CANova: a Hybrid Intrusion Detection Framework based on Automatic Signal Classification for CAN

Alessandro Nichelini[1a], Carlo Alberto Pozzoli[1a], Stefano Longari[a], Michele Carminati[a], Stefano Zanero[a]

[a]*Politecnico di Milano - Dipartimento di Elettronica, Informazione e Bioingegneria, Via Ponzio 34/5, Milan, 20133, Italy*

Abstract

Over the years, vehicles have become increasingly complex and an attractive target for malicious adversaries. This raised the need for effective and efficient Intrusion Detection Systems (IDSs) for onboard networks able to work with the stringent requirements and the heterogeneity of information transmitted on the Controller Area Network. While state-of-the-art solutions are effective in detecting specific types of anomalies and work on a subset of the CAN signals, no single method can perform better than the others on all types of attacks, particularly if they need to provide predictions to comply with the domain's real-time constraints. In this paper, we present CANova, a modular framework that exploits the characteristics of the different Controller Area Network (CAN) packets to select the Intrusion Detection Systems (IDSs) that better fits them. In particular, it uses flow- and payload-based IDSs to analyze the packets' content and arrival time. We evaluate CANova by comparing its performance against state-of-the-art Intrusion Detection Systems (IDSs) for in-vehicle network and a comprehensive set of synthetic and real attacks in real-world CAN datasets. We demonstrate that our approach can achieve good performances in terms of detection, false positive rates, and temporal performances.

Keywords: Automotive Security, Intrusion Detection, Signal Classification, Controller Area Network, Flow and Payload based Detection

[1] Both authors contributed equally to the research work.

1. Introduction

Nowadays, vehicles have become more and more complex not only from a mechanical point of view but especially from Electronic Control Units (ECUs) [1], necessary to implement essential functionalities such as engine performance and power steering, but also entertainment and autonomous drive-related functionalities. However, this increased complexity raises security risks, leaving the room for a bigger attack surface: if once the vehicle was an isolated system, now there are various ways in which the car is connected to the outside world: infotainment via USB and Bluetooth, short-range (e.g., keyless entry, Tyre Pressure Monitoring Systems (TPMS)) and long-range (e.g., 4G/5G) communications interfaces are possible entry points for an attacker. Even when an attacker cannot physically connect to a car's internal network, the vast number of wireless interfaces may allow gaining remote control over the car [2]. Problems reside in how ECUs are connected among each other: even when core vehicle ECUs and communication ECUs are on two separated networks, Koscher et al. [3] demonstrated that by implementing path traversal techniques an attacker would be able to reach any sub-network inside the vehicle. Therefore, the research community has focused on developing Intrusion Detection Systems (IDSs) for CAN, which analyze network packet stream for signs of intrusions.

IDSs for intra-vehicle networks can be divided into flow-based and payload-based approaches. The objective of flow-based approaches is to find anomalies in the flow of packets in the network, independently from the data inside the packets' payload. While these approaches are usually lightweight and reliable, they are limited to detecting specific events (e.g., changes in the arrival frequency of packets, such as increasing or decreasing packets' frequency in the network). Payload-based approaches analyze the data inside the packets and can also detect more sophisticated intrusions that do not modify the flow of the packets but only the information inside their payloads. Consequently, they can detect a wide range of attacks but usually have high computational requirements and are not as effective against frequency-based attacks. Considering attacks to CAN, flow-based approaches cannot be used as a standalone solution since an adversary in a compromised system can send malicious packets without changing their frequency by forcing the transmitting ECU in the bus-off state before communicating [4, 5]. On the other hand, payload-based approaches may not be suited for real-time detection of the entire traffic of the network. In fact, they are usually tested on a small subset of CAN IDs, while the traffic of a real vehicle is composed of packets from tens of CAN IDs (more than 80 in the dataset used in this work). In addition, the characteristics of the packets from different CAN IDs are often heterogeneous. However, state-of-the-art IDSs do not consider the differences between different types of packets or work only on a subset.

Given the limitations of existing IDS, in this paper, we exploit the complementarity of flow- and payload-based approaches by combining them in CANova, a modular IDS that selects the best approach to apply to each packet based on a classification of the CAN IDs. The core module of CANova extracts the signals from CAN packets and performs an attribute-based classification of the signals. Then, on the basis signal's classes and features, it decides the sequence of detection approaches that analyze each packet, focusing on a high detection rate, low computation requirements, and low false positives.

We evaluate CANova by measuring its effectiveness against a comprehensive set of synthetic attacks injected in a real-world CAN dataset. Moreover, to provide a fair evaluation of CANova, we compare its performances against state-of-art Intrusion Detection Systems (IDSs) for in-vehicle network on a public dataset constructed by logging the real-time CAN message via the onboard diagnostic (OBD-II) port of two running vehicles with attack messages. Our experimental results show that CANova outperforms state-of-the-art detection methods with a perfect True Positive Ratio (TPR), an accuracy of up to 0.9997, and lower time requirements on all the attacks under analysis.

The contributions of this paper are the following:

- We propose an attribute-based classifier to analyze and categorize CAN traffic based on flow- and payload-dependent characteristics, which are used to choose the best approach to detect anomalies for each packet.

- We propose a detection module that combines different detection approaches to obtain high detection rates while keeping lower computational requirements. To do so, we propose a new intrusion detection approach based on a Vector Auto Regression (VAR) model and improve state-of-the-art CAN detection techniques, respectively, a flow- and payload-based IDSs.

- We develop CANova, a modular IDS for CAN composed of the attribute-based classifier and the detection module, which exploits its in-depth classification of CAN characteristics to choose the approaches that are used to detect anomalies in each CAN packet.

The rest of the paper is structured as follows: In section 2, we give a brief overview of the concepts related to CAN that are needed to understand our approach. Section 3 provides an overview of existing solutions related to the one proposed in this paper, discussing their limitation and the research goal. In Section 4 we systematically analyze the threat model, while in Section 5 we present CANova. Section 6 presents the experimental evaluation and results. Finally, in Section 7 we draw the conclusion of the paper.

2. Background on CAN

In this section, we present an overview of the CAN protocol. Those concepts are fundamental for the next sections, where we describe the problems we want to solve and how state-of-the-art approaches deal with them.

2.1. Controller Area Network

Controller Area Network (CAN) is a multi-master, message broadcast protocol first introduced by Robert Bosch in 1986, then revised and standardized in 2012 (ISO 11898-1). It is one of the most successful network protocols ever designed, and today it is the de facto standard for vehicle onboard networks, used in the automotive industry for connecting the different ECUs of a vehicle. The CAN bus [6] is a broadcast channel that provides multi-master capabilities. Therefore, any node connected to the bus can read any packet sent on the network. Moreover, when the bus is idle, any node can write on the bus. If multiple nodes need to communicate at the same time, arbitration is won by the message with the lowest ID. This technique is implemented at the physical layer, where logical '0's overwrite logical '1's. Therefore if two nodes start writing the ID of their message (the first segment of the packet) on the bus, the one that has the higher ID is overwritten on the bus by the lowest one and gives up attempting to write its packet. The CAN protocol is also robust, as it features five different methods of error checking: three of them are at the message level, while two are at the bit level. A message failing any of these error detection methods is not accepted and an error frame is consequently generated from the receiving node. In case of repeated errors, the failing node is removed by its controller when the error limit is reached [7].

Table 1: CAN packet structure

Field name	**Description**
Start of Frame (SOF)	1 bit that marks the start of a message.
CAN identifier (CAN-ID)	11-bits (or 29-bits) field that identifies messages and their priority. The lower the values, the higher the priority.
Remote Transmission Request (RTR)	1 bit defining whether this is a Data or Remote frame.
Identifier Extension Bit (IDE)	1 bit that defines if the standard CAN version is being used without any extensions.
r0	1 reserved bit (for future use).
Data Length Code (DLC)	4-bits field that contains the number of bytes of data being transmitted.
Data	Up to 64-bit field encoded with vendors and vehicles dependent rules.
Cyclic Redundancy Check (CRC)	16-bit cyclic redundancy check contains the checksum of the preceding application data for error detection.
Acknowledgement (ACK)	2-bits field: 1-bit acknowledgment, 1-bit delimiter. The acknowledgment bit is overwritten by the receiving ECU once the message is correctly received.
End of Frame (EOF)	7-bits that mark the end of a CAN frame.
Inteframe Space (IFS)	7-bits inter-frame space.

2.1.1. CAN packets

Four different types of messages can be transmitted on CAN, respectively *Data*, *Remote*, *Error*, and *Overload* frames. *Data* frames represent the vast majority of the channel communication and, as the name suggests, carry data in payloads. *Remote* and *Overload* frames are much less common in current CAN networks. The first serves the purpose of requesting a frame from a specific node, while the second would be used by nodes not keeping up with the network to delay other nodes' communication. Finally, *Error* frames are sent by the transmitter or by a receiver when they detect an error on the packet being sent on the bus to notify that the current packet is not valid. In Table 1, we present the standard CAN packet structure. Inside a single data, field are packed multiple values, each with a specific meaning. In this paper, we refer to these different values as signals. Signals usually contain information regarding sensors and commands from actuators, but they can also contain noisy sequences of bits and CRCs that make their interpretation very difficult for security researchers. The translation of the signals of each CAN ID is considered sensible intellectual property by vendors, which hardly disclose the Communication Database for CAN (DBC) files with their specifications to external entities. This security-through-obscurity approach deeply impacts the design of IDSs for CAN

2.1.2. CAN Security Issues

CAN, being designed with network isolation in mind, lacks security systems of any sort and is, therefore, characterized by security weakness [8].

No authentication: Since CAN is a broadcast network where each ECU receives all messages sent by all other ECUs. Any node on the bus continuously listens to the messages being broadcasted, waiting for the IDs of messages it is interested in. Moreover, any node can write any message on the bus. This implies the feasibility of a wide range of attacks that rely upon sending spoofed messages with a tampered CAN ID.

No encryption: Since CAN needs to be light and fast to maintain live communication requirements, CAN protocol never adopted packet encryption of any sort. This problem makes it straightforward for an attacker to sniff data and perform activities with the aim of tampering with the authenticity and integrity of messages.

Protocol misuse: Since the CAN arbitration system relies on CAN IDs and CAN IDs can be easily tampered with, the CAN protocol can be misused by sending messages with low CAN ID and high priority, thus preventing authentic packets from winning arbitration and ultimately implementing a Denial of Service (DoS) attack. Moreover, in the latest years, a novel approach to CAN attacks has focused on exploiting the physical characteristics of the CAN signal and its error-handling mechanism to silence nodes, again ultimately implementing a targeted DoS attack where only the victim node loses access to the bus [9, 4].

Higher level protocols exploitation: Important to mention, although not directly related, some attacks to higher-level protocols sometimes implemented on CAN, such as the Unified Diagnostic Services (UDS), have proven effective at forcing nodes to send attack data or at shutting nodes down [10].

More details about threats and possible attacks will be addressed in Section 4, but it is clear that the CAN bus is insecure by design. With our work, we want to join the choral research effort to find the best possible intrusion detection approach for CAN.

3. Related Works

Multiple researchers and surveys studied the state of the art of CAN intrusion detection systems [11, 12, 13, 14], and proposed methods to categorize them. We focus on software-based intrusion detection classifying IDSs into frequency-based (which we generalize as Flow-based), content-based (which we generalize as Payload-based), and combined IDSs.

3.0.1. Flow-based Detection

Flow-based IDSs monitor the CAN bus, extract distinct features such as message frequency or packet inter-arrival time, and use them to detect anomalous events without inspecting the payloads of the messages.

Song et al. [15] propose an IDS aimed at detecting message injection attacks by analyzing traffic anomalies, in particular time intervals between messages. The authors' objective is to simplify the process of detecting message injection with regard to other statistical methods in order to get a fast response while keeping the detection rate high, avoiding the need to wait up to a full window of analysis latency before detecting the attack. The model has been tested with 2x, 5x, and 10x message injection rates, and in all cases, the model had 100% accuracy with no false positives.

Other approaches apply Machine Learning (ML) algorithms to analyze the flow on the CAN bus. Taylor et al. [16] propose an IDS based on a One Class SVM (OCSVM) classifier that uses as input a feature vector containing the count, mean, and variance of time differences and mean and variance of Hamming distances for each CAN ID. This method shows an Area Under Curve (AUC) between 0.9620 and 0.9905 and outperforms a simple flow-based detector designed by the authors in several scenarios.

GIDS, proposed by Seo et al. [17], is a two-stage IDS composed of two discriminators with the same Deep Neural Network (DNN) architecture. The first discriminator is trained in a supervised way on a labeled dataset composed of both normal and malicious traffic. The second discriminator is trained with a Generative Adversarial Network (GAN) and learns a model of the CAN bus traffic, trying to discern the true messages from the ones created by a generator fed with random noise to detect unknown attacks. The input of the network are images generated by one-hot encoding the CAN IDs of the packets read on the bus traffic, making this a flow-based method, as the author commented that using the entire packet would make the system not suitable for real-time detection. The system shows a detection rate of 100% for the first discriminator and 98% for the second discriminator with an AUC between 0.996 and 0.999.

Olufowobi et al. [18] implemented a system building an anomaly-based, supervised algorithm to obtain a timing model of CAN messages, attempting to predict a window of time for the completion of a can message transmission. The results are promising since all experiments have a recall of 0.97 or more and no false positives. However, the tests have been done on a small real-world dataset, with four attacks and spoofing only two IDs with relatively similar features (Gear and RPMs).

3.0.2. Payload-based IDSs

Payload-based IDSs examine the payload of CAN packets (usually only data frames). Lately, the research in this field has focused on applying ML algorithms to this task, where it is important to distinguish between supervised and unsupervised ML approaches since this distinction has important consequences on the effectiveness and role of the approaches in intrusion detection tasks. Supervised approaches are limited to the detection of the specific anomalies they are trained on, they are generally more effective on specific attacks but have limited capabilities in the detection of different events. On the other hand, unsupervised ML approaches are able to build a representation of the normal behavior of a network and, for this reason, are potentially able to detect any type of anomaly detectable given the input data while being obviously less specialized on the specific attacks. Reduced Inception-ResNet [19] is a supervised deep convolutional neural network based IDS, which takes from Inception-ResNet, a model designed to classify images in up to 1000 classes. The authors simplify the model for the automotive field in an attempt to lower the computation times. CANTransfer [20] is a supervised convolutional LSTM-based IDS designed to apply transfer learning to CAN detection. The goal of this system is that of using a small batch of attacks to train models capable of detecting also other classes of attacks. CAN-ADF [21] is a framework for CAN anomaly detection and ensembles both a rule-based detector and a supervised Recurrent Neural Networks (RNNs)-based detector attempting to provide not only attack detection but also automatic classification of various types of attacks. TSP [22] uses a supervised LSTM-based model and focuses on studying the best loss function form to provide a higher detection rate and lower false positives. O-DAE [23] is a supervised IDS based on a deep denoising autoencoder, basically using an autoencoder to reconstruct an initial data representation while removing the noise in the signal. E-GAN [24] is an unsupervised technique that uses both a GAN and the OEM-provided packet description (e.g., DBC files) to increase the capability of the system to recognize anomalous behavior in packets. Finally, CANnolo [25] and CANdito [26], are unsupervised IDSs based on a Long Short-Term Memory (LSTM) autoencoder. An RNN-based architecture is effective in modeling time series and has been proposed for CAN traffic anomaly detection even without using an autoencoder [27]. A window of packets is fed to the RNN autoencoder, which attempts to reconstruct it following the trained model. A bad reconstruction represents an anomaly since it implies that the stream of data is not behaving as expected. This method performs well on various data field anomalies, but its major limitation is the long computation time.

3.0.3. Combined IDSs

Zhang et al. propose a two-stage IDS for CAN. The purpose of this work is to strike a balance between efficiency and detection rate thanks to a system composed by a first stage that applies a CAN ID validity rule and the time interval rule proposed by the aforementioned IDS by Song et al.[15] and a second stage that is a DNN. The DNN stage analyzes only the traffic that was not considered malicious by the previous stage in order not to overload this more complex stage in case of increased frequency of the messages during an attack. Although this approach is interesting and different from everything proposed before, the way the second stage is designed may not be ideal for detecting complex payload-based attacks. In fact, most of the chosen input features are flow-based (sequence of CAN IDs, number of occurrences on the last second of incoming message CAN ID, relative distance between message CAN ID) and the last one is entropy that, although demonstrated to be a valid feature for flow-based detection methods [28], in our preliminary tests has proven effective only against a very limited set of CAN IDs with very predictable traffic. HyDL-IDS [29] is a supervised, combined IDS for CAN that exploits CNN and LSTM models to extract both temporal and spatial features of the packets. Rec-CNN [30] is a supervised approach that uses Convolutional Neural Network (CNN) by transforming the detection problem into an image-classification one. To do so, they exploit recurrence plots that graphically describe packets over time, enabling the CNN to detect anomalies in the representation. In the attempt to exploit also the correlation between different CAN IDs, Hanselmann et al. propose CANet [31], an IDS based on an autoencoder that has a separate input LSTM network for the traffic of each CAN ID. The inputs of each LSTM sub-network are the signals of the correspondent CAN ID rescaled with a signal-wise 0-1 normalization. The first layer of the autoencoder is a joint latent vector obtained by concatenating the last output of each LSTM input. At each time step, the current packet is fed into the corresponding LSTM sub-network in order to update the latent vector. The joint latent vector is then processed by a dense layer, another smaller dense layer that represents the bottleneck of the autoencoder, and a final dense layer that has the same size as the total number of signals. This approach has the unique advantage compared to every other state-of-the-art method of being potentially able to detect a larger number of anomalies thanks to the fact that every CAN ID is evaluated at the same time. Nonetheless, the time to test each packet has proven unacceptable for real-time detection on a large set of CAN IDs or in series with other IDSs.

3.1. Goals and Challenges

Following the presentation of related works, we briefly comment on our goals and the challenges we attempt to overcome. The main limitation of current state-of-the-art approaches for packet-based intrusion detection is that, while different methods work well on different problems, none of them is able to achieve good results against every attack and packet and at the same time provide results fast in order to process the traffic of the network in real-time. The way this limitation compels a system largely depends on the different type of IDS approach adopted; generally, flow-based approaches are able to provide fast predictions while being limited to detect very specific kinds of vulnerabilities, while payload-based approaches have a broader scope but it is often a problem to make them work in real-time on the entire traffic. Moreover, the traffic on different CAN IDs has different characteristics, but the current state-of-the-art methods do not consider them in order to provide better results.

Our goal, to overcome the limitations of the state of the art, is to build a modular IDS that is able to analyze the entire traffic on the CAN bus while improving the performances of the state of the art, both by taking advantage of the peculiarities of the different existing approaches and by finding and exploiting specific characteristics of the packets, in order to improve the detection capabilities of the system or lighten the computational weight on the more complex modules. To do so, we aim to modify existing state-of-the-art systems to fit our needs and design new ones if necessary.

While the design of a IDS is not per se an easy task, some additional challenges derive from the modularity of our solution. To be able to put different systems together, there are some additional constraints on performances, timing, and synchronization because a module does not only have to work on its own but also alongside the others, affecting the detection rate, false positives, and time to process the data.

4. Threat Model

Modern vehicles have both a wide internal and external attack surface, and while in most cases adversaries do not have direct access to the internal network (except in particular cases as car-sharing services), more and more options are available as the cars become more connected. While securing all the different components of the attack surface is fundamental, it must be considered that all those technologies are heterogeneous and diverse. We focus on what may happen on the CAN bus, assuming that one or more ECUs have been compromised independently from the channel through which the attack has been performed. Our scope is to evaluate the effectiveness of different kinds of IDSs that analyze the packet traffic on in-vehicle networks and, in particular, the CAN bus, as it is the predominant technology in the industry.

We consider an adversary model where the attacker has already compromised at least one ECU. Our attacker can be either a *weak adversary*, which has access to the CAN bus directly or through a compromised ECU, but his target ECU is not compromised in a way that makes it incapable of sending messages on the bus. This means that a weak attacker, whatever his knowledge of the protocol and of the semantics of the messages, needs to change the frequency of the messages of the target CAN ID on the bus in order to succeed in his exploitation. A *strong adversary*, on the other hand, can prevent its victim from sending messages. This means that it has all the capabilities of a weak attacker while also being able to suppress the messages from the target ECU and spoof it without changing the frequency of messages on the bus. As discussed in Section 2.1.2, by design, the CAN protocol lacks some basic security properties. This means that every adversary with access to the network has some basic capabilities: *Sniffing*, as CAN is a broadcast protocol that lacks any cryptography mechanism, anyone that has access to the bus is able to listen to the messages sent by any ECU, and *Packet Injection*, as CAN protocol lacks any form of authentication, any node can send messages on the bus with any CAN ID, being potentially able to impersonate any other ECU on the network. Addressing these issues is difficult because it requires drastic and expensive protocol modifications. This means that on most vehicles, even a weak adversary has access to a wide range of different attacks.

Spoofing Attack: To try to impersonate a target ECU, the attacker floods the network with messages with the same CAN ID as the target. If the injection rate is higher enough than the normal frequency of the target messages, the other ECUs may accept the injected messages.

Fuzzing Attack: This attack consists of injecting random or partially random messages on the network in order to cause malfunctions. This method has also been used to attempt to reverse engineer the semantics of the messages [3].

Denial of Service (DoS): Similar to previous attacks, but the objective is to saturate the network. The injection rate is higher, and the CAN ID of the injected messages is low (e.g., ID 0x0) as the arbitration mechanism of the CAN protocol gives priority to messages with lower CAN ID.

Drop attack: The compromised ECU is prevented from sending messages (e.g., by putting it into Bootrom mode [10] or with a targeted DoS attack aimed at forcing the victim ECU in bus-off state [9, 4]).

The first step for these types of attacks to be successful is to conform to the protocol and to various rules that would make ECUs refuse the new packets. Several rules can be checked by an IDS in order to easily refuse those malformed messages, for example, by checking their formal correctness with regard to CAN specification (correct size of sections, field delimiters, bit stuffing, CRC), the consistency of eventual counters and CRC in the payload (while often if there are problems in these sections, the ECU will just reject the messages) and of bits that are never used. Even for messages that are formally correct, these types of attack are easily and reliably detected by simple classifiers taking into account the increased (for spoofing, fuzzing, and DoS) or decreased (for drop) frequency of messages of the target CAN ID on the bus. This means that for these types of attacks using complex machine learning algorithms is redundant, as they can be detected with much more simple and less computationally intensive methods.

Strong adversaries, besides all the described attacks, can also perform more sophisticated *masquerade attacks*: the attacker may have control of the specific ECU that sends the target packets, or they may implement a drop attack against the transmitting ECU and send malicious packets, in both cases preserving the perceived frequency of the attacked CAN ID. It is important to note that the second implementation of this attack is not trivial and requires a strong adversary with full IDs and packets knowledge (obtained either through CAN DBC files or extensive reverse engineering) and fine-grained packet injection capabilities regarding CAN frame frequency and format, (e.g., the attacks proposed by Tron et al. [4] and Kulandaivel et al. [32]). In fact, placing a legitimate node into bus-off state, or shutting down the legitimate node, implies that all the IDs sent by that particular node will be missing from the bus. Therefore, the attacker has to be capable of replacing the communication of all the missing IDs (as demonstrated by Miller et al. [10]) and maintaining their correct frequency, otherwise it would be trivial to detect the attack due to the misbehavior of packets, easily recognized by rule or frequency-based IDSs. However, if implemented correctly, differently from attacks that widely impact the traffic on the network, this type of intrusion is very difficult to detect through basic detection approaches. An adversary of this type may try to shape the payload of the sent messages in such a way that makes it difficult for more complex systems to detect the attack. We categorize these attacks into the following two categories: Replay and continuous change attacks.

Replay Attack: the payload of the sent messages is replaced with a time series of messages of the same CAN ID previously sniffed on the bus. This type of attack is intended to trick the IDS into considering the injected traffic as valid.

Continuous Change Attack: a signal inside the payload of the messages is slowly modified until the wanted malicious value is reached. This type of attack is more complex as the attacker needs to know the position of a signal inside the payload and its semantics. An attack of this type may make an IDS believe that the traffic is valid as no abrupt changes in the value of the signal can be detected.

5. CANova

In this section, we describe CANova, the modular framework proposed in this work, which exploits the characteristics of the different CAN packets to select the fittest IDSs for detecting attacks. First, we provide an overview of its functioning, and then, we detail the inner workings of each component.

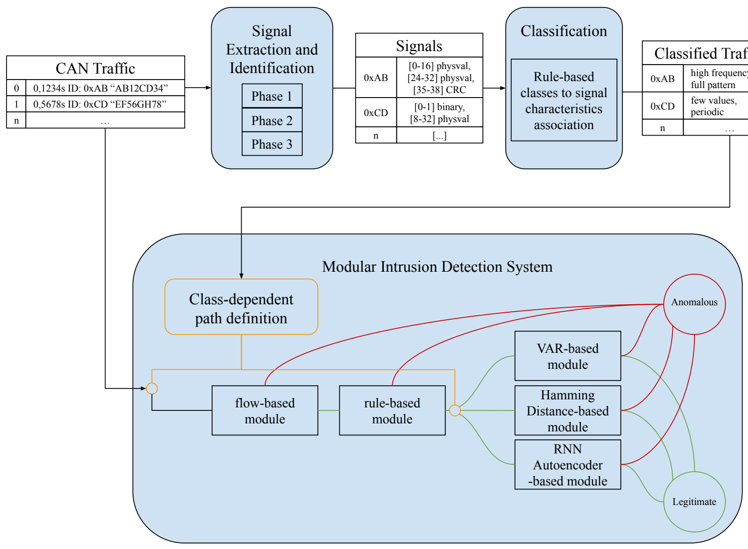

CAN Traffic
0 0,1234s ID: 0xAB "AB12CD34"
1 0,5678s ID: 0xCD "EF56GH78"
n …
Signal Extraction and Identification
Phase 1
Phase 2
Phase 3
Signals
0xAB [0-16] physval, [24-32] physval, [35-38] CRC
0xCD [0-1] binary, [8-32] physval
n [...]
Classification
Rule-based classes to signal characteristics association
Classified Traffic
0xAB high frequency, full pattern
0xCD few values, periodic
n …
Modular Intrusion Detection System
Class-dependent path definition
flow-based module
rule-based module
VAR-based module
Hamming Distance-based module
RNN Autoencoder -based module
Anomalous
Legitimate

Figure 1: The three components of the framework of our work: Signal extraction, classification and modular IDS, and their interactions.

5.1. Approach Overview

As shown in Figure 1, CANova is composed of three components: the signal extractor, the CAN IDs classifier, and the modular IDS. First, CANova extracts the packets' signals using a reverse engineering method based on READ by Marchetti and Stabili [33]. The output of this step is the assignment between CAN IDs and the list of categorized signals. This information is used by the classifier and the IDS modules. The second step of our framework is a classifier, which assigns some attributes to the CAN IDs based on the characteristics and behavior of their packets. These attributes can be either based on frequency-based features (e.g., periodic or non-periodic arrival of packets, high or low packet frequency) or on the content of the payloads (e.g., CAN IDs that have a small number of bit-flips/values of the signals, signals that follow a pattern). Finally, we use these attributes to choose the different IDS modules to assign to each category of packets. In this work, we envision several modules concatenated considering their temporal performance (see Section 6); in other words, "lighter" modules analyze the packets before the more complex ones to lighten the computational burden. Therefore, we first apply "simple" checks on packets' payload and flow to provide fast and precise detection of the less advanced attacks.

Rule-based module: it applies static rules to the traffic (valid packet id, check on a bit that should never flip, check on the correct behavior of pattern sequences).

Flow-based module: it checks the arrival time of packets belonging to CAN IDs showing periodic behavior. This module uses as a reference the inter-arrival time of the packets registered in a real-world scenario to detect anomalous flows on the network.

Packets that are marked as malicious by neither of these modules are then analyzed by one of the different parallel payload-based modules chosen depending on the attributes of the CAN IDs. The idea behind these modules is that they are able to provide the best trade-off between detection accuracy and detection time for at least a category of packets. In particular, we choose the following classes of IDSs, since they performed best in our experimental evaluation depending on packets' extracted features.

Hamming distance-based module: IDS based on the Hamming distance between packets that can detect anomalous events when the value of the packets changes significantly in a small amount of time.

VAR-based module: IDS based on an auto-regressive model that is useful for CAN IDs that have signals that are highly predictable given their last few values.

RNN autoencoder-based module: IDS that can model even complex non-linear relationships inside the traffic of most CAN IDs, at the expense of computational requirements. We have chosen this particular type of architecture because several state-of-the-art works show it to be particularly effective in the same task [27, 25, 31].

It is important to mention that while the modules that we have chosen are the ones that performed best in our experimental evaluation, this may change in the future due to novel detection techniques being proposed. The overarching goal of CANova is not only to propose a viable current solution but also to design and propose a framework that can be easily updateable with novel techniques without re-designing the whole process. In fact, in the event where a novel technique is found that well detects specific types of packets where the current modules underperform, it can be easily inserted by adding the necessary routing rules.

5.2. Signal extraction and identification

To evaluate the characteristics of the signals found in each CAN packet, it is first necessary to extract such signals. To do so, we exploit a reverse engineering method based on READ by Marchetti and Stabili [33]. The output of this step is the assignment between CANs ID and the list of its signals. The second step is a classifier, which assigns attributes to a CAN ID based on the characteristics and behavior of its signals. These attributes can be either based on frequency-related features (e.g., periodic or non-periodic arrival of packets, high or low packet frequency) or on the content of the payloads (e.g., CAN IDs that have a small number of bit-flips/values of the signals, signals that follow a pattern). The output of this classification is then fed to the CANDetector in order to choose the detection modules to use. The extraction of the signals from each CAN ID is necessary since different signals, even if sent alongside the same CAN packet, may behave in completely different ways and, therefore, may require to be handled differently. Therefore, given the full dataset trace, we need to extract signals for each CAN ID. To do so, we expanded Marchetti and Stabili's READ [33]. READ extracts individual signals from CAN traffic by inspecting all the bits of the data field of all observed CAN messages and evaluating their evolution over time. To do so, READ focuses on a CAN ID trace at a time and it does not require any prior knowledge about the nature of the messages' payload. In brief, READ's algorithm is composed of three logical steps. The first step is *PreProcessing*, which computes intermediate data structures needed for the following phases: bit-flip rate and magnitude array. The second step is called *Phase 1*, and in this phase, the magnitude array is used for the definition of preliminary signal boundaries identification. Finally, the last step is called *Phase 2*, and takes as input the preliminary boundaries identified in the previous phase together with the bit-flip rate array to identify and correctly label signals. The output of *phase 2* is the list of signals and their categorization for each CAN ID. READ's *phase2* identifies three types of signals: *PHYSVAL*, *COUNTER*, and *CRC*. Our custom version of *phase 2* adds a fourth one: *BINARY*.

On top of this basic classification made essentially over the analysis of bit flips, we expanded the signal extraction technique by adding another step: *phase 3*. The scope of *phase3* in our modular system is the identification of further characteristics of signals based on features different from the only bit flip ratio and its magnitude used in the first phases. In particular, during *phase3*, we perform two different checks: a *recurrences check*, where signals that present a pattern behavior are marked as *PATTERN*, and an *auto-correlation check*, where signals' auto-correlation coefficients are computed and the ones which present very high values are marked as *HIGH AUTO-*

CORRELATION. This signal-related information is used to eventually enrich the output of *phase 2* with the aforementioned additional categories.

Pre-processing. Our *pre-processing phase* analyzes the messages' payloads and computes the information used by the next phases. The bit-flip rate is evaluated for each bit of the payload, independently of its neighbors. The number of times each bit flips its value is counted for each of the n bits that compose the payloads; the bit-flip rate is then calculated as the ratio between the count of the flips and the total number of packets for the considered CAN ID. Then each element of the magnitude array is calculated as $M_i = \lceil log_{10}(B_i) \rceil, \quad 0 <= i < n$, where B is the array of bit-flip rates of the considered CAN ID. The values of the magnitude array represent the different orders of magnitude of the bit-flip rate for each bit of the payload and they are used in *phase 1* to compute the preliminary boundaries of the signals.

Phase 1. The objective of *phase 1* is to find the preliminary bounds of the signals for each CAN ID (i.e., bounds that contain one or more signals). For each CAN ID the steps of *phase 1* of the original algorithm consist in scanning the magnitude array starting from the first bit, searching for a significant magnitude decrease (which marks the end of a preliminary bound), and repeat from the successive bit to the end of the payload. The idea behind the algorithm is that a decrease of the magnitude indicates the transition from the least significant bit of a signal to the most significant bit of another signal or to a section that does not include any signal. The only novelty that we added in *phase 1* is that every time we find a preliminary bound, we skip all the bits with magnitude $-\infty$ and identify it as the start of the next preliminary bound the first bit with non-infinite magnitude. By doing so we avoid including bits that never flip into the preliminary bounds.

Phase 2. Contrary to the choice for the first two steps of the signal extraction, we more significantly modified the design of *phase 2*. The *Phase 2* function takes as input the preliminary bounds of the signals found in *phase 1* and the bit-flips calculated during *pre-processing*. The purpose of *phase 2* is to find the correct boundaries and type of one or more signals contained in each preliminary bound. Our *phase 2* is designed to be able to identify not only the starting bit of the preliminary bound like in the original approach but also the ending bit, which allows us to be more precise in the recognition of signals. The categories of signals produced by *phase 2* are shown in Table 2.

Phase 3. *Phase 3* is a completely novel contribution to the signal extraction

Table 2: The different types of signal categories identified by phases 2 and 3 of our Extraction and Identification step.

Phase 2	
CRC	Signals that contain the result of a cyclic redundancy check on the message payload to detect transmission errors. These signals have bit-flips that follow a normal distribution with a mean of 0.5.
Counter	Signals whose value always increases or decreases by one with respect to the previous message's counter with the same CAN ID. These signals have bit-flips that approximately double after every bit and have the least significant bit with bit-flip around 1 (e.g., a 4 bits counter has a bit-flip array of values close to [0.125, 0.25, 0.5, 1]).
Physval	Signals that represent a value read from a sensor (e.g., speed, steering angle, gas tank level). We consider *PHYSVAL* signals that do not belong to any of the previous categories.
Binary	One-bit long signals
Phase 3	
Patterns	Signals are considered patterns if they have a periodic pattern that is repeated for the whole dataset.
High Auto-correlation	A signal is considered highly autocorrelated if it is a "physval" and the correlation between the signal and itself, shifted by one time-step, is greater than 0.9.

process, meant to find signals' features that are independent of their bit-flip ratio and magnitude, but that are useful for the following classification step. The outputs of *phase 3* are the same signals found by *phase 2*, with some of those signals enriched by additional attributes. The categories of signals added by *phase 3* are shown in Table 2.

5.3. Classification

Once we extract all the properties from signals we can finish our classification of the packets by identifying distinct categories of CAN IDs on the basis of their attributes. Once the characteristics of the traffic are clear and attributes are assigned, we can define associations between CAN IDs to IDSs on the basis of the set of attributes they are given. We focus the design of our classification and mapping steps on maintaining two characteristics: the first is to build both a modular classification and mapping. In fact, new attributes could come up as meaningful in the future, and our framework would be able to integrate them without requiring a complete reconfiguration. Moreover, new detection techniques could be identified as useful for some specific IDs, and in the same way our framework would be able to integrate them without a complete reconfiguration. Second, we want our system to be automatic, which means that attributes must be assignable without any human supervision. This is fundamental for the framework to be scalable on different vehicles and configurations.

The Classification step takes the output of the signal extractor and calculates different features that we empirically recognized as useful for the successive choice of an IDS. The list of possible features is shown in Table 3.

5.4. Modular Intrusion Detection System

Once the classification of the various packets has been completed, we focus on the design of the actual modular IDS and how the various detection modules interact with each other to obtain the evaluation of each packet. Our focus in the design of the interaction between modules is manifold: The detection system as a whole needs obviously to be effective, as it has to detect the vast majority of attacks; The detection process needs to be fast since the system is designed to be implemented on-board of a vehicle and needs to keep up with the live traffic; Following the same reasoning, the detection system needs to have low latency, even if multiple detection modules process the same packet. Last but definitely not least in terms of importance, the detection system as a whole needs to have low False Positive Ratio (FPR), since the only way for it to be actually implementable in real-world scenarios is by not being more of an annoyance than a solution.

Our intuition to handle these requirements is to implement the intrusion detection modules either in series or in parallel, depending on the module characteristics. We can make various considerations regarding our focus on the design and the properties of modules in series or parallel. In an optimal scenario, where no module has false positives, the solution with the highest detection rate would be the one with all the modules in series, from the computationally fastest one to the slowest. In this way, all modules would have the chance of detecting anomalies, but if the anomaly could be detected by the fast ones, it would not end up taking up unnecessary computation time. Similarly, in an optimal scenario, the solution with the fastest output would be the one where all detection modules are inserted in parallel, and each packet/ID is evaluated only by one module. However, as further discussed in Section 6, some modules are particularly effective in detecting some attacks, independently from many properties of the packet, but are incapable of recognizing others (e.g., flow-based detection cannot recognize

Table 3: All the possible attributes can be assigned to a CAN ID. Note that attributes-ID associations are not unique, meaning that an ID may be assigned multiple attributes if it follows the requirements needed for them.

Classification	
Always constant	CAN IDs for which payloads are always constant in the training set. A CAN ID is marked as *always constant* if the signal extractor indicates there are no signals for that CAN ID.
Few values	CAN IDs for which the payloads have a minimal set of possible values. A CAN ID is marked as *Few values* if the different values assumed by its payloads are less than a predefined threshold (10 values for each dataset).
Few flips	CAN IDs for which the bits of the payloads in the flip a minimal number of times. A CAN ID is marked as *Few flips* if the bits of its payloads flip value a number of times that is less than a predefined threshold (10 flips for each dataset).
Few flipping bits	CAN IDs with a single bit that flips.
Not enough packets	CAN IDs that have a minimal number of packets. A CAN ID is marked as *Not enough packets* if there are less than a certain amount of packets (40 packets for each dataset) from the CAN ID.
Full pattern	CAN IDs with payloads containing only signals that follow a pattern. A CAN ID is marked as *Full pattern* if all its signals are identified as *pattern* by the signal extractor.
Partial pattern	CAN IDs with payloads containing at least one signal that follows a pattern.
High frequency	CAN IDs with a small average packet inter-arrival time (less than 21 ms).
High auto-correlation	A CAN ID is marked as *High auto-correlation* if all its *PHYSVAL* signals are at least 4-bits long, are identified as *high auto-correlation* by the classifier, and has a high packet frequency (period $\leq$ 20 ms).
Full binary	CAN IDs that have only signals composed by a single bit.
Non-periodic	CAN IDs that do not have a periodic behavior. A CAN ID is marked as *non-periodic* if the maximum period calculated on the different training datasets is bigger than two times the minimum period calculated on the training datasets or if the average interval between packets, multiplied for the number of packets, is less than 0.97x the actual duration of the traffic of the CAN ID.
Semi-periodic low	CAN IDs that behave similarly to periodic ones but have some high-frequency events. We classify a CAN ID as *semi-periodic low* if there is at least one packet that satisfies this condition: $(timestamp - last\_timestamp) < 0.3 * average\_interval \land (timestamp - second\_last\_timestamp) < 0.8 * average\_interval$.
Semi-periodic high	CAN IDs that behave similarly to periodic ones but have some low-frequency events. We classify a CAN ID as *semi-periodic high* if there is at least one packet that satisfies this condition: $(timestamp - last\_timestamp) > 2 * average\_interval$.
Semi-periodic	CAN IDs that are both *semi-periodic low* and *semi-periodic high.*
Periodic	CAN IDs that have a strong periodic behavior. A CAN ID is marked as *periodic* if it is neither *non-periodic* nor any type of *semi-periodic.*

masquerade attacks but easily detects DoS). It follows that the real-world best solution is an in-between that both keeps a high detection rate over all possible attacks, and at the same time does not require a series of modules that are excessively long, in order to avoid delays.

Another (although already anticipated) relevant design choice is whether a packet is forwarded to a second detection module when it results anomalous or legitimate. This question, although important, is easily answered. In fact, since as we already mentioned some detection modules are extremely efficient in detecting specific attacks but cannot detect all, choosing to forward only anomalous packets to confirm the result would lead to some attacks passing undetected. Our best choice is to forward packets when they are evaluated as legitimate, to ensure that all types of attack can be detected. This choice, however, has an important drawback, which is that in a standard context the vast majority of packets are legitimate, and therefore the computational overhead and overall delay for each result increases. It is fundamental, therefore, while choosing the detection modules and their layout, to avoid inserting in series a set of modules that are all computationally expensive.

Finally, as previously mentioned, different detection methods are effective against different types of attacks, hence it is necessary to evaluate a packet with detection methods that complement each other, and to avoid instead to insert in a chain of detection multiple detection modules that focus on similar attacks. Our extensive analysis of detection systems led us to assert that it is always effective to insert, where meaningful, as the first two modules a flow-based and rule-based. The reasoning behind this choice is that flow and rule-based modules are orders of magnitude faster than other payload-based modules while having essentially no false positives.

To summarize our modules interaction choices, we define a set of rules:
(1) Modules are inserted in series or in parallel.
(2) A packet that is considered anomalous when it is evaluated as such by one module. A packet is considered legitimate when it is evaluated as such by all the modules that should evaluate it.
(3) Since the system has to be effective in detecting all types of attacks, modules that are effective in the detection of different types of attacks have to be inserted in series.
(4) Since the system has to be fast, and different modules have significantly different testing times, the ones with lower testing time have to be inserted first, so that the simpler modules have the possibility to detect trivial attacks and avoid useless computation.

(5) The latency of a negative prediction is the sum of the latencies of all the modules in series. Therefore, it is fundamental to ensure that such sum of latencies never exceeds the interarrival time between two packets with the same ID.
(6) The FPR of the system is negatively affected by modules in series, since each module has the theoretical possibility of detecting false positives. It is hence necessary to avoid chains of modules with relevant false positive rates for the system to be implementable in real-world scenarios.

Modules mapping. Finally, CANova requires a methodology to map the various IDS modules to the different CAN IDs. Note that this step is the only one in the analysis process that requires complete knowledge of the IDS modules that are being used. Specifically, the modules we implement are a flow-based IDS, a rule-based IDS, an RNN-based autoencoder, a hamming-distance-based IDS and finally a VAR-based IDS.

To choose which ID is assigned to which set of modules, we empirically designed a hierarchy of rules that link an IDS to an ID depending on its attributes. "Hierarchy" means that where a CAN ID is assigned with more than one attribute, only the policy corresponding to the highest rule on the list is applied. The only exception to this is the flow-based module, which is applied first to all IDs that show periodic behavior. The hierarchy of rules is as follows:
(1) *Not enough packets* The CAN IDs marked with this attribute are an exception: no predictions are viable for them because of the limited number of data and the lack of DBC to know their real behavior, therefore no IDSs either in series or parallel is associated with them.
(2) *Always constant* No more than simple checks are feasible. Therefore only the rule-based module is used on them.
(3) *Full pattern* Full pattern CAN IDs traces just need to be checked with the rule-based module because any possible attack would trigger a pattern violation.
(4) *Few flips and Few values* CAN IDs that include this couple of attributes can be seen as a generalization of the *Always constant* class attribute; as a matter of facts packets of the two classes share most of the features and we can see the *always constant* trace as the extreme case: zero flips and just one value. Therefore, besides the rule-based module, we enforce the use of the Hamming distance-based IDS, to check that the packets do not change too much in a short amount of time, in particular, to detect fuzzing attacks.

(5) *Few flipping bits* CAN IDs that include this attribute are handled with the Hamming distance-based IDS (if not already assigned with other IDSs), so that, even if the check upon the hamming distance between two consecutive packets is not particularly effective, the check over the average Hamming distance between a bigger set of packets can detect anomalies associated with unusual bit flipping ratio.
(6) *High-auto correlation* CAN IDs marked with these attributes are analyzed with the VAR-based IDS, besides the rule-based module, which like on all the other CAN IDs enforces some simple rules. On the other hand, the VAR-based IDS can be used on this set of CAN IDs because, as shown in Section 6.3.3, it has a similar effectiveness on them as the more general RNN autoencoder-based model while being an order of magnitude faster in providing the results.
(7) *Others* All the CAN IDs that cannot be classified through the above list of rules are handled with the RNN autoencoder module. This IDS is able to handle the widest range of CAN IDs, and it is therefore used on every packet that cannot be better classified.

We proceed to describe the list of modules that we selected, presenting their design and the role that we envision for them in CANova. As already mentioned, while the following modules were chosen because they have performed the best in our preliminary evaluation for specific categories of packets, the strength of CANova derives from its modularity, which means that if novel techniques are found that perform better against specific categories of packets, adding such modules requires only to update the set of rules that enable to route the given packets towards the new module.

Flow-based Module. An overview of the current state of the art regarding flow-based IDSs has already been provided in Section 3. Amongst the various options, we opted to implement the system based on the measure of time intervals between packets proposed by Song et al. [15], since it has been proven by its authors to have a 100% Detection Rate (DR) on flow-based attacks with no false positives, and low computational overhead, which is exactly what we search for in the module that is inserted first. While implementing this system, however, we noticed a set of CAN IDs with a considerable number of false positives. In fact, upon further inspection of the data that caused the false positives, there is a case that is not considered in the original implementation: some IDs, while generally behaving similarly to the other periodic ones, present some short events with higher traffic frequency (our classification step already categorizes these CAN IDs as *semi-periodic low*) or events with lower traffic frequency (*semi-periodic high*). We have modified the original IDS to deal with this problem thanks to the categorization of the CAN IDs that cause the most false positives. Our new flow-based IDS uses the minimum packet inter-arrival time registered on real CAN traffic for these CAN IDs, instead of the average packet inter-arrival time, in the anomaly detection process. The second modification, to further lower the possibility of false positives, is to extend the check on the message interval from only the last packet to the two packets prior to the current one. This avoids considering anomalous delays of a single message, that may be caused by arbitration and not by an attacker. We made a preliminary evaluation of the Flow-based module alone to ensure that our modifications did not lower its detection capabilities, and confirmed that it has 100% accuracy for injection attacks and extremely close to 100% on drop attacks, provided that such attacks are 2 or more packets long, as expected given the second modification. As also shown by Miller and Valasek in [34, 10], high injection rates are necessary to perform effective injection attacks, and at such injection rates, the system is still effective.

Rule-based Module. The rule-based module, as the name suggests, performs checks on the traffic by enforcing predefined rules. In general, the more details the designer has on the system, the more rule-based checks can be added. However, the main goal of this module in our system is that of freeing other, heavier modules from detecting simple attacks. Therefore, we implemented only the following rules:
(1) CAN IDs are whitelisted (in our case, only IDs that were seen in the training set are allowed).
(2) Fixed bits, which are common in CAN traffic, cannot change.
(3) Signals that follow a fixed pattern, which again are common in CAN traffic, cannot divert from their pattern (we classify them as PATTERN in the classification step).

The reason behind the choice of these rules is that we consider them free from false positives in a real-world scenario. In fact, while we do not have complete knowledge of the various IDs in the dataset and may, for example, blacklist an ID that appears only in the test set, in an industrial setting the list of plausible IDs is defined, similarly to the other rules.

Hamming distance-based Module. The Hamming distance-based IDS approach is inspired by a work by Stabili et al. [35]. Hamming distance is used to measure the minimum number of substitutions required to change one string into another. Since it was first proposed in 1950, it has been used in several disciplines for the most part related to, but not limited to computer science. The generic formula for computing the Hamming distance between two words of equal length k is $H_d(x, y) = \sum_{i=1}^{k} |x_i - y_i|$. The idea behind Hamming distance-based IDS is to compute acceptable hamming ranges for consecutive packets of the same CAN ID trace during the training phase, and consider as anomalies the transitions that violate such range. Stabili et al. consider the Hamming distance of a single transition for detecting the anomaly. However, our empirical analysis has shown that it is more effective to use the sum of the Hamming distances of three consecutive transitions. In fact, this enables the identification of attacks even in CAN IDs that have only a small amount of bits that flip, where instead on a single transition the Hamming range would always be the maximum. Undeniably, this approach has several limitations against some complex attacks such as a replay attack, since they may not violate such ranges. However, as further discussed in our experiments (Section 6.3.4), it causes minimal false positives. From our study of the state of the art and practical analysis, for some classes of CAN IDs there are currently no better detection options.

VAR-based Module. Our VAR-based module is the only one that is not an adaptation of other work but has been designed specifically to lower the overall computation requirements of CANova by assigning a specific class of CAN IDs to a module that is less computationally expensive than others. In fact, VAR is a stochastic process model that generalizes AR models in order to predict multivariate time series. If we assume that the different signals (excluded counters and CRCs) inside the payload of a CAN ID are interrelated, VAR can be a good option to recognize such relationships and predict the future values of the signals. A generic VAR(p) process (i.e., a VAR model in which the variables depend on the past p time steps) with K variables is defined as [36]

$$y_t = v + A_1 y_{t-1} + A_2 y_{t-2} + ... + A_p y_{t-p} + u_t \tag{1}$$

Where y is a K-dimensional vector representing the K variables at the current time step t, v is a k-dimensional vector of constants, y_{t-i} is the K-dimensional vector representing the variables at time step $t - i$, A_i is the K x K matrix of coefficients for time step $t - i$, $u_t \sim \mathcal{N}(0, \Sigma_u)$.

As the error term u_t is unpredictable the one-step ahead predictor can be expressed as:

$$\hat{y} = v + A_1 y_{t-1} + A_2 y_{t-2} + ... + A_p y_{t-p} \tag{2}$$

A similar category of IDSs can be seen as an alternative to more complex payload-based IDSs for CAN IDs that have highly predictable signals. In fact, while the VAR-based IDS does not perform on average, on specific CAN IDs it has an effectiveness comparable to more complex ML methods while providing much faster predictions. We include these highly predictable CAN IDs in our classification in the category *high auto-correlation*. This IDS module is composed of a VAR(1) model for each CAN ID with more than one signal and a AR(1) model for each CAN ID with a single signal. The input variables of the model of each CAN ID are the time series of its non-counter, non-CRC signals rescaled into the 0-1 range. After the models are fitted on a training sequence, the detection threshold of each model is computed on a sequence of other n packets from the corresponding CAN ID. The first packet of the sequence is used to initialize the history window, the others compose an input matrix Y of size $n - 1$ x k, where k is the number of non-counter, non-CRC signals of the considered CAN ID. For each packet the one-step ahead predictor is computed using Equation 2 (with $p = 1$), the predictions compose a matrix $\hat{Y}$ of size $n - 1$ x k. The reconstruction error matrix is computed as $E = \hat{Y} - Y$. Similarly to what was done in [37], [38] and [25], the reconstruction error matrix is fitted to a multivariate Gaussian distribution with mean μ (a k-dimensional vector), and covariance matrix Σ (a matrix of size k x k). These parameters are thus used to compute the anomaly scores, which are the Mahalanobis distance between the one-step ahead reconstruction error on each packet and the computed distribution of non-anomalous packets. Using the Mahalanobis distance has the advantage of considering not only the average value of each of the k variables but also their variances and covariances in order to provide a statistical measure of how much a new prediction is similar to the ones made on clean data [39]. For each packet in the threshold set the anomaly score is computed using the Mahalanobis distance as follows:

$$s = (e - \mu)^T \Sigma^{-1} (e - \mu) \tag{3}$$

Where e is the one-step-ahead prediction error of the tested packet. The detection threshold is computed as the 99.99 percentile of all the anomaly scores computed on the threshold dataset. As soon as a new packet to be tested arrives, the one-step-ahead prediction for the current packet is computed and the testing error is computed as $e = \hat{y} - y$, where $\hat{y}$ is the k-dimensional vector of predictions at current time step and y is the k-dimensional vector of input signals at a current time step. The anomaly score for the current packet is computed using Equation 3, measuring how much the new packet is different from the normal distribution computed on data without anomalies. If the test score is greater than the detection threshold the packet is marked as anomalous. To apply VAR to intrusion detection, a model is fit on a clean (i.e. without anomalies) sequence; the detection threshold is calculated on the basis of the errors of the one-step-ahead prediction on a different clean sequence. Anomalies on new data can be defined as prediction errors on the one-step-ahead prediction that exceed the threshold.

RNN autoencoder-based module. The RNN autoencoder module, based on Longari et al. CANdito [26] is capable of detecting payload-based anomalies on the vast majority of CAN IDs. The main drawback is that it is significantly more computationally expensive than the other modules. Thus, this module is used to evaluate the packets that do not have a faster alternative with comparable or better effectiveness. An RNN-based architecture is effective in modeling time series and has been proposed for CAN traffic anomaly detection even without the use of an autoencoder [27]. On the other hand, an autoencoder-based architecture provides several advantages: autoencoders are simpler to train because the target signals are generated automatically from the input sequence, avoiding the need for a labeled training dataset; autoencoders, thanks to being trained in an unsupervised way, learn a model of the traffic of the network, not the specific anomalies. This means that autoencoders are potentially able to detect attacks not considered by their designers or zero-day attacks. Briefly, the architecture of this module is composed of an input layer of dimension n x k, where $n = 40$ is the dimension of the window of packets, k is the number of signals (excluded counters and CRCs), thus it is variable for the different CAN IDs. The input signals are rescaled in the 0-1 range, a dense layer with 128 units. The activation function of choice is Exponential Linear Unit (ELU) [40], and two LSTM layers with 64 units each. The cell and hidden states of the last LSTM layer of the encoder are used to initialize the states of the first LSTM layer of the decoder. For further details on the architecture and design, we refer the reader to CANdito [26].

6. Experimental Validation

This section presents the experimental evaluation performed to demonstrate the effectiveness of CANova. First, we show the capabilities of our packets' classifier on a public real-world dataset of legitimate CAN traffic. Then, we demonstrate the effectiveness of the intrusion detection modules proposed in this work by comparing their performances on a set of synthetic attacks injected in the aforementioned dataset over the subset of best-fitted CAN IDs. Finally, we show that our CANova is more effective than state-of-the-art solutions on a public dataset with real-world attacks.

6.1. Datasets

We evaluate our work through two public datasets: the ReCAN C-1 dataset [41] used for the first set of experiments and the car-hacking dataset [19] used for the comparison with the state of the art.

6.1.1. ReCAN C-1 dataset

The ReCAN C-1 dataset [41] contains nine logs of real CAN traffic of an Alfa Romeo Giulia Veloce. For repeatability purposes, we report in table 4 how the various parts of the dataset have been used. In brief, datasets 1, 2, 6, 8, and 9 have been used to train the different models. Being the longest log, only dataset 9 has been used for the models trained on a single dataset, while for models trained on more than one dataset, all these 5 datasets have been used. Dataset 4 has been used for computing the detection *thresholds* for the different modules. Dataset 5 has been used for validation and hyperparameter tuning. Finally, dataset 7 has been used only for testing. To generate the attacks used for testing, we use an attack generator tool presented in [26]. Briefly, it is a tool to generate datasets that contain a given anomaly starting from an already existing CAN log. In our case, we generate the attack datasets providing as input to the tool dataset of the ReCAN C-1 dataset, which is the one we selected for testing.

Injection Dataset: The attacks contained in this dataset consist of packets added to the existing log, maintaining the original packets unchanged. The new packets are generated by copying previous traffic and modifying it with simple rules. The anomalous packets are added with periods that range between $\frac{1}{2}$ and $\frac{1}{20}$ of the original one, each sequence has a length between 10 and 50 packets, and at least one physical signal of the packet is modified.

Drop Dataset: This dataset is meant to simulate an attack where the adversary turns off an ECU or its CAN interface and consists of removing sequences of packets from the original CAN traffic. For each attack, 25 packets with the same ID are removed from the log.

Masquerade Dataset: The attacks contained in this dataset consist in substituting already existing packets and changing their payload to avoid modifying their arrival frequency. Aside from this, each anomalous sequence is 25 packets long, at least one physical signal is always modified, and some signals are set to a previously captured random value.

Fuzzed Dataset: This dataset simulate fuzzy attacks against various ECUs. Since detecting attacks if the constant bits are randomized becomes trivial through the rule-based module, these bits are left untouched. Moreover, this attack is not implemented in an injection but in a masquerade fashion.

Seamless Change Dataset: The attacks contained in this dataset simulate a strong attacker that performs a masquerade attack designed to evade IDSs by changing the payload of packets from a legitimate value to a tampered one progressively. As a recall, this attack may be implemented by directly controlling the target ECU or by performing a drop attack and then replacing the payload of the affected packets (i.e., the missing IDs of the shut-down ECU). To consider the worst-case scenario in which the attacker can perfectly perform such an attack (i.e., no frequency misalignment or other inconsistencies), we implement it by directly replacing the packets of the target ID. By doing so, we are simulating an attacker that has full control of the tampered ECU and is simply re-transmitting the other affected IDs. This attack is generated on CAN IDs that have at least one physical signal with a length of 4 or more bits, and only one physical signal is modified per attack. Again the anomalous sequence is 25 packets long.

Replay Dataset: The attacks in this dataset are sequences of packets identical to legitimate ones previously recorded from the same ID. These attacks do not necessarily refer to a real-world scenario since they are not implemented by the adversary with a goal in mind if not that of evading the IDS. However, they are valuable to compare the ability of the various modules to detect sequences that are anomalous due to the context in which they are inserted, but would be valid in another one. This attack has been implemented with the same assumptions as the previous one.

Table 4: The ReCAN C-1 dataset [41] with the details on how each dataset was used

n.	CAN-IDs	Frames	Used for
1	77	3.062.691	Testing
2	76	364.863	Training
3	76	30.005	Not used
4	83	3.227.315	Training (thresholds)
5	83	1.473.625	Validation
6	83	1.684.769	Training
7	83	2.723.484	Testing
8	82	1.569.776	Training
9	88	10.942.747	Training

6.1.2. Car-hacking dataset

The car-hacking dataset [19] is composed of logs of real-time CAN messages via the onboard diagnostic (OBD-II) port of two running vehicles (KIA Soul and Hyundai Sonata) with message attacks. It has four data features, including timestamp, identifier (ID, in hexadecimal format), data length code (DLC, valued from 0 to 8) and data payload (8 bytes), and the label of a CAN message. We refer the reader to [19] for further details on the public dataset under consideration. It contains normal CAN messages (14,237,978) and anomaly messages (2,331,497) belonging to three categories of attacks (for a total of four attacks).

DoS attack: It aims to flood the CAN bus with numerous forged messages with low ID values in a short time interval. Thus, almost all the communication resources are occupied so that messages from other nodes will be delayed or blocked.

Fuzzy attack: Fake messages are sent from malicious ECUs into the CAN bus at a slower rate than the DoS attack.

Impersonation attacks:, They realize unauthorized service access by spoofing legitimate authentication credentials, such as **spoofing the drive gear and the RPM gauze**.

6.2. Evaluation Metrics

Before moving on with the experimental setup and the discussion of the performed experiments, we briefly present metrics and definitions necessary to comprehend the evaluation of the proposed methods' performances.

True Positive Rate: $TPR = \frac{TP}{TP+FN}$. Also known as Detection Rate or Recall, is defined as the ratio of the number of detected intrusions over the total number of actual intrusions. A TPR of 1 means that the model can find all the intrusions, while a TPR of 0 means that the model cannot find any intrusions. Note that the False Negative Rate (FNR) can be obtained as $FNR = \frac{FN}{FN+FN} = 1 - TPR$**.**

False Positive Rate:: $FPR = \frac{FP}{TN+FP}$ FPR refers to the percentage of normal traffic incorrectly classified as an intrusion. A FPR of 1 means that all the normal traffic is incorrectly classified, while a FPR of 0 means that the model commits no errors in classifying normal traffic.

Area Under the Curve: AUC is the area under the Receiver Operating Characteristic (ROC) curve, which is a visualization of the performance of a classifier and can be drawn by plotting the FPR vs. TPR of a classifier and it is a common metric to deal with data imbalance. A perfect classifier has an AUC of 1. We use AUC to evaluate all of CANova's modules that have predictions depending on a score and a threshold.

Precision: $Precision = \frac{TP}{TP+FP}$. It is the ratio of TP to the number of samples detected as attacks.

F1-score: $F1 = \frac{2\ TP}{2\ TP+FP+FN}$. It is the harmonic average of precision and TPR. F1-score values range from 0 (bad classifier) to 1 (good classifier).

Matthews Correlation Coefficient (MCC):

$$MCC = \frac{TP\ TN - FP\ FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

It measures the quality of the detection rate in terms of the correlation coefficient between the observed and predicted classifications. It is a value between -1 (bad classifier) and +1 (good classifier). Its main advantage against other metrics is to be not influenced by dataset imbalances, always providing a sound performance value.

Testing Time per Packet (TTP): $TTP = \frac{Total\ detection\ time}{Number\ of\ packets}$ It is the average time needed to evaluate each packet. All the tests were executed on an Intel(R) Core(TM) i7-8700K Central Processing Unit (CPU) and a Nvidia

Table 5: Periodicity-based attributes (i.e., attribute that defines packets periodicity), their CAN IDs assignments count and their packet percentage count

Attribute	CAN-ID count	Packet count %
NON-PERIODIC	11	0.01%
PERIODIC	57	77,32%
SEMI-PERIODIC	1	0,11%
SEMI-PERIODIC HIGH	8	21,13%
SEMI-PERIODIC-LOW	9	1,44%

Table 6: Attributes not based on periodicity, their CAN IDs assignments count and their packet percentage count. Note that since multiple attributes can be assigned to the same CAN ID, the number of CAN IDs and packets exceeds the real total number.

Attribute	CAN-ID count	Packets count %
NOT ENOUGH PACKETS	7	0,0003%
ALWAYS CONSTANT	15	4,1057%
HIGH FREQUENCY	29	92,5541%
HIGH AUTOCORRELLATION	11	37,3972%
FEW FLIPPING BITS	6	0,9874%
FEW FLIPS	17	3,4046%
FEW VALUES	28	13,3762%
FULL BINARY	13	5,9269%
PARTIAL PATTERN	3	4,4002%
FULL PATTERN	3	4,4002%
NO CLASSIFICATION	21	7,6615%

GeForce GTX 1080 Graphics Processing Unit (GPU) running Ubuntu 20.04.2 LTS (Focal Fossa). This metric provides an estimation of the applicability of an approach in a real-world scenario.

6.3. ReCAN Dataset Experiments

6.3.1. Packet Classification Experiments

CANova is, to the best of our knowledge, the first approach to propose a classification of packets for intrusion detection on CAN. Moreover, this type of classification does not lend itself to provide a metric to maximize for a comparison. However, to explain the evaluation that our classification provides and to discuss on the insights that are achievable through it, we briefly present the results of our classification on the ReCAN dataset.

Our results show that a significant amount, almost 23% of packets (see

Table 7: RNN autoencoder-based module vs Flow-based module performances comparison.

		MCC	FPR	AUC	F1	TPR	TTP
Injection	RNN-based IDS	0.4150	0.0277	0.8716	0.3826	0.6901	0.4715 ms
	Flow-based IDS	**0.9985**	**0.00004**	-	**0.9985**	**1.0**	**0.0050 ms**
Drop	RNN-based IDS	0.3346	0.0238	0.6262	0.3474	0.2300	0.4714 ms
	Flow-based IDS	**0.9997**	**0.00004**	-	**0.9998**	**0.9998**	**0.0056 ms**

Table 5), is not fully periodic. It is therefore necessary, to avoid excessively narrowing the field of applicability of an IDS, to take this into consideration when presenting flow-based solutions.

6.3.2. Flow-based module vs RNN autoencoder-based module

This experiment is meant to show how flow-based approaches are more effective in detecting the anomalies that change the frequency of packets in the network than payload-based approaches, as stated in Section 3. To do so we compare our flow-based module and our RNN autoencoder-based module, that are improved version of state-of-the-art techniques that we have identified as the best alternative for respectively flow-based IDSs and payload-based module. The experiment consists of comparing the performance of our flow-based module and RNN autoencoder module on different tampered datasets. As flow-based IDSs do not analyze the payload of the packets we only focus on the datasets that contain anomalies that either increase or decrease the number of packets in the network (Injection dataset and Drop dataset). The experiment is limited to the CAN IDs that show some periodic behavior (i.e., classified as *periodic* or *semi-periodic*), as flow-based analysis is not possible on the traffic of non-periodic CAN IDs. Finally, the measure of the average AUC among the CAN IDs is not provided for the flow-based module, as it does apply a fixed rule and does not use a threshold to disambiguate between messages that are anomalous or not.

The results (shown in Table 7) show that the RNN autoencoder-based is less effective in the detection of the analyzed anomalies than the Flow-based module. Not only the Flow-based module has almost perfect performance on both the Injection dataset and the Drop dataset, but the TTP is also two orders of magnitude faster. We can easily conclude that adding the Flow-based prediction in the sequence does not add significant computation while providing valuable detection capabilities for specific attacks.

Table 8: VAR-based module and RNN autoencoder-based module performances comparison.

		MCC	FPR	AUC	F1	TPR	TTP
Masquerade	VAR based IDS	0.9180	**0.0089**	0.9600	0.9385	0.9055	**0.0458 ms**
	RNN based IDS	**0.9328**	0.0105	**0.9737**	**0.9502**	**0.9309**	0.4560 ms
Seamless Change	VAR based IDS	0.8523	**0.0090**	0.9782	0.8843	0.8116	**0.0464 ms**
	RNN based IDS	**0.8980**	0.0104	**0.9801**	**0.9230**	**0.8810**	0.4613 ms
Fuzzed	VAR based IDS	**0.9834**	**0.0090**	**0.9997**	**0.9878**	**0.9994**	**0.0472 ms**
	RNN based IDS	0.9802	0.0104	**0.9997**	0.9856	0.9986	0.4513 ms
Replay	VAR based IDS	**0.6582**	**0.0089**	**0.8625**	**0.6920**	**0.5420**	**0.0458 ms**
	RNN based IDS	0.6540	0.0103	0.8584	0.6897	0.5413	0.4483 ms

6.3.3. VAR-based module vs RNN autoencoder-based module

This experiment is meant to present the VAR-based module as a valid alternative to the RNN autoencoder-based module for a set of highly autocorrelated CAN IDs. The experiment consists in comparing the performance of the VAR-based module and our RNN autoencoder over different tampered datasets. As flow-based anomalies can be easily detected by simpler approaches in this experiment we focus on datasets that contain masquerade attacks. The experiment is limited to traces of the CAN IDs that are classified as *High autocorrelation*, which are the ones on which the VAR-based module is effective. Results (see Table 8) show that the VAR-based module has comparable if not better detection performances on the fuzzed and replay datasets, but underperforms in the other two, especially in the seamless change dataset, where it loses more than 0.07 in detection rate. However, the detection rate is not the only metric that should be taken into consideration. In fact, the TTP of the VAR-based module is one order of magnitude smaller, and the FPR is always lower, independently from the error rate. In conclusion, although there is not an evident winner as in the first experiment, we can conclude that the VAR-based module is a comparable solution to the RNN-based one.

6.3.4. Hamming distance-based module vs RNN autoencoder-based module

This experiment is meant to present the Hamming distance IDS as a valid alternative to the RNN autoencoder-based IDS for a set of CAN IDs that we identified to have a really low variability of the packets. The experiment consists of comparing the performance of the Hamming distance module-

Table 9: Hamming distance-based module and RNN autoencoder-based module performances comparison.

		MCC	FPR	F1	TPR	TTP
Masquerade	Hamming distance IDS	0.0	**0.0**	0.0	0.0	**0.0054 ms**
	RNN based IDS	**0.3559**	0.3153	**0.4680**	**0.7792**	0.4560 ms
Seamless Change	Hamming distance IDS	0.1279	**0.0**	0.03390	0.0172	**0.0053**
	RNN based IDS	**0.3253**	0.2944	**0.2679**	**0.9828**	0.4613 ms
Fuzzed	Hamming distance IDS	**0.9439**	**0.0**	**0.9627**	0.9280	**0.0053 ms**
	RNN based IDS	0.6780	0.2871	0.7944	**0.9890**	0.4513 ms
Replay	Hamming distance IDS	0.0	**0.0**	0.0	0.0	**0.0052 ms**
	RNN based IDS	**0.0931**	0.3326	**0.0586**	**0.6970**	0.4493 ms

and the RNN autoencoder over different tampered datasets. As flow-based anomalies can be easily detected by simpler approaches in this experiment we only focus on datasets that contain masquerade attacks. The experiment is limited to traces of the CAN IDs that are classified as both *Few flips* and *Few values*, that are the ones selected for the Hamming distance module into CANova.

The results for this experiment are poor for both modules. The best assumption for the poor behavior of the RNN module is that IDs classified with the *Few flips* and *Few values* do not provide enough information for the detection system to learn their behavior. Interestingly, on this type of data, the RNN is unable to generalize and tends to consider anomalous each unseen sequence, leading to an extremely high FPR, which makes the detection module unfeasible for this type of packets. The Hamming distance-based module, on the other hand, is only able to detect events that heavily change the bits between packets, such as replay attacks. It is incapable of detecting the other, less evident attacks. However, its FPR is 0 even against the fuzzed dataset. None of the two modules is optimal, but while the RNN is not feasible due to the high FPR, the Hamming distance-based module at least provides no downsides to the detection process, having a TTP that is two orders of magnitude lower than the RNN. In conclusion, if possible, on packets classified both *Few flips* and *Few values* it would be optimal to exploit a rule-based IDS, which, however, requires knowledge of the meaning and legitimate behavior of the selected IDs. If not possible, the Hamming distance-based module does not provide any downsides to the detection process while being capable of detecting a subset of the possible

Table 10: CANova and RNN autoencoder-based module performances comparison.

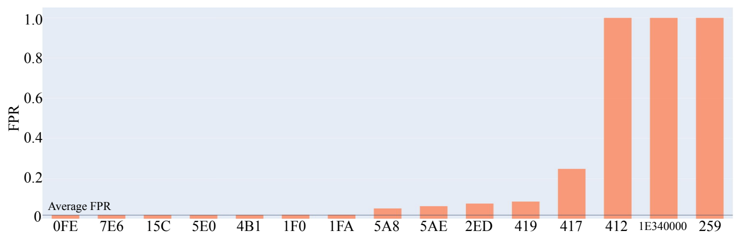
		MCC	FPR	F1	TPR	TTP
Masquerade	CANova	**0.8868**	**0.0161**	0.9143	0.8805	**0.2660 ms**
	RNN based IDS	0.8854	0.0265	**0.9149**	**0.9066**	0.4560 ms
Seamless Change	CANova	0.8652	**0.0156**	0.8944	0.8486	**0.2688 ms**
	RNN based IDS	**0.8684**	0.0263	**0.8996**	**0.8850**	0.4613 ms
Fuzzed	CANova	0.9352	**0.0151**	0.9528	0.9463	**0.2674 ms**
	RNN based IDS	**0.9567**	0.0231	**0.9683**	**0.9954**	0.4513 ms
Replay	CANova	**0.6655**	**0.0158**	**0.7055**	0.5712	**0.2636 ms**
	RNN based IDS	0.6347	0.0286	0.6896	**0.5723**	0.4483 ms
Injection	CANova	**0.6506**	**0.0190**	**0.6036**	**0.9969**	**0.2640 ms**
	RNN based IDS	0.4150	0.0277	0.3826	0.6901	0.4715 ms
Drop	CANova	**0.9608**	**0.0165**	**0.9684**	**0.9998**	**0.2113 ms**
	RNN based IDS	0.3346	0.0238	0.3474	0.2300	0.4714 ms

attacks.

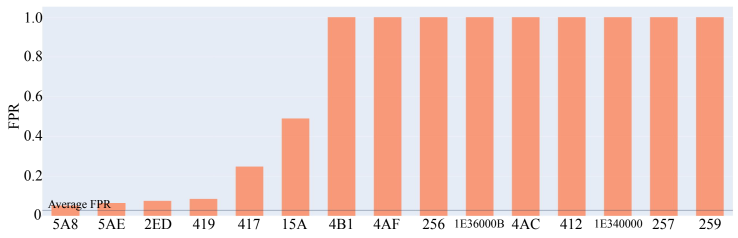
6.3.5. CANova vs the RNN autoencoder-based module

We tested CANova directly against the RNN autoencoder module, which is an improvement of Longari et al.'s work [25], to present the strengths of our modular approach.

The results of the experiment against the RNN-based IDS (shown in Table 10) clearly show the strengths of the modular approach of CANova. On both the analyzed frequency-based anomalies, thanks to the flow-based module, CANova can easily outscore the RNN autoencoder-based module on every metric. Moreover, thanks to the VAR-based and Hamming distance-based modules that lighten the workload of the slower RNN module, CANova's TTP approximately halves the one of the RNN. This becomes even more valueable considering that the average inter-arrival time of packets in our dataset is 0.3808 ms, hence while CANova would be able to keep up with the live network traffic, the RNN IDS would not. Finally, regarding one of the most relevant metrics in a context such as the automotive one, CANova maintains a lower FPR than the RNN on all attack datasets. It is also important to notice that although the detection rate metric of the RNN on the masquerade types of attacks is higher, the MCC, whose main advantage against other metrics is to be not influenced by datasets' imbalances, provid-



(a) CANova



(b) RNN Autoencoder module

Figure 2: False positive rate of CANova's and the RNN module's worst fifteen CAN IDs on a dataset without attacks.

ing a sound performance value, remains comparable if not better on CANova than on the RNN autoencoder IDS. A second consideration, noticeable from Figure 2, is that while the RNN false positives derive from various IDs (as evident by the fact that all the worst fifteen are above the average FPR, which is already high), the false positives for CANova derive mainly from the worse 3 to 8 IDs, that in real-world scenarios could be treated manually.

6.4. Car-hacking dataset Experiment: CANova vs. the State of the art

To provide a fair performance comparison, we evaluate the detection performances of CANova on the public car-hacking dataset [19] against the state-of-the-art IDSs systemized by Wang et al. [14], following the same experimental procedure. Table 11 contains our results alongside the ones provided by Wang et al. [14]. Regarding the DoS and Fuzzy attack, we highlight CANova performances with and without (shown in brackets in Table 11) the frequency module enabled. This is due to the fact that both attacks indirectly delay legitimate packet arrival times. However, since the frequency-based module detects anomalies by analyzing the packets inter-arrival time, it rightfully

[2]This performance was achieved on consumer-level HW, while Wang at al. [14] evaluation was performed on a machine learning dedicated server.

Table 11: Detection Performance Comparison of State-of-the-art IDSs against CANova. In **bold**, the best performance by metric and attack category.

IDSs	Attacks	Accuracy	Precision	TPR	FPR	F1-score	TTP
Reduced Inception-ResNet [19]	DoS Attack	0.9993	**0.9995**	0.9963	**0.0001**	**0.9980**	1.5633
	Fuzzy Attack	0.8730	0	0	0.0002	-	
	Gear Spoofing Attack	0.8223	0	0	**0.0001**	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
CANTransfer [20]	DoS Attack	0.9991	0.9990	0.9951	0.0002	0.9971	1.3264
	Fuzzy Attack	0.8718	0	0	**0.0001**	-	
	Fuzzy Attack (1-shot)	0.8664	0.9794	0.0309	**0.0001**	0.0599	
	Gear Spoofing Attack	0.8223	0	0	0.0004	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
CAN-ADF [21]	DoS Attack	0.9938	0.9826	0.9785	0.0033	0.9805	1.4476
	Fuzzy Attack	0.8715	0.0505	0.0002	0.0006	0.0004	
	Gear Spoofing Attack	0.8222	0	0	0.0004	-	
	RPM Spoofing Attack	0.7769	0.1200	0.0005	0.0012	0.0011	
TSP [22]	DoS Attack	0.9802	0.9100	0.9728	0.0183	0.9403	1.1422
	Fuzzy Attack	0.8714	0	0	0.0005	-	
	Gear Spoofing Attack	0.8221	0	0	0.0005	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
O-DAE [23]	DoS Attack	0.9933	0.9742	0.9843	0.0050	0.9792	1.2130
	Fuzzy Attack	0.8714	0	0	0.0006	-	
	Gear Spoofing Attack	0.8222	0	0	0.0004	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
LDAN [42]	DoS Attack	0.9806	0.9099	0.9756	0.0184	0.9416	0.9283
	Fuzzy Attack	0.8717	0	0	0.0006	-	
	Gear Spoofing Attack	0.8224	0	0	**0.0001**	-	
	RPM Spoofing Attack	0.7775	0	0	**0.0002**	-	
E-GAN [24]	DoS Attack	0.9806	0.9099	0.9756	0.0184	0.9416	1.0331
	Fuzzy Attack	0.8717	0	0	0.0002	-	
	Gear Spoofing Attack	0.8224	0	0	**0.0001**	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
HyDL-IDS [29]	DoS Attack	0.9936	0.9819	0.9781	0.0034	0.9800	0.4395
	Fuzzy Attack	0.8715	0.0612	0.0002	0.0005	0.0005	
	Gear Spoofing Attack	0.8221	0	0	**0.0001**	-	
	RPM Spoofing Attack	0.7769	0.1042	0.0005	0.0011	0.0009	
CANet [31]	DoS Attack	0.9993	0.9992	0.9966	0.0014	0.9979	0.3357
	Fuzzy Attack	0.8717	0	0	0.0002	-	
	Gear Spoofing Attack	0.8223	0	0	0.0001	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
Rec-CNN [30]	DoS Attack	0.9803	0.9097	0.9740	0.0185	0.9408	0.3278
	Fuzzy Attack	0.8714	0	0	0.0006	-	
	Gear Spoofing Attack	0.8221	0	0	0.0005	-	
	RPM Spoofing Attack	0.7774	0	0	0.0003	-	
CANova	DoS Attack	0.8119 (**0.9997**)	0.5784 (0.9987)	**1** (**1**)	0.2535 (0.0004)	0.7329 (0.9936)	**0.2568**[2]
	Fuzzy Attack	0.8631 (**0.9970**)	0.6906 (**0.9903**)	**1** (**1**)	0.1969 (0.0043)	0.8170 (**0.9951**)	
	Gear Spoofing Attack	**0.9983**	**0.9928**	**1**	0.0021	**0.9964**	
	RPM Spoofing Attack	**0.9970**	**0.9872**	**1**	0.0039	**0.9936**	

detects as anomalous all packets subsequent to the attack ones. These delayed packets, even if clearly showing vehicle misbehavior that is obviously detected by a frequency-based detector, are labeled legitimate in the dataset, leading to a high number of FPs. To verify this issue, we tested CANova on the same dataset but with the first packet of each ID after an attack labeled as malicious to show that all FPs are due to the delayed legitimate packets. The results of this experiment (not shown in Table 11) achieved a 0 FP rate. To ensure a fair comparison, we decided not to present the results with the tampered dataset and, instead, show also the performance achieved without the frequency analysis that distorts the results. Overall, CANova achieves better performance than the IDSs under analysis regarding all the metrics with a perfect TPR and accuracy up to 0.9997 on all the attacks. The only exception is the FPR, where *Reduced Inception-ResNet* achieves an FPR of 0.0001 against our of 0.2535 (0.0004) for the DoS attack. However, it should be noticed how *Reduced Inception-ResNet* is six times slower in obtaining this result. Most importantly, since the average packet inter-arrival time of the dataset is 0.7727 ms, a TTP of 1.5633 is not compliant with the real-time requirements of the automotive domain. In addition, differently from all the IDSs under analysis, CANova is able to detect all the categories of attacks.

7. Conclusions

In this paper, we directly addressed the limitations of existing research works on IDSs for CAN, which usually focus on a subset of CAN-ID and on specific attacks, by proposing CANova, a modular IDS framework that exploits the characteristics of the different CAN packets to select the Intrusion Detection Systems (IDSs) that better fits them. In particular, it combines flow- and payload-based IDSs to analyze the packets' content and arrival time. CANova detects simple anomalies (e.g., change in the frequency of the packets or packets from an invalid CAN-ID) through a flow-based module and a rule-based module, while packets that pass these tests are analyzed by one between a Hamming distance-based module, a VAR-based module, and a RNN autoencoder-based module, depending on the attributes of the packet found by the classification. As far as we know, CANova is the first IDS for CAN that is able to exploit a classification of the CAN-IDs to combine different state-of-the-art techniques to improve the performances of the individual techniques and work on the majority of the CAN-IDs.

The experimental results showed that the combination of flow-based modules with RNN autoencoder-based IDS solutions achieves an almost perfect detection rate in scenarios in which RNN-based solutions perform badly. On the other hand, in our tests, the RNN-based solution, while being more than twice as fast as similar state-of-the-art methods, is not able to perform real-time detection on the entire CAN traffic. Nonetheless, CANova performed predictions that are, on average more than 1.5 times faster than the ones provided by the RNN autoencoder-based IDS only, thanks to the addition of the VAR-based module, which analyzes a set of highly autocorrelated CAN-IDs, and the Hamming distance-based module, which analyzes a set of CAN-IDs with very low variability of the payloads.

Moreover, CANova outperforms state-of-the-art detection methods with a perfect True Positive Ratio (TPR) and lower time requirements on all the real-world attacks under analysis.

Future works will focus on improving the overall performances and, in particular, on further reducing the number of False Positives (FPs), a known curse of machine learning-based models, but that is of particular importance in the domain under analysis.

References

[1] S. Longari, A. Cannizzo, M. Carminati, S. Zanero, A secure-by-design framework for automotive on-board network risk analysis, in: 2019 IEEE Vehicular Networking Conference (VNC), 2019, pp. 1–8. `doi:10.1109/VNC48660.2019.9062783`.

[2] S. Checkoway, D. Mccoy, D. Anderson, B. Kantor, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, Comprehensive experimental analyses of automotive attack surfaces (08 2011).

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al., Experimental security analysis of a modern automobile, in: 2010 IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 447–462.

[4] A. de Faveri Tron, S. Longari, M. Carminati, M. Polino, S. Zanero, Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks, in: H. Yin, A. Stavrou, C. Cremers, E. Shi (Eds.), Proceedings of the 2022 ACM SIGSAC Conference on Computer and

Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, ACM, 2022, pp. 711–723. doi:10.1145/3548606.3560618. URL https://doi.org/10.1145/3548606.3560618

[5] S. Longari, M. Penco, M. Carminati, S. Zanero, Copycan: An error-handling protocol based intrusion detection system for controller area network, in: L. Cavallaro, J. Kinder, T. Holz (Eds.), Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy, CPS-SPC@CCS 2019, London, UK, November 11, 2019, ACM, 2019, pp. 39–50. doi:10.1145/3338499.3357362. URL https://doi.org/10.1145/3338499.3357362

[6] Cia, Can data link layers in some detail. URL https://www.can-cia.org/can-knowledge/can/can-data-link-layers/

[7] T. Instruments, Introductionto the controllerareanetwork(can) (2002).

[8] M. Avatefipour, State-of-the-art survey on in-vehicle network communication “can-bus” security and vulnerabilities.

[9] A. Palanca, E. Evenchick, F. Maggi, S. Zanero, A stealth, selective, link-layer denial-of-service attack against automotive networks, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2017, pp. 185–206.

[10] C. Miller, C. Valasek, Can message injection, OG Dynamite Edition (2016).

[11] H. J. Jo, W. Choi, A survey of attacks on controller area networks and corresponding countermeasures, IEEE Trans. Intell. Transp. Syst. 23 (7) (2022) 6123–6141. doi:10.1109/TITS.2021.3078740. URL https://doi.org/10.1109/TITS.2021.3078740

[12] C. Young, J. Zambreno, H. Olufowobi, G. Bloom, Survey of automotive controller area network intrusion detection systems, IEEE Des. Test 36 (6) (2019) 48–55. doi:10.1109/MDAT.2019.2899062. URL https://doi.org/10.1109/MDAT.2019.2899062

[13] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, A. Mouzakitis, Intrusion detection systems for intra-vehicle networks: A review, IEEE Access 7 (2019) 21266–21289. doi:10.1109/ACCESS.2019.2894183.

[14] K. Wang, A. Zhang, H. Sun, B. Wang, Analysis of recent deep-learning-based intrusion detection methods for in-vehicle network, IEEE Transactions on Intelligent Transportation Systems (2022) 1–12doi:10.1109/TITS.2022.3222486.

[15] H. Song, H. Kim, H. K. Kim, Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network, 2016, pp. 63–68. doi:10.1109/ICOIN.2016.7427089.

[16] A. Taylor, N. Japkowicz, S. Leblanc, Frequency-based anomaly detection for the automotive can bus, in: 2015 World Congress on Industrial Control Systems Security (WCICSS), IEEE, 2015, pp. 45–49.

[17] E. Seo, H. M. Song, H. K. Kim, Gids: Gan based intrusion detection system for in-vehicle network, in: 2018 16th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2018, pp. 1–6.

[18] H. Olufowobi, C. Young, J. Zambreno, G. Bloom, Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing, IEEE Transactions on Vehicular Technology 69 (2) (2019) 1484–1494.

[19] H. M. Song, J. Woo, H. K. Kim, In-vehicle network intrusion detection using deep convolutional neural network, Veh. Commun. 21 (2020). doi:10.1016/j.vehcom.2019.100198.
URL https://doi.org/10.1016/j.vehcom.2019.100198

[20] S. Tariq, S. Lee, S. S. Woo, Cantransfer: transfer learning based intrusion detection on a controller area network using convolutional LSTM network, in: C. Hung, T. Cerný, D. Shin, A. Bechini (Eds.), SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020, ACM, 2020, pp. 1048–1055. doi:10.1145/3341105.3373868.
URL https://doi.org/10.1145/3341105.3373868

[21] S. Tariq, S. Lee, H. K. Kim, S. S. Woo, CAN-ADF: the controller area network attack detection framework, Comput. Secur. 94 (2020) 101857.

doi:10.1016/j.cose.2020.101857.
URL https://doi.org/10.1016/j.cose.2020.101857

[22] H. Qin, M. Yan, H. Ji, Application of controller area network (CAN) bus anomaly detection based on time series prediction, Veh. Commun. 27 (2021) 100291. doi:10.1016/j.vehcom.2020.100291.
URL https://doi.org/10.1016/j.vehcom.2020.100291

[23] Y. Lin, C. Chen, F. Xiao, O. Avatefipour, K. Alsubhi, A. Yunianta, An evolutionary deep learning anomaly detection framework for in-vehicle networks-can bus, IEEE Transactions on Industry Applications (2020).

[24] G. Xie, L. T. Yang, Y. Yang, H. Luo, R. Li, M. Alazab, Threat analysis for automotive CAN networks: A GAN model-based intrusion detection technique, IEEE Trans. Intell. Transp. Syst. 22 (7) (2021) 4467–4477. doi:10.1109/TITS.2021.3055351.
URL https://doi.org/10.1109/TITS.2021.3055351

[25] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, S. Zanero, Cannolo: An anomaly detection system based on lstm autoencoders for controller area network, IEEE Transactions on Network and Service Management (2020).

[26] S. Longari, A. Nichelini, C. A. Pozzoli, M. Carminati, S. Zanero, Candito: Improving payload-based detection of attacks on controller area networks (2022). doi:10.48550/ARXIV.2208.06628.
URL https://arxiv.org/abs/2208.06628

[27] A. Taylor, Anomaly-based detection of malicious activity in in-vehicle networks, Ph.D. thesis, Université d'Ottawa/University of Ottawa (2017).

[28] M. Marchetti, D. Stabili, A. Guido, M. Colajanni, Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms, in: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), IEEE, 2016, pp. 1–6.

[29] W. Lo, H. AlQahtani, K. Thakur, A. Almadhor, S. Chander, G. Kumar, A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic, Veh. Commun. 35

(2022) 100471. doi:10.1016/j.vehcom.2022.100471.
URL https://doi.org/10.1016/j.vehcom.2022.100471

[30] A. K. Desta, S. Ohira, I. Arai, K. Fujikawa, Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots, Veh. Commun. 35 (2022) 100470. doi:10.1016/j.vehcom.2022.100470.
URL https://doi.org/10.1016/j.vehcom.2022.100470

[31] M. Hanselmann, T. Strauss, K. Dormann, H. Ulmer, Canet: An unsupervised intrusion detection system for high dimensional can bus data, IEEE Access 8 (2020) 58194–58205.

[32] S. Kulandaivel, S. Jain, J. Guajardo, V. Sekar, CANNON: reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers, in: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021, IEEE, 2021, pp. 195–210. doi:10.1109/SP40001.2021.00122.
URL https://doi.org/10.1109/SP40001.2021.00122

[33] M. Marchetti, D. Stabili, Read: Reverse engineering of automotive data frames, IEEE Transactions on Information Forensics and Security 14 (4) (2018) 1083–1097.

[34] C. Miller, C. Valasek, Adventures in automotive networks and control units, Def Con 21 (260-264) (2013) 15–31.

[35] D. Stabili, M. Marchetti, M. Colajanni, Detecting attacks to internal vehicle networks through hamming distance, in: 2017 AEIT International Annual Conference, 2017, pp. 1–6. doi:10.23919/AEIT.2017.8240550.

[36] H. Lütkepohl, New Introduction to Multiple Time Series Analysis, Springer Berlin Heidelberg, 2007.

[37] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: Proceedings, Vol. 89, Presses universitaires de Louvain, 2015, pp. 89–94.

[38] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, Lstm-based encoder-decoder for multi-sensor anomaly detection, arXiv preprint arXiv:1607.00148 (2016).

[39] K. Wang, S. J. Stolfo, Anomalous payload-based network intrusion detection, in: International workshop on recent advances in intrusion detection, Springer, 2004, pp. 203–222.

[40] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289 (2015).

[41] M. Zago, S. Longari, A. Tricarico, M. Carminati, M. G. Pérez, G. M. Pérez, S. Zanero, Recan–dataset for reverse engineering of controller area networks, Data in brief 29 (2020) 105149.

[42] R. Zhao, J. Yin, Z. Xue, G. Gui, B. Adebisi, T. Ohtsuki, H. Gacanin, H. Sari, An efficient intrusion detection method based on dynamic autoencoder, IEEE Wirel. Commun. Lett. 10 (8) (2021) 1707–1711. `doi:10.1109/LWC.2021.3077946`.
URL `https://doi.org/10.1109/LWC.2021.3077946`