06.
[29] "Fms-standard description version 04," HDEI / BCEI Task Force, Standard, Oct. 2017.
[30] "Can dataset for intrusion detection (otids)," https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset, accessed: 2021-09-06.
[31] G. Dupont, A. Lekidis, J. J. den Hartog, and S. S. Etalle, "Automotive controller area network (can) bus intrusion dataset v2," Nov 2019, accessed: 2021-09-06. [Online]. Available: https://data.4tu.nl/articles/dataset/Automotive_Controller_Area_Network_CAN_Bus_Intrusion_Dataset/12696950/2

TAHSİN C. M. DÖNMEZ received the M.S. degree in information security and cryptography from University of Turku, Turku, Finland, in 2017.

From 2017 to 2021, he was a Project Researcher with the University of Turku, Turku, Finland.

• • •