Yes, CARLA CAN: Extending the CARLA Simulator for In-Vehicle Network Cybersecurity Experimentation

Luigi F. Marques da Luz[1], **Vinicius L. Ventura**[1], **Divanilson R. Campelo**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (CIn - UFPE)
Av. Jorn. Aníbal Fernandes – s/n – Recife – PE – Brazil

{lfml,vlv2,dcampelo}@cin.ufpe.br

Abstract. *Modern vehicles rely on in-vehicle networks (IVNs), but protocols such as CAN lack built-in security, exposing critical systems to cyber threats. Existing platforms often require real vehicles or hardware, limiting accessibility. This work presents a software-based platform for automotive cybersecurity research and experimentation. It integrates a virtual CAN bus with dynamic driving scenarios using the CARLA simulator. A modular design enables ECUs (Electronic Control Units), attackers, and intrusion detection systems to operate as independent nodes. The platform enables monitoring and manipulation of CAN traffic without hardware, providing a reproducible environment for research, teaching, dataset generation, and evaluation of IVN security.*

1. Introduction

In recent years, vehicles have evolved from purely mechanical systems to complex cyber-physical platforms [Wu et al. 2020]. A typical in-vehicle network (IVN) integrates different network protocols technologies and electronic control units (ECUs). This integration and enhanced connectivity enable advanced functions, including remote software updates, driver assistance, and real-time data exchange. However, this same connectivity introduces new attack surfaces that expose vehicles to cybersecurity risks that can compromise both safety and reliability [Häckel et al. 2022].

For this reason, automotive cybersecurity has become a critical research domain, dedicated to identifying vulnerabilities and proposing mitigation strategies for IVN [Checkoway et al. 2011]. In this context, experimental validation plays a fundamental role, providing empirical evidence for the effectiveness of defensive approaches and supporting the development of new attack scenarios. Nevertheless, conducting experiments directly on real vehicles is highly costly, as it often requires access to proprietary systems, can involve invasive modifications to the controller area network (CAN) bus, and introduces significant financial and logistical challenges [Koscher et al. 2010]. Although there are a variety of proprietary tools and software suites [Intrepid Control Systems 2025, Vector Informatik GmbH 2025], these platforms typically require dedicated interfaces, ECUs, and laboratory infrastructure, making them costly and less accessible for flexible or large-scale educational use.

Meanwhile, automotive simulators such as CARLA [Dosovitskiy et al. 2017] provide high-fidelity vehicle dynamics and interactive environments suitable for software-based experimentation. However, the lack of native in-vehicle network support limits their use for research and educational activities involving automotive communication protocols. As a result, software-only environments remain limited, hindering exploration of

concepts such as message arbitration, signal encoding, and bus-level behavior under dynamic driving conditions.

In this direction, this work extends the CARLA driving simulator by proposing a modular platform that connects CARLA to a virtual CAN bus. This integration allows vehicle control signals to be generated and transmitted as CAN frames, closely simulating communication in real in-vehicle networks. Beyond enabling realistic vehicle control, the inclusion of a virtual CAN bus creates a flexible environment for experimentation and research in automotive cybersecurity. The platform offers ready-to-use cyberattack and defense modules, allowing direct experimentation and observation of their impact on vehicle behavior. At the same time, the modular architecture enables learners to progressively expand the system by enhancing existing modules or adding new network nodes through lightweight Python scripts, facilitating hands-on exploration of additional attacks and defensive strategies. By relying entirely on simulation, the proposed approach lowers the entry barrier to automotive cybersecurity education and research, eliminating the need for specialized hardware while preserving realism.

In a nutshell, the main contributions of this work are:

- An extension of the CARLA simulator that enables vehicle control signals to be exchanged over a virtual CAN bus, emulating real in-vehicle network communication;
- A configurable CAN network environment that allows the experimentation of different message formats, transmission periods, and arbitration scenarios, and to analyze their impact on network performance;
- A cyberattack module implementing state-of-the-art CAN bus attacks, enabling the demonstration and visualization of their effects on a simulated vehicle;
- An extensible intrusion detection module that includes an intrusion detection system based on message periodicity, and can be readily expanded to support more advanced techniques, such as machine learning–based intrusion detection systems;
- Support for generating custom datasets under diverse network configurations and both existing and novel cyberattacks, facilitating the development and evaluation of new automotive network defense mechanisms.

This paper is organized as follows. Section 2 reviews related work in automotive cybersecurity and experimentation platforms. Section 3 describes the proposed platform architecture and how its modules interact within the virtual CAN bus. Section 4 presents the available features and their use in demonstrating fundamental concepts in automotive cybersecurity. Section 5 discusses the limitations in comparison with real-world scenarios. Finally, Section 6 concludes the paper and outlines future work.

2. Related work

Existing work in automotive cybersecurity largely focuses on novel attacks and defenses. For instance, [Evenchick 2015] shows that the lack of native security and the broadcast nature of the CAN bus allow a compromised node to inject arbitrary frames, such as high-priority messages that block legitimate traffic and cause denial-of-service. More advanced attacks, such as spoofing and replay, exploit the absence of authentication to inject forged or previously recorded messages that can alter vehicle behavior [Wen et al. 2020]. Taking

a practical perspective, [Nie et al. 2017] demonstrates a full attack chain from wireless compromise to CAN injection, using crafted frames to control critical vehicle functions.

Intrusion detection systems (IDS) have gained significant attention for enhancing CAN bus security due to their low implementation cost and ability to leverage machine and deep learning techniques [Wu et al. 2020]. In [Yang et al. 2021], the authors propose a hybrid IDS combining rule-based and anomaly-based methods using algorithms such as k-means and random forest. Similarly, [Seo et al. 2018] introduces a GAN-based IDS capable of detecting unknown attacks using only normal data, while [Song et al. 2020] adapts a convolutional neural network (CNN) to CAN traffic for malicious activity detection.

This growing reliance on data-driven IDS highlights the need for high-fidelity and diverse datasets that accurately capture CAN behavior. However, existing publicly available datasets are often static and tailored to specific vehicles or collection scenarios, limiting their broader applicability [Lee et al. 2017, Han et al. 2018, Lampe and Meng 2024].

The difficulty of obtaining high-quality datasets and reproducing realistic attack scenarios highlights the need for controlled environments to generate reliable CAN traffic. To address this, several hardware-based testbeds have been proposed to emulate automotive communication with sufficient fidelity. One example is RAMN [Gay et al. 2020], an open-source hardware and software platform that emulates automotive networks using multiple virtual ECUs. It provides a cost-effective alternative to real vehicles by replicating engine, dashboard, and sensor communication patterns, enabling the generation of both benign and malicious CAN traffic in a controlled setting.

More advanced experimental environments have emerged through industry collaborations. The Toyota PASTA [Toyama et al. 2018] platform provides a modular automotive testbed with dedicated hardware and software that realistically emulates in-vehicle communication and integrates with the CARLA simulator. This enables synchronization of real CAN messages with virtual driving scenarios, bridging the gap between physical testbeds and simulations. Complementing this approach, VALUE-PASTA [VALUE-PASTA Contributors 2024] offers a lower-cost alternative with simplified hardware, improving accessibility while maintaining compatibility with the same software ecosystem.

Despite the availability of public datasets and hardware-based platforms, fully software-based environments tailored for learning and experimentation in automotive cybersecurity remain limited. To address this gap, we introduce a platform based on a virtual CAN bus that runs on a personal computer, removing the need for specialized hardware. The platform integrates a driving simulator to demonstrate the impact of CAN message exchanges on vehicle behavior and provides an industry-compatible mechanism for defining network messages. It also includes ready-to-use modules for experimenting with state-of-the-art cyberattacks and intrusion detection.

3. Proposed platform

The proposed platform extends the CARLA driving simulator [Dosovitskiy et al. 2017] with support for realistic in-vehicle communication via the CAN protocol, enabling the study of CAN traffic, cyberattacks, and defense mechanisms in a fully virtual environ-

ment. Figure 1 provides a high-level overview of the platform architecture and its main modules, which are detailed in the following subsections.

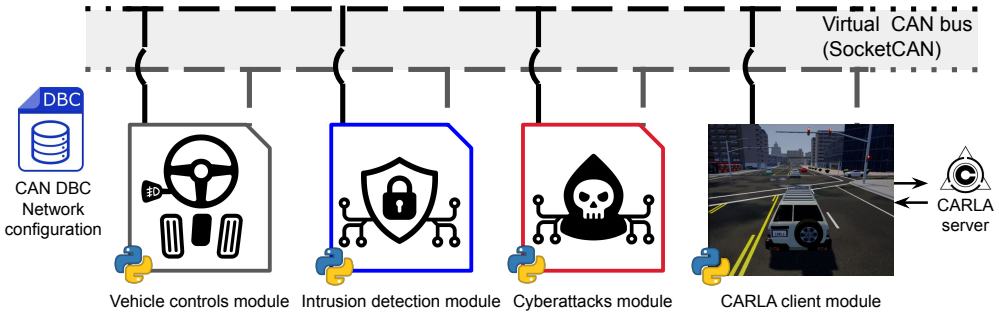


Figure 1. Proposed platform architecture with the modules connected to the virtual CAN bus.

3.1. Virtual CAN bus using SocketCAN

The platform is built upon a virtual CAN bus, a core in-vehicle communication protocol and a key technology in automotive cybersecurity. To enable realistic interaction among software modules, CAN communication is simulated on a standard personal computer using SocketCAN, a Linux utility that provides a virtual CAN interface through which modules exchange messages similarly to real vehicle ECUs.

3.2. Network messages specification

Before configuring network nodes, the platform requires the specification of CAN messages using CAN Database (DBC) files, the industry standard for defining CAN communication. A DBC file describes how raw CAN frames are translated into physical values by specifying message identifiers and signal parameters. These parameters include bit start, bit length, scale, and offset, which define how signals (e.g., sensor readings or actuator states) are decoded from raw data (Refer to the Appendix A for a better visualization of the parameters and DBC file syntax).

By modifying DBC definitions, users can experiment with different CAN IDs and signal configurations and observe their impact on bus arbitration and overall traffic behavior. Additionally, we use DBC attributes to specify the message transmission period, allowing users to configure custom transmission intervals directly within the DBC file. Adjusting these intervals enables observation of their effects on bus occupancy and the potential starvation of lower-priority frames.

3.3. Vehicle controls module

This module generates control commands for the simulated vehicle by translating user inputs from the keyboard into CAN signals, encapsulating them into frames according to the DBC specification, and transmitting them on the virtual CAN bus at configured intervals. For instance, the user can send vehicle steering and acceleration control messages using the keyboard arrow keys.

3.4. CARLA client module

The CARLA client module modifies a default control script to change the source of vehicle commands. Instead of receiving inputs directly from the keyboard, the client operates as a node connected to the virtual CAN bus, listening for messages with identifiers defined in the DBC file and executing the corresponding actions upon reception.

As an end-to-end example, consider a throttle command. A key press is translated into a control signal, encapsulated into a CAN frame, and transmitted over the virtual bus. The CARLA client, continuously listening, receives the message and actuates the vehicle accordingly. This process mirrors real in-vehicle message flow while enabling direct observation of how CAN communication influences vehicle behavior in a dynamic environment.

3.5. Cyberattacks module

The platform also supports the exploration of automotive cybersecurity concepts. As the CAN protocol lacks native security and operates in a broadcast manner, compromised nodes can inject malicious messages. To model this threat, the platform includes a cyberattacks module that acts as a standard CAN node on the virtual bus, enabling the execution of attacks such as denial-of-service, fuzzing, and spoofing. A key benefit is the ability to directly observe their impact on the simulated vehicle, which is often difficult to infer from CAN traffic logs alone.

The module provides several pre-implemented attack scenarios, while its open-source design allows users to modify existing attacks or implement new ones. This flexibility encourages experimentation, supports project-based learning, and fosters research in automotive cybersecurity.

3.6. Intrusion detection module

The platform includes an intrusion detection module designed as a baseline implementation for exploring automotive IDS concepts. As discussed in Section 2, intrusion detection is a widely studied mechanism for CAN networks, making it well suited for instructional use. The module is fully extensible, allowing users to modify the implementation, explore alternative strategies, or develop custom IDS using the generated CAN traffic.

The default implementation provides a simple data-driven IDS based on message timing. Under normal conditions, CAN messages are periodic, resulting in stable inter-arrival times for each identifier. The platform first records normal traffic to estimate the mean and variability of these intervals. During operation, messages are flagged as anomalous when their timing deviates significantly from this baseline. This lightweight approach facilitates understanding of timing-based detection and provides a foundation for more advanced data-driven IDS techniques.

4. Features demonstration

This section demonstrates the functional capabilities of the proposed platform, including the experimental environment, CAN message specification via DBC files, execution of representative cyberattacks, real-time IDS operation, and dataset generation for experimentation, training, and evaluation of defense mechanisms.

4.1. Experimental setup

The platform was developed and tested using CARLA 0.9.15 and Python 3.9 on Ubuntu 24.04 LTS, requiring native support for `SocketCAN` to enable virtual CAN interfaces. "Yes, CARLA CAN" was designed as an open-source tool, and its codebase is publicly available at https://github.com/luigiluz/yes-carla-can, supporting reproducibility and further extensions.

4.2. Network messages specification

Network behavior is defined through a DBC file (Appendix A), which specifies messages and signals such as throttle, brake, steer, and reverse. Message transmission periods are configured using a custom "GenMsgCycleTime" DBC attribute. For example, brake messages can use shorter periods than reverse signals, reflecting their higher criticality. This setup enables experimentation with different parameterizations, allowing observation of their impact on bus occupancy and arbitration behavior based on CAN IDs.

4.3. Cyberattacks demonstration

The platform provides a CLI-based script to execute cyberattacks, enabling configurable injection of malicious traffic into the virtual CAN bus. Users select attack modes via simple arguments, each implemented as an independent routine, supporting modularity, reproducibility, and easy customization.

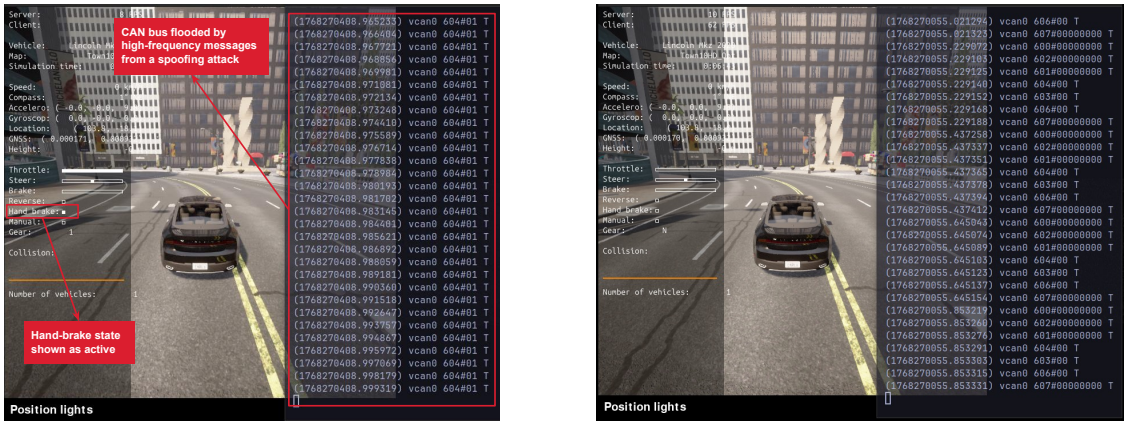
To demonstrate this capability, Figure 2(a) shows a spoofing attack on the hand brake system. The attack repeatedly injects forged messages with valid identifiers and payload definitions, producing a continuous stream that overrides legitimate control frames. As the receiver processes the most recent valid message, the injected values dominate the bus, causing the hand brake to remain engaged. As a result, the vehicle remains stationary despite throttle input. In contrast, during normal operation (Figure 2(b)), the hand brake is disengaged and vehicle behavior correctly follows driver input. This comparison highlights how manipulation of a single signal is reflected in both CAN traffic and vehicle dynamics.

The cyberattacks module also includes other state-of-the-art attacks, such as fuzzing, which injects random CAN frames with synthetic data targeting multiple messages. This produces unpredictable bus traffic that may trigger vehicle functions, such as door operation and lighting, in a random manner.

4.4. Running intrusion detection

The platform includes a lightweight intrusion detection module implemented as a standalone script for real-time monitoring of the virtual CAN bus. Users select a detector via CLI arguments, which instantiates the corresponding class and loads the required statistical model. This modular design simplifies extensibility, allowing new detectors to be added without modifying the existing structure.

The IDS runs concurrently with the simulation, enabling real-time observation of CAN frame classification under different conditions. Figure 3(a) shows operation under normal traffic, where a small number of packets are incorrectly flagged as malicious. These false positives are acceptable given the module's demonstrative purpose and provide a baseline for comparison with more advanced IDS implementations. On the other



(a) Vehicle under spoofing attack. (b) Vehicle under normal conditions.

Figure 2. Comparison of vehicle behavior under spoofing attack (left, see video) and normal conditions (right, see video). The spoofing attack targeted the hand brake (CAN ID 0x604) with an attack period of 0.001 seconds.

hand, Figure 3(b) shows the IDS during a hand brake spoofing attack. As observed on the right side of the figure, 3967 frames with CAN ID 0x604 (hand brake) are classified as malicious, indicating anomalous behavior for this message.

(a) IDS running under normal conditions. (b) IDS under spoofing attack.

Figure 3. Comparison of IDS behavior under spoofing attack (left, see video) and normal conditions (right, see video). The attack targeted the hand brake (CAN ID 0x604) with an attack period of 0.001 seconds.

4.5. Dataset generation

The platform supports the generation of reproducible datasets for evaluating intrusion detection systems, including both benign and adversarial CAN traffic. All communication between simulated ECUs passes through the virtual `vcan0` interface, allowing SocketCAN tools (e.g., `candump`) to capture complete, timestamped traffic as in a physical network. Baseline datasets are collected during normal operation, while adversarial datasets are generated by activating attack modules that introduce anomalous patterns such as abnormal frequencies or timing irregularities. Captured data can be post-processed using

the same DBC specifications to decode frames into signals, ensuring consistency with the simulated vehicle state.

By centralizing communication and enabling controlled traffic manipulation, the platform facilitates the creation of diverse and reproducible datasets covering both normal and attack scenarios. These datasets support analysis, visualization, and IDS evaluation, providing a comprehensive view of ECU behavior under different conditions.

5. Limitations

The current version of the "Yes, CARLA CAN" platform supports only a limited subset of messages related to vehicle control modules. Specifically, it includes a total of nine messages with distinct CAN IDs and associated signals. This constitutes a limitation when compared to real-world vehicles, which typically involve dozens or even hundreds of CAN messages [Lee et al. 2017, Han et al. 2018, Lampe and Meng 2024]. However, the number of messages on the platform can be increased by incorporating additional sensor data available in CARLA (e.g., GNSS, radar, collision detection, IMU, among others), thereby improving realism. Additionally, as the platform operates entirely at the software level via SocketCAN, it does not capture physical-layer effects, hardware-induced timing variations, or the bit-level arbitration mechanisms of a real CAN bus, which are abstracted by the operating system.

6. Conclusions and future work

This work presents a modular, fully software-based platform that extends the CARLA simulator with native CAN communication via a virtual bus, enabling realistic message exchange, adversarial traffic injection, intrusion detection, and traffic monitoring without physical hardware. By decoupling communication from vehicle control and using DBC files to externalize message definitions, the architecture remains easily adaptable to different vehicle profiles and network configurations. The platform provides full visibility into CAN traffic and supports the generation of benign and malicious datasets, facilitating the study of arbitration, timing, attack detection, and data-driven security analysis. Operating entirely in software, it reduces cost, improves accessibility, and enables controlled, repeatable experiments. Overall, it offers an extensible environment that bridges theory and practice while supporting research and prototyping in automotive network security.

As future work, we plan to extend the platform with additional in-vehicle network technologies, such as automotive Ethernet protocols (e.g., Audio Video Transport Protocol), enabling camera stream transmission and heterogeneous network traffic in autonomous driving scenarios. We also intend to enhance dataset generation with automatic labeling for intrusion detection experiments. Finally, we plan to integrate the platform into an undergraduate automotive networks course to support hands-on learning in CAN communication, cyberattack analysis, and IDS development.

Acknowledgements

This paper was partly supported by CNPq (Grant 306157/2025-0) and is part of the INCT of Intelligent Communications Networks and the Internet of Things (ICoNIoT) funded by CNPq (proc. 405940/2022-0) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) Finance Code 88887.954253/2024-00.

References

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium (USENIX Security 11)*.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Evenchick, E. (2015). An introduction to the CANard toolkit. In *Blackhat Conference*, pages 1–8.

Falch, M. (2024). CAN DBC File Explained – A Simple Intro. `https://www.csselectronics.com/pages/can-dbc-file-database-intro`. Accessed: 2026-01-14.

Gay, C., Toyama, T., and Oguma, H. (2020). Resistant automotive miniature network. In *Proceedings of the Chaos Computer Congress, Leipzig, Germany*, pages 27–30.

Häckel, T., Meyer, P., Korf, F., and Schmidt, T. C. (2022). Secure time-sensitive software-defined networking in vehicles. *IEEE Transactions on Vehicular Technology*, 72(1):35–51.

Han, M. L., Kwak, B. I., and Kim, H. K. (2018). Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular Communications*, 14:52–63.

Intrepid Control Systems (2025). Vehicle spy. `https://www.intrepidcs.com/products/software/vehicle-spy/`. Accessed: 2025-12-08.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., and Savage, S. (2010). Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462.

Lampe, B. and Meng, W. (2024). can-train-and-test: A curated CAN dataset for automotive intrusion detection. *Computers & Security*, 140:103777.

Lee, H., Jeong, S. H., and Kim, H. K. (2017). OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709.

Nie, S., Liu, L., and Du, Y. (2017). Free-fall: Hacking tesla from wireless to CAN bus. *Briefing, Black Hat USA*, 25(1):16.

Seo, E., Song, H. M., and Kim, H. K. (2018). GIDS: GAN based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6.

Song, H. M., Woo, J., and Kim, H. K. (2020). In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198.

Toyama, T., Yoshida, T., Oguma, H., and Matsumoto, T. (2018). PASTA: Portable automotive security testbed with adaptability. *London, blackhat Europe*.

VALUE-PASTA Contributors (2024). VALUE-PASTA: Low-Cost Automotive Security Testbed. `https://github.com/mintynet/value-pasta-auto`. Accessed: 2025-12-08.

Vector Informatik GmbH (2025). Vector automotive software tools. `https://www.vector.com/`. Accessed: 2025-12-08.

Wen, H., Chen, Q. A., and Lin, Z. (2020). Plug-N-Pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new Over-the-Air attack surface in automotive IoT. In *29th USENIX security symposium (USENIX Security 20)*, pages 949–965.

Wu, W., Li, R., Xie, G., An, J., Bai, Y., Zhou, J., and Li, K. (2020). A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933.

Yang, L., Moubayed, A., and Shami, A. (2021). MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, 9(1):616–632.

Appendices

A. CAN DBC File

CAN DBC File

```
VERSION "1.0"
BU_ : ECU

BO_ 1536 THROTTLE: 4 ECU
SG_THROTTLE_signal : 0|8@1+ (1,0) [0|255] "" ECU

BO_ 1537 BRAKE: 4 ECU
SG_BRAKE_signal : 0|8@1+ (1,0) [0|255] "" ECU

BO_ 1538 STEER: 4 ECU
SG_STEER_signal : 0|8@1+ (1,0) [0|255] "" ECU

BO_ 1540 HAND_BRAKE: 1 ECU
SG_HAND_BRAKE_signal : 0|8@1+ (1,0) [0|255] "" ECU

BA_DEF_ BO_ "GenMsgCycleTime" INT 0 10000;
BA_DEF_DEF_ "GenMsgCycleTime" 500;

BA_ "GenMsgCycleTime" BO_ 1536 100;
BA_ "GenMsgCycleTime" BO_ 1537 100;
BA_ "GenMsgCycleTime" BO_ 1538 100;
BA_ "GenMsgCycleTime" BO_ 1540 200;
```

Figure 4. DBC file specifying messages, signals, and message periods for throttle, brake, steering, and hand brake (other messages omitted for simplicity)

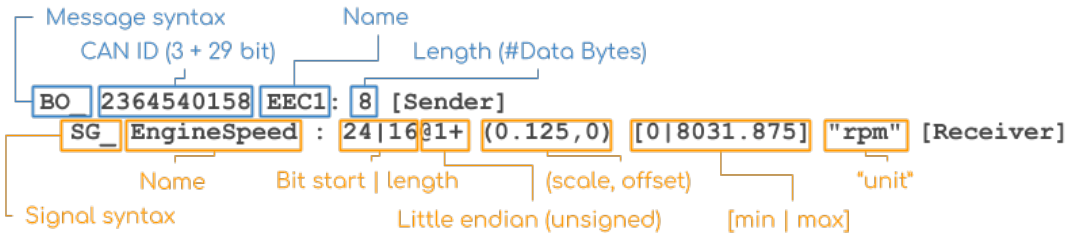


Figure 5. CAN DBC syntax. Source: [Falch 2024]

B. Cyberattack module command line interface

```
Terminal

CAN network attacks CLI
usage: attacks_cli.py [-h]
                  [--feature {hand_brake,doors,reverse,
                     high_beam,internal_lights,low_beam,
                     fog_lights,lights_off,position_lights,
                     left_blink,right_blink,fuzzy,
                     denial_of_service}]
                  [--period PERIOD]

Perform CAN network attacks.

options:
  -h, --help show this help message and exit
  --feature {hand_brake,doors,reverse,high_beam,
     internal_lights,low_beam,fog_lights,lights_off,
     position_lights,left_blink,right_blink,fuzzy,
     denial_of_service}
                  Feature to attack
  --period PERIOD Period between messages in seconds
```

Figure 6. Help command output of the `attacks_cli.py` CLI highlighting the available cyberattacks and which vehicle functionality they target.

C. Intrusion detection module algorithm specification

The IDS module employs a classical statistical approach to detect intrusions. The algorithm relies on historical CAN_ID inter-arrival time statistics and is formalized in Eq. (1).

$$IDS(ts\_diff^{(k)}_{CAN\_ID_i}) = \begin{cases} \text{normal}, & ts\_diff^{(k)}_{CAN\_ID_i} \leq \tau_i \\ \text{attack}, & ts\_diff^{(k)}_{CAN\_ID_i} > \tau_i \end{cases} \quad (1)$$

where:

- k is the current CAN message index that is going to through the IDS;
- $ts\_diff^{(k)}_{CAN\_ID_i} = ts^{(k)}_{CAN\_ID_i} - ts^{(k-1)}_{CAN\_ID_i}$ is the inter-arrival time of the k-th message for CAN_ID$_i$;
- $\tau_i = \mu_{ts\_diff_{CAN\_ID_i}} + 3\sigma_{ts\_diff_{CAN\_ID_i}}$, where μ and σ are computed from historical inter-arrival times of CAN_ID$_i$ messages under normal conditions.

D. Fuzzy attack demonstration

The Fuzzy attack randomly injects CAN frames with synthetic data targeting multiple message IDs, producing unpredictable bus traffic. As shown in the CAN log in Figure 7(a), successive frames rapidly alternate between unrelated functions without semantic consistency. In CARLA, these messages cause nondeterministic actuation of several subsystems; at the captured moment, actions such as door activation and fog light illumination were triggered.

When the Fuzzy attack is active, the IDS responds immediately to the irregular traffic. Figure 7(b) shows intrusion counts rapidly accumulating across multiple identifiers as arbitrary messages are injected at inconsistent intervals. The correlation between IDS alerts and the vehicle's erratic behavior demonstrates that anomalies on the bus are directly reflected in the simulation.

(a) Vehicle under fuzzy attack.

(b) IDS responses during fuzzy attack.

Figure 7. Fuzzy attack on vehicle (left, see video) and IDS response (right, see video).