CANPak: An Intrusion Detection System against Error Frame Attacks for Controller Area Network

Sikandar Mehmood Abbasi and Stefano Longari

Politecnico di Milano, Milano, Italy
sikandarmehmood.abbasi@mail.polimi.it
stefano.longari@polimi.it

Abstract

The automotive industry has experienced significant evolution and expansion in recent years, resulting in increasingly complex in-vehicle networks and a growing number of external communication interfaces and on-board Electronic Control Units (ECUs). Despite advancements, the Controller Area Network (CAN) protocol and its enhanced version, the CAN with Flexible Data-rate (CAN FD) protocol, continue to be widely used due to their reliable and efficient real-time transmission capabilities. However, the CAN protocol was not originally designed with security in mind, lacking authentication mechanisms for communications. This vulnerability allows attackers to send spoofed messages across the bus. While application-level Intrusion Detection Systems (IDSs) can identify these spoofed messages, sophisticated attackers can bypass such security measures by disconnecting the target ECU before initiating the spoofing attack. This disconnection can be achieved through error frame injection attacks, a known vulnerability of the CAN protocol. In this work, we propose an IDS that defends against error frame injection attacks, recognizing an attacker's attempt to force a victim ECU to disconnect itself from the network. Our approach detects these attacks with up to 0.97 accuracy, without requiring any modifications to existing ECUs or the network architecture.

1 Introduction

The automotive industry has experienced a significant evolution over the past decades, transforming from purely mechanical systems to advanced, digitally integrated vehicles. This digitalization has enabled a range of functionalities, including enhanced safety features, infotainment systems, and autonomous driving capabilities, marking a new era of automotive innovation. However, this increased integration of digital components and networked systems has also increased the security risk. As vehicles now rely heavily on complex electronic control systems, the potential for security vulnerabilities has escalated, creating critical challenges in ensuring the safety and reliability of modern vehicles.

In this context, the Controller Area Network (CAN) protocol is the de facto standard for in-vehicle communication. Developed in the 1980s, CAN was designed for efficiency and real-time data transmission, making it highly suitable for automotive applications. However, due to the lack of inherent security features in CAN's original design, the protocol is highly vulnerable to various forms of attacks. Research has demonstrated numerous attack vectors targeting CAN, including spoofing, injection, and denial-of-service (DoS) attacks [25, 21, 37, 36, 10, 12, 22, 6]. As a consequence, CAN's pervasive use across the automotive industry, combined with its security limitations, renders it a prime target for exploitation.

A specific subset of attacks that leverage CAN's vulnerabilities is bus-off attacks [31], which pose a unique and particularly stealthy threat to in-vehicle communication. In a bus-off attack, an attacker induces a series of error frames that temporarily disconnect an Electronic Control

Unit (ECU) from the CAN bus. These errors, however, are not transmitted to the application layer, making the attack effectively invisible at this level. This covert quality allows a bus-off attack to momentarily deactivate an ECU without triggering immediate alerts. Such attacks may both directly impact the vehicle's functionality and safety, and be used to implement more complex spoofing attacks [26].

Despite the significant threat posed by bus-off attacks, most existing CAN intrusion detection systems (IDS) focus on recognizing the aftermath of such attacks rather than directly preventing them. Many IDS solutions aim to identify anomalous behavior on the bus once an ECU is already silenced, indirectly inferring that an attack may have occurred [20, 32]. To the best of our knowledge, only one solution, CopyCAN [19], directly addresses bus-off attacks; however, it can only detect these attacks after their completion, rendering impossible to develop reaction measures to avoid the attack's effects.

In this paper we introduce CANpak, a lightweight IDS for CAN bus-off and generally error frame injection attacks, that overcomes the limitation of previous works by exploiting the concept of polyglot frames presented in [8] to recognize the attacker before the victim ECU has been deactivated. The intuition behind CANpak is that the injection of errors on the bus by the attacker is hardly perfectly synchronized with the victim's clock. Unlike other solutions, which require changes to the network architecture or ECUs, our IDS only requires the monitoring unit to be installed within the existing network, and does not require knowledge of other ECUs signals. We demonstrate our approach in a real-world test-bed, operating the IDS on an embedded system, demonstrating its capabilities to work with a variety of frame and timing configurations. Our solution obtains a 0.97 detection rate, remarkably without obtaining false positives, a crucial characteristic for cyber-physical systems intrusion detection, enabling the implementation of active reaction measures.

In summary, our contributions are the following:

- We study the effectiveness of exploiting polyglot frames [8] in defensive solutions for CAN attacks.
- We present CANpak, a lightweight, effective, specification-based IDS against CAN error frame attacks, based on the oversampling capabilities of SPI.
- We demonstrate CANpak capabilities in a real-world test-bed, obtaining 0.96 detection rate, 0.95 f1-score, and no false positives.

2 CAN Primer

In this section, we present an overview of the necessary protocol details for the presented solution. Additional details regarding protocol specifications are available at [11, 1].

2.1 CAN Nodes

CAN nodes are composed of a transceiver and a controller, as illustrated in Figure 1, connected to a two-wire differential bus that forms a broadcast topology. The **CAN controller**, embedded within the host microcontroller, encodes and decodes data according to the CAN protocol's data link layer and manages frame transmission from application-level messages. The **CAN transceiver** converts logical data to physical signals, with its selection based on speed requirements. Low-speed transceivers support up to 125 Kbit/s and offer fault-tolerant communication, while high-speed transceivers handle bit rates up to 1 Mbit/s. Every typical CAN data frame includes multiple fields as illustrated in Figure 2.

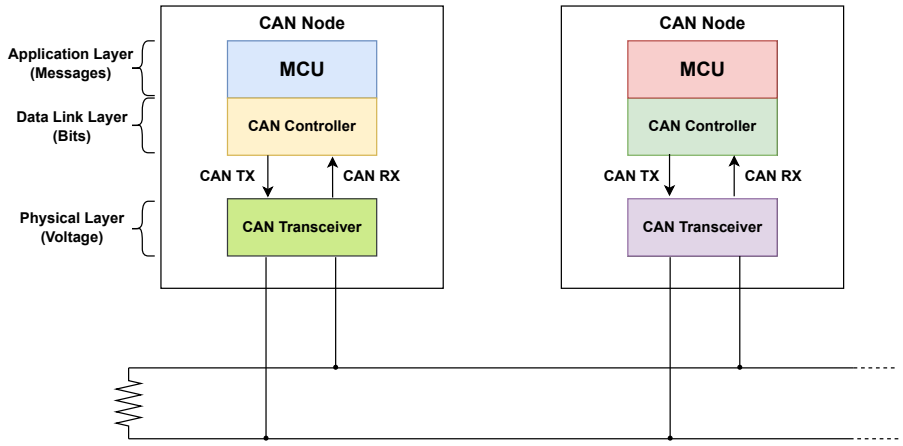


Figure 1: Two CAN nodes connected to the same CAN bus.

Transmission of Messages. CAN's communication protocol operates on the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) principle, where nodes on the network monitor the bus for idle time before attempting to transmit messages. In case of simultaneous transmission attempts, collision detection is executed mostly at arbitration time, ensuring that nodes detect and respond to collisions appropriately.

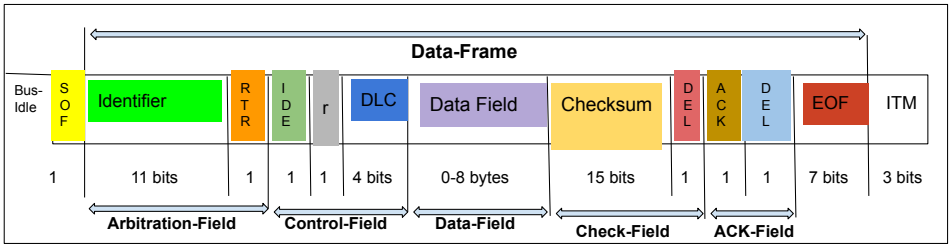
Arbitration Mechanism. CAN arbitration mechanism, based on dominant (0) and recessive (1) bits at the physical layer, ensures collision avoidance through CSMA/CA. During transmission, nodes monitor the bus bit by bit, particularly the ID field, which is the first after the Start of Frame, is used for arbitration. If multiple nodes transmit simultaneously, each continues until the data sent differs from the data received. A mismatch causes the node to cease transmission, preventing collisions. Since dominant bits override recessive ones, lower-priority messages (higher IDs) lose arbitration, allowing higher-priority messages to be transmitted first.

Bit Stuffing. CAN employs bit stuffing to maintain bus synchronization. After every five consecutive identical bits, a bit of the opposite value is inserted, regardless of the sixth bit's value. This avoids desynchronization between nodes, since synchronization is obtained during the raising or falling edge of a bit.

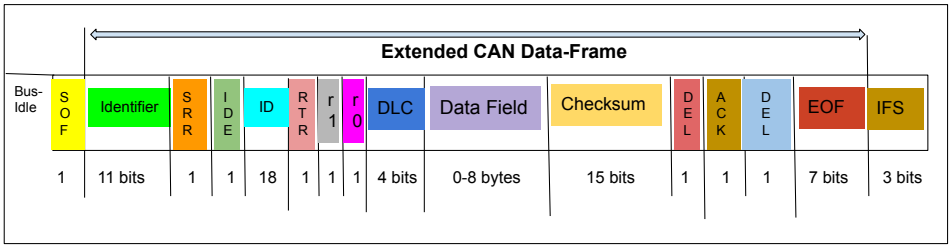
Error Handling. When a transmitter or receiver detects an error, it sends an error frame starting with an Error Flag (composed of six consecutive dominant or recessive bits) violating the bit-stuffing rule. This forces all nodes to detect the error, ignore the currently transmitted message, and respond with their own Error Flags. The frame ends with an error delimiter of eight recessive bits. The transmitter then retries sending the corrupted frame once the bus becomes idle, competing again for arbitration.

Fault Management. When a node detects and handles an error while transmitting or receiving, it increases its transmit or receive error counter of a value of 8. Once the transmit error counter increases over 255, the node moves in bus off state and shuts itself out of communication

Figure 2: Standard CAN and CAN-FD Data Frame Format.



(a) Standard CAN Frame



(b) Extended CAN Frame

until enough packets have passed.

2.2 Differences between CAN and CAN-FD

In 2011, Bosch started working with automakers and CAN experts to develop the Controller Area Network with Flexible Data-rate, or CAN FD. With the help of this new protocol, data can be transferred at rates higher than 1 Mbit/s and with payload lengths of up to 64 bytes. ISO 11898-1:2015 is the international standard for CAN FD [1].

Although CAN FD was initially developed for high-performance automotive ECUs, it is also suitable for non-automotive applications needing reliable, high-speed data transmission. With its enhanced communication capabilities, CAN FD is expected to become standard in new vehicles as manufacturers transition from traditional CAN bus systems [35, 2].

Frame format. The frame format is one of the key distinctions between CAN and CAN FD. The two-frame format is displayed in Figure 2.

Bit Rate. Higher bit-rates are needed to reduce the transmission time and maintain an acceptable throughput as payload sizes expand. Although the arbitration phase is still only allowed to proceed at a maximum of 1 Mbit/s for compatibility, CAN FD increases the bit rate during the transmission of the payload up to eight times (8 Mbit/s). Nevertheless, there is a notable deviation from the theoretical and practical bit-rate [15].

CRC Calculation. CAN with Flexible Data-rate (CAN FD) uses different CRC polynomials based on frame formats and includes stuffed bits across the entire frame. For all standard CAN

and CAN FD frames, a CRC-15 polynomial is used. CAN FD frames with data fields up to 16 bytes use CRC-17, while those with fields larger than 16 bytes use CRC-21.

Note that the bit-stuffing rule also applies to the CRC field, adding bits at specific points: at the start and after every 4th bit of the CRC sequence, with values opposite to the previous bit. If a receiver detects a stuff error (when a fixed bit matches the previous bit's value), it drops these bits before running the CRC check.

3 Related Works and Threat Model

Multiple researchers have demonstrated over the years both how attacks against automotive CAN are safety critical [5, 13, 23, 26], and how it is possible to reach the CAN buses of a vehicle through physical [24, 5, 21, 37, 36, 10, 12, 22] or remote access [25], [24, 38, 16, 30].

A vast literature is also present regarding countermeasures against CAN attacks. One of the most viable and common mechanisms to mitigate such events are IDSs [14]. Such systems can be classified depending on the technology used for detection, e.g., model-based or machine-learning based, or on the features and network level at which they operate: *Signature-based IDSs* identify attacks by comparing real-time activity to a list of recognized signatures, harmful events, or rules derived from known attack patterns. To detect attack fingerprints for present and future intrusion detection systems, Muter et al. [28] deployed eight anomaly detection sensors. On a similar vein, Studnia et al. [34] identified signatures of an anomalous CAN message sequence on a network. *Time or Frequency-based IDSs* [27, 33, 17] work based on the knowledge that the majority of the vehicle signals are transmitted at fixed intervals to comply with safety standards, and that attacks that inject or remove messages would modify the inter-arrival time between two packets with the same ID. *Feature-Based IDSs* evaluate network parameters like anomalous messages, payloads, bus load, frequency, and dropped messages. They compare these features against a baseline, flagging deviations that exceed a predefined threshold. These solutions often exploit machine-learning techniques. Finally, *specification-based IDSs* work based on the assumption that untampered messages follow protocol rules, while attack packets may not do so. Parrot, developed by Dagan et al. [7], operates by monitoring specific CAN IDs broadcasted by the ECU where the IDS is installed. If another ECU trasmits its IDs, Parrot deletes the message. However, it requires knowledge of all the broadcasted IDs and to be installed on each defended node. Longari et al. [19, 17] propose two IDSs that detect when an ECU is in bus-off state and do not allow its IDs to be sent in that period. However, they cannot detect attacks that trigger single error frames.

Note that while IDSs are effective, attacks have been studied to bypass them. In particular, many intrusion detection techniques are vulnerable to masquerade attacks, or attacks that - instead of injecting additional packets on the data stream - manage to modify the content of valid packets [29]. In fact, frequency-based detection becomes uneffective, but also feature- and specification-based detection may at this point be exploited by adversarial attacks [18, 4]. However, to execute a masquerade attack from a CAN node different from the one that should generate a given packet (source), it is necessary to silence the source. This has been demonstrated feasible -although it is implementation dependent - through diagnostic services [26] or, more consistently, by exploiting the CAN fault management process. If the attacker manages to generate bus errors while another node is transmitting, it is possible to delete its messages, eventually send such node in bus off state, and start communicating on its behalf, effectively implementing a masquerade attack. This has been demonstrated feasible initially through physical access [6, 31] and successively from remote [8] through the exploitation of Polyglot frames.

Polyglot Frames. Polyglot frames are based on the notion that a signal's meaning is not

intrinsic but instead depends on how it is interpreted, since its true meaning emerges only through interpretation based on decoding conventions or rules. Thus, a signal's meaning is tied not to its physical properties but to the system interpreting it. In the context network protocols, a bitstream transmitted over the bus is considered a polyglot frame if it may be interpreted by multiple protocols. Specifically, De Faveri et al. [8] demonstrated that the SPI protocol is capable of interpreting and generating any CAN message due to its much less constrained protocol rules. Thanks to this property, if an attacker manages to modify which peripherals are connected to the physical pins of the embedded board, it is possible to connect the CAN transceiver (which is commonly not embedded in the microcontroller) to the SPI interface, allowing the attacker to bypass CAN protocol rules and generate bus off and masquerade attacks against other nodes.

3.1 Threat model

Since gaining direct control over a safety-critical ECU is typically hindered by secure gateways [3, 9], the adversary's strategy involves influencing the operation of a separate ECU via a vulnerable one, all without triggering suspicion. While physical attacks are now deemed impractical due to the necessity of prior physical network access, the risk of remote attacks remains significant. Remote exploitation is viable through various attack vectors as mentioned previously, Bluetooth, and cellular networks [25]. In our threat model, we assume the attacker possesses complete knowledge of the CAN bus network and control over an ECU. The attacker targets a specific ECU using a targeted error-frame attack by injecting six consecutive dominant bits, aiming to delay the victim's messages or disable the ECU by driving its TEC beyond 256.

Motivation. CAN and CAN FD lack built-in security. While multiple IDSs techniques have been designed, many are not effective against adversarial masquerade attacks, which however require the attacker to delete the source frame. Our objective with this research is to build a simple IDS that can be used to recognize error frame and bus off attacks and therefore mitigate masquerade attacks. To do so, we follow the polyglot frames approach presented by De Faveri et al. [8] but implement it in a defensive setting.

4 Approach: CANPAK

The goal of our approach is to design a lightweight, consistent IDS for error frame attacks against CAN that meets the requirement of generating zero false positives, without relying on additional hardware or an ECU with high computational capabilities. To do so, our system is based on a simple intuition, which is that while the victim ECU, generating frames and being silenced by the attacker, has a consistent bit period, the attacker can hardly manage to avoid a phase shift while starting the injection of its dominant bits. To recognize this event, we oversample the reading by a factor of eight through the use of the SPI interface, thanks to a mechanism called "polyglot frames" designed and previously presented by de Faveri et al. [8].

Our approach can be divided in two phases: in the first phase the objective of the detection system is that of recognizing a potential attack event, which can be identified by an error flag. CANpak functions as a routine that consistently reads traffic from the CAN bus: a Finite State Machine (FSM) is utilized to parse the bit sequence, adhering to the frame format and fault confinement regulations, for the purpose of identifying erroneous bits or error sequences. Each field within nearly every frame corresponds to a FSM state, with the transition from one state to another determined by the bit count.

Once an error flag is identified, the second phase of the detection process aims to check whether the error flag appears to be generated by a malicious actor or not, and it does so by evaluating whether the oversampled error bits are consistent with the authentic traffic or not. If any 8-bit sequence starting from where the error-flag is located fails to comply with a combination of eight consecutive dominant or recessive bits, our algorithm generates an attack alert. Additionally, we monitor the Transmit Error Count (TEC) of the protected ECU or multiple ECUs based on the occurrence of error frames and successfully transmitted frames.

4.1 Polyglot Frames and Bus Reading

The IDS functions as a continuous reading routine, capturing CAN bus traffic bit by bit through the high-speed Serial Peripheral Interface (SPI). The key advantage of SPI is that of being perfectly "polyglot" in relation to CAN. In fact, thanks to the lack of start and stop bits, SPI allows for perfectly oversampling the CAN bus, even at its maximum 8 Mbit/s bit rate for CAN FD. By sampling the bus N times per bit, SPI captures an equivalent bit value composed of repeated samples. This oversampling enables the analysis of polyglot frames—bitstreams interpretable across multiple protocols. By bypassing the CAN controller, SPI captures raw CAN frames as digital signals, offering fine-grained, bit-level data without significant hardware modifications.

4.2 Reading Routine and Error Frame Detection

This section outlines the algorithm driving the CANPak device, focusing on standard 11-bit frames for simplicity, though it can be easily adapted for 29-bit extended frames.

The algorithm functions as a custom CAN controller, continuously reading the bus bit by bit and interpreting the bit sequence following a Finite State Machine (FSM) that identifies valid CAN or CAN FD frames. Data is directly retrieved from the CAN transceiver. Inspired by Longari et al. [19], the FSM tracks frame fields, occasionally grouping them for simplification. The relevant variables are defined as follows: *The Bit Counter*, or BC, tracks the number of bits processed in the current frame field, increasing by 1 with each new bit. *The Polarity Counter*, or PC, counts consecutive identical bits, resetting when a different bit appears. It's mainly used for fields with bit stuffing and also for validating CRC stuffed bits. *The Stuff Counter* counts stuffed bits by incrementing when five identical bits are detected (PC == 4). The variable *DL* represents the data field length in bits, based on the frame's DLC field. *The CRC Counter*, or CC, counts CRC sequence bits to ensure the CRC bit stuffing rule is followed. The actual length of the CRC sequence is known as CRC LEN. It tracks the CRC sequence length, which varies (15, 17, or 21 bits) based on data length. The variable called *TEC* is required to maintain track of the protected ECU's Transmit Error Count.

Error-free parsing. The algorithm begins in the IDLE state, incrementing the Bit Counter (BC) upon detecting a dominant bit, moving to the Start of Frame (SOF) and then the Identifier (ID) state. Here, stuff bits and errors are tracked by the Polarity Counter (PC) according to the bit stuffing rule. The Identifier bits (2nd to 13th) identify the transmitter, with possible stuff bits noted by the Stuff counter. In the RTR/RRS state, both dominant and recessive bits are stored. Although CAN FD lacks remote frames, the RTR bit is saved for compatibility with other protocols. The IDE bit determines whether the ID is 11 or 29 bits long. The FDF bit directs the algorithm to either the RES state (for CAN FD) or DLC state (for CAN). If CAN FD is detected, a dominant reserved bit precedes the BRS state, which switches the data bit-rate based on the bit value. The algorithm then parses the DLC, Data Length (DL),

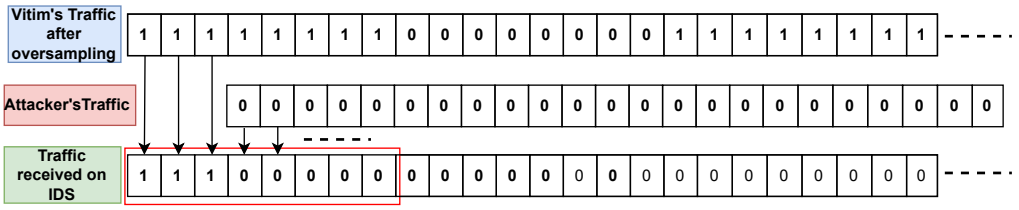


Figure 3: Attack scenario: The red rectangle represents the 8-bit SPI sequence visible by the IDS, interrupted halfway by the attacker's error flag.

CRC Length (CRC LEN), and CRC Stuff (CRC STUFF). In the DATA state, the payload is processed, followed by parsing the SBC and CRC sequence. The CRC counter increments as each bit is parsed, updating the PC as needed. Once bit stuffing is no longer relevant, the PC counter is deactivated. The final 10 bits, including the CRC delimiter, ACK slot, ACK delimiter, and EOF, are read, ensuring all are recessive except the ACK slot, which confirms successful reception. If no errors occur, the TEC decreases by 1, and after an IFS (3 recessive bits), the system resets to IDLE, awaiting the next frame. If any of the 3 IFS bits is dominant, the algorithm handles overload frames, parsing up to twelve dominant bits followed by eight recessive bits. For CAN frames, the FDF bit triggers a unique transition, but the general operations remain the same. The DLC and DL are extracted, and if the RTR bit is 0, the payload is processed; if 1, indicating a remote frame, the algorithm skips to CRC parsing. After parsing 15 CRC bits, the final frame bits are read, and if error-free, the TEC is decremented, returning to the IFS state.

Error Handling and Attack Detection. The algorithm's error detection process begins by observing the bus for an error-flag: if six consecutive identical bits are detected (PC == 5) during bit stuffing, the error flag commences. Once the error flag finishes, the algorithm transitions to the ERROR FLAG state. Once in the Error-flag state, we collect the last 96 SPI bits (or 12 CAN bits, since we oversample CAN signals by a factor of eight) from the currently processed index.

Once the last twelve CAN bits are identified, our intuition is straightforward: if the error is not the result of an attack, the error flag initiated by the source node will consistently follow a predefined pattern, starting precisely at the beginning of a byte or clock signal in a fully synchronous manner. Conversely, if the CAN bits fail to align with the 8-bit SPI sequences, it can be assumed that this desynchronization is caused by an ongoing attack, which imposes an error flag on top of the victim's traffic, as visible in Figure 3.

If any byte, starting from where the error-flag sequence begins, fails to comprise a combination of eight 0s or eight 1s (indicating an interruption or discrepancy in the expected error pattern), our algorithm triggers an attack message and current state will transition to error-DEL marking the end of the error analysis phase and alerting that an attack may be underway. Additionally, we continuously monitor and update the TEC of the protected ECU or multiple ECUs based on two conditions i.e incremented (+8) for error frames detected and decremented (-1) for successfully transmitted frames:

Subsequently, if the TEC reaches 256 (indicating a "bus-off" state), the algorithm scans for 11-bit-long sequences of recessive bits. Once 128 such sequences are detected, the TEC is reset, and the ECU is considered reconnected.

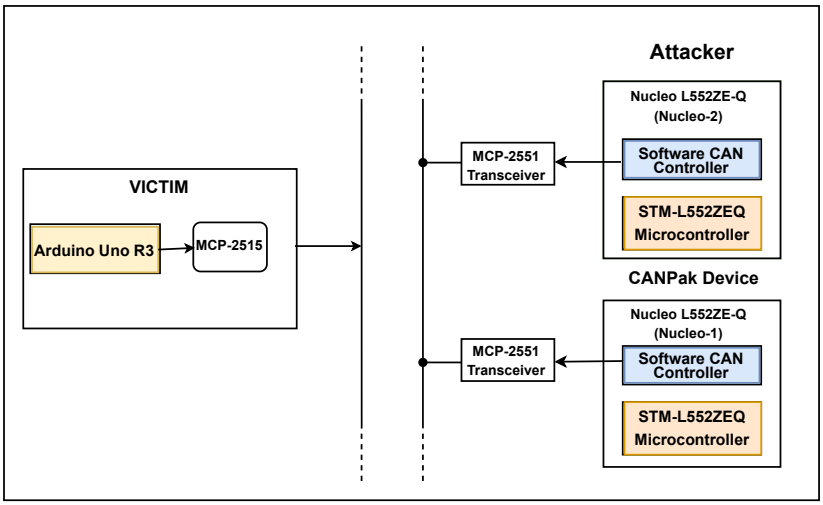


Figure 4: Schematic view of the final test setup.

Table 1: Confusion matrix for the attack detection experiment.

	True Positives	True Negatives
Predicted Positive	111	0
Predicted Negative	11	129

5 Experimental Validation

Our main experimental goals are to validate our synchronization assumption and to evaluate our approach performances in real-world scenarios. Specifically, we aim to demonstrate that the IDS can reliably detect fraudulent error-flag sequences composed of six consecutive dominant bits (0s) without mistaking legitimate error flags. By examining detection rates, false positives, and response times, we will validate the system's capability to distinguish attack-generated error-flags from genuine network errors.

Our setup is composed of three devices connected to a CAN bus test-bed, as shown in Figure 4. An Arduino board with a CAN shield using an MCP2515 CAN controller acts as the victim of the attack, generating messages with various IDs on the bus. A first Nucleo L552ZE-Q board runs the CANpak code, monitoring the bus. Finally, second Nucleo L552ZE-Q board runs the attacker code, performing attacks by injecting dominant error frames against the victim's frames on the bus. Both Nucleo boards are equipped with an external MCP2551 CAN transceiver.

Experiment 1: Attack detection.

We run our experiment multiple times, with the attacker attempting to inject error frames over multiple packets with various IDs generated by the victim. Our results are promising, as presented by the confusion matrix in Table 1. With an accuracy of 0.97, a precision of 1.0, and an f1-score of 0.95, while not achieving perfect detection rate, the performances suggest that our IDS may be implemented to recognize error-frame attacks, especially as an complementary measure for those techniques that are vulnerable to masquerade attacks.

Experiment 2: Error-free traffic.

The objective of the second experiment is to ensure that our assumption is correct, and that there are no instances of correctly generated error frames that are not synchronous and therefore end up being flagged as false positives by our system. For this evaluation, we let our test-bed run in an attack-free scenario for hours, in order to allow for errors to "naturally" arise and error flags to (hopefully) not be detected. In fact, while unfortunately only in 9 instances errors were generated, none was detected by our system, ending up with a perfect 0.0 false positive rate.

6 Conclusion

In this study, we introduced CANpak, an innovative Intrusion Detection System (IDS) for CAN and CAN FD networks. Our research commenced drawing inspiration from both CopyCAN [19] for the reading routine and CANflict [8] for the idea of polyglot frames. Our design was born from the necessity of a robust and lightweight specification-based IDS capable of mitigating adversarial masquerade attacks. However, current state-of-the-art specification-based IDSs that work towards that goal either have too many requirements, such as having to be implemented on all nodes, or cannot directly detect a single error frame, and instead need the attack to lead the victim in bus-off state to be effective. Our IDS addresses these shortcomings while providing lightweight, real-time detection against error frame attacks and all the attacks enabled by silencing a victim node. Specifically, the system detects instances where an attacker inserts an error flag on the bus to disrupt the traffic of a specific target. We validated the viability of CANPak through the implementation of a proof-of-concept testbed. The experimental outcomes confirm the effectiveness of the algorithm and demonstrate its reliability achieving 0.97 accuracy, 0.95 f1-score, and 0.0 false positive rate.

Undoubtedly, our solution has its own limitation, specifically in the assumption that an attacker cannot synchronize its error flag with the rising or falling edge of the valid CAN packet they are attempting to invalidate. While - without oversampling themselves - the attacker cannot synchronize willingly, there is a chance dependent on the oversampling frequency that the attack ends naturally being synchronous with the valid traffic. To mitigate and minimize the chance that this event happens, future work includes studying the realistic increase in the oversampling rate that maintains the 0.0 false positive rate while increasing the detection rate, and the evaluation of different hardware that may allow for the increase of the maximum SPI clock frequency above the 40Mhz of the current one, potentially allowing for additional oversampling.

7 Acknowledgments

Stefano Longari is supported by the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). CUP MICS D43C22003120001.

References

[1] Iso central secretary. road vehicles — controller area network (can) — part 1: Data link layer and physical signalling. *Standard ISO 11898-1:2015, International Organization for Standardization, Geneva, CH,*, 2015.

[2] Cia can in automation. canopen fd. *Available: https://www.can-cia.org/news/cia-in-action/view/canopenfd- cia-1301-released/.*, Cia 1301 released, 2017.

[3] Jonas Berg, Jens Pommer, Chuan Jin, Fredrik Malmin, and Johan Kristensson. Secure gateway-a concept for an in-vehicle ip network bridging the infotainment and the safety critical domains. *13th Embedded Security in Cars (ESCAR'15)*, pages 1–12, 2015.

[4] Paolo Cerracchio, Stefano Longari, Michele Carminati, Stefano Zanero, et al. Investigating the impact of evasion attacks against automotive intrusion detection systems.

[5] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011.

[6] Kyong-Tak Cho and Kang G. Shin. Error handling of in-vehicle networks makes them vulnerable. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1044–1055. ACM, 2016.

[7] Tsvika Dagan and Avishai Wool. Parrot, a software-only anti-spoofing defense system for the can bus. *ESCAR EUROPE*, 34, 2016.

[8] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 711–723. ACM, 2022.

[9] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 711–723. ACM, 2022.

[10] Ian D. Foster, Andrew Prudhomme, Karl Koscher, and Stefan Savage. Fast and vulnerable: A story of telematic failures. In Aurélien Francillon and Thomas Ptacek, editors, *9th USENIX Workshop on Offensive Technologies, WOOT '15, Washington, DC, USA, August 10-11, 2015*. USENIX Association, 2015.

[11] Robert Bosch GmbH. Can specification. *Available: http://esd.cs.ucr.edu/webres/can20.pdf.*, 1991.

[12] Hyo Jin Jo, Wonsuk Choi, Seoung Yeop Na, Samuel Woo, and Dong Hoon Lee. Vulnerabilities of android os-based telematics system. *Wirel. Pers. Commun.*, 92(4):1511–1530, 2017.

[13] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[14] Brooke Lampe and Weizhi Meng. Intrusion detection in the automotive domain: A comprehensive review. *IEEE Commun. Surv. Tutorials*, 25(4):2356–2426, 2023.

[15] Jo Laufenberg, Thomas Kropf, and Oliver Bringmann. Can simulation framework - from classic can to can xl. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3343–3348, 2023.

[16] Yousik Lee, Samuel Woo, Jungho Lee, Yunkeun Song, Heeseok Moon, and Dong Hoon Lee. Enhanced android app-repackaging attack on in-vehicle network. *Wirel. Commun. Mob. Comput.*, 2019, 2019.

[17] S Longari, G Galletti, J Holle, S Zanero, et al. Canter: data-link layer detection of drop-and-spoof attacks on can and can fd. In *Proceedings of the Italian Conference on Cyber Security (ITASEC 2024)*, pages 1–13. CEUR, 2024.

[18] Stefano Longari, Francesco Noseda, Michele Carminati, and Stefano Zanero. Evaluating the robustness of automotive intrusion detection systems against evasion attacks. In Shlomi Dolev, Ehud Gudes, and Pascal Paillier, editors, *Cyber Security, Cryptology, and Machine Learning - 7th International Symposium, CSCML 2023, Be'er Sheva, Israel, June 29-30, 2023, Proceedings*, volume 13914 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2023.

[19] Stefano Longari, Matteo Penco, Michele Carminati, and Stefano Zanero. Copycan: An error-handling protocol based intrusion detection system for controller area network. In Lorenzo Cavallaro, Johannes Kinder, and Thorsten Holz, editors, *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy, CPS-SPC@CCS 2019, London, UK, November 11, 2019*, pages 39–50. ACM, 2019.

[20] Stefano Longari, Carlo Alberto Pozzoli, Alessandro Nichelini, Michele Carminati, and Stefano Zanero. Candito: Improving payload-based detection of attacks on controller area networks. In Shlomi Dolev, Ehud Gudes, and Pascal Paillier, editors, *Cyber Security, Cryptology, and Machine Learning - 7th International Symposium, CSCML 2023, Be'er Sheva, Israel, June 29-30, 2023, Proceedings*, volume 13914 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2023.

[21] Amit Kr Mandal, Federica Panarotto, Agostino Cortesi, Pietro Ferrara, and Fausto Spoto. Static analysis of android auto infotainment and on-board diagnostics II apps. *Softw. Pract. Exp.*, 49(7):1131–1161, 2019.

[22] Sahar Mazloom, Mohammad Rezaeirad, Aaron Hunter, and Damon McCoy. A security analysis of an in-vehicle infotainment and app platform. In Natalie Silvanovich and Patrick Traynor, editors, *10th USENIX Workshop on Offensive Technologies, WOOT 16, Austin, TX, USA, August 8-9, 2016*. USENIX Association, 2016.

[23] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[24] Charlie Miller and Chris Valasek. A survey of remote automotive attack surfaces. *black hat USA*, 2014:94, 2014.

[25] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91):1–91, 2015.

[26] Charlie Miller and Chris Valasek. Can message injection. *OG Dynamite Edition*, 2016.

[27] Michael Roy Moore, Robert A. Bridges, Frank L. Combs, Michael S. Starr, and Stacy J. Prowell. Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In Joseph P. Trien, Stacy J. Prowell, John R. Goodall, Justin M. Beaver, and Robert A. Bridges, editors, *Proceedings of the 12th Annual Conference on Cyber and Information Security Research, CISRC 2017, Oak Ridge, TN, USA, April 4 - 6, 2017*, pages 11:1–11:4. ACM, 2017.

[28] Michael Müter, André Groll, and Felix C. Freiling. A structured approach to anomaly detection for in-vehicle networks. In *Sixth International Conference on Information Assurance and Security, IAS 2010, Atlanta, GA, USA, August 23-25, 2010*, pages 92–98. IEEE, 2010.

[29] Alessandro Nichelini, Carlo Alberto Pozzoli, Stefano Longari, Michele Carminati, and Stefano Zanero. Canova: A hybrid intrusion detection framework based on automatic signal classification for CAN. *Comput. Secur.*, 128:103166, 2023.

[30] Sen Nie, Ling Liu, and Yuefeng Du. Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA*, 25(1):16, 2017.

[31] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. In Michalis Polychronakis and Michael Meier, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment - 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings*, volume 10327 of *Lecture Notes in Computer Science*, pages 185–206. Springer, 2017.

[32] Md Hasan Shahriar, Yang Xiao, Pablo Moriano, Wenjing Lou, and Y. Thomas Hou. Canshield: Deep-learning-based intrusion detection framework for controller area networks at the signal level. *IEEE Internet Things J.*, 10(24):22111–22127, 2023.

[33] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 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 2016, Kota Kinabalu, Malaysia, January 13-15, 2016*, pages 63–68. IEEE Computer Society, 2016.

[34] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaâniche, and Youssef Laarouchi. A language-based intrusion detection approach for automotive embedded networks. *Int. J. Embed. Syst.*, 10(1):1–12, 2018.

[35] Aoran Wang, Jie Fang, Yinan Xu, Yihu Xu, Yubing Wang, Yujing Wu, and Jin-Gyun Chung. Anomaly information detection and fault tolerance control method for can-fd bus network. In *2022 19th International SoC Design Conference (ISOCC)*, pages 308–309, 2022.

[36] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. Plug-n-pwned: Comprehensive vulnerability analysis of OBD-II dongles as A new over-the-air attack surface in automotive iot. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 949–965. USENIX Association, 2020.

[37] Haohuang Wen, Qingchuan Zhao, Qi Alfred Chen, and Zhiqiang Lin. Automated cross-platform reverse engineering of CAN bus commands from mobile apps. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[38] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Trans. Intell. Transp. Syst.*, 16(2):993–1006, 2015.