CANTXSec: A Deterministic Intrusion Detection and Prevention System for CAN Bus Monitoring ECU Activations

Denis Donadel[1], Kavya Balasubramanian[2], Alessandro Brighente[1], Bhaskar Ramasubramanian[3], Mauro Conti[1], and Radha Poovendran[2]

[1] University of Padova, Padua, Italy
[2] University of Washington, Seattle, USA
[3] Western Washington University, Seattle, USA
denis.donadel@phd.unipd.it, kavyab25@uw.edu, alessandro.brighente@unipd.it, ramasub@wwu.edu, mauro.conti@unipd.it, rp3@uw.edu

Abstract. Despite being a legacy protocol with various known security issues, Controller Area Network (CAN) still represents the de-facto standard for communications within vehicles, ships, and industrial control systems. Many research works have designed Intrusion Detection Systems (IDSs) to identify attacks by training machine learning classifiers on bus traffic or its properties. Actions to take after detection are, on the other hand, less investigated, and prevention mechanisms usually include protocol modification (e.g., adding authentication). An effective solution has yet to be implemented on a large scale in the wild. The reasons are related to the effort to handle sporadic false positives, the inevitable delay introduced by authentication, and the closed-source automobile environment that does not easily permit modifying Electronic Control Units (ECUs) software.

In this paper, we propose *CANTXSec*, the first deterministic Intrusion Detection and Prevention system based on physical ECU activations. It employs a new classification of attacks based on the attacker's need in terms of access level to the bus, distinguishing between Frame Injection Attacks (FIAs) (i.e., using frame-level access) and Single-Bit Attacks (SBAs) (i.e., employing bit-level access). *CANTXSec* detects and prevents classical attacks in the CAN bus, while detecting advanced attacks that have been less investigated in the literature. We prove the effectiveness of our solution on a physical testbed, where we achieve 100% detection accuracy in both classes of attacks while preventing 100% of FIAs. Moreover, to encourage developers to employ *CANTXSec*, we discuss implementation details, providing an analysis based on each user's risk assessment.

Keywords: CAN bus · Intrusion Detection System · Intrusion Prevention System · Automotive Security · Cyber-Physical Systems.

1 Introduction

Nowadays, automobiles are equipped with hundreds of ECUs controlling vehicle components and functionalities. The de facto standard to allow these devices to communicate is the CAN bus protocol [21]. Officially released in 1986 by Robert Bosch GmbH, it is still employed in almost every vehicle in the world. Moreover, it finds applications in other domains such as Industrial Control Systems (ICSs) [65] and maritime vehicles such as ships [47]. Despite CAN clear advantages, such as high resilience to faults and the requirements of two wires only [21], it suffers from intrinsic security issues [5,29,32,36,56]. For instance, the absence of authentication allows effortless spoofing [20], while the fault-tolerant error handling protocol allows Denial of Service (DoS) attacks against specific vehicle components [7]. Tackling these issues with an easily deployable solution is tough due to the sensitivity of networks in safety-critical Cyber-Physical System (CPS), where availability is the main concern. Moreover, the diffusion of the CAN bus technology in the automotive industry and beyond makes it difficult for manufacturers to accept radical changes to the standard.

Over the years, researchers have proposed different solutions to fix security issues, although none are widely implemented at the current time. A class of mitigations employs cryptographic primitives to authenticate frames and hide transmitted information through encryption [12, 16, 17, 19, 31, 40, 57]. However, these approaches suffer from several drawbacks. While it is difficult to deploy authentication mechanisms supporting retro-compatibility with already employed devices [12], it is even more complicated to imagine imposing a standard that mandates legacy ECUs to manage cryptographic functions, effectively making all ECUs already on the market unusable. Moreover, the close source automotive environment makes it cumbersome to reverse proprietary frame formats and reduces the number of people that may get involved [45]. As the CAN bus serves as a safety-critical communication channel, it is imperative to control delays tightly. Cryptographic solutions typically influence performance, even if symmetric encryption can minimize them [69], thus presenting limitations.

To deal with the limitations of this environment, researchers started looking at it from different angles, implementing solutions working on top of the untouched bus. A classic approach is to employ an additional node to monitor the network and detect attacks [30]. It results in IDSs that monitor the format and content of frames to detect anomalies [11,53,60,64] or match signatures of known attacks [23,27]. Researchers proposed solutions employing voltage-based security systems that monitor the precise voltage each ECU imposes on the bus to identify the transmitter [51,68]. These solutions require no additional computational costs for ECUs but have been proven vulnerable to attacks [3, 50].

The inability to autonomously stop attacks is a considerable limitation of IDSs, which need to rely on external systems or human action. Countermeasures to attacks are often developed ad-hoc, presenting mitigations to newly discovered attacks [14,29,55,63]. However, applying several defense mechanisms could be expensive for a manufacturer, who may need to deal with incompatibilities. Moreover, it does not guarantee that the vehicle will be completely secure

from attacks. Another approach is represented by Intrusion Prevention Systems (IPSs), offering the capabilities to detect and react to attacks, preventing or stopping entire categories of threats. Unfortunately, there has been limited research investigating IPSs on the CAN bus [10, 33, 54]. Although attacks can be effectively prevented [43], this could be explained by the dangers of blocking actions incorrectly identified as attacks (false positives), which may endanger the safety of the vehicle and its passengers. Moreover, in the automotive scenario, false positives also represent a problem in a detection-only paradigm. Even a low number of false positives may result in alerts being ignored by drivers or several stops to have the vehicle unnecessarily inspected, deteriorating the reputation and trustworthiness of the manufacturer. Therefore, developing a comprehensive solution that is unaffected by detection error is essential in this field.

Most of the attacks researchers proposed are based on CAN protocol-compliant attacks, exploiting the bus structure and the error handling mechanisms [7, 20, 55]. Less attention has been directed to attacks exploiting actions that do not conform to the CAN specification. Connecting a malicious device to a vehicle's bus [67] gives the attacker complete freedom in their actions, even injecting single bits instead of entire frames [4]. Despite ad-hoc devices, even standard compromised ECUs may suffer from design flaws that make launching these kinds of attacks possible [29]. If the offensive side of these attacks has been little investigated, to the best of our knowledge, the detection of such attacks has seldom been discussed.

There is, hence, the need for a solution that overcomes current limitations. It needs to offer retro-compatibility without increasing the computational burden on legacy ECUs, whil at the same time achieving no false positives. This makes it difficult to employ Machine Learning (ML) models or other probabilistic approaches that are inherently open to rare but nonetheless present false alarms. Employing physical properties of the CPS allows the development of deterministic security solutions resistant to false positives. While an attacker can easily craft frames [20, 66], mimicking physical properties is at least harder than imitating network behaviors, if not impossible at all.

Contributions. In this paper, we bridge these gaps by proposing *CANTXSec*, the first deterministic Intrusion Detection and Prevention System (IDPS) in the CAN bus based on the co-presence of activity on the transmitting line of an ECU and one of its frame on the bus. Specifically, after building a list of message IDs in the bootstrap phase, they are linked to the ECU activated when each ID is transmitted in the bus. Then, we employ this list to detect malicious traffic. Our approach relies on deterministic comparisons, representing a unique and essential improvement over probabilistic IDS and IPS in the literature (e.g., ML-based [10, 37, 53]), virtually getting rid of any false positives. Moreover, while an extra effort related to wiring is needed, *CANTXSec* does not require any modification to the standard, resulting in a solution compatible with legacy devices. It achieves 100% detection and prevention of common attacks compliant with the CAN standard discussed in the literature while being able to precisely detect advanced attacks based on deviations from the protocol specifications.

Our contributions can be summarized as follows:

- We propose a new classification system to separate known and future CAN bus attacks between classical FIAs (i.e., using frame-level access) and advanced SBAs (i.e., employing bit-level access). The two classes represent different access levels by the attacker and can be employed to classify future IDSs in the field.
- We introduce *CANTXSec*, a deterministic Intrusion Detection and Prevention System (IDPS) in the CAN bus based on the co-presence of frame IDs on the bus and corresponding ECU activations. Our solution only requires cheap additional hardware, without any software modifications, cryptographic protocol, or time-consuming training processes, while preventing FIA attacks and detecting SBA, zeroing out false positives.
- Through the development of a real-world testbed, we demonstrate that *CANTXSec* achieves 100% accuracy in detecting both categories of attacks. Moreover, we prove that our system can prevent FIA with a success rate of 100% without disrupting other communications in the bus. The deterministic detection ensures that no legitimate frames are stopped, thus making *CANTXSec* applicable in commercial devices without risks.
- We provide an analysis to facilitate the implementation of *CANTXSec* based on each user's risk assessment. In particular, we show that monitoring the activation of 30% of the overall number of ECUs in a car ensures attack detection on the majority of safety-critical functions.

Organization. The paper starts by providing background insights in Section 2. Section 3 describes and categorizes attacks in the CAN bus, while Section 4 depicts the threat model we consider in our paper. Then, Section 5 describes *CANTXSec*, our IDPS, and discusses its implementation. Section 6 illustrates the testbed we employed to obtain the results we discuss in Section 7. Relevant related works are presented in Section 8, while in Section 9, we analyze *CANTXSec* with respect to other papers and discuss implementation details. Section 10 concludes the paper with some final insights.

2 Background

In this section, we discuss some background topics needed to understand the rest of the paper. We overview the CAN bus protocol in Section 2.1 and introduce the architecture of ECUs in Section 2.2.

2.1 CAN bus

CAN is a broadcast-based bus employed in CPSs and, in particular, in the automotive environment [21]. The bus works as a logical *AND*: dominant values (zeros) win over recessive values (ones). After transmitting a bit, each node senses the bus to ensure the transmitted value is actually on the bus. If not, a collision is identified.

CAN frames start with a Start-Of-Frame (SOF) bit followed by the ID, which is used to identify message content and as an arbitration mechanism to decide which node is allowed to transmit. Lower IDs correspond to higher priority. During the ID transmission, each node sends one bit at a time and senses the bus immediately afterward. If a node sending a one senses a zero, it loses the arbitration and thus stops transmitting. Through this mechanism, CAN guarantees that, beyond the ID, only one node is allowed to transmit.

Whenever a collision is identified after the ID, an error is raised. Every node keeps two error counters. A Trasmitter Error Counter (TEC) counts errors during transmission, while a Receiver Error Counter (REC) monitors reception errors. Every transmitting error increases TEC by 8, while receiving error increases REC by 1. Every correctly sent or received frame decreases the respective counter by 1. Upon encountering an error, the node signals it broadcasting an *error frame*.

Error frames have different aspects based on the *error state* of the node. Error states are defined by TEC and REC values. Nodes start in the *error-active state*. Once their REC or TEC exceeds 127, they enter the *error-passive state*. If the TEC exceeds 255, they enter the *bus-off state*, where they stop communicating with the bus.

When in active error state, error frames are called *active* and are composed of 6 dominant bits. On the other hand, during the passive error state, *passive* error frames are composed of 6 recessive bits. This represents the only occasion when a stream of 6 identical bits could be transmitted in the bus. During the transmission of other frames, this is not possible because of *bit stuffing* mechanism. It states that whenever a node sends five bits of the same logic level (dominant or recessive), it must send one bit of the opposite level. This extra bit is automatically removed by receivers. This process helps ensure continuous synchronization of the network.

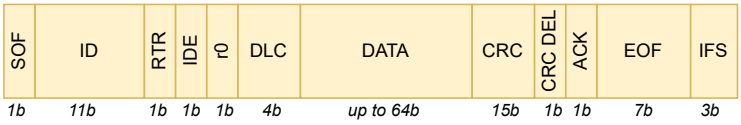


Fig. 1: A standard CAN frame showing the bit size of each field.

Figure 1 illustrates the different sections of a CAN frame. After SOF and ID, two control bits are sent: the Remote Transmission Request (RTR) indicates remote frame requests, while the Identifier Extension Bit (IDE) signals extended IDs. Then, a dominant reserved bit (r0) is sent. Follows the Data Length Code (DLC), a 4-bit value indicating the length of the data in bytes. Then, the actual content is sent, followed by a Cyclic Redundancy Check (CRC) and a recessive bit used as a delimiter (CRC DEL). Then, the transmitter sends a recessive Acknowledge (ACK) bit, during which receivers can confirm the reception of the

packet by imposing a dominant value. After the ID, this is the only bit where a node that lost the arbitration is supposed to send data. Finally, seven recessive bits called End-of-frame (EOF) conclude the frame. Before the start of a new frame, three recessive bits are sent and are called Inter-Frame Spacing (IFS).

2.2 Electronic Control Unit

An ECU is an embedded system employed in automotive electronics to control one or more of the electrical systems in a vehicle. It is usually connected to a variety of sensors and actuators. The process running on an ECU varies from simple and well-defined tasks (e.g., a brake system) to more complex and power-consuming operations (e.g., the infotainment ECU). The majority of ECUs need to communicate with each other to exchange information about the vehicle's state through the CAN bus. ECUs are therefore required to comply with the CAN standard [21] to allow this communication channel to work properly. A controller is usually employed as an interface between the ECU's microcontroller (MCU) and the bus, as shown in Figure 2. It implements the standard and is in charge of all the CAN related operations, such as buffering and queuing CAN frames that need to be transmitted, packet filtering, arbitration management, and error handling, including the management of error counters [35]. The controller can be included in the MCU Printed Circuit Board (PCB) or added as an external device [43]. In this latter case, the MCU employs general-purpose communication protocols (e.g., Serial Peripheral Interface (SPI)) to communicate with the controller. This approach allows developers to increase the range of MCUs they can employ, including boards without CAN bus support.

The logical output of the controller needs to be converted to the differential signaling employed by the CAN protocol. A transceiver converts the controller output (i.e., CANTX) to a signal compliant to the CAN standard [21]. Moreover, it reads bits in the bus and transmits values to the controller through the CANRX wire [34]. While it is possible to implement the controller as software [4], the transceiver is essential to convert digital values to signals that can be understood by the bus.

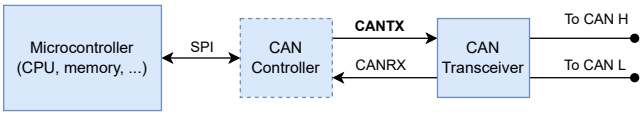


Fig. 2: The architecture of a standard ECU.

3 Attacks on CAN bus

Being employed in safety-critical systems, the CAN bus has been the target of numerous attacks, both from researchers and from malicious actors [7, 29, 36, 54,

Table 1: Attacks available in the CAN bus in the literature and effectiveness of detection (Det.) and prevention (Prev.) of *CANTXSec*. [1]: detectable for packets with spoofed IDs; [2]: starting from the first spoofed packet.

Attacks	**Type**	**Det.**	**Prev.**
Flooding/DoS [44]	FIA	✓	✓
Frame Spoofing [66]	FIA	✓	✓
Adaptive Spoofing [66]	FIA	✓	✓
Replay Attack	FIA	✓[1]	✓[2]
Original Bus-off Attack (BOA) [7, 9]	FIA	✓	✓
Stealthy BOA "single-frame" [55]	FIA	✓	✓
Error Passive Spoofing [13]	SBA	✓	✗
Double Receiving Attack [66]	SBA	✓	✗
Stealthy BOA "one-packet" [54]	SBA	✓	✗
Freeze Doom Loop Attack [66]	SBA	✓	✗
Shutdown via Clock Gating [29]	SBA	✓	✗
Selective DoS [43]	SBA	✓	✗

55, 66, 67]. Table 1 summarizes most of the attacks documented in the literature targeting CAN, indicating the type and the ability of *CANTXSec* to detect and prevent them.

In usual threat models in the literature [3, 5, 15, 26, 54], the attacker gains access to an ECU, which is employed to attack the bus. However, ECUs are not all the same. They can be deployed in different subnetworks while being connected to different sensors and actuators. Moreover, different ECUs are equipped with different capabilities that could allow attackers to compromise different parts of the system. The ECU's firmware and its hardware architecture are also part of the game: experienced attackers may find and exploit bugs in ECUs through reverse engineering that results in sophisticated access to the CAN bus, while unskilled attackers may rely only on high-level access to the bus provided by the ECU frontend. All of these factors are critical in assessing the types of attacks that can be carried out and, consequently, the effectiveness of identification and prevention methods. In this paper, we introduce a distinction between two fundamental attack classes in the CAN bus: *Frame Injection Attack (FIA)* and *Single-Bit Attack (SBA)*.

Frame Injection Attacks. FIAs include the most common attacks in the literature, where requirements on the compromised ECU are relaxed since the attack does not require any particular capability other than the ability to ask the controller to send frames. A malicious entity can obtain such access by compromising an ECU with malware [22], exploiting software bugs [36], or installing a malicious device on the bus [67].

Many attacks can be adapted from the IT scenario within this threat model. Flooding attacks with low IDs (i.e., high priority) slow down periodic frames and may lead to DoS [44]. Through frame spoofing, an attacker can send tampered information on the bus, possibly taking care of avoiding collisions with other

frames with the same ID using an adapting spoofing attack [66]. Network captures from the bus can also be replayed later and several times to hide malicious activities or masquerade other attacks performing a replay attack. Moreover, the original Bus-Off Attack (BOA) does not require any special access to the bus [7,9]. BOA exploits the error handling mechanisms to increase error counters in a victim ECU, pushing it to a bus-off state and effectively stopping the device from sending any frames. The main challenge is related to synchronization because the attacker's message must overlap with the victim's one. Researchers found different ways to synchronize the attacker device with the bus without any special access to the bus, such as exploiting the periodicity of CAN bus messages [7], the knowledge of the legitimate transmitter [9], or the preceding IDs [55].

Single Bit Attacks. If these kinds of ECU are the most spread, there exist cases where the attacker needs more *fine-grained access* to the bus, monitoring it not frame-by-frame but bit-by-bit. This can be the case when the adversary needs precise synchronization in the bus and the ability to inject single bits during other ECU transmissions, possibly disregarding the CAN bus specification [21]. Different ways exist to achieve this: 1) installing a malicious device developed ad-hoc without a controller, 2) bypassing or hacking the controller of an ECU [29], or 3) obtaining access to an ECU which does not employ a controller and which, therefore, allows fine-grained access to the bus. Apart from the installation of a malicious device, it is worth noticing that the effort needed to obtain this level of access is considerable, and it is not always possible. While it is theoretically feasible with some MCUs [61] that allow multiplexing the CAN controller output in standard GPIO pins, it is more complex if the output pins are not reconfigurable or the controller is connected via serial connections (e.g., SPI). Other strategies may be possible by exploiting wrong design choices or by interfering with advanced ECU functions such as the clock [29]. We call attacks requiring such an access SBAs since, usually, this control on the bus allows attackers to generate errors by injecting a *single bit* in specific instants during the transmission of frames by other ECUs [29, 54, 55].

With such precise access, many other advanced attacks are possible. For instance, Error Passive Spoofing [13] is a two-stage attack that requires 1) forcing an error passive mode victim into generating an error frame and 2) replacing the recessive data and CRC bits with the spoofed payload. A Double Receiving Attack forces the retransmission of a frame by imposing a dominant value in the last EOF bit [66]. Moreover, direct bus access enables stealthier versions of already discussed FIAs, such as the one-packet BOA [54] or the Selective DoS [43]. A legacy feature of the CAN bus offers a technique to decrease the bandwidth of the bus and gain time to finish computations for slow ECUs. It works by imposing a dominant value in the first IFS bit [66]. An adversary may exploit it with a Freeze Doom Loop Attack, forcing the dominant bit in a frame and then again to the consequent overflow error frame that will be generated. A peculiarity of these kinds of errors is that they do not increase error counters, making the attack stealthier while keeping the bus busy as long as the attack is

ongoing. Kulandaivel *et al.* [29] proposed an attack to shutdown ECUs exploiting design errors of certain ECUs in the wild.

Since our approach is tied to the physical aspects of the attack, we decided to separate attacks into FIAs and SBAs. As extensively described in Section 5, *CANTXSec* can detect both kinds of attacks, but in two different ways strictly depending on the category. On the other side, using our approach, prevention is possible for the former category only. Moreover, this categorization is important for future work on risk assessment. Even though it may happen that SBAs have more severe effects, they are usually less likely to happen because of the challenging environment needed to carry them out.

From our classification, we intentionally left out *modification attacks*, where an attacker compromises an ECU and uses it to replace the content of legitimate frames, i.e., without spoofing another ECU's ID. Being an attack on the data level and not on the network level, it is independent of the communication protocol employed and, therefore, not in the scope of this paper. In addition, it is worth noting that attackers usually exploit vehicles through over-exposed ECUs, such as the infotainment system, which in normal scenarios is not required to send critical messages on the bus. Therefore, to create effective damage, attackers usually need to forge message IDs to spoof the identity of sensitive ECUs. To defend the system against these attacks, various techniques exist in the literature, even directly targeting CPS [38] or vehicle [6] environments. Promising techniques employ lightweight feature extraction and contextual information to detect anomalies in transmitted data [2, 24].

4 Threat Model

In this paper, as common in the CAN bus security literature [3, 5, 15, 26, 54], we consider a malicious attacker with the capabilities to compromise ECUs software remotely (e.g., via unsecured over-the-air firmware update mechanisms [48], via Internet [36]) or physically (e.g., exploiting insecure OBD-II ECU software update mechanisms [42]). We do not consider an attacker able to swap a legitimate ECU with a malicious one, while we increase the attacker's strength including physical attackers able to connect new malicious devices to the bus [67]. Furthermore, compared to similar recent works [54], the attacker can hack or bypass the ECU's controller to act on the transceiver directly and thus on the bus. This is trivial for a physical attacker but also possible for remote attackers [29]. However, this advanced threat model has some consequences on which attacks can be detected and prevented, as we will discuss in Section 9.2.

5 *CANTXSec*

In this paper, we propose *CANTXSec*, the first IDPS on CAN bus, which employs fine-grained measurements from ECUs activities to detect and prevent network attacks in the bus. *CANTXSec* collects data on the CANTX pin of each ECU

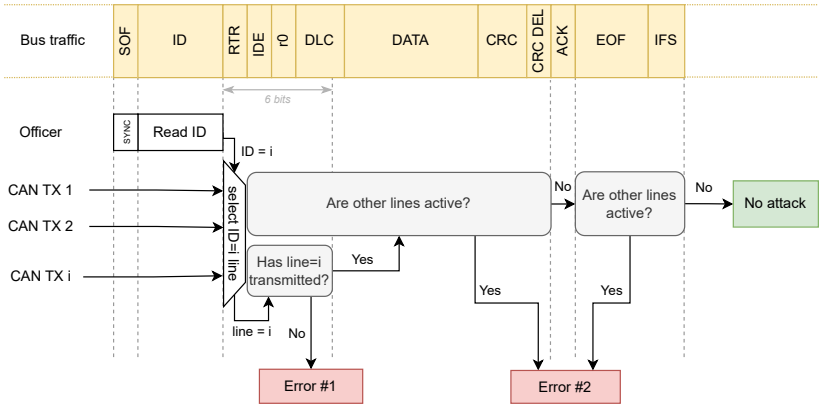


Fig. 3: The flow chart explains the different checks *CANTXSec* performs during each bit of every frame sent through the bus.

and correlates these values with a bit-by-bit reading of the bus. When a malicious communication is detected originating from a compromised ECU and stopping the frame is safe, *CANTXSec* blocks the transmission of the message by generating errors, eventually forcing the compromised ECU to a bus-off state. The architecture of a system employing *CANTXSec* is depicted in Figure 3. A new device, the so-called *officer*, is connected to the bus to detect and react in case of attacks.

5.1 Officer

An officer with bit-level access to the bus is the core component of *CANTXSec*. It should be an MCU connected to a CAN transceiver. For the officer, a CAN controller is unnecessary since the device should be able to take action on frames almost in real-time without waiting for the end of the packet. To this aim, the officer MCU demodulates frames on the fly at the software level. A transceiver is, hence, everything required to enable the MCU to interface with the bus if not already provided by the board.

Moreover, the officer is connected via a single dedicated wire to the CANTX lines of all the monitored ECUs to detect their activities. This wire is only intended to measure digital voltage levels and signal the activation of the CANTX line to the officer. As depicted in Figure 2, the CANTX line represents the connection between the controller and the transceiver or, if the controller is not present, between the transceiver and the MCU. The CANTX line shows a high value when either a recessive bit needs to be sent on the bus or when the transmitter is idle. It instead shows a low value when a dominant bit should be imposed on the bus.

5.2 Setup

The first part of the *CANTXSec* setup phase requires physical access to the vehicle and its components. In fact, we need to connect a wire from the officer to the CANTX of all the ECUs to be protected. A perfect time to perform the setup is during automobile manufacturing while assembling the vehicle, to make it easier to reach the various ECUs and to ensure that no stealthy attacks are ongoing (e.g., malicious ECU spoofing IDs during setup). However, a technician can also perform the setup a posteriori, ensuring as much as possible a clean state of the ECU software or relying on manufacturer specifications to collect the legitimate IDs. All the ECUs must be wired to protect against the highest possible number of attacks. However, a lot of attacks can be prevented by connected ECUs even without having all the devices connected to the officer, as we discuss in Section 9.2. Based on the risk assessment conducted on each vehicle, the developer and/or the owner can decide how many and what ECUs should be protected [39].

Each CANTX pin connected to the officer represents a unique ECU. For each ECU, the officer maintains a list indicating which IDs are allowed to be transmitted from that particular ECU. The second step of *CANTXSec* setup is hence the creation of this list. The manufacturer can populate the list by manually inspecting the specifications. Otherwise, it can be achieved by running the automobile in a controlled environment for a sufficient time to ensure that all relevant ECUs activate at least one time. In the end, the officer should maintain a table where each ECU pin corresponds to a list of allowed IDs. As discussed in Section 2.1, each ID is strictly linked to a particular ECU, and there should never be two ECUs with the same ID in the bus, as per standard [21].

With respect to other IPSs [10], *CANTXSec* only requires this setup step to start working properly without any time-consuming data collection and training of ML models. Updates on the vehicle, such as the replacement or installation of a new ECU, require connecting the appropriate cable to the CANTX line and updating the list on the officer. These kinds of actions are straightforward for a technician, and no additional expertise is needed, making *CANTXSec* very easy and fast to upgrade.

5.3 Attack Detection

The intuition on how *CANTXSec* works is as follows. Consider a frame with arbitration ID = i being transmitted on the bus. For each transmitted bit, we assess whether 1) the correct ECU (i.e., the one associated with ID i) is transmitting, and 2) all the other ECUs are quiet. Implementing this approach using loops is inconvenient due to the real-time nature of the problem. A better solution involves the employment of interrupts. They are available in modern MCUs and can be triggered by a voltage change in a pin [62]. It is worth recalling that the CANTX lines are set to 1 while the connected ECU is in idle state. This makes it more difficult to assess if a certain ECU is transmitting a 1 (recessive value) or waiting. However, because of bit stuffing [21], a 0 should be transmitted

periodically, even if not included in the original message. This makes it possible to detect attacks using edge interrupts with reasonable delays and always within 6-bit time, as discussed in Appendix A.1. These interrupts fire when a voltage variation is registered in the line, both from a high value to a low value and vice versa.

The SOF is used by the officer for synchronization, as shown in Figure 3. During the arbitration, *CANTXSec* is silent, waiting to know which ID wins the arbitration. After arbitration, only the CANTX of the winner ECU is authorized to transmit dominant values (except for the Acknowledge bit, as discussed later). The officer performs the first check after 6 bits following the end of the arbitration ID. In that period, the CANTX line i should have transmitted at least a 0 (because of bit-stuffing, as detailed in Section 2.1). Otherwise, *CANTXSec* raises an Error #1. The second check starts again after the arbitration ID but continues up to the end of the packet, checking if some of the other CANTX lines (i.e., all except for CANTX line i) are transmitting a dominant value. If so, *CANTXSec* raises an Error #2. This ensures that no ECU except the one who wins the arbitration sends bits in the bus.

As depicted in Figure 3, there is one bit that is not considered in this check. In fact, the ACK bit is meant to be set dominant by other ECUs to acknowledge the reception of the frame. Therefore, the check for Error #2 is paused during that bit since other ECUs are allowed to transmit. It is also worth mentioning the existence of a legitimate behavior where the first IFS bit is set to zero to signal an overflow. This was used in legacy systems to allow a receiver to delay the next message while executing computation on the previous message [21]. However, it is no longer employed today and can be exploited to launch a Doom Freeze Attack on the bus [66]. Therefore, we do not consider this possibility in our system. However, it can be easily implemented by excluding the check for Error #2 during that bit.

Compared to other IDS and IPS in the literature, *CANTXSec* does not rely on statistical or ML models to detect and prevent attacks. Instead, it is based on *physical measurements* and *co-presence* between ID values and corresponding ECU *activation* in real-time. This ensures a deterministic solution that leads to perfect attack detection.

5.4 Attack Prevention

When the officer detects an attack, it can act in two different ways. One option is just to detect it and alert the driver, who can then decide what action to take. Another option is instead to stop the frame and, therefore, the attack. If detection cannot prevent attacks in real time, it allows alert verification and avoids blocking legitimate packets when false positives are detected. This is one reason why such systems have not found huge implementation in the literature: even if the false positive rate is usually low [51, 53, 59], it may have an impact on the driving experience. *CANTXSec*, instead, reaches 100% accuracy without false positives: this allows us to safely implement a prevention mechanism, being sure not to compromise the reliability of the bus.

To stop a frame on the bus, the officer *injects a dominant value until a recessive bit is overwritten*, which happens at most every 6 bits thanks to the bit stuffing mechanisms (see Appendix A.1). This generates an error on the transmitter, which stops the frame transmission and starts sending an error frame. The error frame depends on the ECU state. In error active mode, it is composed of 6 dominant bits, while in passive mode, it comprises 6 recessive bits. However, even if this is enough to stop the frame [10], it may not be the smartest strategy to block the attack completely. By default, controllers encountering a transmission error will reschedule the same packet again in the next frame time, which will be stopped by the officer again [35]. Such a process increases the busload, and the attacker may exploit this behavior to slow down the bus, performing a DoS.

A more efficient strategy aims to force the compromised ECU into a bus-off state. While there exist different ways to perform the so-called *bus-off attack* [7, 55], the most efficient is the *Instant Bus-Off* proposed by Serag *et al.* [54]. It forces an ECU into a bus-off state, targeting a single packet in $510\mu s$ on a $500kbps$ bus. The idea behind this attack is to target one packet with an error and then target the error frames generated by the first error injection. By iterating this process, error counters on the compromised ECU will increase, eventually resulting in a bus-off state. More details are available in the paper [54].

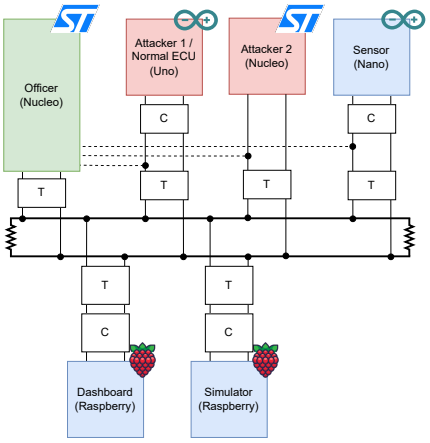
While FIAs can be easily stopped with this strategy, SBAs cannot. SBAs generate errors while transmitting of another frame by a legitimate ECU, violating the CAN protocol [21]. Since, when the attack is detected, the frame being transmitted is the legitimate one, the same strategy cannot be applied. Otherwise, it will target the legitimate ECU. For this reason, the safest solution is only to detect the attack and notify the user, who should decide the most suited action based on the context. Discriminating between attacks to be stopped and attacks to be detected is easy since Error #1 is related to FIAs that can be prevented, while Error #2 indicates that a user action is required to avoid jeopardizing the vehicle's safety.

5.5 Development

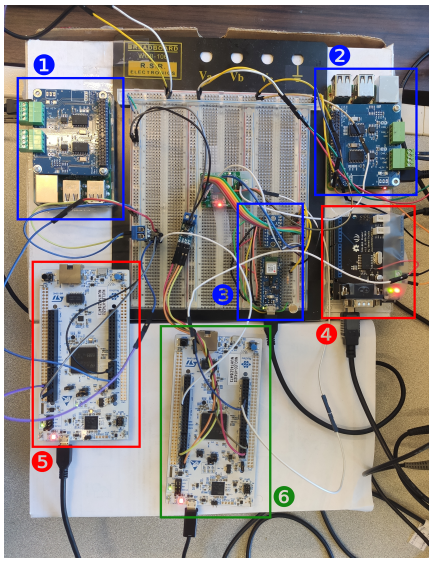
A suitable officer to implement *CANTXSec* should reproduce the logic summarized in Figure 3. Since no ML or other computationally expensive calculation is involved, a cheap MCU is enough as long as it provides edge interrupts and enough GPIO pins to connect all the monitored ECUs. Precise analog voltage measurement capability by GPIO pins is not needed since the officer only needs to obtain the digital level of CANTX lines. Moreover, the officer should include a connection to the bus, which allows reading each singular bit in real-time (i.e., without a controller). Because of the controller's absence, the software is responsible for decoding each frame ID and deciding which CANTX lines should be monitored. To identify errors, a computationally lightweight approach includes the employment of interrupts to detect changes in the CANTX lines connected to the officer's GPIO pins. Errors and prevented attacks should be logged and reported to the user so that further analysis can be performed. The code of our

prototype and all the devices employed in the testbed are available in our Github repository[4].

6 Testbed



(a) The officer is connected to the CANTX of the monitored ECUs, which is the wire connecting the transceiver (T) and the controller (C), when available.



(b) ❶: Dashboard; ❷: Simulator; ❸: Sensor; ❹: Attacker 1 / Normal ECU; ❺: Attacker 2; ❻: Officer.

Fig. 4: The architecture (Figure 4a) and picture (Figure 4b) of the employed testbed.

To test *CANTXSec*, we developed a testbed employing different components to simulate a real CAN bus as closely as possible. We employed different microcontrollers and microprocessors to create a more realistic environment. The testbed schema is depicted in Figure 4a, and the use of each component is described in the following.

– **Officer**: an STM32 Nucleo H743ZI2 board [61] employed to run *CANTXSec*. It is connected to the bus via a transceiver without any CAN controller and to the ECUs under control through GPIO pins.

[4] `https://github.com/donadelden/CANTXSec/`

- **Attacker 1/Normal ECU**: an Arduino Uno [1] with Seeed Studio shield [52] for the connection to the bus employed as a data transmitter. During the frame spoofing attack, this ECU is considered compromised. Otherwise, it behaves as a normal ECU broadcasting a random value every 100 milliseconds. The Seeed Studio shield included both a transceiver and a controller.
- **Attacker 2**: an STM32 Nucleo H743ZI2 board [61] emulating an ECU with direct access to the bus (i.e., not mediated by a controller) through a transceiver. It represents the optimal target for an attacker aiming to perform a SBA.
- **Sensor**: developed with an Arduino Nano 33 BLE [1] acts as a sensor broadcasting a physical parameter (the light received by a photoresist) every 100 milliseconds. It represents one of the many sensors a modern vehicle contains to monitor different vehicle and environmental parameters (e.g., the temperature of the engine, external temperature, and tire pressure).
- **Dashboard**: running in a Raspberry Pi 3b [49] with a PICan Duo hat [58], it emulates a dashboard, essentially receiving and logging the traffic on the bus. The shield includes both a controller and a transceiver.
- **Simulator**: running in a Raspberry Pi 3b [49] with a PICan Duo hat [58], it created background noise on the bus by continuously sending traffic from a dataset. The dataset has been created by collecting 10 minutes of traffic while using an instrumentation cluster simulator [46].

All the devices are connected to a simulated bus with terminal 120Ω resistors as mandated by the standard [21]. Figure 4b illustrates the final testbed with all its components.

7 Results

Even if the attacks presented in Table 1 have different goals, we can summarize them into two categories based on the physical mechanisms needed to perform the attack successfully, as described in Section 3. Since *CANTXSec* is focused on the bit level, the end goal of the attack is not relevant to the detection. In other words, independently of the final target of the attack, our system is able to identify malicious injections that compose the attack. Therefore, we implemented some representative attacks using different technical methodologies, which we believe are comprehensive for the majority of the attacks in the literature.

Table 2 summarizes the attack's scores and *CANTXSec*'s accuracy in detecting and preventing attacks. The Attack Success Rate (ASR) is measured depending on the attack's goal, and it is explained in the following sections. The dashboard collects frames successfully sent through the bus while all the relevant ECUs log the packets transmitted for further analysis and correlation. We ran each experiment for at least 10 minutes, generating several attacks during that timeframe. Furthermore, Appendix A.2 proposes a simple example showing how *CANTXSec* can effectively block an attack and restore the system's normal behavior.

Table 2: Effectiveness of *CANTXSec* in detecting and preventing the implemented attacks, which are presented with their Attack Success Rate (ASR). Preventing Selective DoS is not possible with this approach.

Attack	Type	ASR	Detection	Prevention
Frame Spoofing without traffic	FIA	100%	100%	100%
Frame Spoofing with traffic	FIA	100%	100%	100%
Selective DoS	SBA	100%	100%	NA

7.1 Frame injection

As a representation of FIAs, we implemented two different frame spoofing attacks, considering that the spoofed ID is also transmitted from another ECU or is only sent by the attacker's device.

Frame spoofing without legitimate traffic. In this scenario, the attacker is able to stop a monitored ECU from sending data. For instance, they can exploit a BOA [7] or use malware to compromise the ECU [22]. Moreover, the attack has the control of an ECU through which they can send frames on the bus with arbitrary ID. We measure the ASR as the percentage of malicious frames transmitted by the attacker that are correctly received by the dashboard. An attack of this kind has a 100% ASR since there are no countermeasures applied by default against spoofing attacks.

Without detection or defense mechanisms, the recipient of the ID would not notice a difference in the packets since the ID is spoofed. If the officer is activated in detection mode, instead, the driver could be notified of every spoofed packet sent on the bus. In the testbed, this happens for 100% of the attacks, matching with theory since attack detection is deterministic, as explained in Section 5.

Furthermore, frame spoofing attacks can be prevented by setting the officer into prevention mode. In this case, when a malicious packet is sensed in the bus, the officer will launch a BOA against that ID and, therefore, against the compromised ECU. Figure 5 shows on an oscilloscope the prevention mechanisms firing against a malicious frame. Even in this case, the success rate is 100%, and no malicious packets are ever completely sent in the bus. Thus, the receiver does not receive any spoofed data.

Frame spoofing with legitimate traffic. Similarly to the previous attack, the malicious actor has compromised an ECU through which they can send spoofed frames with any arbitration ID. Compared to the previous scenario, this time, the attacker did not stop the legitimate ECU from sending frames on the bus. Therefore, if the attacker sends packets continuously without caring about what is happening on the bus, there may be collisions with the legitimate packets. However, a smart attacker can easily sense the bus and send packets just after legitimate ones [66]. With these techniques, they can ensure a 100% of ASR, which is measured as the percentage of malicious frames correctly received by the dashboard.

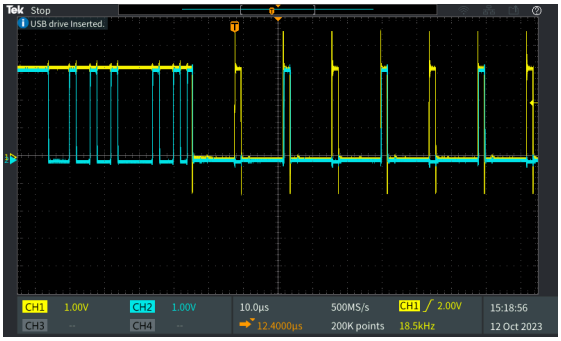


Fig. 5: Start of a BOA as seen in an oscilloscope. The blue signal is the target frame (i.e., the malicious frame). The yellow signal represents the injection made by the officer. The first injection of a dominant value stops the malicious frame. Then, the following injections target error frames automatically generated by the attacker's ECU. This increases the attacker's ECU error counters, eventually reaching the bus-off state.

Detecting this attack may seem more challenging since the officer needs to differentiate between malicious and legitimate packets with the same ID. However, since the officer knows the state of the CANTX of the monitored ECUs, it can tell the source of a frame. With this strategy, during detection mode, 100% of the malicious packets are detected, while 0% of the legitimate packets are wrongly classified as malicious.

In this scenario, prevention shows the same difficulties as detection. When the officer is in prevention mode, it can block all the malicious frames, pushing the attacker's ECU into the bus-off state without modifying the behavior of legitimate frames that are received by other ECUs correctly. In our experiments, *CANTXSec* got a success rate of 100% in stopping malicious frames.

7.2 Single bit access attacks

If detecting and partially preventing frame spoofing has already been investigated by other works, SBAs are intrinsically more challenging to detect. These kinds of attacks bypass the controller with different techniques, for instance, using a modified ECU, hacking the controller, or exploiting design errors (e.g., bus clock gating [29]).

As a representative example of these attacks, we implemented the *Selective DoS*, first proposed by Palanca *et al.* [43]. Usually, these kinds of attacks exploit the generation of errors in the bus to maliciously stop a packet or increase error counters in ECUs. Since these bits are injected during the transmission of legitimate packets, trying to stop them by launching BOAs is counterproductive for two reasons. First, this will likely stop the legitimate frame and possibly help the attacker by increasing even more victims' error counters. Second, to control single bits in the bus, the attacker has probably bypassed the controller

in some way and, therefore, they no longer need to behave to the standard. For these reasons, we just investigate the detection of these kinds of attacks since prevention is not possible with our strategy. After detection, the victim should stop the automobile and ask for assistance to identify and fix the problem.

The Selective DoS [43] works by injecting a dominant value during the transmission of a recessive value, thus generating an error that stops the packet transmission. The transmitting ECU reacts by sending an error frame and rescheduling the transmission of the packet as soon as possible. Then, the attacker uses the same strategy to stop the retransmitted frame, and the cycle begins again. In some cases, the victim ECU may go into the bus-off state, completely stopping packet transmissions.

We implemented the attack on a Nucleo H743ZI2 board [61] (Attacker 2 in our testbed) by monitoring the bus for frames with the target arbitration ID and then injecting a dominant value during the first transmission of a recessive bit. To check the effectiveness of the attack, we monitor the reception of packets from the dashboard while launching the attack. We measure the ASR as the percentage of packets correctly stopped by the attacker. We assessed that no packets are received when the attack is active, indicating an ASR of 100%.

The detection of Selective DoS goes through the identification of injections of single bits after the transmission of the arbitration ID. Since, by design, the attack will generate a lot of consecutive traffic (i.e., the retransmissions), the officer MCU is not fast enough to print an alert for each message. Therefore, we employed a different strategy. We program the attacker to target exactly 200 frames (regardless of whether they are first sends or retransmissions), and we program *CANTXSec* to alert every 200 detection. We avoid notifying each attack on the serial port since this action is time-consuming compared to the attack frequency, and it may result in some attacks passing unnoticed during the notification. By repeating this process several times, we were able to ensure that all the attacks were identified by *CANTXSec*, thus reaching a 100% detection rate for SBA. In a production environment, there are different strategies to mitigate this issue. First, a faster MCU may be employed, possibly implementing the logic on a Field Programmable Gate Array (FPGA) to maximize performances. Second, another MCU may be delegated to collect alerts and queue the printing of errors to the driver. Third, since getting the precise number of attacks is usually not essential, the driver may be alerted only for the first attack received.

8 Related Works

IDS on CAN bus. Plenty of works in the literature propose to detect attacks in the CAN bus by analyzing frame content employing ML and Deep Learning (DL) models [53, 60]. Different models have been employed, such as Random Forest [37], CNN [11], GAN [53] and LSTM [60]. GIDS [53] is an IDS proposed by Seo *et al.* [53] together with a dataset including DoS, fuzzing (i.e., spoofing of packets with random content and ID), and modification. A GAN is trained with benign data and used to detect attacks with 98% of accuracy. The same

dataset is also used by Song *et al.* [60] to test an IDS based on different DL models achieving higher accuracy.

Other approaches have been discussed in the literature as well. Five different features have been used by Xiu *et al.* [23] to detect spoofing attacks, including the ID, time interval between packets, and three features related to the frame content (correlation, changing amplitude, and value range). Other works [51,68] try to fingerprint ECUs based on the voltage level imposed on the bus. However, these mechanisms exploit an average over different measurements during the transmission of the same packet, making them unreliable when dealing with SBAs. Moreover, they have been proven vulnerable to attacks [3, 50].

Time intervals are another feature employed in the literature to detect attacks in the CAN bus [23, 59, 64]. However, these techniques have some limitations and cannot provide protection against the injection of single bits during the transmission of legitimate frames (i.e., SBAs).

IPS on CAN bus. Because of the bus nature of the CAN protocol, IPSs are not frequent in the literature. The first to theorize the utilization of the error handling mechanism to prevent the reception of malicious messages have been Matsumoto *et al.* [33]. They theorize a modification of ECUs to include a flag to be set when transmitting so as to enable a security device to send an error if someone else is transmitting in the same period. However, a software and hardware modification of each ECU is requested, together with the wiring cost of flags and the extra device. It is suitable for new networks and devices but not for legacy systems. Moreover, the authors did not tackle the implementation challenge. De Araujo-Filho *et al.* [10] propose an IPS based on an unsupervised Isolation Forest, which requires some unlabelled training data without attacks in it. Moreover, through the injection of error frames, malicious packets can be stopped. However, only fuzzing and modification attacks are discussed without including any SBA. A different solution is called Parrot [9], a spoofing prevention system based on the fact that the legitimate ECU being spoofed can easily detect it and launch a bus-off attack against the attacker. However, other attacks, such as SBAs, are not discussed, and some limitations hold, for instance, related to the speed of the CAN transmitter and the impossibility of preventing the first spoofed frame. Another approach has been proposed with ZBCan [54], a security solution able to authenticate messages exploiting the intra-packet time and stop attacks using a SBA. However, the implementation requires a new device and a modification of ECUs' software to include authentication management. Moreover, SBAs are discussed only as a countermeasure, and the system cannot detect or prevent them.

9 Discussion

In this section, we discuss some aspects of our work. We compare *CANTXSec* with other approaches in the literature in Section 9.1. Then, we investigate the ECUs coverage requirement in Section 9.2 and discuss limitations in Section 9.3.

9.1 Comparison with other works

Despite the huge amount of IDS in the literature, only a few papers describe an effective prevention system that can block attacks in the CAN bus [10, 54], as described in Section 8. Table 3 summarizes the capabilities of our work with respect to other relevant papers in the literature. Detection of FIAs is usually one of the main targets of papers and is thus always satisfied, even if with differences. For instance, papers such as GIDS [53] and Song *et al.* [60] employ ML models to detect attacks. However, they only consider certain specific attacks (i.e., DoS, Fuzzy, Gear Spoofing, RPM Spoofing) that were included in their datasets. Scores vary based on the detected attack, with precisions reaching 99.9%. Another approach employing voltage levels, VALID [51], only reaches 99.5% of accuracy and has been proven vulnerable by certain attacks [3, 50]. Detection of SBAs is, instead, a completely new topic, and, to the best of our knowledge, *CANTXSec* is the first paper in the literature addressing this issue.

Moreover, our solution is also able to stop attacks, which is instead a topic discussed in just a couple of other works. Compared to our solution, both Parrot [9] and ZBCan [54] require software modification of each ECU, which may be cumbersome when dealing with proprietary software in the automotive scenario. On the other side, the hardware modification required by *CANTXSec* can be applied on every ECU by inspecting the PCB for the CANTX line, which is easier than dealing with integrity protection and digital signatures applied in all modern ECU firmware [25, 28]. Moreover, both approaches can only detect FIA, while SBAs will pass undetected without *CANTXSec*. While both ZBCan and *CANTXSec* are based on almost deterministic solutions, the approach discussed by De Araujo-Filho *et al.* [10] includes training of ML models, resulting in increased setup time and need for computation resources. Finally, it is worth mentioning the development of secure CAN transceivers [41] providing certain security measures such as spoofing and flooding protection. However, they cannot protect against threats that are transparent for the transceiver, such as SBAs, which are still possible with such devices.

Table 3: Our solution compared to other IDSs and IPSs in the literature. $\sim$ indicates a lightweight not-ML training process.

	Detection		**Prevention**		**Training**
	FIA	**SBA**	**FIA**	**SBA**	
GIDS [53]	✓	✗	✗	✗	✓
Song *et al.* [60]	✓	✗	✗	✗	✓
Xu *et al.* [68]	✓	✗	✗	✗	$\sim$
VALID [51]	✓	✗	✗	✗	$\sim$
Jin *et al.* [23]	✓	✗	✗	✗	✓
De Araujo-Filho *et al.* [10]	✓	✗	✓	✗	✓
Parrot [9]	✓	✗	✓	✗	✗
ZBCan [54]	✓	✗	✓	✗	✗
CANTXSec	✔	✔	✔	✘	✘

9.2 ECUs coverage

The most significant requirement of *CANTXSec* is the need to wire the ECUs to the officer, which is somehow a limiting feature in a CPS like an automobile, while it can be more easily applicable in ICSs or ships. However, it is worth noting that not all the ECUs in a vehicle have the same risk model, and *CANTXSec* can be easily implemented and deployed to cover only the most critical ECUs. For instance, it is essential to protect safety-critical ECUs (e.g., brakes) to ensure reliable operation and to avoid malicious sudden actions that might harm the driver. On the other side, devices such as the infotainment system are less critical and, if compromised, cannot physically harm the passengers. An iteration of such an analysis prioritizes the ECUs that most require protection.

The number of ECUs in a vehicle is quite variable. Cheap automobiles are equipped with only a bunch of them, while luxury cars may have up to 150 ECUs [18]. Accessing information about the number of ECUs in a vehicle and the related messages is tough because of the closed source of the automotive environment. According to openDBC [8], a collection of reverse-engineered database of CAN messages for different vehicles, they have a mean of 35 ECUs, even though the number is quite variable, and databases are not always complete. Usually, each ECU sends various messages through different frame IDs, indicating the content of the message and its priority. Since, in the worst-case scenario, attacks target the entire ECU (e.g., bus-off attacks will stop the ECU from sending any message), in our analysis, we considered only the lowest ID for each ECU.

Even though the repository [8] does not offer a complete collection of vehicle messages and ECUs, we analyzed the most complete dataset to estimate the number of ECUs *CANTXSec* should be connected to in different scenarios. Since the ID indicates the priority of the frame, we suppose that lower IDs correspond to safety-critical messages, while high IDs are reserved for infotainment and other non-critical applications. Therefore, to roughly estimate the number of safety-critical ECU, we sort them by IDs and threshold them with the first clearly not-critical ECU. A utility automobile such as a Hyundai Kia contains 43 ECUs. Out of them, we identify the 12 ECUs with lower IDs to be considered safety-critical, using as a threshold the ID of the parking assistance ECU, which can be considered non-critical. More expensive vehicles, such as BMW E9X, contain slightly more ECUs (50). We identify 14 of them as safety-critical, using as a threshold the IDs adaptive front lighting system, which represents the set of the most important ECUs to be covered by *CANTXSec*. The effects of limited coverage are discussed deeper in Appendix A.3.

Based on the simple risk assessment we propose, or others in the literature [39], each vehicle owner can decide how many ECUs should be covered by *CANTXSec*. Just by connecting less than 30% of ECUs to the officer, it is possible to protect safety-critical ECUs from FIAs from both remote and physical attackers. If more than one CAN bus is deployed in a vehicle, another solution could be to cover only the ECUs connected to the safety-critical buses (e.g., the power train), leaving unprotected less essential networks (e.g., the infotainment systems). While this could be the most widely adopted scenario, high-risk usages

may require that all the ECUs should be covered with *CANTXSec* to ensure full detection of both FIAs and SBAs.

9.3 Limitations

This work introduced for the first time a novel strategy to reduce the long-discussed issue of false positives in the field of attack detection for CPSs and propose the first strategy to eradicate them. However, some limitations should be addressed in future works.

First, the requirement to cover all the ECUs with *CANTXSec* to comprehensively detect all the discussed attacks may restrain some producers who could see production prices and vehicle weight rise. However, we offer the reader an analysis to consciously decide how to deploy the system based on the personal risk assessment. Moreover, the most common and straightforward attacks (i.e., FIAs) are detected and prevented with 100% accuracy just by connecting to *CANTXSec* the most safety-critical ECUs, relaxing the full wired requirement for the standard user while providing the possibility to increase security for high-risk cases.

Another weakness is that the capability of stopping attacks is limited to FIAs. SBAs represent a huge challenge for prevention since they may be effective after a single injected bit, rendering the time frame to act and stop the attack virtually zero. Mitigation strategies could include firmware and architecture hardening to prevent single-bit access to the bus. Another approach could be to physically prevent ECUs other than the one winning the arbitration from transmitting on the bus when not allowed, for instance, forcing the CANTX pin to a high value. Moreover, forensics tools may be deployed to secure the system after the attack.

Finally, this research focuses on network-level threats, leaving aside data-level attacks such as modification attacks. *CANTXSec* correlates IDs on the bus with ECUs activations to spot attacks without looking at the actual data transmitted. Due to this architecture, modification attacks are not possible to be detected. Further research should be conducted to port the proposed deterministic approach to the data level.

10 Conclusions

In this paper, we bridged the gap in the detection and prevention of attacks in the CAN bus by proposing *CANTXSec*. Our solution is 1) deterministic, 2) covers advanced attacks never discussed in the literature before, and 3) does not require any software modification of ECUs and is thus easier to implement. To test and analyze its characteristics, we introduced a novel categorization of attacks in the CAN bus based on the access an attacker needs to carry them out. In fact, we introduced FIAs requiring typical frame-level access and SBAs employing more sophisticated bit-level access. Through the usage of a physical testbed we developed, we demonstrated that *CANTXSec* achieves 100% accuracy

in detecting both FIAs and SBAs. Moreover, we demonstrate how our solution is even able to prevent FIAs with 100% accuracy.

Future works include further tests through the deployment of *CANTXSec* in a real vehicle and augmenting the number of connected devices. Such an improved testbed will allow for new experiments to test performance and latency under high-load conditions (e.g., heavy traffic on the bus). Increasing the number of ECUs could also allow testing the practical effects of partial ECUs cover by *CANTXSec*. The development of such a testbed could also allow considerations about the complexity of deploying *CANTXSec*, such as evaluating the ease of wiring the CANTX lines of legacy and modern ECUs. Moreover, this work opens a new research direction aiming at the prevention of the newly introduced category of threats we called SBAs, which is a topic not addressed in the literature up to now.

Acknowledgments This work was supported by the European Commission under the Horizon Europe Programme, as part of the project LAZARUS (`https://lazarus-he.eu/`) (Grant Agreement no. 101070303). The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors. This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. This research was also partially funded by the Research Fund for the Italian Electrical System under the Contract Agreement "Accordo di Programma 2022–2024 between ENEA and Ministry of the Environment and Energetic Safety – Project 2.1. Moreover, this work has received support from grants N00014-23-1-2386 from US Office of Naval Research and CNS-2153136 from US National Science Foundation.

References

1. Arduino: Arduino: open-source electronic prototyping platform enabling users to create interactive electronic objects. `https://www.arduino.cc/` (February 2023)
2. Balasubramanian, K., Baragur, A.G., Donadel, D., Sahabandu, D., Brighente, A., Ramasubramanian, B., Conti, M., Poovendran, R.: CANLP: A NLP-based intrusion detection system for CAN. In: Proceedings of the 2024 ACM/ SIGAPP Symposium on Applied Computing. pp. 212–214 (2024)
3. Bhatia, R., Kumar, V., Serag, K., Celik, Z.B., Payer, M., Xu, D.: Evading voltage-based intrusion detection on automotive can. In: NDSS (2021)
4. bitbane: CANT (2016), `https://github.com/bitbane/CANT`, defcon 26 Car Hacking Village
5. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: 20th USENIX security symposium (USENIX Security 11) (2011)
6. Chiscop, I., Gazdag, A., Bosman, J., Biczók, G.: Detecting message modification attacks on the can bus with temporal convolutional networks. arXiv preprint arXiv:2106.08692 (2021)

7. Cho, K.T., Shin, K.G.: Error handling of in-vehicle networks makes them vulnerable. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1044–1055 (2016)
8. commaai: opendbc. `https://github.com/commaai/opendbc` (2023), accessed: Aug. 13, 2023
9. Dagan, T., Wool, A.: Parrot, a software-only anti-spoofing defense system for the can bus. ESCAR EUROPE **34** (2016)
10. De Araujo-Filho, P.F., Pinheiro, A.J., Kaddoum, G., Campelo, D.R., Soares, F.L.: An efficient intrusion prevention system for can: Hindering cyber-attacks with a low-cost platform. IEEE Access **9**, 166855–166869 (2021)
11. Desta, A.K., Ohira, S., Arai, I., Fujikawa, K.: Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots. Vehicular Communications **35**, 100470 (2022)
12. Doan, T.P., Ganesan, S.: Can crypto fpga chip to secure data transmitted through can fd bus using aes-128 and sha-1 algorithms with a symmetric key. Tech. rep., SAE Technical Paper (2017)
13. Elend, B., Adamson, T.: Cyber security enhancing CAN transceivers (2017)
14. de Faveri Tron, A., Longari, S., Carminati, M., Polino, M., Zanero, S.: Canflict: exploiting peripheral conflicts for data-link layer attacks on automotive networks. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 711–723 (2022)
15. Foster, I., Prudhomme, A., Koscher, K., Savage, S.: Fast and vulnerable: A story of telematic failures. In: 9th USENIX Workshop on Offensive Technologies (WOOT 15) (2015)
16. Groza, B., Murvay, S., Van Herrewege, A., Verbauwhede, I.: Libra-can: A lightweight broadcast authentication protocol for controller area networks. In: Cryptology and Network Security: 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings 11. pp. 185–200. Springer (2012)
17. Halabi, J., Artail, H.: A lightweight synchronous cryptographic hash chain solution to securing the vehicle can bus. In: 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET). pp. 1–6. IEEE (2018)
18. Hammerschmidt, C.: Number of automotive ecus continues to rise. `https://www.eenewseurope.com/en/number-of-automotive-ecus-continues-to-rise/` (May 2019), accessed: Dic. 13, 2023
19. Hartkopp, O., Reuber, C., Schilling, R.: Macan-message authenticated can. In: 10th Int. Conf. on Embedded Security in Cars (ESCAR 2012) (2012)
20. Iehira, K., Inoue, H., Ishida, K.: Spoofing attack using bus-off attacks against a specific ecu of the can bus. In: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC). pp. 1–4. IEEE (2018)
21. International Standard Organization (ISO): CAN Standard ISO 11898-1:2015. `www.iso.org/standard/63648.html` (2015), accessed: Aug. 23, 2023
22. Iqbal, S., Haque, A., Zulkernine, M.: Towards a security architecture for protecting connected vehicles from malware. In: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). pp. 1–5. IEEE (2019)
23. Jin, S., Chung, J.G., Xu, Y.: Signature-based intrusion detection system (ids) for in-vehicle can bus network. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5. IEEE (2021)
24. Kalutarage, H.K., Al-Kadri, M.O., Cheah, M., Madzudzo, G.: Context-aware anomaly detector for monitoring cyber attacks on automotive can bus. In: Proceedings of the 3rd ACM Computer Science in Cars Symposium. pp. 1–8 (2019)

25. Karthik, T., Brown, A., Awwad, S., McCoy, D., Bielawski, R., Mott, C., Lauzon, S., Weimerskirch, A., Cappos, J.: Uptane: Securing software updates for automobiles. In: International Conference on Embedded Security in Car. pp. 1–11 (2016)
26. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al.: Experimental security analysis of a modern automobile. In: 2010 IEEE symposium on security and privacy. pp. 447–462. IEEE (2010)
27. Kruegel, C., Toth, T.: Using decision trees to improve signature-based intrusion detection. In: International workshop on recent advances in intrusion detection. pp. 173–191. Springer (2003)
28. KT Secure: Software Code Signing (2023), `https://ktsecure.co.uk/services/software-code-signing/`, accessed: Jan. 15, 2024
29. Kulandaivel, S., Jain, S., Guajardo, J., Sekar, V.: Cannon: Reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 195–210. IEEE (2021)
30. Lokman, S.F., Othman, A.T., Abu-Bakar, M.H.: Intrusion detection system for automotive controller area network (can) bus system: a review. EURASIP Journal on Wireless Communications and Networking **2019**, 1–17 (2019)
31. Lotto, A., Marchiori, F., Brighente, A., Conti, M.: A survey and comparative analysis of security properties of can authentication protocols. IEEE Communications Surveys & Tutorials pp. 1–1 (2024). `https://doi.org/10.1109/COMST.2024.3486367`
32. Maggi, F.: A vulnerability in modern automotive standards and how we exploited it. Trend Micro (2017)
33. Matsumoto, T., Hata, M., Tanabe, M., Yoshioka, K., Oishi, K.: A method of preventing unauthorized data transmission in controller area network. In: 2012 IEEE 75th Vehicular Technology Conference (VTC Spring). pp. 1–5. IEEE (2012)
34. Microchip: SN65HVD230: 3.3 V CAN Transceiver with Standby Mode (4 2018), rev. 0
35. Microchip: MCP2515: Stand-Alone CAN Controller with SPI Interface (4 2021), rev. K
36. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. Black Hat USA **2015**(S 91), 1–91 (2015)
37. Minawi, O., Whelan, J., Almehmadi, A., El-Khatib, K.: Machine learning-based intrusion detection system for controller area networks. In: Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications. pp. 41–47 (2020)
38. Mitchell, R., Chen, I.R.: A survey of intrusion detection techniques for cyber-physical systems. ACM Computing Surveys (CSUR) **46**(4), 1–29 (2014)
39. Nilsson, D.K., Phung, P.H., Larson, U.E.: Vehicle ecu classification based on safety-security characteristics. In: IET Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference. pp. 1–7. IET (2008)
40. Nürnberger, S., Rossow, C.: –vatican–vetted, authenticated can bus. In: Cryptographic Hardware and Embedded Systems–CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings 18. pp. 106–124. Springer (2016)
41. NXP: Secure TJA115x CAN Transceiver Family (1 2020), rev. 3
42. Onuma, Y., Terashima, Y., Nakamura, S., Kiyohara, R.: A method of ecu software updating. In: 2018 International Conference on Information Networking (ICOIN). pp. 298–303. IEEE (2018)

43. Palanca, A., Evenchick, E., Maggi, F., Zanero, S.: A stealth, selective, link-layer denial-of-service attack against automotive networks. In: Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14. pp. 185–206. Springer (2017)
44. Park, S.B., Jo, H.J., Lee, D.H.: Flooding attack mitigator for in-vehicle can using fault confinement in can protocol. Computers & Security **126**, 103091 (2023)
45. Pesé, M.D., Stacer, T., Campos, C.A., Newberry, E., Chen, D., Shin, K.G.: Libre-can: Automated can message translator. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2283–2300 (2019)
46. phil-eqtech: CH-Workshop. `https://github.com/phil-eqtech/CH-Workshop` (2020), accessed: Dec. 13, 2023
47. Piętak, A., Mikulski, M.: On the adaptation of can bus network for use in the ship electronic systems. Polish Maritime Research **16**(4), 62–69 (2009)
48. Punde, A.: Understanding risks in over the air firmware upgrade for automotives including evs. `https://www.einfochips.com/blog/understanding-risks-in-over-the-air-firmware-upgrade-for-automotives-including-evs/` (January 2023)
49. Raspberry Pi Foundation: Raspberry Pi: Putting the power of computing and digital making into the hands of people all over the world. `https://www.raspberrypi.org/` (February 2023)
50. Sagong, S.U., Ying, X., Poovendran, R., Bushnell, L.: Exploring attack surfaces of voltage-based intrusion detection systems in controller area networks. In: Proc. ESCAR Eur. pp. 1–13 (2018)
51. Schell, O., Kneib, M.: Valid: Voltage-based lightweight intrusion detection for the controller area network. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 225–232. IEEE (2020)
52. SeeedStudio: CAN-BUS Shield V2 - high-performance MCP2515 controller & MCP2551 transceiver. `https://www.seeedstudio.com/CAN-BUS-Shield-V2.html` (February 2023)
53. Seo, E., Song, H.M., Kim, H.K.: Gids: Gan based intrusion detection system for in-vehicle network. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST). pp. 1–6. IEEE (2018)
54. Serag, K., Bhatia, R., Faqih, A., Ozmen, M.O., Kumar, V., Celik, Z.B., Xu, D.: ZBCAN: A zero-byte can defense system. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 6893–6910 (2023)
55. Serag, K., Bhatia, R., Kumar, V., Celik, Z.B., Xu, D.: Exposing new vulnerabilities of error handling mechanism in can. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 4241–4258 (2021)
56. Serag, K., Kumar, V., Celik, Z.B., Bhatia, R., Payer, M., Xu, D.: Attacks on can error handling mechanism. In: International Workshop on Automotive and Autonomous Vehicle Security (AutoSec) (2022)
57. Siddiqui, A.S., Gui, Y., Plusquellic, J., Saqib, F.: Secure communication over can-bus. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). pp. 1264–1267. IEEE (2017)
58. SK Pang Electronics: PiCAN2 Duo CAN-Bus Board for Raspberry Pi. `https://copperhilltech.com/pican2-duo-can-bus-board-for-raspberry-pi/` (February 2023)

59. Song, H.M., Kim, H.R., Kim, H.K.: Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In: 2016 international conference on information networking (ICOIN). pp. 63–68. IEEE (2016)
60. Song, H.M., Woo, J., Kim, H.K.: In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications **21**, 100198 (2020)
61. STMicroelectronics: Stm32 nucleo-144 development board with stm32h743zi mcu. `https://www.st.com/en/evaluation-tools/nucleo-h743zi.html/` (February 2023)
62. STMicroelectronics: Interrupt overview - stm32mpu (2024), `https://wiki.st.com/stm32mpu/wiki/Interrupt_overview`, accessed: Jan. 15, 2024
63. Takada, M., Osada, Y., Morii, M.: Counter attack against the bus-off attack on can. In: 2019 14th Asia Joint Conference on Information Security (AsiaJCIS). pp. 96–102. IEEE (2019)
64. Taylor, A., Japkowicz, N., Leblanc, S.: Frequency-based anomaly detection for the automotive can bus. In: 2015 World Congress on Industrial Control Systems Security (WCICSS). pp. 45–49. IEEE (2015)
65. Thompson, S.: Application of controller area network bus and CANopen protocol in Industrial Automation. Ph.D. thesis, Murdoch University (2018)
66. Tindell, K.: CAN Bus Security - Attacks on CAN bus and their mitigations. Canis Labs White Paper (2019)
67. Tindell, K.: Can injection: keyless car theft. `https://kentindell.github.io/2023/04/03/can-injection/` (April 2023)
68. Xu, T., Lu, X., Xiao, L., Tang, Y., Dai, H.: Voltage based authentication for controller area networks with reinforcement learning. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). pp. 1–5. IEEE (2019)
69. Zhang, M., Masrur, A.: Improving timing behavior on encrypted can buses. In: 2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). pp. 1–6. IEEE (2019)

A Appendix

A.1 Attack Detection Delay

The fastest way for a MCU to monitor several lines is through the usage of interrupts. Therefore, we employed changing edge interrupts to check every CANTX line our officer is monitoring. We set the interrupts to fire every time a rising or falling edge is observed on the line. This is perfect for detecting Error #2, which is generated by an ECU passing from idle (recessive value) to a dominant value on the bus. However, it can also be used to identify if the correct ECU is transmitting some data. It can be done by monitoring the first bits after the arbitration ID, looking for a bit of change. Theoretically, in the space of binary messages, there exist cases where all the bits in the frames are the same. Practically, this is not possible in a CAN bus because of bit stuffing that forces a different bit after five consecutive equal bits.

Even if we understand that Error #1 will eventually trigger, it is interesting to know the mean delay for having such an error raised. The theoretical limit is given by the definition of bit stuffing: $5 + 1$ bit times. However, we investigated the probability of waiting that time before detecting Error #1 by computing the time to wait for an edge after the completion of the arbitration ID. We extracted data from a dataset of 10 minutes of CAN traffic to compute the number of bits to wait for an Error #1 to be raised. Figure 6 shows the probability of waiting up to a certain amount of bit times to have an alert. As shown, after 4 bits time, almost all the frames have transmitted a dominant value. However, 6 bits is the maximum number of bits to wait to have a 0, as mandated by the bit stuffing mechanism. This guarantees an upper bound on the time to wait for the detection, which is anyway negligible, and ensures detection and prevention of attacks before the end of the frame.

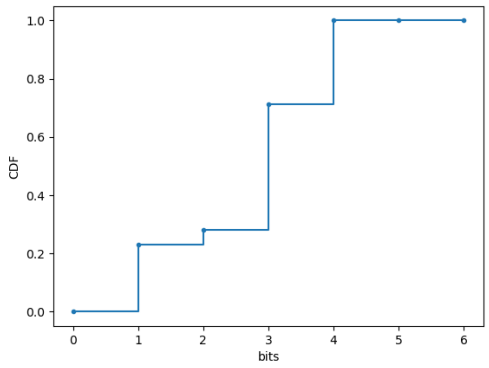


Fig. 6: CDF of maximum bit times to wait for an Error #1 to be raised.

A.2 A toy example of attack prevention

To demonstrate the capabilities of *CANTXSec*, we conducted an experiment in a scenario that could happen in a real vehicle. In particular, we imagine an attacker launching a spoofing attack against a sensor in order to tamper with the real value. We targeted a light sensor that is available in our testbed. Similar sensors are everywhere in vehicles, for instance, to measure engine temperatures or tire pressure. As explained in Section 6, the environment light value is broadcasted every 100ms. The value is almost constant and exhibits small changes through time since the environment brightness does not usually vary with high frequency.

We started with a bus without any security measures. At the time $t = t_0$, we imagine that a compromised ECU monitored by *CANTXSec* starts sending frames at a very high frequency containing a tampered value and spoofing the light sensor ID. The effect is visible in Figure 7, where it is easy to spot the malicious high value being imposed. In particular, from t_0, the flooding of tampered messages with an abnormally high value tries to push the reading out of the normal scenario. The consecutive drops are caused by the legitimate reading that is still periodically sent from the sensor and, therefore, received from the other ECUs. An advanced adversary could exploit a BOA against the legitimate sensor to avoid the drops [7, 20].

To prevent the attack, we activated *CANTXSec* in prevention mode at time $t = t_1$. Blocking malicious frames restores the normal value, and the attack is stopped. This highlights our system's capabilities to stop spoofing attacks, which are the starting point for all FIAs. Of course, with respect to this example, in a real vehicle, the system will be activated when the vehicle is turned on, so the attack will be stopped from the beginning.

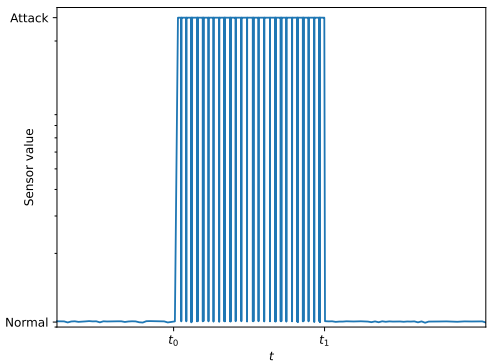


Fig. 7: Values of a light sensor over time received by the dashboard. At $t = t_0$, a compromised ECU initiates an adaptive spoofing attack, forcing the sensor value to be abnormally high. At $t = t_1$, *CANTXSec* is activated in prevention mode, and the attack is defeated.

A.3 Effect of partial ECU covering on CANTXSec capabilities

Covering more or less ECUs with *CANTXSec* impacts the attacks that can detected and, possibly, prevented. The detection of FIAs is quite resilient to narrow coverage of ECUs by *CANTXSec*, as shown in Table 4a. In fact, spoofing is detected and prevented even if the compromised ECU is not connected to the security system. This happens because, during spoofing, the legitimate ECU's CANTX will be idle while the officer detects its ID on the bus. It is worth noticing that the malicious ECU could be a not monitored ECU or a completely new device connected from the attacker to the bus [67]. Therefore, for normal use cases, covering safety-critical ECUs is enough to secure the system against common FIAs.

Table 4: Detectability of a spoofing attack (Table 4a) and a SBA (Table 4b) for different configurations of monitored or unmonitored ECUs. $\bar{X}$ means the ECU sending ID=X is monitored by *CANTXSec*.

		Spoofed ID			
		A	B	C	D
Origin ECU	$\bar{A}$	–	✓	✓	✓
	$\bar{B}$	✓	–	✓	✓
	C	✓	✓	–	✗
	D	✓	✓	✗	–

(a) Spoofing attack.

		Victim frame ID			
		A	B	C	D
Attacker ECU	$\bar{A}$	–	✓	✓	✓
	$\bar{B}$	✓	–	✓	✓
	C	✗	✗	–	✗
	D	✗	✗	✗	–

(b) SBA.

When dealing with SBAs, the situation is more complicated, as shown in Table 4b. In this class of attacks, identifying the transmitter of the malicious bits is essential to detect attacks. All the attacks originated from ECUs connected to *CANTXSec* are identified since the officer will notice a dominant value during the transmission of a frame not linked to the transmitting ECU. However, this does not apply to unmonitored ECUs since they are outside the control of the officer. This implies that to have complete detection of SBAs, all the ECUs must be connected to *CANTXSec*.