Evaluating the Robustness of Automotive Intrusion Detection Systems against Evasion Attacks

Stefano Longari[1], Francesco Noseda[1], Michele Carminati[1], and Stefano Zanero[1]

Politecnico di Milano, Milan, Italy
{stefano.longari,michele.carminati,stefano.zanero}@polimi.it,
{francesco.noseda}@mail.polimi.it

Abstract This paper discusses the robustness of machine learning-based intrusion detection systems (IDSs) used in the Controller Area Networks context against adversarial samples, inputs crafted to deceive the system. We design a novel methodology to deploy evasion attacks and address the domain-specific challenges (i.e., the time-dependent nature of automotive networks) discussing the problem of performing online attacks. We evaluate the robustness of state-of-the-art IDSs on a real-world dataset by performing evasion attacks. We show that, depending on the targeted IDS and the degree of the attacker's knowledge, our approach achieves significantly different evasion rates.

1 Introduction

Machine learning (ML) techniques have been successfully exploited in several tasks from image identification [5] to the detection of malware [11] and intrusions [13]. In particular, ML methods have been applied in the automotive domain [1, 16, 26], where they are implemented in IDSs that aim at detecting anomalies in the CAN bus data stream, which is composed of network packets transmitted by the Electronic Control Units (ECUs). Although effective in detecting malicious behaviors, machine learning models are vulnerable to adversarial attacks: it is possible to significantly worsen their performance through adversarial samples [6, 36], which are inputs crafted by iteratively perturbing legitimate samples until the detector misclassifies them. In addition, due to the transferability property [12, 36], an attack against a machine learning system can be effective against a different, potentially unknown, target system [35]. Adversarial attacks have been widely studied in the image classification domain, but, no study has yet targeted the automotive domain, whose security, similarly to other CPSs, directly affects users' safety. In fact, the automotive domain is characterized by inherent challenges that make existing solutions not directly applicable.

In this paper, we study the feasibility of the application of adversarial machine learning to the automotive context by developing a novel approach to perform evasion attacks against IDSs. Our approach extends existing methods

for crafting adversarial samples (i.e., CAN packets) and considers the challenges of the domain under analysis. Differently from related works in which the adversarial samples are generated offline independently from the current status of the data stream (i.e., perturbed pixels in an image), in this work, we focus on generating adversarial samples online, which must be coherent with the constraints imposed by the automotive data stream (e.g., the generated sample must "follow" the dynamic of the vehicle) while maintaining the properties of known automotive attacks, which generally require multiple malicious packets to achieve their goal. In other words, we generate evasive attacks that depend on the history of data transmitted until the injection while concealing multiple attack packets. To do so, starting from known attacks, we iteratively perturb each of their payloads to produce a new but evasive sequence of packets that evades a surrogate detection system (i.e., Oracle). When a generated packet evades the Oracle, exploiting the transferability property, we inject it into the data stream of the target IDS. In addition, we model an attacker with different degrees of knowledge: *Black Box*, with zero knowledge of the target system; *Gray Box*, with partial knowledge of the system; *White Box*, with complete knowledge of the system. Using a real-world dataset of CAN packets [40], we evaluate the security of state-of-the-art IDSs against our approach, demonstrating that it is possible to reduce the detection capabilities of the IDS without heavily modifying the attack's effects on the vehicle. However, the effectiveness of the adversarial attack substantially depends on the targeted IDS and the degree of the attacker's knowledge. Interestingly, our results demonstrate that the constraint imposed by the automotive domain severely limits the attacker's room for maneuver, directly impacting the effectiveness of the attacks.

Our contribution are the following:

- We present a novel approach for performing evasion attacks against IDSs under different degrees of attacker's knowledge and simulating the behavior of different attacks. To do so, we generate evasive adversarial inputs from existing attacks.
- We study the robustness of automotive IDSs against evasive inputs generated using our novel approach.

2 Background and Related Works

Controller Area Network (CAN) has been the de-facto standard communication protocol for vehicular on-board networks since '80. In brief, it is a bus-based multi-master communication protocol whose data packets are composed mainly of an ID and a payload (and a set of flags and control fields). For further details on the CAN specification, we refer the reader to [37]. **CAN security** weaknesses are nowadays well known and discussed in multiple works [23, 39]. As demonstrated by Miller and Valasek in [30, 31], one of the most common known vulnerabilities derives from the lack of authentication of messages on CAN. A node should not be allowed to send IDs that it does not own, but there is no mechanism to enforce this rule. Therefore, an attacker that takes control of an ECU that has access

to a CAN bus can ideally send any ID and payload. In worst-case scenarios, the attacker is also capable of silencing the owner of the packet to avoid conflicts, as presented in [25]. In brief, an attacker may affect the payload and flow of packets on the bus.

To recognize in-vehicle network attacks, researchers have proposed multiple Intrusion Detection Systems (IDSs), which can be classified in flow- and payload-based approaches following Al Jarrah et al.'s [1]. Flow-based approaches [9, 21, 29, 32] extract distinct features, such as frequency and period, from messages on the network. However, the main weakness of such systems is their incapability to comprehend the content of the packet's payload, making them blind to attacks that do not modify the flow of the network. To fill such a gap, payload-based approaches study the content of the payload of frames to recognize patterns and, from those, anomalies. Aside from rule-based anomaly recognition, which is limited by the necessity of writing specific rules for each ID and each anomaly, researchers have started designing systems that attempt to recognize such patterns automatically. Stabili et al. [38] propose a solution based on the hamming distance between the payloads of different packets. More often, to automatize the recognition of patterns and thresholds, solutions are based on machine learning algorithms: LSTM autoencoders [26], deep neural networks [19], RNN [27] have all been proposed as anomaly detectors for CAN. Finally, researchers have proposed various multi-stage IDSs that propose ensembles of flow- and payload-based systems [33].

2.1 Related Works on Adversarial Machine Learning

Adversarial machine learning attacks can be divided into *Evasion* and *Poisoning* attacks [2]. Evasion attacks occur at detection time, and their goal is to perturb a malicious input to convince the model to misclassify it as non-malicious. In poisoning attacks, instead, the attacker injects carefully crafted malicious samples into the training set to mis-train the model and create a *'backdoor'* that can be exploited in subsequent attacks. Many studies and examples have been published in recent years, such as [4, 6, 8, 15, 34]. In this work, we focus on *evasion attacks*. Ibitoye et al. [18] survey various adversarial machine learning approaches, most of which modify each input element just once, tailored to attack, for instance, image analysis algorithms. Instead, our target IDSs use a self-supervised approach: they analyze a *sliding window* of payloads. Once one of the payloads of the input window has been modified, the subsequent input will be the previous window shifted by one, thus including the just modified payload. Such algorithms, if applied to our context, will modify each payload more than once, rapidly leading to a complex combination of modifications to deal with. Huang et al. [17] propose two algorithms for evading point generation. They are called *Genetic Attack* (GA) and *Probability Weighted Packet Saliency Attack* (PW-PSA). The paper specifically focuses on evading LSTM-based IDSs, and it is the most similar work to our approach. Li et al. [22] adapt to the automotive domain FGSM [15] and BIM [20], which are two fast gradient-based method to compute adversarial samples. The attacker increases the classification error by applying

the gradient ascent of the cost function: it adds random noise in the direction of the gradient ascent of the cost function. Their approach is white-box only and assumes to modify a pre-recorded CAN log before injecting it.

Discussion w.r.t. the state of the art. Most algorithms are designed to operate on images. They assume that the input features can be slightly modified without substantially changing the meaning of what they represent. As the features used by our models are single bits, this assumption does not hold, and such algorithms cannot be directly applied. Other approaches modify cyber-attacks and deceive IDSs. Nevertheless, such approaches do not apply to our threat model. For instance, the GA and PWPSA approaches are thought to perturb DoS attacks. This is a major constraint since, in a packet of a DoS attack, the payload content is irrelevant. On the contrary, our approach modifies precisely the payload of the packets. Moreover, such approaches do not modify the original attack's payloads but focus on other features. Finally, Li et al. [22] and Huang et al. [17] are designed to modify the original attack *offline*, meaning that they turn an attack into an adversarial point before injecting such an attack into the target system. This is an important difference because such approaches alter each packet of the input, considering all the other packets that precede or follow it in the attack. On the contrary, our approach is an *online* approach, which alters one packet at a time, only considering the ones already transmitted and ignoring all the attack payloads yet to be transmitted.

3 Threat model

Informally, we imagine an attacker generating an automotive attack and testing various evasion techniques on it to bypass an automotive IDS and have its attack successfully executed. At this point, we require to make an assumption regarding the behavior of CAN packets: given a CAN attack consisting of a sequence of payloads, switching a few bits for each of its payloads only barely changes the effect that the attack has on the vehicle; moreover, such change is proportional to the number of bits switched. This assumption does not always hold, depending on the type of packet, attack, and position of the switched bit. However, to obtain an extended evaluation of the capabilities of different adversarial attacks on multiple types of packets without having access to multiple DBC files (automakers proprietary files that describe the meaning of each packet in the network), we assume that this concept, although not perfectly, describes the general behavior of the network. We can define different scenarios according to which details of the target IDS the attacker knows. We formalize the threat model following the framework proposed by Biggio et al. in [5], which is, in turn, derived from [2, 3].

Attacker's goal. The attacker's goal can be defined along three dimensions: *security violation*, *attack specificity* and *error specificity*. The security violation is against *integrity*, as our goal is to inject attacks that evade detection without compromising the normal operations of the IDS. The specificity is *targeted*, as our main goal is to modify malicious samples while leaving the non-anomalous

ones untouched. Finally, the error specificity is *specific*, as we want the modified inputs to be specifically misclassified as non-anomalous.

Attacker's knowledge. This aspect defines what the attacker knows about the target IDS. It can be defined as a tuple $(\mathcal{D}, f, \mathbf{w})$, where $\mathcal{D}$ is the training set, f is the learning algorithm, and $\mathbf{w}$ is the set of hyperparameters. These elements can be either *known* or *unknown*. We will mark an unknown element with a hat (e.g., $\hat{\mathcal{D}}$ means that the training set is unknown by the attacker). We formalize three different scenarios: white-box, gray-box, and black-box.

In the **white-box scenario**, the attacker fully knows target system: $(\mathcal{D}, f, \mathbf{w})$. In the **black-box scenario**, the attacker knows nothing about the target system: $(\hat{\mathcal{D}}, \hat{f}, \hat{\mathbf{w}})$. In the **gray-box scenario**, the attacker knows the learning algorithm: $(\hat{\mathcal{D}}, f, \hat{\mathbf{w}})$.

Attacker's capability. It defines the *attack influence* and a set of *data manipulation constraints*. In evasion attacks, also called *exploratory* attacks, the attacker manipulates only the test set [7]. Among the manipulation constraints, we specify how the attack payloads should be manipulated: the attacker can only modify the payloads already belonging to the attack and not the others. The manipulated payloads should be as close as possible to the original payloads to have a similar effect on the vehicle. The distance metric used is the Hamming distance. Moreover, the attacker operates in a *real-time* fashion: she modifies the payloads in the same order in which they appear in the attack. In this way, each payload is modified, considering previous modifications already applied to previous payloads. As previously mentioned, this is a requirement for CAN since the attacker must transition from an authentic data stream to a tampered one.

Attack strategy. The strategy defines the attack in terms of an objective function. Our objective function can be formalized as $\hat{x} = argmin_{x^*} \delta(x^*, x') \quad s.t. \mathcal{F}(x^*) = y_{normal}$, where $\hat{x}$ is the manipulated payload, $\mathcal{F}$ is an automotive IDS trained according to the adopted knowledge scenario, x' is the original attack payload, δ is the Hamming distance, and y_{normal} is the output of $\mathcal{F}$ when no attack is detected. The attacker optimizes this function for each payload belonging to the attack. The distance we want to minimize is the Hamming distance, i.e., the number of switched bits in each payload, because we want the effect of the modified payload to be as close as possible to the effect of the original attack payload, as mentioned in the assumption discussed in Section 3.

4 Approach

The goal of this work is to study the feasibility of evasion attacks against CAN-based automotive IDSs, depending on the attacker's knowledge of the defending system. In particular, we start from non-evasive automotive attacks (i.e., primary attacks) and perturb them into evasive ones (i.e., secondary attacks). We based our approach on [7] and [10], where the idea of using an oracle was effective. An oracle can be thought of as an IDS controlled by the attacker, used to emulate a real-world IDS and test the evasive properties of an attack before actually

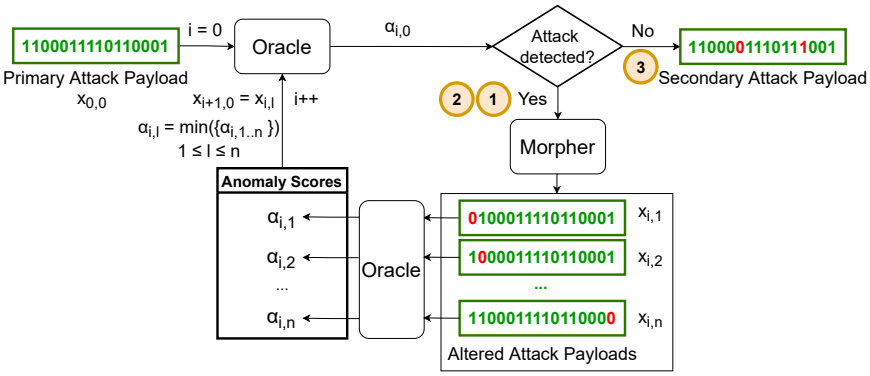

Figure 1: Overview of our approach: The attack payload is morphed by switching one of its bit (1,2) until it is not detected by the Oracle (3).

injecting it. To generate the candidate evasive points to test, we used a *Morpher* as introduced in [10]. A Morpher is a component that, given an input x_0, returns a set $\{x_1, ..., x_n\}$ of n different elements, each of which is obtained by *slightly modifying* x_0. The generated elements are then tested against the Oracle. An evasive point is an element $x_l \in \{x_1, ..., x_n\}$ s.t. x_l is not classified by the Oracle as malicious. Denoting the Morpher as $\mathcal{M}$ and writing $\mathcal{M}(x_0) = \{x_1, ..., x_n\}$, we can write $x_l = argmin_{x^* \in \mathcal{M}(x_0)} \delta(x^*, x_0) \quad s.t.\ \mathcal{F}(x^*) = y_{normal}$. A very similar approach is proposed by Huang et al. in [17], where the concepts of Oracle and Morpher are used in a similar way as in our definition. However, their work focuses specifically on Denial of Service (DoS) attacks in a standard network and would not be applicable *"as is"* to the automotive domain.

The idea behind our approach is to start from a non-modified automotive attack that is likely to be detected by an IDS (*primary attack*) and modify it into a second attack which is not only very similar to the one in input but is also able to evade the target IDS (*secondary attack*). To do so, we use the Morpher to generate a given amount of slight modifications of the primary attack. Specifically, once the Morpher modifies the attack, each modified attack is tested against the Oracle. If the oracle does not detect one of the attacks, we consider that attack as an evading point. If none of the modified attacks is an evading point, we proceed in a greedy way selecting the *'less detected'* output element (the one that generated the lowest anomaly score while still being over the detection threshold) and modifying it once again, repeating the process until an evading point is found or a termination condition is met. Figure 1 presents an overview of our approach.

ML-based IDSs and Oracle. An ML-based IDS is used to recognize different attacks which may be detected in a network. To do so, it analyzes the traffic and computes a measure of how likely the input is to contain an attack (*anomaly score*). This score is then compared with a *threshold*, generally fixed upon training: if the score is below the threshold, the input is classified as *benign*, otherwise

as *anomalous*. An Oracle is an IDS in its own right but is not used for defensive purposes. Instead, it is created and trained by an attacker to understand how likely it is for an attack to be detected. In more formal terms, it returns the *anomaly score*, an index of how likely the input is to be classified as *anomalous* by the target IDS. Given our approach, it is necessary to obtain the anomaly score and not only a binary detection value since we sort the various inputs by anomaly score. It is immediate to notice that the proposed approach is more successful the more the Oracle is similar to the IDS.

Morpher. As sketched in Figure 1, the Morpher is a component that returns a set of different alterations of the same input. To do so, it switches all the bits of the given input, one for each output element. Assuming that the input packet payload $x_{0,0}$ is composed of n bits, our implementation of the Morpher returns n different modifications $\{x_{0,1}, ..., x_{0,n}\}$ of the input, each one obtained by switching a different bit of $x_{0,0}$. Let $\alpha_{0,0}$ be the anomaly score of $x_{0,0}$ and θ be the Oracle's classification threshold. All the elements of the output set are tested against the Oracle, and the one with the lowest anomaly score is selected. Let $x_{0,l}$ be such element and $\alpha_{0,l}$ its anomaly score, with $1 \leq l \leq n$. If $\alpha_{0,l} < \alpha_{0,0}$ and $\alpha_{0,l} < \theta$, then $x_{0,l}$ is an evasive point and is returned; otherwise, if $\alpha_{0,l} < \alpha_{0,0}$ but $\alpha_{0,l} > \theta$, then we set $x_{1,0} = x_{0,l}$ and $x_{1,0}$ is in turn fed to the Morpher, and another iteration takes place. The particular case in which, at a certain iteration i, $\alpha_{i,l} > \alpha_{i,0}$ is discussed later on. It is worth noting that there are no guarantees on the termination of the algorithm. This is due to the algorithm potentially ending in a local minimum that remains over the detection threshold or, depending on the IDS taken into consideration, due to the global minimum of the function not being under the threshold. To address this, we apply a standard termination policy, setting a maximum number of iterations before the process is forcefully stopped without finding an evading point. It is then necessary to address the issue of what the algorithm should do if this case happens and an evading point is not found. There are two possible courses of action: raise a failure and stop the process or continue the attack with the best alteration that was produced, even if it is not sufficiently evasive. Finally, another particular case to analyze is the one in which $\alpha_{i,l} > \alpha_{i,0}$. As we want the anomaly score of the altered attack to eventually go below the threshold θ, the most straightforward way is to select a monotonic non-increasing sequence of anomaly scores. When, at the end of a generic iteration i, the best score $\alpha_{i,l}$ is higher than the anomaly score of the input $\alpha_{i,0}$ (which is the lowest anomaly score in the previous iteration, $\alpha_{i-1,l}$), it means that the anomaly score cannot be lowered further by just switching a single bit. Once again, different policies can be adopted: we can raise a failure and decide that the input cannot become an evasive point, or we can select $x_{i,l}$ as input for another iteration. To avoid moving away from a local minimum point of the function and with the computation overhead in mind, we choose the latter for our approach.

5 Experimental Validation

The goal of our experiments is twofold: (1) Evaluate the impact on the detection performances of evasive attacks by comparing the detection rate of the IDSs against primary attacks (i.e., baseline performance) with the detection rate of the same IDSs against secondary attacks, in black- grey- and white-box scenarios; (2) Statistically measure the number of perturbated bits required to obtain a secondary evasive attack. To simulate the white, grey, and black box scenarios, we build the secondary attacks using different Oracles. We then test the secondary attacks against all the IDSs, measuring Recall and FPR. Evaluating FPR is necessary since it is important to understand whether a given IDS is detecting the secondary attacks due to the attacks not being evasive enough or because it has the general tendency of considering many packets as anomalous even when they are not.

Datasets. We execute our experiments on ReCAN [40], an automotive CAN dataset gathered from real-world vehicles. We chose the data related to the vehicle with dataset ID C-1 (an Alfa Giulia Veloce) since it has the longest recordings (more than 20 million rows in total). We exploit the three longest continuous logs (1,3, and 9) for our scope: to perform training and validation we use datasets 9 and 1, and divide the data in a standard 70% - 30% split. More in detail, we use dataset 1 to train both the defending IDSs and the Oracles in the White-box scenario and dataset 9 to train the surrogate Oracles in the Black-box and Gray-box scenarios. From now on, we refer to dataset 1 as "DEF" and dataset 9 as "ATK". To test the attacks, we use dataset 4, in which we inject the attacks generated through the attack tool from [24].

Selected CAN IDs. To perform our experiments, we select candidate CAN IDs representing the types of signals most commonly transmitted on CAN. In particular, we categorize CAN IDs by extracting their signals with READ [28] a signal extraction tool that analyses a record of CAN traffic and, for each CAN-ID, tries to reconstruct the signals contained in its payloads. Then, we analyze their features (i.e., frequency of bit flipping, presence of patterns, signal variance, autocorrelation, signal length). Each detected signal can be of one of the following types: **PHYSVAL**, if signals that carry readings from sensors; **COUNTER**, if signals have a value that constantly increases or decreases by 1 w.r.t. the value in the previous payload; **CRC** if signals that contain a checksum of the payload; **BINARY**, if signals are only 1 bit long. The bits of a payload are not constrained to belong to a signal. Bits that never change are not assigned to any signal and are thus ignored. In our work, only bits belonging to signals are used as features of the IDSs, are modified when building an attack, and are considered by our approach in the search for an adversarial point. We obtain the six ID classes presented in Table 1. For each class, we choose the ID with the highest number of payloads from the dataset in order to improve the quality of the trained model. Moreover, in the reminder, we refer to the set of bits of a payload belonging to a signal as the set of *switchable bits*, which represents the set of features of our IDSs and Oracles.

Table 1: Different classes of CAN IDs depending on the identified properties of their payloads, including the number of bits that change in each of the payloads.

Class Number	**Payload properties**	**ID**	**# bits**
Class 1	Payloads assume up to 10 different values	420	15
Class 2	Bits of the payload flip up to 10 times	226	1
Class 3	All signals of the ID follow a pattern	416	38
Class 4	Only one-bit signals in the payload	1FC	2
Class 5	PHYSVAL are ≥ 4 bits long and highly auto-correlated	0EE	56
Class 6	None of the above (unclassified)	2ED	23

5.1 Attacks generation

Attacks against CAN can be divided into injection, masquerade, and denial of service attacks [1, 26]. In the first case, the malicious packets are new packets added to the dataset, while in the second, the malicious packets are generated by modifying already existing packets. Denial of Service attacks can be implemented by means of flooding (i.e., injection) or by disconnecting a node from the network [14] (i.e., drop). In this work, we avoid considering attacks that can be easily detected through frequency-based analysis and focus on masquerade and drop attacks.

Drop. A sequence of packets is removed from the dataset, and the first packet after the attack is marked as tampered, as it is the point at which the IDS should detect the attack.

Fuzzy. Every signal in every attack payload is replaced with a random string of bits.

Replay. Simulates a replay attack. The target sequence of payloads is replaced with a sequence of the same length sniffed from previous traffic.

Seamless Change. In the target sequence of payloads, the signals that represent physical values are replaced with a sequence of values that continuously increase or decrease until a target value is reached. In our implementation, the target value is a random binary string.

5.2 Intrusion Detection Systems and Oracles

We evaluate our approach against various classes of IDSs found in the literature [1, 26, 38]. We implemented total of 7 payload-based IDSs for CAN, which we deem representative of the various identified classes:

Small-LSTM. RNN with two layers of LSTM cells and one Dense output layer.

Small-GRU. RNN with two layers of GRU cells and one Dense output layer.

Large-GRU. RNN with four layers of GRU cells and one Dense output layer.

CANnolo [26]. Autoencoder based on LSTM cells that takes as input a sliding window of 40 packets.

NN. Simple Neural Network with two hidden layers of the Dense type. The input is the entire dataset to be tested, which is compressed and decompressed.

VAR. The input is composed of a window of 2 consecutive payloads. Using a Vector Auto Regressive model, the first payload is used to compute the one-step-ahead predictor of the next payload, and such prediction is then compared with the second payload in input.

Hamming (adapted from [38]). Model that takes as input a window of 3 consecutive payloads, and the analyzed property is the sum of the Hamming distances between the payloads. If the result is included in a range of accepted values, the input is marked as *non-anomalous*, otherwise as *anomalous*.

5.3 Exp. 1: Impact of AML Attacks on Detection Performances

We consider all the available primary attacks and turn them into secondary attacks using all the available Oracles. For each CAN-ID, we have 4 different primary attacks and 14 different Oracles (7 trained with the DEF train-set and 7 trained with the ATK train-set). In total, we can generate 56 secondary attacks for each CAN-ID. The tested defending IDSs are only trained on the DEF train-set. For the sake of simplicity, in the following subsections, we report the average performances per ID in Table 2a and attack in Table 2b. The different scenarios we take into consideration are:

White-box. The attacker can build an Oracle which has both the same architecture and train-set as the IDS. Thus, Oracles are trained on the DEF dataset, and we tested each secondary attack only against the IDS having the same architecture as the Oracle used to generate it.

Black-box. The attacker can only build an Oracle that share neither the architecture nor the train-set with the IDS. Thus, Oracles are trained on the ATK dataset, and we tested each secondary attack only against the IDSs having a different architecture as the Oracle used to generate it.

Gray-box. The attacker can build an Oracle that has the same architecture of the IDS but is trained on another dataset. Thus, Oracles are trained on the ATK dataset, and we tested each secondary attack only against the IDS having the same architecture as the Oracle used to generate it.

Discussion on results. The results of Experiment 1 are presented in Table 2b and Table 2a. The easiest detected primary attacks are Fuzzy and Seamless Change, as they actively modify the payloads. The impact on the detection performances of the secondary attacks depends more on the targeted CAN ID than the type of attack: Attacks on **Class 2** IDs (few flips in the payload) achieve a small baseline (primary attack) recall; thus the reduction of the performances caused by the secondary attacks is limited. Attacks on **Class 4** IDs (one-bit signals payloads), on the contrary, show a high baseline recall, and thus the impact of the secondary attack is more evident, with a reduction of the detection rate up to 43%. Attacks on **Class 1** and **Class 3** IDs often obtain a very high baseline Recall. However, the reduction caused by the secondary attacks is not

Table 2: Average performance of evasion attacks over all the attack types. The grey row represents the recall baseline of the IDSs against primary attacks. The performance is represented as the delta between the secondary and the primary attacks.

		DEF (IDS)							DEF (IDS)						
		Class 1							Class 4						
		[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR	[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR
	Baseline	**0.80**	**0.81**	**0.78**	**0.81**	**0.81**	**0.38**	**0.50**	**0.98**	**0.98**	**0.12**	**0.98**	**0.98**	**0.43**	**0.00**
ATK (Oracle)	W.B.	0.00	-0.01	0.00	-0.18	-0.18	-0.38	-0.31	0.00	-0.25	-0.12	-0.25	-0.25	-0.43	0.00
	[26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.25	-0.25	0.00	-0.25	-0.25	0.00	0.00
	L. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.12	0.00	0.00	-0.18	0.00
	NN	0.00	0.00	0.00	0.00	0.00	+0.01	0.00	-0.25	-0.27	-0.12	-0.25	-0.31	-0.43	0.00
	S. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.25	-0.25	-0.01	-0.25	-0.25	-0.03	0.00
	S. LSTM	0.00	0.00	0.00	0.00	0.00	+0.01	+0.01	-0.25	-0.27	-0.12	-0.25	-0.31	-0.43	0.00
	Hamm.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.04	0.00	0.00	-0.08	0.00
	VAR	-0.02	-0.03	-0.17	-0.02	-0.08	-0.35	-0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		Class 2							Class 5						
		[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR	[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR
	Baseline	**0.56**	**0.00**	**0.28**	**0.00**	**0.00**	**0.19**	**0.00**	**0.97**	**0.97**	**0.48**	**0.75**	**0.84**	**0.00**	**0.02**
ATK (Oracle)	W.B.	-0.13	0.00	-0.28	0.00	0.00	-0.19	0.00	-0.34	-0.48	-0.36	-0.50	-0.48	0.00	-0.02
	[26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	L. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.20	-0.38	-0.04	-0.18	-0.23	0.00	0.00
	NN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	-0.08	-0.01	-0.01	0.00	+0.02
	S. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.19	-0.34	-0.16	-0.31	-0.30	0.00	-0.01
	S. LSTM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.21	-0.28	-0.05	-0.15	-0.17	0.00	+0.01
	Hamm.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	VAR	-0.45	0.00	-0.19	0.00	0.00	-0.19	0.00	0.00	-0.02	0.00	0.00	0.00	0.00	-0.01
		Class 3							Class 6						
		[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR	[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR
	Baseline	**0.91**	**0.98**	**0.65**	**0.95**	**0.91**	**0.70**	**0.67**	**0.72**	**0.36**	**0.17**	**0.37**	**0.39**	**0.04**	**0.05**
ATK (Oracle)	W.B.	0.00	0.00	0.00	0.00	0.00	-0.70	0.00	-0.01	-0.35	-0.17	-0.36	-0.39	-0.04	-0.05
	[26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	L. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.20	-0.27	-0.04	-0.28	-0.30	-0.04	-0.05
	NN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	+0.01	-0.02	+0.01	+0.02	-0.04	-0.05
	S. GRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.14	-0.23	-0.06	-0.21	-0.23	-0.04	-0.05
	S. LSTM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.24	-0.29	-0.09	-0.32	-0.33	-0.04	-0.05
	Hamm.	0.00	0.00	+0.13	0.00	0.00	-0.03	+0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	VAR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	+0.01	0.00	0.00	0.00	-0.05

(a) Divided by CAN ID class.

		DEF (IDS)							DEF (IDS)						
		Drop							Replay						
		[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR	[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR
	Baseline	**0.49**	**0.53**	**0.10**	**0.50**	**0.49**	**0.11**	**0.11**	**0.87**	**0.67**	**0.22**	**0.54**	**0.60**	**0.03**	**0.11**
ATK (Oracle)	W.B.	-0.16	-0.18	-0.02	-0.16	-0.17	-0.11	0.00	-0.06	-0.18	0.00	-0.30	0.00	-0.03	0.00
	[26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.17	-0.17	0.00	-0.17	0.00	0.00	0.00
	L. GRU	-0.13	-0.16	0.00	-0.14	-0.16	0.00	0.00	-0.11	-0.02	+0.07	+0.07	+0.04	0.00	0.00
	NN	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	-0.17	-0.17	0.00	-0.17	-0.17	0.00	0.00
	S. GRU	-0.13	-0.16	0.00	-0.15	-0.15	0.00	0.00	-0.26	-0.17	+0.02	-0.15	-0.15	0.00	0.00
	S. LSTM	-0.14	-0.17	0.00	-0.15	-0.15	0.00	+0.01	-0.30	-0.17	+0.05	-0.10	-0.11	0.00	+0.01
	Hamm.	0.00	0.00	+0.06	0.00	0.00	-0.03	0.00	0.00	0.00	+0.07	0.00	0.00	+0.02	0.00
	VAR	0.00	0.00	0.00	0.00	0.00	0.00	+0.01	-0.01	-0.02	-0.04	-0.01	-0.04	0.00	+0.01
		Fuzzy							Seamless Change						
		[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR	[26]	L. GRU	NN	S. GRU	S. LSTM	Hamm.	VAR
	Baseline	**0.98**	**0.82**	**0.70**	**0.83**	**0.83**	**0.56**	**0.38**	**0.96**	**0.71**	**0.64**	**0.71**	**0.72**	**0.46**	**0.38**
ATK (Oracle)	W.B.	-0.04	-0.24	-0.32	-0.23	-0.23	-0.56	-0.10	-0.05	-0.12	-0.28	-0.16	-0.15	-0.46	-0.10
	[26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	L. GRU	-0.01	-0.17	-0.12	-0.15	-0.15	-0.09	-0.04	0.00	-0.08	-0.08	-0.08	-0.09	-0.06	-0.04
	NN	0.00	0.00	-0.07	0.00	-0.02	-0.14	+0.01	0.00	0.00	-0.08	0.00	-0.01	-0.14	+0.01
	S. GRU	-0.01	-0.13	-0.06	-0.13	-0.12	-0.04	-0.04	+0.01	-0.09	-0.11	-0.09	-0.09	-0.01	-0.04
	S. LSTM	-0.01	-0.15	-0.12	0.16	-0.17	-0.17	-0.03	-0.02	-0.07	-0.10	-0.08	-0.11	-0.13	-0.03
	Hamm.	-0.15	0.00	-0.10	0.00	0.00	-0.10	0.00	-0.15	0.00	-0.10	0.00	0.00	-0.10	0.00
	VAR	0.00	0.00	-0.03	0.00	-0.01	-0.15	-0.06	0.00	-0.01	-0.04	0.00	-0.01	-0.08	-0.06

(b) Divided by attack type.

Table 3: Switched bits in the Replay attack by the training set, Oracle, and CAN ID Class. We report the minimum, the maximum, and the average number of bits switched in a single payload, the variance of the distribution, and the number of payloads left unmodified. A row is set to '-' when no bit has been switched in all payloads.

	Class 1	Min	Max	Avg	Var	Unmodified	Class 5	Min	Max	Avg	Var	Unmodified
	CANnolo	-	-	-	-	173.0	**CANnolo**	5.00	14.00	6.89	3.07	152.88
	L. GRU	2.00	2.00	2.00	0.00	172.00	**L. GRU**	2.00	12.00	6.00	6.15	221.88
	NN	-	-	-	0.00	173.00	**NN**	-	-	-	-	0.00
DEF	**S. GRU**	2.00	2.00	2.00	0.00	51.00	**S. GRU**	1.00	8.00	1.81	1.32	30.88
	S. LSTM	2.00	2.00	2.00	0.00	51.00	**S. LSTM**	1.00	20.00	4.13	12.67	87.00
	Hamming	-	-	-	0.00	0.00	**Hamming**	-	-	-	-	0.00
	VAR	1.00	2.00	1.29	0.21	0.00	**VAR**	-	-	-	-	0.00
	CANnolo	-	-	-	-	173.00	**CANnolo**	1.00	1.00	1.00	0.00	120.00
	L. GRU	-	-	-	-	173.00	**L. GRU**	1.00	25.00	6.77	17.74	46.00
	NN	-	-	-	-	173.00	**NN**	-	-	-	-	0.00
ATK	**S. GRU**	-	-	-	-	173.00	**S. GRU**	1.00	12.00	2.46	2.79	48.00
	S. LSTM	-	-	-	-	173.00	**S. LSTM**	1.00	15.00	4.15	7.80	97.00
	Hamming	-	-	-	-	0.00	**Hamming**	-	-	-	-	0.00
	VAR	1.00	2.00	1.29	0.21	0.00	**VAR**	4.00	11.00	5.39	4.39	7.00
	Class 3	**Min**	**Max**	**Avg**	**Var**	**Unmodified**	**Class 6**	**Min**	**Max**	**Avg**	**Var**	**Unmodified**
	CANnolo	-	-	-	-	172.00	**CANnolo**	1.00	3.00	1.60	0.80	214.00
	L. GRU	-	-	-	-	172.00	**L. GRU**	1.00	3.00	1.19	0.17	1.00
	NN	-	-	-	-	172.00	**NN**	-	-	-	-	0.00
DEF	**S. GRU**	-	-	-	-	172.00	**S. GRU**	1.00	2.00	1.37	0.24	0.00
	S. LSTM	-	-	-	-	172.00	**S. LSTM**	1.00	2.00	1.38	0.24	0.00
	Hamming	1.00	37.00	10.54	124.41	0.00	**Hamming**	-	-	-	-	0.00
	VAR	-	-	-	-	9.00	**VAR**	-	-	-	-	0.00
	CANnolo	-	-	-	-	171.00	**CANnolo**	1.00	2.00	1.50	0.50	25.00
	L. GRU	-	-	-	-	141.00	**L. GRU**	1.00	2.00	1.16	0.14	0.00
	NN	-	-	-	-	172.00	**NN**	-	-	-	-	0.00
ATK	**S. GRU**	-	-	-	-	83.88	**S. GRU**	0.00	0.00	1.11	0.10	0.00
	S. LSTM	-	-	-	-	0.59	**S. LSTM**	0.00	0.00	1.31	0.21	0.00
	Hamming	1.00	30.00	9.52	88.63	0.00	**Hamming**	-	-	-	-	0.00
	VAR	-	-	-	-	9.00	**VAR**	-	-	-	-	0.00

significant. This is due to the peculiar properties of such CAN-IDs: **Class 3** payloads follow strict patterns, while **Class 1** payloads can only assume values from a restricted set. In both cases, modifying even a few bits results in obvious deviations from the pattern or the set of valid values, easily detected by both IDSs and Oracle. Finally, attacks on **Class 5** and **Class 6** IDs do not have any property that forces the payloads to assume specific values. Therefore, the attacker has more room for maneuver, and the secondary attacks are more effective, with a reduction in the detection rate up to 50%. As expected, the IDSs recall reduction in the white-box scenario is generally much higher than in the other two. However, the reduction obtained in the Gray-box scenario is very similar to the one obtained in the Black-box scenario, especially using Oracles based on RNNs. This suggests a good transferability of the adversarial attacks between different models. Independently from the attack type and CAN ID, from a defense perspective, autoencoder-based LSTMs (e.g., CANnolo) are the most effective in mitigating evasion attacks based on the approach presented in this paper. From the attacker's perspective, instead, the Small-LSTM is overall the best oracle to generate evasive attacks.

5.4 Exp. 2: Required Perturbation Analysis

For each payload in the secondary attacks, we measure the hamming distance from its unperturbed version in the primary attack. Due to the very high number of payloads in each attack, we measure the average distribution using the minimum, maximum, average, and variance values. Such metrics are measured only for the payloads whose hamming distance is greater than zero (i.e., perturbed payloads). We do not consider classes 2 and 4 for this experiment since they have a very low number of switchable bits.

Summary of results. It is not always possible to successfully turn a payload into an evading point. This happens when the algorithm meets a termination condition before an evading point is found. In such cases, we return the original input payload without modifications. When, instead, an evasive modification is found, we can measure how many bits the algorithm switched to obtain it. In reporting the obtained results, we write $\mu \pm \sigma$, where μ is the average number of switched bits and σ is the square root of its variance. For brevity, in Table 3 we present the results only for the Replay attack, which well summarizes the overall results for the various attacks. Regarding attacks on **Class 5** IDs (high autocorrelated payloads), our approach generally needs to switch, on average, more than 5 bits per payload. This CAN-ID type is thus very expensive to modify. Attacks on **Class 6** IDs (unclassified), instead, can be perturbed with far less effort: the highest number of average switched bits is 2.8 ± 1.18. For attacks on **Class 3** IDs (repeated pattern payloads), almost every Oracle fail in modifying even a single payload. The only exception is Hamming, which, however, needs to switch up to 13.8 ± 11.96 or 14.49 ± 9.56 bits per payload on average. Finally, in attacks on **Class 1** IDs (few changed values payloads), the number of switched bits ranges from 1.29 ± 0.46 to 5.60 ± 0.55. However, we notice that many Oracles failed to modify even a single payload in many cases.

6 Conclusions

In this paper, we studied the feasibility of adversarial machine learning attacks against intrusion detection systems (IDSs) in the automotive field, and designs a methodology to perform evasion attacks. The approach involves perturbing attack payloads and testing them against an Oracle to generate evasive payloads, attempting to evade a target IDS exploiting the transferability property. The results show that the constraints imposed by the automotive domain limit the attacker's capabilities, and the effectiveness of the adversarial approach is reduced. The average perturbation needed for concealing an attack is between 1 to 3 bits per payload, and in some scenarios, the attacker can significantly reduce the detection power of the IDS under analysis.

References

1. Al-Jarrah, O.Y., Maple, C., Dianati, M., Oxtoby, D., Mouzakitis, A.: Intrusion detection systems for intra-vehicle networks: A review. IEEE Access **7**, 21266–

21289 (2019). https://doi.org/10.1109/ACCESS.2019.2894183, https://doi.org/10.1109/ACCESS.2019.2894183
2. Barreno, M., Nelson, B., Joseph, A., Tygar, J.: The security of machine learning. Machine Learning **81**(2), 121–148 (Nov 2010). https://doi.org/10.1007/s10994-010-5188-5, https://doi.org/10.1007/s10994-010-5188-5
3. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Lin, F., Lee, D., Lin, B.P., Shieh, S., Jajodia, S. (eds.) Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006. pp. 16–25. ACM (2006). https://doi.org/10.1145/1128817.1128824, https://doi.org/10.1145/1128817.1128824
4. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 387–402. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018). https://doi.org/https://doi.org/10.1016/j.patcog.2018.07.023, https://www.sciencedirect.com/science/article/pii/S0031320318302565
6. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017. pp. 39–57. IEEE Computer Society (2017). https://doi.org/10.1109/SP.2017.49, https://doi.org/10.1109/SP.2017.49
7. Carminati, M., Santini, L., Polino, M., Zanero, S.: Evasion attacks against banking fraud detection systems. In: Egele, M., Bilge, L. (eds.) 23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2020, San Sebastian, Spain, October 14-15, 2020. pp. 285–300. USENIX Association (2020), https://www.usenix.org/conference/raid2020/presentation/carminati
8. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: A survey. CoRR **abs/1810.00069** (2018), http://arxiv.org/abs/1810.00069
9. Cho, K., Shin, K.G.: Fingerprinting electronic control units for vehicle intrusion detection. In: Holz, T., Savage, S. (eds.) 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. pp. 911–927. USENIX Association (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho
10. Dang, H., Huang, Y., Chang, E.: Evading classifiers by morphing in the dark. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 119–133. ACM (2017). https://doi.org/10.1145/3133956.3133978, https://doi.org/10.1145/3133956.3133978
11. Demetrio, L., Coull, S.E., Biggio, B., Lagorio, G., Armando, A., Roli, F.: Adversarial exemples: A survey and experimental evaluation of practical attacks on machine learning for windows malware detection. ACM Trans. Priv. Secur. **24**(4), 27:1–27:31 (2021). https://doi.org/10.1145/3473039, https://doi.org/10.1145/3473039
12. Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., Roli, F.: Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In: Heninger, N., Traynor, P. (eds.)

28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019. pp. 321–338. USENIX Association (2019), https://www.usenix.org/conference/usenixsecurity19/presentation/demontis

13. Erba, A., Taormina, R., Galelli, S., Pogliani, M., Carminati, M., Zanero, S., Tippenhauer, N.O.: Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In: ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020. pp. 480–495. ACM (2020). https://doi.org/10.1145/3427228.3427660, https://doi.org/10.1145/3427228.3427660
14. de Faveri Tron, A., Longari, S., Carminati, M., Polino, M., Zanero, S.: Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022. pp. 711–723. ACM (2022). https://doi.org/10.1145/3548606.3560618, https://doi.org/10.1145/3548606.3560618
15. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6572
16. Hanselmann, M., Strauss, T., Dormann, K., Ulmer, H.: Canet: An unsupervised intrusion detection system for high dimensional CAN bus data. IEEE Access **8**, 58194–58205 (2020). https://doi.org/10.1109/ACCESS.2020.2982544, https://doi.org/10.1109/ACCESS.2020.2982544
17. Huang, W., Peng, X., Shi, Z., Ma, Y.: Adversarial attack against LSTM-based DDoS intrusion detection system. In: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). pp. 686–693 (2020). https://doi.org/10.1109/ICTAI50040.2020.00110
18. Ibitoye, O., Khamis, R.A., Matrawy, A., Shafiq, M.O.: The threat of adversarial attacks on machine learning in network security - A survey. CoRR **abs/1911.02621** (2019), http://arxiv.org/abs/1911.02621
19. Kang, M.J., Kang, J.: A novel intrusion detection method using deep neural network for in-vehicle network security. In: IEEE 83rd Vehicular Technology Conference, VTC Spring 2016, Nanjing, China, May 15-18, 2016. pp. 1–5. IEEE (2016). https://doi.org/10.1109/VTCSpring.2016.7504089, https://doi.org/10.1109/VTCSpring.2016.7504089
20. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=HJGU3Rodl
21. Lee, H., Jeong, S.H., Kim, H.K.: OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In: 15th Annual Conference on Privacy, Security and Trust, PST 2017, Calgary, AB, Canada, August 28-30, 2017. pp. 57–66. IEEE Computer Society (2017). https://doi.org/10.1109/PST.2017.00017, https://doi.org/10.1109/PST.2017.00017
22. Li, Y., Lin, J., Xiong, K.: An adversarial attack defending system for securing in-vehicle networks. In: 18th IEEE Annual Consumer Communications & Networking Conference, CCNC 2021, Las Vegas, NV, USA, January 9-12, 2021. pp. 1–6. IEEE (2021). https://doi.org/10.1109/CCNC49032.2021.9369569, https://doi.org/10.1109/CCNC49032.2021.9369569

23. Longari, S., Cannizzo, A., Carminati, M., Zanero, S.: A secure-by-design framework for automotive on-board network risk analysis. In: 2019 IEEE Vehicular Networking Conference, VNC 2019, Los Angeles, CA, USA, December 4-6, 2019. pp. 1–8. IEEE (2019). https://doi.org/10.1109/VNC48660.2019.9062783, https://doi.org/10.1109/VNC48660.2019.9062783
24. Longari, S., Nichelini, A., Pozzoli, C.A., Carminati, M., Zanero, S.: Candito: Improving payload-based detection of attacks on controller area networks. In: Dolev, S., Gudes, E., Paillier, P. (eds.) Cyber Security Cryptography and Machine Learning - First International Conference, CSCML 2023, Online, June 29-30, 2023, Proceedings. Lecture Notes in Computer Science, Springer (2023)
25. Longari, S., Penco, M., Carminati, M., Zanero, S.: Copycan: An error-handling protocol based intrusion detection system for controller area network. In: Cavallaro, L., Kinder, J., Holz, T. (eds.) Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy, CPS-SPC@CCS 2019, London, UK, November 11, 2019. pp. 39–50. ACM (2019). https://doi.org/10.1145/3338499.3357362, https://doi.org/10.1145/3338499.3357362
26. Longari, S., Valcarcel, D.H.N., Zago, M., Carminati, M., Zanero, S.: Cannolo: An anomaly detection system based on LSTM autoencoders for controller area network. IEEE Trans. Netw. Serv. Manag. **18**(2), 1913–1924 (2021). https://doi.org/10.1109/TNSM.2020.3038991, https://doi.org/10.1109/TNSM.2020.3038991
27. Loukas, G., Vuong, T., Heartfield, R., Sakellari, G., Yoon, Y., Gan, D.: Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access **6**, 3491–3508 (2018). https://doi.org/10.1109/ACCESS.2017.2782159, https://doi.org/10.1109/ACCESS.2017.2782159
28. Marchetti, M., Stabili, D.: READ: reverse engineering of automotive data frames. IEEE Trans. Inf. Forensics Secur. **14**(4), 1083–1097 (2019). https://doi.org/10.1109/TIFS.2018.2870826, https://doi.org/10.1109/TIFS.2018.2870826
29. Marchetti, M., Stabili, D., Guido, A., Colajanni, M.: Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In: 2nd IEEE International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow, RTSI 2016, Bologna, Italy, September 7-9, 2016. pp. 1–6. IEEE (2016). https://doi.org/10.1109/RTSI.2016.7740627, https://doi.org/10.1109/RTSI.2016.7740627
30. Miller, C., Valasek, C.: Adventures in automotive networks and control units. Def Con **21**(260-264), 15–31 (2013)
31. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. Black Hat USA **2015**(S 91) (2015)
32. Müter, M., Asaj, N.: Entropy-based anomaly detection for in-vehicle networks. In: IEEE Intelligent Vehicles Symposium (IV), 2011, Baden-Baden, Germany, June 5-9, 2011. pp. 1110–1115. IEEE (2011). https://doi.org/10.1109/IVS.2011.5940552, https://doi.org/10.1109/IVS.2011.5940552
33. Nichelini, A., Pozzoli, C.A., Longari, S., Carminati, M., Zanero, S.: Canova: a hybrid intrusion detection framework based on automatic signal classification for can. Computers & Security p. 103166 (2023)
34. Papernot, N., McDaniel, P.D., Goodfellow, I.J.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR **abs/1605.07277** (2016), http://arxiv.org/abs/1605.07277

35. Papernot, N., McDaniel, P.D., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Karri, R., Sinanoglu, O., Sadeghi, A., Yi, X. (eds.) Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017. pp. 506–519. ACM (2017). https://doi.org/10.1145/3052973.3053009, https://doi.org/10.1145/3052973.3053009
36. Papernot, N., McDaniel, P.D., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016. pp. 372–387. IEEE (2016). https://doi.org/10.1109/EuroSP.2016.36, https://doi.org/10.1109/EuroSP.2016.36
37. Robert Bosch GMBH: Can specification, version 2.0. Standard, Robert Bosch GmbH, Stuttgart, Germany (1991)
38. Stabili, D., Marchetti, M., Colajanni, M.: Detecting attacks to internal vehicle networks through hamming distance. In: 2017 AEIT International Annual Conference. pp. 1–6. IEEE (2017)
39. Young, C., Zambreno, J., Olufowobi, H., Bloom, G.: Survey of automotive controller area network intrusion detection systems. IEEE Des. Test **36**(6), 48–55 (2019). https://doi.org/10.1109/MDAT.2019.2899062, https://doi.org/10.1109/MDAT.2019.2899062
40. Zago, M., Longari, S., Tricarico, A., Carminati, M., Gil Pérez, M., Martínez Pérez, G., Zanero, S.: ReCAN – dataset for reverse engineering of controller area networks. Data in Brief **29**, 105149 (2020). https://doi.org/https://doi.org/10.1016/j.dib.2020.105149, https://www.sciencedirect.com/science/article/pii/S2352340920300433