https://doi.org/10.31436/ijpcc.v11i1.530

Digital Twin-Based Evaluation of Vehicular Controller Area Network Intrusion Detection Systems

Shaila Sharmin, Hafizah Mansor, Andi Fitriah Abdul Kadir, Amelia Ritahani Ismail

Department of Computer Science, International Islamic University Malaysia, Kuala Lumpur, Malaysia

*Corresponding author shailasharmin@protonmail.com

(Received: 26th December, 2024; Accepted: 15th January, 2025; Published on-line: 30th January, 2025)

Abstract— The functions and operations of a modern automobile are becoming increasingly computerised, with this transformation made possible by Electronic Control Units (ECUs) that communicate and coordinate with each other on the in-vehicle network. Controller Area Network (CAN) is one of the most popular protocols for the in-vehicle network, supporting low latency and reliable communications. However, the CAN protocol does not have provisions for security, such as encryption, authentication, and authorisation, which makes it vulnerable to cyberattacks, particularly in today's automotive landscape characterised by extensive connectivity with external devices, vehicles, and infrastructure. While intrusion detection systems (IDS) for CAN have emerged as a key security measure, assessing their performance against realistic attacks remains a challenge since testing with real vehicles poses significant costs and safety risks and testbeds suffer from a lack of fidelity in terms of the CAN frame transmission timings and generated payloads. This work proposes a digital twin (DT)-based framework for CAN IDS evaluation that replicates the functionality of real-world ECUs and CAN bus of a vehicle with real-time flow of data from the physical bus to its virtual representation. The main contribution of this work is a CAN DT that can not only enable the generation of realistic attack traffic for simple and sophisticated attack scenarios but also the generation of diverse combinations of attack and real driving scenarios. This DT can facilitate the evaluation of both the detection capability and performance of CAN IDS. This work presents the methodology for generating the proposed DT and discusses current findings as well as future work.

Keywords— In-vehicle network, Controller Area Network, Intrusion detection, Digital twin

I. Introduction

The modern vehicle is capable of more than just moving passengers and cargo from one point to another – it has a myriad of features and functionalities that facilitate driving, enable safety and comfort, and support navigation, communication, and entertainment. These systems, which are increasingly computerised in modern vehicles, are enabled by as many as 150 microcontrollers called Electronic Control Units (ECUs) [1]. ECUs coordinate with each other by communicating the current state of the vehicle on internal vehicular networks or in-vehicle networks. Numerous protocols are implemented for in-vehicle networks with the most common being Controller Area Network (CAN).

The CAN protocol enables low-latency, reliable communications but does not provide mechanisms for encryption, authentication, or authorisation. This makes vehicular CAN bus vulnerable to a variety of cyberattacks that can allow car theft or cause dangerous accidents. This is especially true in today's automotive landscape where vehicles are equipped with a wide range of communication interfaces to enable vehicle-to-everything (V2X) connectivity. These interfaces, which include Wi-Fi, Bluetooth, and radio, enable external entities access to the in-vehicle network and become potential attack vectors. While Koscher et al. [2] established the vulnerability of vehicular CAN bus to injected CAN frames, Checkoway et al. [3] demonstrated the possibility of remotely attacking a vehicle's CAN bus. More recent work on CAN bus hacking underscores the continued need to secure the vehicular CAN bus [4], [5].

The development of intrusion detection systems (IDS) has emerged as a key effort towards securing the vehicular CAN bus. CAN IDS vary in the technique used as well as the feature of CAN bus traffic used for detection. Unlike conventional computer networks, a vehicular network represents a safety-critical system where a CAN IDS would need to detect attacks accurately and as quickly as possible to minimise harm to occupants and surroundings. As such, all CAN IDS need to be evaluated against realistic CAN bus traffic and attack scenarios to ensure their performance in a real in-vehicle network.

However, generating realistic attack scenarios remains a challenge since using real vehicles for security testing is a costly option and poses a safety risk to personnel and the environment. While testbeds and simulations do not have these issues and are an attractive alternative for security testing [6], [7], current proposals have limitations in fidelity, specifically in the behaviour and interactions of ECUs. This impacts the realism of the generated CAN frame payloads, particularly in attack scenarios and undermines the evaluation of payload-based CAN IDS carried out on such solutions.

The main contribution of this work is to propose a digital twin (DT)-based framework for generating a virtual representation of a real-world CAN bus, capable of accurately emulating the behaviour and interactions of real ECUs. This DT can, thus, be used to simulate both simple and sophisticated attacks and generate CAN attack traffic that is realistic in terms of frame transmission timings and payloads. Establishing the flow of data from the physical CAN bus to the virtual CAN bus can further enable the generation of countless combinations of attack and driving scenarios, resulting in robust security and performance assessment of CAN IDS.

The rest of the work is organised in the following manner: Section II provides an overview of the CAN protocol as well as the concept of Digital Twins. Section III discusses current proposals for CAN testbeds, simulations, and DT. Section IV outlines the proposed framework while Section V presents current results. Finally, Section VI concludes this work and discusses future work.

II. Background

A. *Controller Area Network (CAN)*

The Controller Area Network (CAN) protocol was introduced in the 1980s with the aim of enabling efficient, reliable communication for in-vehicle networks. It uses a bus architecture to connect Electronic Control Units (ECUs) within a vehicle that control and coordinate the various operations of the vehicle. This bus architecture significantly reduces the weight and complexity of the in-vehicle network compared to older point-to-point connections [8], [9]. CAN finds usage in critical vehicular subsystems such as powertrain and chassis that enable functions like power steering, braking and transmission [10].

CAN is a multi-master, message-based communication protocol, which means that any node on a CAN bus can transmit frames, and all frames are received by all nodes on the bus. A CAN data frame consists mainly of an arbitration identifier (AID), a data length code (DLC), and a data field [11]. The AID identifies a data frame and the information contained in the data field. While an ECU may broadcast multiple AIDs, a particular AID is typically broadcast by only one ECU [12]. Every ECU also subscribes to a list of AIDs and only reads the data frames of received frames that match these AIDs. The DLC specifies the number of bytes in the subsequent data field of the frame. The data field, which can be up to 64 bits, encodes the information being conveyed by the frame and represents the frame's payload.

An ECU encodes values of a particular set of signals in the data field of each AID and transmits the latest signal values to update all other nodes of the bus and coordinate the operations of the vehicle. While most AIDs are broadcast at fixed frequencies, some AIDs may be event-triggered and broadcast occasionally. The signals associated with each AID and the way they are encoded in the data field are specified in the form of rules in a CAN database (DBC). Unlike the format of a CAN frame which is specified by the CAN protocol, the CAN DBC may vary among different vehicle makes and models and is often kept proprietary and confidential.

Since any node on the CAN bus can transmit frames, CAN implements an arbitration mechanism when two nodes attempt to broadcast frames at the same time. In the event of bus contention, a frame with a lower-valued AID has higher priority and is broadcast first, while the higher-valued AID has lower priority and is retransmitted later. CAN has an error-handling mechanism implemented through cyclic redundancy checks and acknowledgement bits in the frame. It also has a method for error confinement to prevent errors from propagating in the bus whereby each node implements an error counter and is removed from the bus when the value of the error counter becomes too high (bus-off) [13], [14].

B. *CAN Attack Model*

CAN was designed for in-vehicle networks at a time when they were isolated systems. The security of the in-vehicle network was less of a concern while low latency and reliability were prioritised for the protocol. As a result, CAN lacks key security features like encryption, authentication, authorisation and integrity [8]. The modern vehicle also has many communication interfaces that allow external access to the in-vehicle network and thus act as attack vectors. These two factors combine to make the vehicular CAN bus vulnerable to a range of cyberattacks.

Cho & Shin [15] as well as Verma et al. [13] propose an attack model for the CAN bus which begins with the distinction between a weakly compromised node and a strongly compromised node. A weakly compromised node is one that an adversary has stopped from transmitting frames, while a strongly compromised node is one that the adversary has complete control over and can use to transmit malicious frames on the bus. This attack model categorises CAN bus attacks in the following manner:

1) *Fabrication attacks*: Fabrication attacks represent the most common type of CAN bus attacks, whereby an attacker uses a strongly compromised node to inject malicious frames. These include the following attacks:

- Denial of Service (DoS): A DoS attack is carried out by injecting frames with AID 0x000 and an arbitrary data field at a high frequency. This attack takes advantage of the CAN arbitration mechanism and prevents the broadcast of other legitimate frames. To evade security mechanisms in newer vehicles that prevent the transmission of frames with invalid AIDs, a DoS can be carried out by injecting the lowest-valued valid AID that appears in normal CAN traffic [8].

- Fuzzing: A fuzzing attack involves the injection of frames with random AIDs and payloads at a high frequency. While the injection of random AIDs generally disrupts the transmission of legitimate frames, the random AIDs can include valid AIDs which can confuse ECUs about the real values of signals.
- Targeted ID or Spoofing: In a targeted ID attack, the adversary injects frames with a specific valid AID and manipulated payloads to cause ECUs that subscribe to the AID to malfunction. The fabricated frames may be injected at a high frequency in what is called a *flooding* delivery, or immediately following the appearance of legitimate frames of the same AID in a *flam* delivery. Both flooding and flam delivery achieve the same effect, but the latter does so with fewer fabricated frames.
- Replay: A replay attack is carried out by capturing a sequence of frames from the CAN bus and injecting them again at a later point in time when the vehicle is in a different state.

2) Suspension: A suspension attack involves weakly compromising a node on the CAN bus so that the node stops broadcasting CAN frames. This results in frames of the associated AIDs being missing from the bus traffic. A node can be prevented from broadcasting frames using any technique, such as by forcing it into the bus-off state [16].

3) Masquerade: A masquerade attack can be thought of as combining a suspension and a spoofing attack: legitimate transmissions of an AID are first suspended and then a malicious node injects spoofed frames of the same AID with manipulated payloads. This scenario is different from a spoofing attack where the fabricated frames appear alongside legitimate frames of the same AID, making it a more subtle, difficult-to-detect attack.

C. *Digital Twin*

The concept of digital twin (DT) was introduced by Grieves in 2003 who described it as "rich representations of products that are virtually indistinguishable from their physical counterparts" [17]. The DT concept model was described as including three components: physical objects, their virtual representations, and the data and information that connect these counterparts. From this early definition, the idea of what constitutes a digital twin has evolved [18], [19] but Guo et al. [20] find no consistent definition of DT in their survey. In the current literature, there are three levels of understanding of what a digital twin is that vary on the kind of interaction between the physical and virtual counterparts [18], [19]. The first may be described as a *digital model* where the virtual representation is built of a specific physical object, but there is no persistent flow of data between the two counterparts. The second may be described as a *digital shadow*, where there is a unidirectional flow of data from the physical twin to the virtual twin, with changes in the physical twin resulting in changes to the virtual twin. Finally, there is the *fully integrated digital twin*, represented in Fig. 1, where there is a bidirectional flow of data between the physical and virtual twin such that the virtual twin adapts to changes in the physical twin and provides feedback to the physical twin.

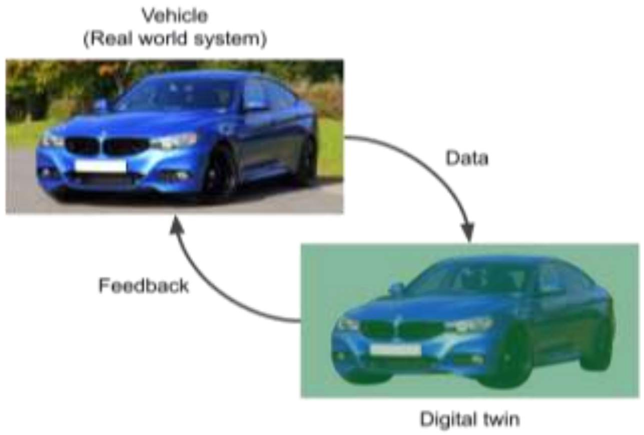

Fig. 1 Interactions in a fully-integrated DT

While some works [18], [19] emphasise the real-time, bidirectional flow of data between the virtual and physical counterparts as a key component of digital twins, such stipulations are considered restrictive in [21]. VanDerHorn and Mahadevan [21] consider two factors that distinguish a DT from a digital model or a simulation: a DT represents a particular instance of a physical object (e.g. a specific vehicle) instead of the entire class of the physical object, and the flow of data from the physical object to the virtual object over time. Other requirements on the digital twin may be considered with respect to the use case for which a particular implementation is aimed.

Digital twin technology is envisioned as an enabler of smart manufacturing and Industry 4.0. In the review of digital twin applications in the industry, [18] finds extensive applications of digital twins for product design, production, and product health management. Apart from these, DT can also be used for security applications such as in [22]. In this 'virtual testing' application of DT, DT is akin to a "more realistic and accurate" simulation and is used to simulate and explore different attack scenarios that would cause damage to real systems [19].

III. Literature Review

Intrusion detection systems (IDS) have emerged as a key mechanism for securing the vehicular CAN bus alongside encryption and authentication schemes. Current CAN IDS show a wide variety in the technique used for attack detection, ranging from relatively simpler statistical methods to advanced methods based on traditional machine learning (ML) as well as deep learning (DL) [23], [24]. CAN IDS also vary in terms of the features utilised for intrusion detection – an IDS may use the timing of CAN frames, sequences of AIDs, the payload of CAN frames, or any combination of these features for analysis and attack detection [10]. The in-vehicle network where a CAN IDS operates is distinct from conventional computer networks due to the limited computing capability of ECUs. The safety-critical nature of the in-vehicle network also necessitates low false-positive and false-negative rates, as well as fast attack detection. Therefore, in addition to meeting detection capability requirements measured using security metrics such as accuracy and F1-score, a CAN IDS should also meet non-functional requirements characterised by performance metrics such as detection latency [23].

Many CAN IDS proposals are evaluated in offline experiments using publicly available CAN intrusion datasets whereby the proposed methods are used to analyse the dataset. Using such datasets allows reproducibility of results and comparison of different methods under similar experimental settings [23]. However, evaluation using datasets are restricted to the attack scenarios contained in these datasets, which do not contain realistic samples of advanced attacks such as suspension and masquerade [13], [23]. Furthermore, evaluations with these datasets do not allow robust assessment of non-functional properties like detection latency and resource consumption in a realistic environment. As such, it is necessary to move towards online methods of evaluations which can allow the assessment of both detection capability and performance in varied attack scenarios and in experimental settings closely resembling real in-vehicle networks.

The best option for online testing is real vehicular CAN buses which are closest to real operating environments. Stachowski et al. [25] present an assessment methodology where CAN IDS products under test are integrated into a real vehicle for real-time performance evaluation. Three undisclosed anomaly-based CAN IDS products were evaluated in a vehicle on which various targeted ID attacks were performed while the vehicle was both stationary and in motion in a private test track. Their methodology encompasses both qualitative and quantitative metrics: while the quantitative metrics measure detection capability, the qualitative metrics include the effort required to integrate the IDS solution in a vehicle, flexibility of the solution, forensic capabilities, etc. However, only quantitative security metrics were reported, and performance metrics were not measured. There are further challenges associated with using real test vehicles. Conducting attacks on real vehicles runs the risk of permanently damaging the internal electronics. There is also a safety risk towards drivers, passengers, bystanders and surroundings [13], [26]. Furthermore, safely conducting security tests with a real vehicle also incurs significant costs, an example of which is a dynamometer used by Verma et al. [13] during CAN traffic collection. On the other hand, testbeds and simulations can mitigate these challenges by minimising the safety risks associated with running attack scenarios as well as minimising financial costs [26].

A. CAN Security Testbeds and Simulations

Numerous CAN testbeds and simulations have been developed for cybersecurity applications, such as for testing encryption schemes, reverse engineering, and penetration testing. Cros et al. [27] present a simulation platform called Cacao, aimed towards the evaluation of encryption and signature solution for CAN communications. Raspberry Pi devices are used to simulate nodes on a CAN network that has been used for monitoring bandwidth usage as a means of detecting brute-force attacks. Mundhenk et al. [28] also propose a discrete event simulator for assessing encryption schemes for CAN. Unlike Cacao, this platform was used to analyse real-time performance aspects like computation time and memory usage for authentication protocols.

Zheng et al. [29] propose a testbed architecture for security analysis of a vehicular CAN network, which can be used to capture CAN bus traffic for analysis and to simulate attacks. It consists of a real-time CAN bus simulation along with an emulated infotainment system that was used to simulate a DoS attack. Fowler et al. [30] also propose a CAN testbed based on a commercial Hardware-in-the-loop (HIL) solution, which they use to perform a penetration test in a case study, where vehicle network vulnerabilities are exploited using a dongle connected to the On-Board Diagnostic (OBD-II) port. Instead of simulating complete vehicle functionality, Granata et al. [31] aim to simplify security testing by emulating the minimum set of components to effectively reproduce security vulnerabilities. Their hybrid CAN bus simulation system, called HybridgeCAN, is proposed as a low-cost testbed alternative to expensive hardware-in-the-loop (HIL) testing systems and real vehicles.

Everett & McCoy [32] provide a software package and hardware framework as part of the Open Car Testbed and Network Experiments (OCTANE) testbed geared towards reverse engineering and testing of automotive networks. The software has a layered architecture, making it flexible and adaptable, while the hardware framework does not require specific hardware components. Portable Automotive Security Testbed with Adaptability (PASTA) [33] is another CAN testbed that focuses on white box ECUs which can be reprogrammed to set up the development environment as well as implement and test security solutions. It either disposes of or uses scaled-down versions of actuators and does not use expensive sensors to reduce cost and enhance safety and portability. A limitation of this testbed is that the software vehicle simulator does not reflect actual vehicle behaviour accurately.

While these testbeds are suitable for reverse-engineering, penetrating testing, and evaluating encryption methods, they do not focus on emulating realistic interaction among ECU nodes, hindering direct application for IDS evaluations.

B. Testbeds and Simulations for CAN IDS Evaluation

Compared to other cybersecurity applications, fewer testbeds and simulations are geared towards testing and evaluation of CAN IDS. A platform for evaluating CAN IDS employing various detection techniques and CAN bus traffic features would entail accurate simulation of CAN bus communications, not only in terms of the timing of messages but also the generation of realistic message payload data.

A real-time vehicular CAN bus testbed is provided by Jadidbonab et al. (2021) which can be used for training and testing CAN IDS. A virtual car in the CARLA autonomous vehicle simulator serves as the source of physical data input to simulated ECUs in a virtual CAN bus implemented in Vector CANoe that generates CAN bus traffic. While the virtual car enables the generation of realistic driving scenarios, the virtual CAN bus can be connected to a physical CAN bus consisting of an attack and IDS nodes. A clustering-based intrusion detection algorithm was tested against a targeted ID attack in two ways: offline, against a previously collected CAN bus log; and online, as a plug-and-play addition to the testbed. The classifier yielded lower accuracy and precision in the latter evaluation, which the authors discuss could be due to an overfitted model, inadequately representative training data, or issues with data parsing. However, the differences in the results underline the importance of performing online tests with CAN IDS. A limitation of this testbed is that it does not include bidirectional communication with the virtual car, i.e. the driving behaviour is not influenced by attacks on the virtual CAN bus.

Jichici et al. [7] also use Vector CANoe in their framework that integrates adversary model and intrusion detection nodes in a simulated CAN bus. CAN bus logs collected from a real vehicle are replayed in the virtual CAN bus, while an application interface is developed that allows configuration and launching of fuzzing and targeted ID attacks. MATLAB is used in this framework to enable implementation of CAN IDS, with a k-Nearest Neighbor (kNN) classifier used to demonstrate CAN IDS evaluation. Both message interval and data fields of the CAN bus traffic were used as features for the classifier, which generally yielded very good detection results in terms of sensitivity, specificity, false negative rate (FNR) and false positive rate (FPR). While the usage of real-world data in an industry-standard simulator makes for a realistic testbed, this testbed does not emulate ECUs. Furthermore, performance metrics like detection latency are not measured.

Another CAN bus security testbed is provided by Shi et al. [34] which focuses on maintaining similarity in timing between real-world CAN messages and those generated in the testbed. A real CAN bus log is fed into an ECU Operation Centre which in turn feeds corresponding time series data to each emulated ECU on the testbed CAN bus. A collector module is also implemented which reads messages broadcast by the emulated ECUs to a testbed database. The simulation is evaluated for stability and effectiveness, with the testbed messages demonstrating a relative delay of 0.8% and a negligible packet loss. Using a dynamic time warping (DTW) algorithm, it is also found that the similarity between the real CAN log and the CAN log collected from the testbed is very high. While all the fabrication attacks as well as suspension and masquerade attacks are described and implemented in this testbed, they have not been analysed or used for any form of security testing in this work.

An important limitation of these works is that they do not emulate the behaviour of ECUs. In other words, the simulations do not involve emulated ECUs that read data from the CAN bus. Therefore, attacks like fuzzing, targeted ID, and masquerade attacks that manipulate payloads of certain AIDs would not affect the payloads of other related AIDs. Analysis conducted in [35] using data from a CAN digital twin indicates that even during attacks, there is a correlation between messages containing related signals. In their example, an attack on messages containing vehicle speed is correlated with the change in engine speed. This implies that attacks targeting a particular AID affects messages of related AIDs as well, which would have a bearing on the performance of CAN IDS that analyse payloads. As such, a platform for simulating attacks on a CAN bus should be able to emulate the interactions between related ECUs for effective assessment of CAN IDS.

C. Digital Twin for Automotive CAN

Digital twin already finds diverse applications within the automotive field. Bhatti et al. [36] identify seven areas of application of digital twin technology in the automotive industry in their survey: (a) intelligent driver assistance, (b) autonomous navigation, (c) converters and inverters, (d) consumer centered development, (e) digital design and manufacturing, (f) health monitoring, and (g) battery management systems. In these application areas, DT is utilised to model all or some aspects of a vehicle's functions and operations in a virtual representation, which can be used to predict and analyse the behaviour and state of the replicated functions in various scenarios.

In the domain of automotive cybersecurity, digital twin-based approaches have been presented for privacy assessment and enhancement [37] as well as automated software security testing [38]. However, while the concept of a digital twin has already been applied for intrusion and anomaly detection in industrial cyber-physical systems [39],

[40], it has not been sufficiently explored in the literature with regard to utility for intrusion detection for automotive systems.

A DT-based approach to enable the design, implementation, and maintenance of vehicular wiring harnesses has been proposed in [41]. However, the use of DT for the simulation of in-vehicle networks remains a relatively nascent area of study. To enable use cases such as analysing effects of cyberattacks on in-vehicle networks and the development of security countermeasures, a digital twin of a real-world vehicular CAN bus called CarTwin is proposed by Popa et al. [35]. While previous work in this area focuses on replicating vehicle dynamics, this work replicates a real CAN bus in details like wire lengths, stub lengths, number of nodes, as well as data transmitted on the network. Seven different ECUs of the real CAN bus, related to power steering, instrument panel cluster, powertrain, etc., are emulated using MATLAB Simulink models implemented on development boards. These emulated ECUs not only broadcast CAN messages but also read CAN messages from the bus, thus simulating interactions of related subsystems on the CAN network. A software application with a user interface is used to provide input signals required by the ECU models. Experiments using signals from real CAN logs as input reveal a high correlation between output computed by the digital twin ECUs and the data in the real CAN log. The utility of this digital twin for security analysis is further demonstrated by an analysis of a targeted ID attack on the vehicle speed, where the authors find that messages communicating engine speed are also influenced. This is in contrast to a generic attack-free CAN log manipulated to simulate an attack, where there is no correlation between the targeted vehicle speed and the engine speed. However, this work does not focus on attack implementation or using the proposed DT for IDS evaluation. While CarTwin replicates a real-world CAN bus, it does not use the corresponding DBC for the CAN bus communications. Furthermore, an automatic flow of data from the physical CAN bus to the virtual representation is absent in this proposal, which makes it the 'digital model' level of DT.

D. *Research Gap*

An important limitation of prior CAN testbeds for IDS evaluation is that they lack ECU behaviour emulation. Since ECUs read and use data transmitted by other ECUs on the CAN bus, changes in a signal (e.g. braking signal) may result in changes in related signals (e.g. vehicle speed). As such, a spoofing or masquerading attack that targets a particular AID does not impact only signals of the targeted AID but also related signals in other AIDs. Therefore, if this interaction among ECUs is not replicated in the CAN testbed, the generated CAN data payloads would not reflect real-world data and would not be appropriate for evaluating CAN IDS. The fidelity of generated payloads is especially significant for the evaluation of CAN IDS that utilise frame payloads and leverage correlation among signals for anomaly detection.

While some works like [29], [35] implement ECU models to simulate ECU interactions, they are not geared towards IDS evaluations and do not implement diverse attack scenarios such as fuzzing or spoofing attacks. Simulations of driving scenarios in prior testbeds are also limited to replaying previously captured CAN traffic or generating CAN traffic with unrealistic signal values.

The present work aims to address these gaps by not only emulating the functionality and interactions of a real-world vehicular CAN bus but also implementing unidirectional, real-time data flow from the physical CAN bus to its virtual representation. By emulating ECU behaviours, we can generate realistic CAN bus data in both normal and attack scenarios. Furthermore, the data flow from the physical to the virtual twin would allow us to examine the impact of different attacks in any driving scenario the physical vehicle is in. The present work is thus a step closer to a true digital twin which can be used for robust evaluation of CAN IDS that is reflective of their performance in a real car.

IV. Proposed Evaluation Framework

The DT-based evaluation framework proposed in this work seeks to address the need for high fidelity, low risk, and low-cost alternatives for evaluating CAN IDS against diverse attack scenarios. The scope of the proposed CAN DT is to simulate the behaviour of ECUs on the real-world CAN bus to enable the generation of CAN bus traffic that is realistic in terms of timing and frame payloads, in both normal and attacks scenarios. Towards achieving this, data and specifications from a real-world vehicular CAN bus are collected and used to understand architecture of the target CAN bus as well as the bus traffic that is to be simulated. This information is used to implement a virtual CAN bus with virtual ECUs that simulate the behaviour and interactions of their physical counterparts. This would enable the generation of realistic CAN bus traffic, particularly under attack scenarios that are too risky to be conducted on a real vehicle. The generated CAN traffic can thus be used to perform detection capability and performance assessments of CAN IDS. The proposed DT framework allows not just repeatable experiments for IDS evaluation, but also has the potential to generate attack traffic for different driving scenarios using unidirectional flow of data from the physical CAN bus to its virtual twin. The proposed CAN DT-based framework is described in further detail in the following subsections.

A. *Data Collection*

To implement a realistic DT simulation of a selected vehicular CAN bus, we identify the following information and data that should be collected:

1) Sample CAN bus traffic: A sample of bus traffic collected from the target vehicular CAN bus is required to obtain the set of valid AIDs that are observed during normal operation as well as their normal observed transmission frequencies. In combination with the vehicle's DBC, this sample also serves as the source of bus traffic for running repeatable simulations of normal and attack scenarios. For vehicles that allow it, this sample may be collected from a vehicle's CAN bus via the OBD-II port. For other CAN buses not accessible via the OBD-II port, it is possible to tap the CAN bus and collect this data. We collected data from a Hyundai Sonata 2018, which provides direct access to a CAN bus via its OBD-II port. A sample of CAN bus traffic was collected with the aid of a Korlan USB2CAN adapter *[42]* which was used to connect a Linux laptop to the vehicle's CAN bus via the OBD-II port. The SocketCAN package in Linux provides the can-utils library which includes the functionality to log traffic from a CAN bus with not just the AID, DLC, and data field but also the timestamp.

2) DBC: The DBC for a vehicle specifies the rules for how signal data are encoded in frames of each AID. As such, it provides information such as the signals that are encoded by each AID, along with the ECUs that transmit each AID and the expected receivers. Although the best case would be to obtain the DBC for the vehicle from the Original Equipment Manufacturer (OEM), DBCs are often proprietary and commonly confidential to hinder CAN bus hacking. In such a situation, open source DBCs contributed to repositories like *opendbc [43]* may be leveraged. For the Hyundai Sonata 2018, we find a corresponding DBC in *opendbc*, *hyundai_2015_ccan.dbc*, that is applicable to the vehicle's CAN data.

3) Wiring diagram: The wiring diagram for the vehicle model, often part of auto mechanic manuals and available online, supplements the information that can be obtained from the DBC with respect to the wiring harness – the CAN bus segments that are present along with the number and functionality of ECUs on each segment. These details inform the architecture of the virtual CAN bus as well as the computational model that needs to be implemented for each virtual ECU.

B. Data Analysis

In this stage, the CAN bus traffic is analysed to examine (1) transmission frequencies of each AID, and (2) the relationships among signals transmitted by each AID.

Frames of each AID are typically broadcast by only one ECU and at regular intervals [13]. The time interval for each AID, which is not specified in the DBC, should be obtained from the collected CAN bus sample by analysing each stream of AIDs so that the virtual representations of the respective ECUs can be modelled to perform transmissions at similar intervals.

The DBC for the vehicle may be used to decode the signals transmitted in the captured CAN traffic. A pairwise correlation test performed among all the decoded signals from the dataset should reveal groups of signals showing high correlation among each other. Attacks targeting a particular AID, such as in a spoofing or masquerade attack, should result in changes not just to the signals of that AID, but also other AIDs with highly-correlated signals. These groups of correlated signals and AIDs can allow us to select a subset of ECUs if we are interested in a smaller scale simulation that can produce realistic changes under attack scenarios.

The data analysis may be performed with any statistical packages, such as the *pandas* and *numpy* Python packages in our case. For the deserialising signals from CAN frames using the DBC, we use the *cantools* Python library that provides utilities for parsing DBC files, encoding and decoding signals, monitoring and plotting CAN signals [44].

C. ECU Modelling

After identifying the signals and AIDs of interest and the corresponding ECUs, the behaviour of these ECUs needs to be emulated with respect to their functionalities and data transmission. For each ECU, given the set of input and output signals, we need to implement the computation model that can generate output signals from input. The virtual counterpart of each ECU, thus, uses the DBC to decode input signals from received CAN frames, compute output signals, and then encode the output signals in frames for transmission. The virtual ECUs transmit their respective AIDs at the time intervals determined in the data analysis stage. The virtual ECUs are connected on a virtual CAN bus which serves as the digital twin of the real-world CAN bus. The virtual CAN bus is interfaced with the physical counterpart so that signals generated on the real CAN bus can be sent to the virtual CAN bus as input.

We implement the virtual ECUs and CAN bus using the Vehicle Network Toolbox from MathWorks [45], which provides functions and blocks for CAN communication simulations. This virtual CAN bus is bridged to the real-world CAN bus, which could be via the vehicle's OBD-II port or, in the case of attack simulations repeated with a single driving scenario, a CAN bus prototype with an appropriate connector. If we consider a smaller scale DT whereby only a subset of ECUs are emulated, then frames transmitted by the other ECUs are replayed from the real CAN bus in the manner of a restbus simulation [35], [46].

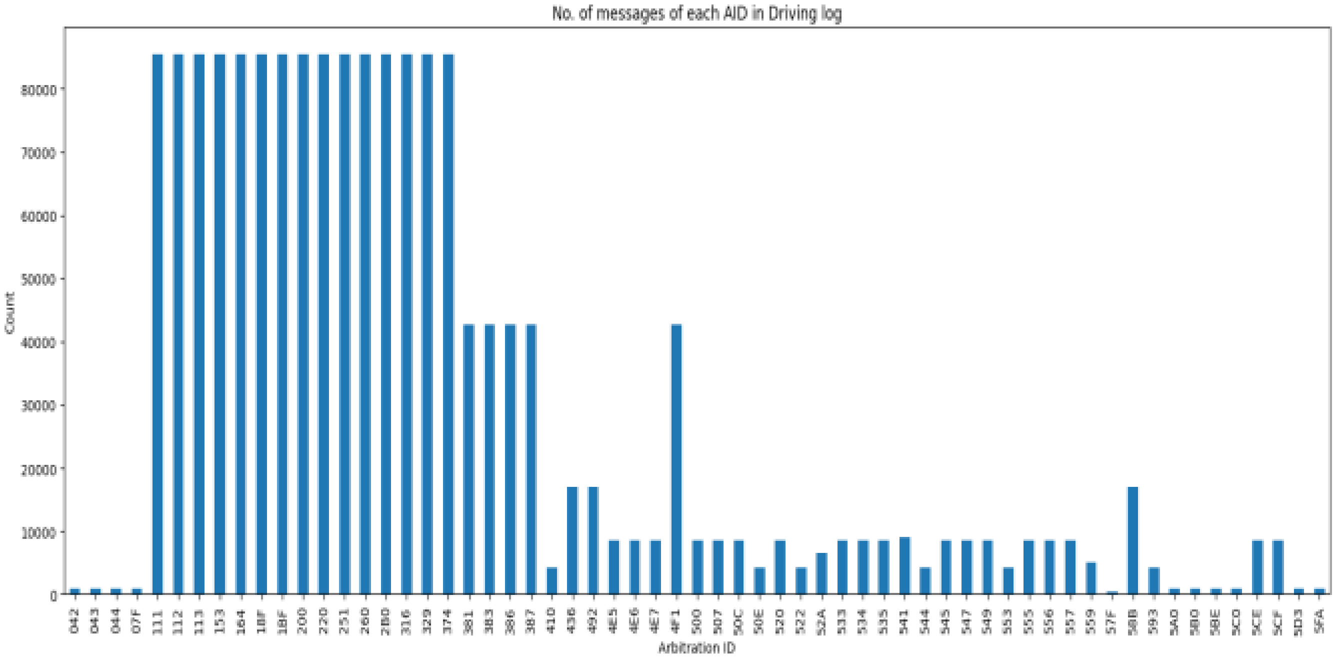

Fig. 2 Number of CAN frames by AID

D. *Attack Implementation*

Once the virtual CAN bus twin is operational, the attack scenarios required for IDS evaluation may be implemented. In the attack model that we consider, most attacks require a strongly compromised node that is capable of injecting frames on the CAN bus. As such, an attack node is added to the virtual CAN bus which can be programmed to inject frames at appropriate frequencies and suspend transmissions for fabrication, suspension, and masquerade attacks. Given that the virtual CAN bus represents a white box system where the AIDs and signals associated with all functionalities are known, the attack node can be set to execute spoofing, suspension, and masquerade attacks that target specific functionalities and inject malicious frames with the targeted AID. Executing DoS and fuzzing attacks are relatively simpler, and so is a replay attack, which entails capturing and replaying bus traffic from the real or virtual CAN bus.

E. *IDS Evaluation*

At this stage, the CAN DT may be used for IDS evaluation against different attack scenarios and in different driving scenarios. A node running the IDS as well as measuring evaluation metrics is added to the virtual DT for testing against the generated CAN traffic. The DT simulation can be run using either recorded sample of CAN bus traffic or real-time CAN traffic from the physical CAN bus. The former case allows data collected during a particular driving scenario to be used to drive a simulation multiple times with different attack scenarios. With this, one or more IDS can be evaluated against multiple attacks to obtain statistically significant results. In the case of using real-time CAN traffic from the physical CAN bus, attacks can be run against the vehicular CAN bus while the vehicle is in different driving situations, e.g. stationary, driving at high speeds. This can allow the evaluation of any CAN IDS against a wide variety of attack and driving scenario combinations.

While the DT can be used for real-time assessment of CAN IDS to measure both security and performance metrics, it can also be used to generate realistic attack datasets. Generated datasets used to evaluate a particular CAN IDS can also be made available along with the IDS so that future IDS proposals can be directly compared or benchmarked using the same datasets.

V. Findings and Discussion

As mentioned previously, we begin with collecting a sample of data from a Hyundai Sonata 2018. Approximately 14 minutes of driving data was collected while it was driven on urban roads. The collected data consisted of a total of 1,754,253 CAN frames, averaging 2054.82 frames transmitted per second. A total of 61 unique AIDs were found in this log. In Fig. 2 which shows the number of frames of each AID, we can see that, with some exceptions, frames with lower-valued AIDs appear the most on the CAN bus. This can be a result of the arbitration mechanism whereby lower-valued AIDs have higher priority for transmission and higher-valued AIDs have to wait for retransmission in the event of bus contention. Lower-valued AIDs are therefore always able to broadcast, while high-valued AIDs are transmitted fewer times due to lost arbitration in some situations.

A. *Timing Analysis*

We divided the collected CAN bus log into streams of frames of each AID and calculated the time interval for each frame. The time interval for a frame can be described as the period of time between the transmission of the frame and the transmission of the previous frame of the same AID. From these, we calculated the average time interval as well as the maximum percentage of deviation from the mean for each AID.

Apart from a few AIDs that appear at intervals of 1-2 seconds, a majority of the AIDs (50) in the CAN bus log were broadcast at time intervals under 0.2 seconds. In Fig. 3 which provides a distribution of these AIDs by average time interval, we can see that the time intervals even among these AIDs vary in magnitude and scale, ranging from 0.01 to 0.2 second.

The maximum percentage error from the mean time interval was calculated for each AID stream to understand the regularity of the CAN bus transmissions. As can be observed in Fig. 4, the largest number of AIDs show under 40% deviation from the average time interval, indicating that these are transmitted at reliably regular intervals. A smaller number of AIDs show greater variation, which is indicative of irregular or event-triggered transmissions. Variations in time intervals also arise from ECUs losing arbitration to higher-priority frame and having to wait to attempt retransmission of lower-priority frames.

It is important for any CAN bus modelling effort to take into consideration these timing features in CAN traffic. While lower-valued AIDs may always be transmitted at regular intervals without losing arbitration and without having frames delayed, higher-valued AIDs may lose arbitration and show delayed transmissions more often. These differences in timing characteristics are relevant considerations for timing- and frequency-based CAN IDS, which detect deviations from normal patterns in time intervals or frequencies. In the event of DoS or fuzzing attacks, it is expected for time intervals of legitimate frames to increase as the injected frames hinder normal transmissions, while a spoofing attack would cause time intervals of the targeted AID to decrease. Overall, a faithful CAN bus model should not only incorporate the dynamics of frame timing under normal operation, but also during different attack scenarios, for more accurate CAN IDS evaluations.

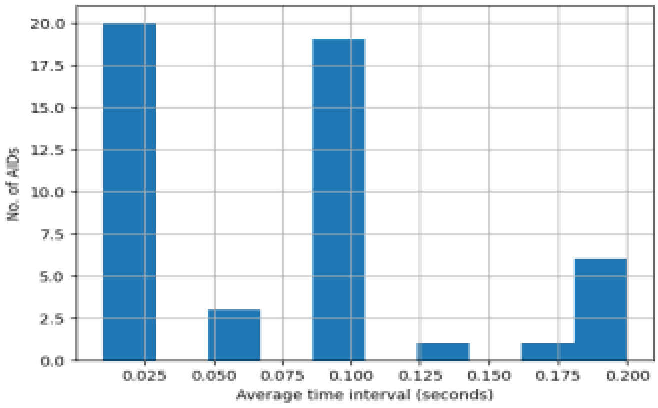

Fig. 3 Distribution of AIDs by average time interval, excluding AIDs with average time intervals greater than 0.25 second

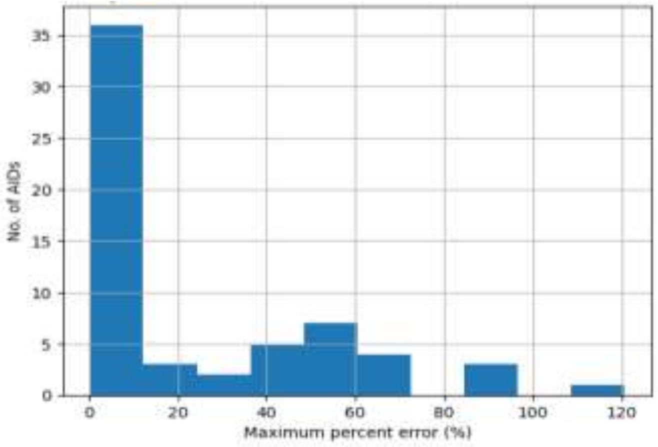

Fig. 4 Distribution of AIDs by maximum percent error from average time interval

TABLE 1
SENDER ECUS FOUND IN COLLECTED CAN LOG

Acronym in DBC	Full name
DATC	Dual Automatic Temperature Control
BCM	Body Control Module
TCU	Transmission Control Unit
ESC	Electronic Stability Control
EMS	Engine Management System
MDPS	Motor Driven Power Steering
ABS	Anti-lock Brake System
CLU	Cluster Module
ACU	Airbag Control Unit
FPCM	Fuel injection Pump Control Module
LCA	Lane Centering/Change Assist
ODS	Occupant Detection System

B. Signal Analysis

The data collected from the Hyundai Sonata 2018 consisted of CAN data frames with payloads in raw bytes. To deserialise the signals encoded in these data frames, we use the *hyundai_2015_ccan*.db file from *opendbc*, using which we are able to deserialise signals for 48 AIDs. These 48 AIDs are transmitted by 12 ECUs responsible for different subsystems, which are listed in Table 1.

The *cantools* library was used to deserialise signals from the CAN log using the aforementioned DBC file to yield a total of 627 signals. Of these signals, 493 signals remain constant throughout the CAN bus log, leaving 134 signals for analysis. We visualise the steering angle signal from AID 0x2B0, transmitted by the MDPS ECU in **Error! Reference source not found.** as well as the individual wheel speeds from AID 0x386, transmitted by the ABS ECU in Fig. 6.

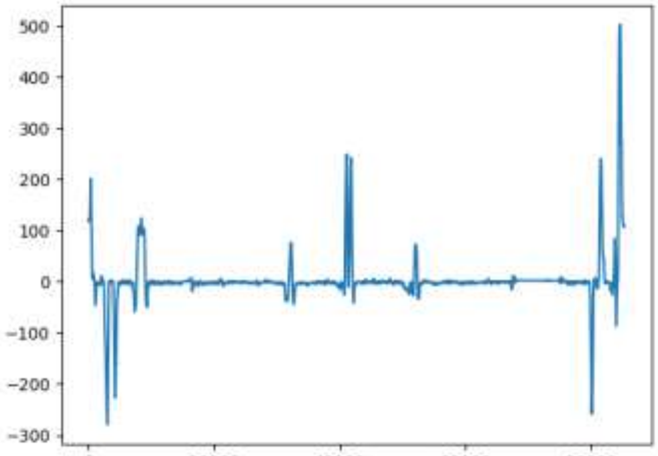

Fig. 5 Steering angle signal from AID 0x2B0

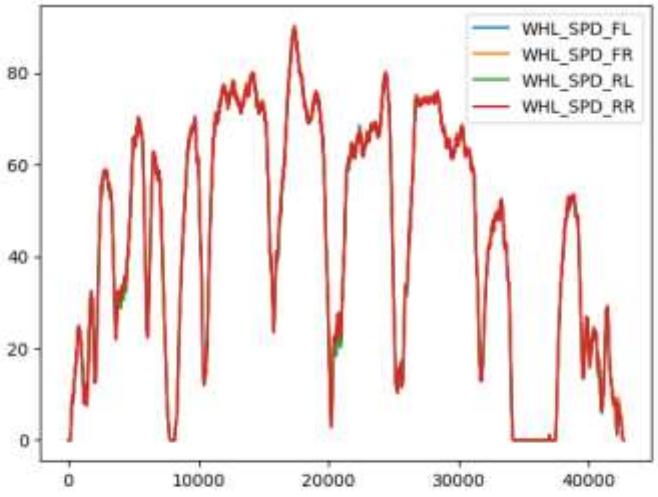

Fig. 6 Individual wheel speed signals from AID 0x386

To understand how different signals are related to each other and identify groups of correlated signals, we perform a pairwise Pearson correlation test on the non-constant 134 signals and generate a correlation heatmap, which is available at [47]. Signals showing a magnitude of correlation coefficient higher than 0.5 are listed in Table 2. The signals in the heatmap are reorganised by applying hierarchical agglomerative clustering with complete linkage, to facilitate the identification of signal clusters showing high correlations. While it may be expected to find correlations among signals originating from the same ECUs, we see in this heatmap significant correlations among signals originating from *different* AIDs and ECUs. An example is wheel speed signals from ABS AIDs showing a high positive correlation with signals from TCU and EMS ECUs. These correlations indicate that it is possible for changes in a signal to result in changes in other related signals. This is due to the fact that each ECU uses data transmitted by other ECUs as input for its respective functions and in turn, transmits signals that are used by other ECUs.

The correlation among related signals should also be maintained in attack scenarios like spoofing and masquerade where fabricated frames with manipulated payloads are injected to provide false information to ECUs. In this situation, when ECUs read and use the spoofed values of signals in the malicious frames, the anomaly cascades into the data transmitted by these ECUs. Such effects of attacks on CAN bus traffic are not captured in other methods of generating CAN bus data such as augmentation of benign CAN bus logs or testbeds that do not emulate ECU behaviour [35]. In such methods, the injected frames do not produce any changes in related frames, which is different from what would be observed in a real CAN bus and is thus not useful for evaluating CAN IDS particularly based on analysing frame payloads.

VI. Conclusions and Future Work

Using real vehicles for attack simulation and security testing can be restrictive in terms of the associated costs, safety risks, and attack scenarios that can be conducted. While testbeds and simulations do not have these challenges, they do not provide sufficient fidelity for the assessment of CAN IDS that use different features of CAN bus traffic. An important limitation of current testbeds and simulations is that they do not emulate the interaction of ECUs or generate realistic CAN traffic. This work aims to address these gaps by proposing a DT-based evaluation framework for CAN IDS which can be used to generate diverse attack scenarios and perform detection capability and performance evaluations of CAN IDS. This CAN DT models a real-world CAN bus at the ECU level and interfaces with the real-world bus for data to drive the DT simulation. Not only can it allow repeatable simulations of attack

TABLE 2
SIGNALS WITH ABSOLUTE CORRELATION COEFFICIENT GREATER THAN 0.5

No.	Signal	AID	Sender	Highly correlated signals
1	CR_Datc_OutTempF	044	DATC	59, 83
2	CF_Tcu_Alive1	111	TCU	10, 15
3	N_TC	111	TCU	8, 7, 12, 21, 23, 45, 42, 51, 50, 65, 64, 63, 62, 69, 71, 80, 84, 85, 86, 89
4	SWI_CC	111	TCU	9, 7, 12, 36, 47, 42, 65, 64, 63, 62, 69, 71, 85
5	SWI_GS	111	TCU	16
6	TEMP_AT	111	TCU	53, 68, 74, 75, 77, 90
7	VS_TCU	112	TCU	3, 4, 9, 8, 12, 21, 23, 36, 45, 42, 65, 64, 63, 62, 69, 71, 80, 84, 85
8	N_TC_RAW	112	TCU	3, 7, 12, 21, 23, 45, 42, 51, 50, 65, 64, 63, 62, 69, 71, 80, 84, 85, 86, 89
9	CUR_GR	112	TCU	4, 7, 12, 36, 42, 65, 64, 63, 62, 69, 71, 85
10	CF_Tcu_Alive	112	TCU	2, 15
11	N_INC_TCU	112	TCU	13
12	CF_Tcu_TarGr	113	TCU	3, 4, 9, 8, 7, 21, 36, 42, 65, 64, 63, 62, 69, 71, 85
13	N_TGT_LUP	113	TCU	11
14	CF_Tcu_ShfPatt	113	TCU	38, 54
15	CF_Tcu_Alive3	113	TCU	2, 10
16	CF_Tcu_ITPhase	113	TCU	5
17	AliveCounter_TCS1	153	ESC	18
18	CheckSum_TCS1	153	ESC	17
19	CF_Esc_AliveCnt	164	ESC	20, 29
20	CF_Esc_Chksum	164	ESC	19, 29
21	R_NEngIdlTgC	18F	EMS	3, 8, 7, 12, 23, 45, 42, 51, 50, 65, 64, 63, 62, 69, 71, 80, 84, 85, 86, 89
22	R_PAcnC	18F	EMS	60, 61, 73, 76
23	CF_Ems_PumpTPres	200	EMS	3, 8, 7, 21, 40, 45, 42, 51, 50, 65, 64, 63, 62, 69, 71, 80, 84, 85, 86, 89
24	FCO	200	EMS	35, 33, 45, 44, 43, 51, 50, 82, 84, 86, 88, 89
25	LONG_ACCEL	220	ESC	40, 35, 37, 33, 45, 44, 43, 51, 50, 82, 84, 86, 88, 89
26	YAW_RATE	220	ESC	27, 30, 31, 32, 41, 56
27	LAT_ACCEL	220	ESC	26, 30, 31, 32, 41, 56
28	CYL_PRES	220	ESC	52, 72, 87
29	ESP12_Checksum	220	ESC	19, 20
30	CR_Mdps_OutTq	251	MDPS	27, 26, 31, 32, 41, 56
31	CR_Mdps_StrColTq	251	MDPS	27, 26, 30, 32, 41, 56
32	CR_Mdps_StrTq	251	MDPS	27, 26, 30, 31, 41, 56
33	TQI_TARGET	260	EMS	24, 25, 40, 35, 37, 45, 44, 43, 51, 50, 82, 84, 86, 88, 89
34	TQI_MAX	260	EMS	46, 48, 87
35	TQI	260	EMS	24, 25, 40, 37, 33, 45, 44, 43, 51, 50, 82, 84, 86, 88, 89
36	SPK_TIME_CUR	260	EMS	4, 9, 7, 12, 47, 42, 65, 64, 63, 62, 69, 71, 81, 85
37	TQI_MIN	260	EMS	25, 35, 33, 44, 43, 51, 50, 82, 86, 88, 89
38	CRUISE_LAMP_S	260	EMS	14, 54
39	CRUISE_LAMP_M	260	EMS	75, 77, 90
40	CF_Ems_AclAct	260	EMS	23, 25, 35, 33, 45, 44, 43, 51, 50, 82, 84, 86, 88, 89
41	SAS_Angle	2B0	MDPS	27, 26, 30, 31, 32, 56
42	VS	316	EMS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 65, 64, 63, 62, 69, 71, 80, 84, 85
43	TQI_ACOR	316	EMS	24, 25, 40, 35, 37, 33, 45, 44, 51, 50, 82, 84, 86, 88, 89
44	TQI	316	EMS	24, 25, 40, 35, 37, 33, 45, 43, 51, 50, 82, 84, 86, 88, 89
45	N	316	EMS	3, 8, 7, 21, 23, 24, 25, 40, 35, 33, 44, 43, 42, 51, 50, 65, 64, 63, 62, 69, 80, 82, 84, 85, 86, 89

scenarios for statistically significant IDS evaluations, but, with the flow of data from the real to the virtual CAN bus, it can be used to generate any combination of attack and driving scenarios for a thorough assessment of CAN IDS that is reflective of performance in the real-world. Towards building the DT of a real CAN bus, we collected data from a Hyundai Sonata 2018 and analysed timing and signal data to understand patterns that are relevant to intrusion detection and ECU modelling.

There are several challenges with the proposed CAN DT for CAN IDS evaluation. Firstly, it relies on the vehicle DBC, which is not always available for all vehicle models. While we

TABLE 2 (CONTD.)
SIGNALS WITH ABSOLUTE CORRELATION COEFFICIENT GREATER THAN 0.5

No.	Signal	AID	Sender	Highly correlated signals
46	RATIO_TQI_BAS_MAX_STND	316	EMS	34
47	PUC_STAT	316	EMS	4, 36, 81
48	TQFR	316	EMS	34
49	TEMP_ENG	329	EMS	59, 83
50	TPS	329	EMS	3, 8, 21, 23, 24, 25, 40, 35, 37, 33, 45, 44, 43, 51, 82, 84, 86, 88, 89
51	PV_AV_CAN	329	EMS	3, 8, 21, 23, 24, 25, 40, 35, 37, 33, 45, 44, 43, 50, 82, 84, 86, 88, 89
52	BRAKE_ACT	329	EMS	28, 72, 87
53	MAF_FAC_ALTI_MMV	329	EMS	6, 68, 74, 75, 77, 90
54	ACC_ACT	329	EMS	14, 38
55	MUL_CODE	329	EMS	66
56	CR_Mdps_DrvTq	381	MDPS	27, 26, 30, 31, 32, 41
57	CF_Fatc_ChkSum	383	DATC	58
58	CF_Fatc_MsgCnt	383	DATC	57
59	CR_Fatc_OutTemp	383	DATC	1, 49, 83
60	CR_Fatc_OutTempSns	383	DATC	22, 61, 73, 76
61	CR_Fatc_TqAcnOut	383	DATC	22, 60, 73
62	WHL_SPD_RR	386	ABS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 65, 64, 63, 69, 71, 80, 84, 85
63	WHL_SPD_RL	386	ABS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 65, 64, 62, 69, 71, 80, 84, 85
64	WHL_SPD_FR	386	ABS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 65, 63, 62, 69, 71, 80, 84, 85
65	WHL_SPD_FL	386	ABS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 64, 63, 62, 69, 71, 80, 84, 85
66	WHL_SPD_AliveCounter_MSB	386	ABS	55
67	CF_Ems_ModeledAmbTemp	492	EMS	70, 76
68	CR_Ems_EngOilTemp	492	EMS	6, 53, 74, 77, 90
69	CF_Clu_Vanz	4F1	CLU	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 65, 64, 63, 62, 71, 80, 84, 85
70	CF_Clu_DTE	50C	CLU	67, 73, 76
71	CF_Clu_VehicleSpeed	52A	CLU	3, 4, 9, 8, 7, 12, 21, 23, 36, 42, 65, 64, 63, 62, 69, 80, 84, 85
72	BAT_Alt_FR_Duty	545	EMS	28, 52, 87
73	TEMP_FUEL	545	EMS	22, 60, 61, 70, 76
74	AMP_CAN	545	EMS	6, 53, 68, 75, 77, 90
75	CTR_CDN_OBD	547	EMS	6, 39, 53, 74, 77, 90
76	IntAirTemp	547	EMS	22, 60, 67, 70, 73
77	STATE_DC_OBD	547	EMS	6, 39, 53, 68, 74, 75, 90
78	BAT_SOH	549	EMS	79
79	BAT_SOC	549	EMS	78
80	CR_Fpcm_LPActPre	555	FPCM	3, 8, 7, 21, 23, 45, 42, 65, 64, 63, 62, 69, 71, 84, 85
81	PID_03h	556	EMS	36, 47
82	PID_04h	556	EMS	24, 25, 40, 35, 37, 33, 45, 44, 43, 51, 50, 84, 86, 88, 89
83	PID_05h	556	EMS	1, 49, 59
84	PID_0Ch	556	EMS	3, 8, 7, 21, 23, 24, 25, 40, 35, 33, 45, 44, 43, 42, 51, 50, 65, 64, 63, 62, 69, 71, 80, 82, 85, 86, 89
85	PID_0Dh	556	EMS	3, 4, 9, 8, 7, 12, 21, 23, 36, 45, 42, 65, 64, 63, 62, 69, 71, 80, 84
86	PID_11h	556	EMS	3, 8, 21, 23, 24, 25, 40, 35, 37, 33, 45, 44, 43, 51, 50, 82, 84, 88, 89
87	PID_07h	557	EMS	28, 34, 52, 72
88	PID_0Bh	557	EMS	24, 25, 40, 35, 37, 33, 44, 43, 51, 50, 82, 86, 89
89	PID_23h	557	EMS	3, 8, 21, 23, 24, 25, 40, 35, 37, 33, 45, 44, 43, 51, 50, 82, 84, 86, 88
90	CF_Clu_Odometer	5B0	CLU	6, 39, 53, 68, 74, 75, 77

use an open-source DBC for our vehicle, such DBC is not available for all car models, which restricts the number of vehicles to which this methodology can be applied.

Moreover, the implementation of data flow between the physical CAN bus and the device hosting its virtual twin may introduce a new attack vector and raise security and privacy concerns. This attack vector could potentially be exploited to sniff the physical CAN bus and steal data revealing information such as driving behaviour or conduct attacks on the physical CAN bus that interfere with normal operation. It is necessary to implement security measures while establishing communication between the physical and digital twin to ensure that these threats do not become a reality. Finally, it is also pertinent to explore resource utilisation of the virtual twin, especially if we are interested in scaling up the virtual representation by increasing the number of ECUs that are emulated in the virtual CAN bus.

Implementation of the CAN DT using insights gathered from data analysis and validating the accuracy of generated CAN bus traffic remain as future work.

Acknowledgment

The authors hereby acknowledge the review support offered by the IJPCC reviewers who took their time to study the manuscript and find it acceptable for publishing.

Conflict of Interest

The authors declare that there is no conflict of interest

References

[1] R. N. Charette, "How Software Is Eating the Car," IEEE Spectrum. Accessed: Sep. 08, 2022. [Online]. Available https://spectrum.ieee.org/software-eating-car

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, and T. Kohno, "Experimental Security Analysis of a Modern Automobile," *2010 IEEE Symp. Secur. Priv.*, pp. 1–16, 2010.

[3] S. Checkoway *et al.*, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," *Proc. 20th USENIX Secur. Symp.*, pp. 77–92, 2011.

[4] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," in *Black Hat USA*, Las Vegas, 2015.

[5] "Tencent Keen Security Lab: Experimental Security Assessment on Lexus Cars," Keen Security Lab Blog. Accessed: Jun. 14, 2022. [Online]. Available: http://keenlab.tencent.com/2020/03/30/Tencent-Keen-Security-Lab-Experimental-Security-Assessment-on-Lexus-Cars/index.html

[6] H. Jadidbonab, A. Tomlinson, H. N. Nguyen, T. Doan, and S. A. Shaikh, "A Real-Time In-Vehicle Network Testbed for Machine Learning-Based IDS Training and Validation," 2021, p. 16.

[7] C. Jichici, B. Groza, and P.-S. Murvay, "Integrating Adversary Models and Intrusion Detection Systems for In-vehicle Networks in CANoe," in *Innovative Security Solutions for Information Technology and Communications*, vol. 12001, E. Simion and R. Géraud-Stewart, Eds., in Lecture Notes in Computer Science, vol. 12001. , Cham: Springer International Publishing, 2020, pp. 241–256. doi: 10.1007/978-3-030-41025-4_16.

[8] B. Lampe and W. Meng, "can-train-and-test: A curated CAN dataset for automotive intrusion detection," *Comput. Secur.*, vol. 140, p. 103777, May 2024, doi: 10.1016/j.cose.2024.103777.

[9] F. Pollicino, D. Stabili, and M. Marchetti, "Performance Comparison of Timing-Based Anomaly Detectors for Controller Area Network: A Reproducible Study," *ACM Trans. Cyber-Phys. Syst.*, vol. 8, no. 2, pp. 1–24, Apr. 2024, doi: 10.1145/3604913.

[10] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and Countermeasures for In-Vehicle Networks," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–37, Apr. 2021, doi: 10.1145/3431233.

[11] K. Tindell, "The canframe.py tool," Ken Tindell's blog. Accessed: Jan. 27, 2023. [Online]. Available: https://kentindell.github.io/2020/01/03/canframe_py_tool/

[12] D. Stabili, F. Pollicino, and A. Rota, "A benchmark framework for CAN IDS," presented at the Italian Conference on Cyber Security, Apr. 2021.

[13] M. E. Verma *et al.*, "A comprehensive guide to CAN IDS data and introduction of the ROAD dataset," *PLOS ONE*, vol. 19, no. 1, p. e0296879, Jan. 2024, doi: 10.1371/journal.pone.0296879.

[14] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of CAN Bus Security Challenges," *Sens. Switz.*, vol. 20, no. 8, 2020, doi: 10.3390/s20082364.

[15] K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *25th USENIX Security Symposium (USENIX Security 16)*, Austin, Texas: USENIX Association, 2016, pp. 911--927. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

[16] 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*, Vienna Austria: ACM, Oct. 2016, pp. 1044–1055. doi: 10.1145/2976749.2978302.

[17] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," 2014.

[18] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Trans. Ind. Inform.*, vol. 15, no. 4, pp. 2405–2415, 2019, doi: 10.1109/TII.2018.2873186.

[19] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *J. Manuf. Syst.*, vol. 58, no. October 2019, pp. 346–361, 2021, doi: 10.1016/j.jmsy.2020.06.017.

[20] J. Guo, M. Bilal, Y. Qiu, C. Qian, X. Xu, and K.-K. Raymond Choo, "Survey on digital twins for Internet of Vehicles: Fundamentals, challenges, and opportunities," *Digit. Commun. Netw.*, vol. 10, no. 2, pp. 237–247, Apr. 2024, doi: 10.1016/j.dcan.2022.05.023.

[21] E. VanDerHorn and S. Mahadevan, "Digital Twin: Generalization, characterization and implementation," *Decis. Support Syst.*, vol. 145, p. 113524, Jun. 2021, doi: 10.1016/j.dss.2021.113524.

[22] M. Eckhart and A. Ekelhart, "Towards security-aware virtual environments for digital twins," *CPSS 2018 - Proc. 4th ACM Workshop Cyber-Phys. Syst. Secur. Co-Located ASIA CCS 2018*, pp. 61–72, 2018, doi: 10.1145/3198458.3198464.

[23] S. Sharmin, H. Mansor, A. F. Abdul Kadir, and N. A. Aziz, "Benchmarking frameworks and comparative studies of Controller Area Network (CAN) intrusion detection systems: A review," *J. Comput. Secur.*, vol. 32, no. 5, pp. 477–507, Nov. 2024, doi: 10.3233/JCS-230027.

[24] G. Karopoulos, G. Kambourakis, E. Chatzoglou, J. L. Hernández-Ramos, and V. Kouliaridis, "Demystifying In-Vehicle Intrusion Detection Systems: A Survey of Surveys and a Meta-Taxonomy," *Electronics*, vol. 11, no. 7, p. 1072, Mar. 2022, doi: 10.3390/electronics11071072.

[25] S. Stachowski, R. Gaynier, and D. J. LeBlanc, "An Assessment Method for Automotive Intrusion Detection System Performance," National Highway Traffic Safety Administration, Washington, D.C., DOT HS 812 708, Apr. 2019.

[26] S. Mahmood, H. N. Nguyen, and S. A. Shaikh, "Automotive Cybersecurity Testing: Survey of Testbeds and Methods," in *Digital Transformation, Cyber Security and Resilience of Modern Societies*, vol. 84, T. Tagarev, K. T. Atanassov, V. Kharchenko, and J. Kacprzyk, Eds., in Studies in Big Data, vol. 84. , Cham: Springer International Publishing, 2021, pp. 219–243. doi: 10.1007/978-3-030-65722-2_14.

[27] O. Cros, A. Thiroux, and G. Chênevert, "Cacao, a CAN-Bus Simulation Platform for Secured Vehicular Communication," in *Ad Hoc Networks*, vol. 345, L. Foschini and M. El Kamili, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 345. , Cham: Springer International Publishing, 2021, pp. 213–224. doi: 10.1007/978-3-030-67369-7_16.

[28] P. Mundhenk, A. Mrowca, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Open source model and simulator for real-time performance analysis of automotive network security," *ACM SIGBED Rev.*, vol. 13, no. 3, pp. 8–13, Aug. 2016, doi: 10.1145/2983185.2983186.

[29] X. Zheng, L. Pan, H. Chen, R. Di Pietro, and L. Batten, "A Testbed for Security Analysis of Modern Vehicle Systems," in *2017 IEEE*

Trustcom/BigDataSE/ICESS, Sydney, NSW: IEEE, Aug. 2017, pp. 1090–1095. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.357.

[30] D. S. Fowler, M. Cheah, S. A. Shaikh, and J. Bryans, "Towards a Testbed for Automotive Cybersecurity," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, Tokyo, Japan: IEEE, Mar. 2017, pp. 540–541. doi: 10.1109/ICST.2017.62.

[31] D. Granata, M. Rak, and G. Salzillo, "Towards HybridgeCAN, a hybrid bridged CAN platform for automotive security testing," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, Rhodes, Greece: IEEE, Jul. 2021, pp. 249–254. doi: 10.1109/CSR51186.2021.9527969.

[32] C. E. Everett and D. McCoy, "OCTANE: Open Car Testbed And Network Experiments Bringing Cyber-Physical Security Research to Researchers and Students," in *6th Workshop on Cyber Security Experimentation and Test*, 2013, p. 8.

[33] T. Toyama, T. Yoshida, H. Oguma, and T. Matsumoto, "PASTA: Portable Automotive Security Testbed with Adaptability," presented at the Black Hat Europe, London, 2018.

[34] D. Shi, L. Kou, C. Huo, and T. Wu, "A CAN Bus Security Testbed Framework for Automotive Cyber-Physical Systems," *Wirel. Commun. Mob. Comput.*, vol. 2022, pp. 1–11, Aug. 2022, doi: 10.1155/2022/7176194.

[35] L. Popa, A. Berdich, and B. Groza, "CarTwin—Development of a Digital Twin for a Real-World In-Vehicle CAN Network," *Appl. Sci.*, vol. 13, no. 1, p. 445, Dec. 2022, doi: 10.3390/app13010445.

[36] G. Bhatti, H. Mohan, and R. Raja Singh, "Towards the future of smart electric vehicles: Digital twin technology," *Renew. Sustain. Energy Rev.*, vol. 141, no. January, p. 110801, 2021, doi: 10.1016/j.rser.2021.110801.

[37] V. Damjanovic-Behrendt, "A Digital Twin-based Privacy Enhancement Mechanism for the Automotive Industry," 2018, pp. 272–279.

[38] S. Marksteiner, S. Bronfman, M. Wolf, and E. Lazebnik, "Using Cyber Digital Twins for Automated Automotive Cybersecurity Testing," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Vienna, Austria: IEEE, Sep. 2021, pp. 123–128. doi: 10.1109/EuroSPW54576.2021.00020.

[39] F. Akbarian, E. Fitzgerald, and M. Kihl, "Intrusion Detection in Digital Twins for Industrial Control Systems," *2020 28th Int. Conf. Softw. Telecommun. Comput. Netw. SoftCOM 2020*, 2020, doi: 10.23919/SoftCOM50211.2020.9238162.

[40] A. Pokhrel, V. Katta, and R. Colomo-Palacios, "Digital Twin for Cybersecurity Incident Prediction: A Multivocal Literature Review," *Proc. - 2020 IEEEACM 42nd Int. Conf. Softw. Eng. Workshop ICSEW 2020*, pp. 671–678, 2020, doi: 10.1145/3387940.3392199.

[41] R. Tharma, R. Winter, and M. Eigner, "An Approach for the Implementation of the Digital Twin in the Automotive Wiring Harness Field," presented at the 15th International Design Conference, 2018, pp. 3023–3032. doi: 10.21278/idc.2018.0188.

[42] "Korlan USB2CAN - 8devices." Accessed: Jun. 21, 2024. [Online]. Available: https://www.8devices.com/products/usb2can_korlan

[43] "commaai/opendbc: democratize access to car decoder rings." Accessed: Jan. 20, 2024. [Online]. Available: https://github.com/commaai/opendbc

[44] "CAN BUS tools — cantools 39.4.3.dev10+gcc02988 documentation." Accessed: Dec. 26, 2024. [Online]. Available: https://cantools.readthedocs.io/en/latest/

[45] "Vehicle Network Toolbox." Accessed: Dec. 26, 2024. [Online]. Available: https://www.mathworks.com/help/vnt/index.html

[46] "Accelerating Testing with Advanced ECU Restbus Simulation." Accessed: Dec. 24, 2024. [Online]. Available: https://www.ni.com/en/solutions/transportation/hardware-in-the-loop/vehicle-communication-software-suite.html

[47] S. Sharmin, "Pairwise Pearson correlation heatmap for non- constant signals extracted from a Hyundai Sonata 2018." Zenodo, Jan. 2025. doi: 10.5281/zenodo.14627705.