POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Lightweight Multi-Method Intrusion Detection for Automotive CAN Bus Networks

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Edoardo Galtieri

Advisor: Prof. Stefano Longari

Academic year: 2024-2025

1. Introduction

Modern vehicles have evolved from purely mechanical systems into sophisticated cyber-physical platforms, integrating hundreds of Electronic Control Units (ECUs) that manage critical functions ranging from engine performance and braking systems to infotainment and driver assistance features. Due to this transformation vehicles are no longer isolated machines but connected systems that communicate internally through in-vehicle networks (IVN) and externally through wireless interfaces such as Bluetooth or Wi-Fi. The **Controller Area Network (CAN)** protocol is the standard for in-vehicle communication due to its reliability, real-time performance and cost-effectiveness. CAN enables ECUs to exchange messages over a shared bus without requiring point-to-point wiring between every component. However, the protocol was designed without security as a design requirement. Indeed, CAN messages lack authentication mechanisms, encryption or sender verification: any node on the bus can transmit messages claiming to originate from any ECU and all nodes must trust received messages without validation. Due to this security gap Intrusion Detection Systems (IDSs) emerged as one of the most practical approach for securing CAN networks. Various IDS methodologies were studied: e.g. voltage-based fingerprinting that identifies ECUs through electrical signal characteristics, frequency-based analysis that detects timing anomalies in periodic message transmission and machine learning approaches that model normal traffic patterns to identify deviations. Especially, Machine learning-based IDSs have demonstrated high detection rates, often approaching or exceeding 99% true positive rates across different attack scenarios. However, deep learning models require costly and heavy hardware for inference and possibly processing latencies measured in hundreds of milliseconds. On the other hand, existing lightweight IDSs operate within ECU resource constraints but cannot obtain the same performance as ML algorithms, or they do not have enough attack coverage. For these reasons this thesis addresses three specific objectives: (1) demonstrating that combining complementary lightweight detection methods achieves competitive detection rates without machine learning; (2) determining the optimal architectural configuration (parallel versus serial) for maximizing attack coverage while minimizing computational overhead; (3) validating practical deployability through performance comparison against state-

of-the-art systems and embedded implementation on resource-constrained hardware representative of automotive ECUs.

2. Background

The CAN has become the standard in modern vehicles since its introduction by Robert Bosch in 1986 and its standardization in 2012 (ISO 11898-1). This protocol was implemented with broadcast packets in mind, which means that all connected nodes receive every transmission. This is performed thanks to a dual-wire twisted pair interface consisting of two lines designated as CAN High (CAN_H) and CAN Low (CAN_L). When the bus is in a Recessive state (logical bit 1) the nominal voltage is common and equal to 2.5V. On the contrary during a Dominant state (logical bit 0) a voltage differential is created by the CAN_H being at 3.5V and the CAN_L at 1.5V. Data transmission within the CAN protocol is organized into structured packets known as frames. The standard defines four distinct frame types: data, remote, overload and error frames. Data frames being organized as in Figure 1:

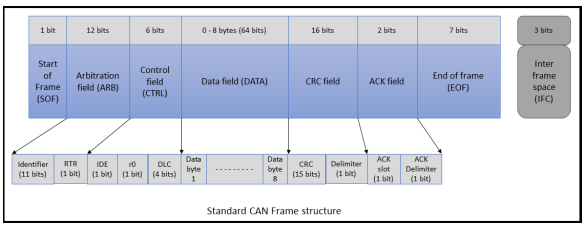

Figure 1: Data Frame format

Starting with the Start of Frame (SOF), this is followed by the Arbitration Field, which determines the priority of the message and is divided into an 11- or 29-bit Identifier (ID) that identifies the message content and an RTR bit that distinguishes a dominant Data Frame from a recessive Remote Frame. Next is the Control Field followed by the Data Field, the actual payload containing the application data, which can range from 0 to 8 bytes in length. After the data, the 16-bit CRC functions as the verification process for nodes to check whether a packet is valid or not. Following this is the ACK, consisting of two bits set to recessive by the sending node; upon positive reception, other ECUs will set it to dominant. Finally, the End of Frame (EOF) marks the conclusion of the data packet with a sequence of seven consecutive recessive bits. To handle errors CAN Bus has a specific protocol that implies using the error frames: when an error is detected on a frame, an Error Frame is transmitted. This frame is composed of six consecutive bits of the same polarity that intentionally break the bit-stuffing rule[1], followed by an error delimiter: eight recessive bits, providing a stall that allows the bus to return to an idle state before any node attempts to re-transmit. To make use of this frames, the protocol use a finite state machine with three states: Error Active, Error Passive, Bus-off, the first being the default state of every ECU. From there, after 2 registers have passed a certain threshold, the ECU enters the second state, which means each node has to wait exactly 8 bits (Suspend Transmission Time) after a transmission to try and send another message. If the errors increase the load of the system over to a certain threshold, the ECU enters the last state, where it is effectively disconnected from the bus and can only re-enter after 128 occurrences of 11 consecutive recessive bits.

3. Can Bus Security

As mentioned in section 1, CAN Bus suffers from some inherent vulnerabilities that an attacker may exploit to gain control of ECUs or send malicious messages on the channel. We've established a potential adversary model that can attack these vulnerabilities:

- Impersonation: The attacker injects messages on the bus impersonating as the victim node, which requires access to the physical channel.
- Masquerading: The attacker disrupts the communication with a Bus-Off or targeted injection attack (these will be analyzed in the following section) and assumes the victim's identity but with another physical location. It requires the knowledge of the target ECU's parameters and possibly remote access to it.
- Compromised ECU: The attacker has full control over the victim ECU and can send

[1]The protocol dictates a strict "5-bit rule": whenever a transmitting node sends five consecutive bits of the same logic level (whether dominant or recessive), it must automatically insert a sixth bit of the opposite level into the stream

valid messages on its behalf.

These model can perform a series of attack, we focused only of a relevant subsection:

- **Fabrication**: Attackers inject messages with valid CAN ID and data fields onto the bus which makes them indistinguishable from legitimate traffic except for their payload content. [3]
- **Fuzzing**: Packets sent on the channel, with both ID and payload being completely randomly generated. It is exploited as an attack vector to cause malfunctions and disruptive behavior in ECUs.
- **Replay**: Valid CAN packets are replayed exactly as recorded without any alteration.
- **Denial-of-Service (DoS)**: Bus is flooded with packets to prevent all legitimate nodes from communicating on the channel.
  - **Suspension**: Specific type of targeted Denial-of-Service attacks in which a targeted ECU is prevented from communicating on the bus. One common method of performing such an attack is with a Bus-Off attack
  - **Bus-Off**: Attacker continuously injects a dominant bit during transmission while the targeted node is sending a recessive one creating bit errors and making the node enter Bus-off state.
- **Masquerade**: Masquerade attacks are a combination of the fabrication and suspension attack types. The attacker first suspends the target ECU then injects messages with the same CAN ID, effectively impersonating the legitimate sender. [2]

4. IDS in automotive

Intrusion Detection Systems (IDSs) are one of the most common defense against attacks on in-vehicle networks. Over the years, researchers have developed many different types of IDS, each using different methods to detect malicious activity. We will take some categories as an example of the current state-of-the-art on IDS in automotive.

- **Entropy-based**: In the context of automotive IDS, entropy quantifies the predictability of CAN traffic. Entropy-based IDS methods leverage information theory to detect anomalies in CAN network traffic. These approaches measure the randomness, uncertainty or complexity of message characteristics such as ID distribution, payload values or timing patterns. They can detect both known and unknown attacks by identifying statistical deviations from normal patterns but normal driving produces varying entropy patterns depending on the scenario (e.g., highway vs. city driving), potentially causing false positives.
- **HW-based**: They detect attacks by analyzing the electrical and physical characteristics of CAN signals rather than message content. These methods provide device authentication, identifying attack sources and work independently of CAN protocol details. Their main limitation is that physical characteristics are environmentally sensitive (temperature affects clock skew, aging changes voltage, cables degrade) necessitating recalibration.
- **Message Frequency**: Message frequency-based IDS detect attacks by monitoring how often messages with specific CAN IDs appear on the bus since most in-vehicle messages are transmitted periodically at fixed rates. IDSs are extremely lightweight, requiring only message counters and basic arithmetic operations, making them ideal for resource-constrained automotive ECUs. However, frequency-based methods cannot detect sophisticated attacks that preserve normal transmission frequencies such as frequency masquerading or low-rate replay attacks.

5. Pipeline Approach

The selection of the pipeline components followed a systematic review of the state-of-the-art, starting from a pool of 4,141 papers reduced to two final candidates through filtering criteria based on performance, computational overhead, detection time and absence of external hardware requirements. The selected systems are ErrIDS by Lee et al. [4], focusing on timing analysis, and an IDS based on Hamming distance between message payloads by Marchetti et al. [5]. This multi-layered approach enables detection of both content-based attacks and timing-based attacks.Since suspension attacks are not detectable in time through payload or timing analysis alone, an indepen-

dent watchdog timer was added to complement both systems regardless of the chosen configuration. The three detection methods are described below:

- **Hamming Distance**: The research use a method to detect malicious CAN messages by computing the Hamming distance between consecutive payloads of the same message ID [2]. In particular, here the hamming distance was interpreted as the bitwise XOR between the two payloads as shown in equation 1:

$$\mathcal{H}_d(p_t, p_{t+1}) = \sum_{i=1}^{n} p_t^i \oplus p_{t+1}^i \qquad (1)$$

  where p_t and p_{t+1} are consecutive payloads of the same CAN ID and n is the payload length in bits (typically 64 for 8-byte messages) [5]. During training max and min ranges of each ID are learned: if during detection one packet exceed these thresholds it is flagged as anomalous.
- **ErrIDS**: This method exploits the residual[3] between nominal and actual inter-arrival times to perform detection. For each CAN ID, the nominal transmission period t is determined from benign training data and rounded to standard frequencies. The i-th residual is calculated as:

$$r_i = at_i - t \qquad (2)$$

  where t is the nominal period and at_i is the actual measured timestamp interval between consecutive messages i and $i-1$. After the residual calculation, for each ID, an upper and lower thresholds are calculated using a fixed parameter K, mean and standard deviation. During detection if one of these threshold is exceeded the packet is flagged as anomalous.
- **Watchdog Suspension Check**: for each monitored message ID, we maintain a timer that resets upon message reception: if the expected message does not arrive within a timeout threshold (calculated by multiplying the nominal transmission period by a fixed value of 5), the watchdog triggers a suspension alert.

We decided to test for a Serial and a Parallel configuration to determine which is the best in terms of performances and computational overload. In both configurations the Watchdog timer runs independently and continuously.

5.1. Collateral Effect

During testing we observed a high rate of FP that initially appeared to indicate poor system performance. However, detailed analysis revealed that the majority of these false positives were direct consequences of the attacks themselves rather than algorithmic errors. So we decided to classify them as **collateral false positives**, meaning bening packets flagged as a consequence of the attack and not detection failures. Both detection methods exhibit this behaviour through different mechanisms:

Hamming Distance: When an attack packet is processed, its payload becomes the baseline for the next comparison. This corrupts the baseline, the next benign packet is compared against the attack payload rather than a legitimate payload, producing an Hamming distance that falls out of range for that ID. That triggers detection even though the benign packet itself is normal.

ErrIDS Timing: When attacks flood the CAN bus, legitimate messages go through queuing delays. Benign packets may arrive outside their expected timing windows and trigger detection not because they are malicious, but because the attack disrupted normal bus timing.

6. Experiments and Results

As our main testing Dataset, we decided to use X-CANIDS, a publicly available dataset that contains both raw CAN messages and deserialized signals collected from a real commercial vehicle (2017 Hyundai LF Sonata e-VGT). We run our first tests in a Python environment to validate our approach and then we ported the code to Wokwi on a STM32 Nucleo C031C6 (an online IoT and embedded systems simulator supporting ESP32, STM32, Arduino and Raspberry Pi platforms with built-in peripheral simu-

[2] The Hamming Distance between two strings of equal length is the number of positions at which the corresponding symbols differ. It is a metric that only counts substitutions and assumes that the two strings being compared are of the same length [5]

[3] The residual is defined as the difference between the actual timestamp interval and the nominal period, where the timestamp interval is the time elapsed between two consecutive transmissions of the same CAN ID.

lation [1]) to calculate detection time and memory occupation. As our first test we decided to run the Serial Configuration: the detection components were put in a sequential, two-stage pipeline, where The Hamming distance method serves as the initial filter: each incoming CAN message is first scored against the learned per-ID payload ranges. Based on the resulting anomaly score and two decision thresholds, the system has three possible choices to make. If the score exceeds the upper threshold (θ_{high}), the message is immediately flagged as an attack with high confidence. If it falls between the lower and upper thresholds ($\theta_{low} \leq s < \theta_{high}$), the message is seen as suspicious and sent to the second stage for timing-based analysis via ErrIDS; otherwise, it is accepted as normal.
Instead, for the Parallel Configuration, we used both detection methods to examine every message independently. Hamming Distance method and ErrIDS work together on the CAN traffic: a packet can be "detected" or not and the systems combine their detection outputs with a logical OR: a message is flagged if either method identifies it as anomalous. Each method contributes its strengths without being blocked by the other's blind spots.

6.1. Serial Configuration Results

Stage 1 (Hamming distance) detected 7,286,546 attacks with high confidence, yielding a direct TPR of 42.21% and an FPR of 0.0000%. The remaining 9,976,755 attack packets were either classified as suspicious and forwarded to Stage 2, or missed entirely because their anomaly scores fell below θ_{low}.
Stage 2 (ErrIDS timing) received 2,062,744 suspicious packets; from which it recovered only 15,332 additional attacks, for a Stage 2 efficacy of 0.15%. The timing module also generated 2,046,728 false positives on the suspicious benign packets it processed.
The combined system achieved: TPR = 42.30%, FPR = 0.3907%, F1 = 0.5488.
It is important to notice performance varied substantially across attack types due to the systems characteristics. ErrIDS generates far more false positives in serial than parallel not because it behaves differently, but because in serial it receives a pre-filtered suspicious subset that is disproportionately populated by baseline-corrupted benign packets that Hamming already identified as borderline anomalous.

6.2. Parallel Configuration Results

The combined system achieved (with packet-level detection metrics): TPR = 90.97%, FPR = 0.0438% and F1 score of 0.9476, with a precision of 98.88%

Attack Type	TPR	F1 Score	Precision
Masquerade	99.99%	98.08%	96.24%
Fuzzing	97.53%	98.50%	99.47%
Fabrication	99.57%	97.80%	96.08%
Replay	81.58%	89.81%	99.88%
Suspension	100.00%	n/a%	100.00%
Overall System FPR: 0.0438%			

Table 1: Parallel Configuration Performance by Attack Type

From the table above we can understand how the systems are able to detect the majority of malicious packets on almost every type of attacks, except for replay which is the reason for the seemingly "low" overall TPR. The replay attacks are on average 6.1M packets while the other attacks have on average 4.5M packets, therefore this discrepancy greatly influences the final TPR shifting all the weight on the replay attacks. Regarding false positives, the system maintains a consistent intrinsic FPR of 0.0438% across most attack types, with minor variations attributable to the collateral damage effect described in Section 5.1 and to ErrIDS's elevated false positive rate on periodic-and-on-event (PE) messages.

6.3. STM32 Results

The main objective of these tests was to prove that our pipeline was fast enough for the peak rate of CAN traffic and did not require much memory on the resource constrained ECU. For both configurations we replayed a representative subset of the X-CANIDS test dataset: 500 messages taken from fuzzing, fabrication and masquerade dumps. The dataset was pre-loaded into program memory as a static array, with each entry containing timestamp, CAN ID, DLC, payload and ground truth label. The serial configuration obtained an average detection time of 657μs with the minimum being 472 μs and max-

imum being 8352 μs. The throughput was 1522 msg/s which is well below the 2000/2500 msg/s of the X-CANIDS dataset. For memory measures, RAM usage was 5760 B while Flash usage was 1248 B. For parallel configuration we obtained better results: the average detection time dropped to 229 μs with the minimum being 54 μs and the maximum 373μs: This implies a much better throughput of 4366 msg/s. Memory occupation remained the same since it is composed only of thresholds and ranges that do not change between configurations.

6.4. Comparison with State-of-the-Art

Comparison against recent state-of-the-art IDS systems evaluated on the same X-CANIDS dataset reveals that our system achieves competitive or superior detection rates on several attack types, with performance comparable to or better than some existing approaches. While some systems (including both machine learning-based and signal-based methods) achieve near-perfect detection rates (approaching 100% TPR) and false alarm rates (near 0% FPR) that exceed our performance, our approach outperforms them in detection latency and deployment feasibility: competing systems either suffer from processing delays orders of magnitude slower than our 229 μs average or depend on proprietary vehicle specifications unavailable in production deployment.

7. Conclusions

This research addressed a critical challenge in automotive cybersecurity: developing a practical intrusion detection system for CAN bus networks that balances high detection accuracy with real-time performance and deployment feasibility on resource-constrained Electronic Control Units by taking already developed systems from the state-of-the-art and combining them in the best possible configuration. Through a pipeline of complementary lightweight detection methods this work demonstrates that comprehensive attack coverage comparable to machine learning systems is achievable without requiring costly and heavy hardware. The evaluation also showed that architectural configuration impacts detection effectiveness: the parallel OR-logic configuration achieves comprehensive coverage whereas serial two-stage filtering creates numerous blind spots, demonstrating that design of method integration matters as much as the individual algorithms themselves. The embedded validation on STM32 microcontroller hardware proved sub-millisecond detection latency and throughput exceeding peak automotive CAN bus rates.

8. Limitations and Future Work

Key limitations include: single-dataset evaluation (2017 Hyundai LF Sonata), per-vehicle Hamming training requirements, omission of PE message filtering (to avoid DBC dependency), and functional simulation rather than production hardware validation. Based on these limitations, future work should pursue multi-dataset validation, field testing on instrumented vehicles, and expansion with additional lightweight detection methods to address sophisticated attack patterns not represented in X-CANIDS.

References

[1] Wokwi - World's most advanced ESP32 Simulator.

[2] Seonghoon Jeong, Sangho Lee, Hwejae Lee, and Huy Kang Kim. X-canids: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network. *IEEE Transactions on Vehicular Technology*, 73(3):3230–3246, 2024.

[3] Hyo Jin Jo and Wonsuk Choi. A survey of attacks on controller area networks and corresponding countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6123–6141, 2022.

[4] Seyoung Lee, Wonsuk Choi, Hyo Jin Jo, and Dong Hoon Lee. Errids: An enhanced cumulative timing error-based automotive intrusion detection system. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):12406–12421, 2023.

[5] Dario Stabili, Mirco Marchetti, and Michele Colajanni. Detecting attacks to internal vehicle networks through hamming distance. In *2017 AEIT International Annual Conference*, pages 1–6, 2017.