Evaluation Framework for Network Intrusion Detection Systems for In-Vehicle CAN

Guillaume Dupont*, Jerry den Hartog*, Sandro Etalle* and Alexios Lekidis*†
Dept. of Mathematics and Computer Science, Eindhoven University of Technology
Forescout Technologies, Eindhoven
Email: *{g.f.c.dupont, j.d.hartog, s.etalle, a.lekidis}@tue.nl, †alexis.lekidis@forescout.com

***Abstract*—Modern vehicles are complex safety critical cyber physical systems, that are connected to the outside world, with all security implications it brings. Different network intrusion detection systems (NIDSs) proposed for the CAN bus, the predominant type of in-vehicle network, to improve security are hard to compare due to disparate evaluation methods adopted. In this paper we provide the means to compare CAN NIDSs on equal footing and evaluate the ones detailed in the literature. Based on this we observe some limitation of existing approaches and why in the CAN setting it is intrinsically difficult to distinguish benign from malicious payload. We argue that "meaning-aware" detection (a concept we define) which is challenging (but perhaps not impossible) to create for this setting.**

***Index Terms*—In-vehicle networks, CAN bus, intrusion detection, car hacking**

I. Introduction

A modern vehicle is a complex cyber physical network rather than purely a mechanical device. Over time vehicles became more intelligent due to manufacturers deploying innovative applications and connecting cars with diverse parties such as personal devices, VANETs and the Internet. The increased connectivity, while offering many benefits, also come with evident security risks. Vehicle complexity offers a large attack surface, allowing the execution of (remote) attacks that could have life threatening consequences. Researchers have already demonstrated the ability to remotely take over the control of diverse vehicles at speed [8]–[11].

Implementing security measures in a safety critical complex cyber physical network is difficult as one has to guarantee the security measures do not impact the existing functionalities. Invasive solutions that could impact availability of safety critical functions or require a major redesign of the vehicle or its components will not be acceptable to the manufacturer. Network intrusion detection systems (NIDS) provide a non-invasive security measure that is well-established in other fields (general IT, ICS) and thus seems potentially suitable for the automotive setting. In fact, in the literature we find a number of NIDS proposals for the predominant type of in-vehicle network: the Controller Area Network (CAN) bus (see Table I). The question raises how these systems compare to each other and in general which kind of coverage they provide against attacks on CAN systems.

In this paper we address this question of how CAN NIDSs perform. Our contributions are: (1) A unified evaluation framework enabling comparison of approaches with its datasets made publicly available. (2) Evaluation of proposed systems (which provide sufficient details for faithful reproduction) using the framework. (3) A discussion of observed limitations of existing CAN NIDS limitations (such as relying on narrow indicators of compromise or only addressing a specific type of attack) and the lack of *content aware* NIDS for CAN bus.

Below we introduce the CAN protocol and known attack (Section II) and cover existing NIDS and their validation approaches (Section III). Next we provide a uniform evaluation framework (Section IV) and use it to compare NIDS approaches (Section V). We provide a general discussion based on the results (Section VI) before concluding (Section VII).

II. Preliminaries

In this section we provide background information: an overview of the Controller Area Network (CAN) protocol, followed by known CAN bus attacks.

A. Controller Area Network

CAN is a message-oriented transmission protocol originally designed for the automotive industry in an attempt to reduce the wiring complexity of automobiles [12]. An *Electronic Control Unit* (ECU) uses CAN to broadcast its frames onto a bus [13]. There is no addressing scheme. Instead each frame is assigned an 11 bit unique identifier, known as the "arbitration ID" or "CAN ID", which defines both the function and the priority of the frame where 0x000 has highest priority and 0x7FF lowest. The priority resolves conflict when two or more nodes try to send a frame at the same time. A frame may also contains up to 8 bytes of data. While an ECU may broadcast multiple CAN IDs, each CAN ID is bound to a single ECU; two ECUs cannot send data frames with the same CAN ID. Every time a frame is transmitted, all ECUs on the bus will receive it, and will determine, based on the CAN ID whether they should accept and further process the message [14].

Generally speaking, ECUs use two types of data frames: normal messages and diagnostic messages [9]. Normal messages follow a proprietary format and are transmitted by the ECUs during regular operations. Diagnostic messages are defined by diagnostic communication protocol such as Unified Diagnostic Services (ISO 14229-1): they are special messages normally sent by mechanic's tool (or "tester") during maintenance operations. Depending on the services implemented on

978-1-7281-0142-2/19/$31.00 ©2019 IEEE

an ECU, diagnostic messages can be used to perform various actions such as querying information or updating its firmware.

Car makers deploy the reliable CAN protocol on high-speed buses inside their vehicles to interconnect safety-critical ECUs. However, its simplicity implies that many common security measures such as communication encryption or ECU authentication are not possible. In addition, the broadcasting nature of the protocol allows anyone on the bus to read and send messages, which can have critical consequences as explained in the next section.

B. CAN bus attacks

As demonstrated in previous research [8], [9], [11], [15], an attacker has a broad range of entry points which can be leveraged to gain access to the CAN bus. The context of in-vehicle CAN bus determines the types of attacks that can be performed once access is obtained. We cover some aspects of this context before going into the attacks themselves.

(A1) Recall that the CAN bus is an unauthenticated and unencrypted broadcast bus where anyone can read and send any message. (A2) ECUs may have *message confliction* resolution mechanisms. When receiving conflicting values (e.g. a real value and a spoofed value from a compromised ECU) an ECU will react differently depending on the mechanism implemented [10]. (A3) As mentioned above, ECUs have a "diagnostic mode" allowing maintenance operations. (A4) The mapping from CAN IDs to their respective ECUs and functionalities varies across different car models. Car manufacturers design their cars in their own specific ways, making the architectural ecosystem of vehicles very diverse.

To achieve his/her goal, an attacker may need to know the architecture of the vehicle and which CAN ID is linked to which ECU. We see that, when first gaining access to the CAN bus, this is not straightforward without prior knowledge (see A4). Thus, during a reconnaissance phase, an attacker may perform a *fuzzing attack* by sending messages with random CAN IDs and payload values and observing the reaction. This attack can allow an attacker to learn about the architecture and the behavior of the vehicle and its ECUs.

As demonstrated in [9], an attacker can send a diagnostic message (see A3) in order to open a diagnostic session on an ECU. Depending on the services implemented on that ECU, diagnostic messages can be extremely powerful as shown, for example, by attacks on Toyota Prius and a Ford Escape [16].

An attacker can send frames with the CAN ID and payload of his/her choice (see A1). The attacker can *replay* observed messages or, with sufficient knowledge of the car's CAN implementation (e.g. how the payload for a given CAN ID is encoded), an attacker could *spoof* messages with a payload modified to achieve the desired outcome (e.g. making the instrument cluster display an arbitrary speed). However, as the legitimate ECU would keep on sending its frames, such spoofed messages are likely to conflict with legitimate ones. As such the attacker must also deal with the message conflict resolution in the ECUs (see A2). ECUs will handle conflicting values in different ways. Simple ECUs may accept the messages that are the most predominant. Against such ECUs an attacker could use *flooding*: send his/her crafted frames faster (in practice 20 to 100 times faster [1]) than the legitimate messages to ensure his/her frames are accepted.

A straightforward *Denial of Service (DoS)* attack preventing ECUs from communicating, consists in flooding the bus with frames with CAN ID set to 0x000. The CAN ID of a frame determines its priority (see Section II-A) so repeatedly sending messages with this highest possible priority CAN ID will prevent other ECUs from transmitting their frames. Such an attack can put the car in an unstable state [16].

Low level attacks relying on bus signal tempering, e.g. [17]–[20] can achieve *bus-off* attacks which will cause an ECU to stop sending frames. As these papers explain, these attacks are not detectable by current frame-based NIDS since they do not involve injecting whole CAN frames, and are thus out of scope for this study. However, we do consider a consequence of this attack that frame-based NIDS might detect: in *suspension attacks* the attacker somehow causes an ECU to suddenly stop emitting its frames [21], [22]. A frame-based NIDS may be able to detect a silenced ECU.

TABLE I: Validation approaches of proposed CAN NIDs

	Attacks covered					Validation data			
Papers	Diagnostic	Fuzzing	Replay	Suspension	DoS	Real car	Prototype	Simulation	Dataset (in min)
[1]*		✔	✔		✔	✔			-
[2]*		✔	✔	✔	✔	✔	✔		-
[3]*		●	✔			-	-	-	-
[4]*			✔	✔		✔			5
[5]*			✔		✔	✔			-
[6]*	●		✔		✔	✔			40
[7]*		✔	✔			✔			240
[23]		✔	✔				✔		-
[24]			✔			✔	✔		-
[25]				●	●			✔	-
[21]		✔	✔	✔		✔	✔		2 $\sim$ 33
[26]			✔		✔	✔			0.25 $\sim$ 5
[27]		✔	✔			✔			-
[22]			✔	✔		✔			1140
[28]		✔	●		✔	✔			8
[29]		●	✔			✔			240
[30]			✔					✔	-
[31]		✔	✔			✔		✔	-
[32]		✔	✔			✔			-
[33]			✔		✔	✔			8
[34]		●				✔			-
[35]		✔	✔			✔			3.4
[36]		✔	✔		✔	✔			30 $\sim$ 40
[37]		●				✔			-
[38]		●						✔	-

✔ Explicitly mentioned in paper ● Implicitly mentioned in paper
* We implemented and tested it - Unclear or not mentioned at all

III. CAN Network Intrusion Detection Systems

We surveyed and classified CAN NIDS in our technical report [39]. To draw conclusions regarding the performance

of these systems one would need to provide a quantitative comparison addressing standard NIDS performance metrics: false positives and detection rate. However, as can be seen from Table I, CAN NIDS authors use disparate validation strategies, varying in attacks covered as well as in nature and size of data sets. Table I presents:

Attacks covered For each of the NIDS the authors considered a subset of the attacks presented in Section II-B. These columns show which of the five attacks the paper considers.

Validation data We classify the validation data into in three categories: *real car* - authors captured data from a licensed vehicle, *prototype* - they build a prototype of a CAN bus (see Figure 1 for an example) or *simulation* - they used a CAN bus simulator such as CANoe[1]. We also indicate the approximate amount of data (in min) they gathered.

To compare the performance of these systems we need to compare them on common ground. To make this possible we create a unified evaluation framework, which is presented in the following section.

IV. Unified Evaluation Framework

In this section we propose a framework for evaluation of CAN NIDSs. As CAN implementations can vary a lot from one car to another, it is essential to include data from different cars. Also data covering the different types of attacks (see Section II-B) is needed. With such data in hand standard NIDS testing methodology can be used. We recorded data from two cars, an Opel Astra and Renault Clio, and a CAN bus prototype we built. The data we recorded (along with our implementations, see Section V) are available online on the TUe Security Group website[2]. We also consider Kia Soul data provided online[3] by the Hacking and Countermeasure Research Lab (HCRL).

a) Driving data collection: We collect data from both cars and the prototype. For the cars we record CAN traffic through a CAN-to-USB interface (CANtact[4]) connected to the OBD-II port of the car. CAN bus traffic is logged on our laptop with CAN sniffer `candump` from the *canutils* tool suit[5]. These vehicles do not have a firewall or gateway filtering CAN messages. We drive the Opel for approximately 20 minutes (analysis of this set shows the data is already comprehensive after a few minutes) and the Renault for about 4 minutes in urban environments (university campus and city). We believe this data to be sufficiently representative for normal urban driving for our purpose, though it obviously does not cover rare/specific events (e.g. empty tank). Such corner cases could provide additional challenges to NIDS, meaning that the results in our framework should be interpreted as best case scenario for the NIDS.

[1]https://www.vector.com/int/en/products/products-a-z/software/canoe/
[2]https://security1.win.tue.nl (under *Research/Software and Data*)
[3]http://ocslab.hksecurity.net/
[4]https://linklayer.github.io/cantact/
[5]https://github.com/linux-can/can-utils

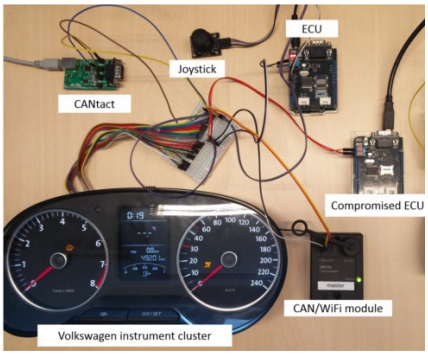

Fig. 1: "Homemade" CAN bus prototype

The prototype we built is depicted in Fig. 1. It comprises 2 ECUs (Arduino boards mounted with CAN shield), a Volkswagen instrument cluster and a joystick which is used to replicate a vehicle's throttle: when pushed forward, one can see the speed increasing on the instrument cluster. (The CAN/Wifi module is not used in this study.) We connect the CANtact to the bus and we record the data of the prototype for about 4 minutes, while using the joystick to emulate a driver accelerating and decelerating.

We partition the data into *training datasets* (70%) for training NIDSs and *testing dataset* (30%) for evaluating their performance with respect to false positives. Table II gives an overview of the resulting datasets.

b) Attack dataset creation: To create attack datasets covering the attacks from Section II-B, we perform the attacks on the prototype and simulate them on the two cars. This simulation is done by modifying normal driving data that is not in the training set in different ways. For diagnostic attack, 10 packets with CAN IDs greater than 0x700 [40] are injected at random position in the files. In a similar fashion, two different types of fuzzing attacks are performed: one consisting of injecting 10 packets with unknown CAN IDs, i.e. CAN IDs not present in the training dataset, and below CAN ID 0x700. The other fuzzing attack involves the replacement of the payload of 10 frames belonging to a legitimate CAN ID, without injecting new packets. The bytes of the payload are set to values not used for this CAN ID in the training dataset.

The replay attack for the two cars is replicated by injecting 30 times an arbitrary packet seen in the dataset. By adjusting the timestamps, we simulate the packets being sent 10 times faster than normal onto the bus. Since we do not know the functionality of this CAN ID and its payload's semantic, we injected it without modification. On our prototype, we want to spoof the speed displayed by the speedometer. Since we know the payload semantic of the corresponding CAN ID, we can attack the speedometer to force it to display any speed value, as long as we send the spoofed frames faster than the legitimate ECU. We program the attacking ECU ("compromised ECU" on Fig. 1) to make the speedometer display a speed of 220 Km/h on the instrument cluster for 10 seconds.

To simulate a DoS attack, we replace all messages within a 10 second section by messages with a CAN ID of 0x000 sent

TABLE II: Summary of the normal driving data collected

Car model	Opel Astra		Renault Clio		Prototype		Kia Soul	
Dataset	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Duration (sec)	967.6	414.6	192.5	82.5	189.3	82.0	1331.4	572.8
Number of packets	1883K	806K	270K	115K	70K	30K	259K	1113K
Number of unique CAN IDs	85	85	55	55	17	17	28	27

at a rate of 4 packets per milliseconds. This simulates flooding a 500 Kbps CAN bus (same bus speed as our cars). Finally we simulate a suspension attack by deleting all messages with a certain CAN ID over a period of 10 seconds.

c) HCRL online data sets: The HCRL data contain an attack-free log trace captured from a Kia Soul during 30 minutes. They also address attacks similar to the ones we described previously, logging CAN traffic through the Kia's OBD-II port while message injection attacks are performed. They include a fuzzing attack, in which they injected frames with random CAN ID and payload every 0.5 milliseconds; an impersonation attack injecting frames with CAN ID 0x164; a spoofing attack, consisting in injecting certain CAN IDs related to RPM/gear information every 1 millisecond; and finally a DoS attack in which they injected messages with CAN ID 0x000 every 0.3 milliseconds. These data have also been used in previous NIDS research [33], [36], and some analysis of the data is presented in [31].

V. Evaluation

We use our evaluation framework to compare NIDSs on equal footing. We implement all proposed NIDSs from papers in Table I published before mid 2018 and providing sufficient amount of detail to allow a faithful reproduction. Some papers propose hybrid NIDSs combining different modules, each with their own detection method. For such papers we implement all modules described in sufficient detail. This results in 8 NIDSs from 7 papers (marked by a star in the table). The Python source code of our implementation of these NIDSs is included in the public framework repository (see Section IV).

NIDS N1 reproduces the diagnostic messages detection module suggested by Miller and Valasek in [1] (simply looking for CAN IDs above 0x700 as mentioned in [40]). N2 is a combination of the two pattern-matching algorithms proposed by Ujiie et al. in [2]: one algorithm looks for invalid CAN IDs (also proposed by Abbott-McCune in [23]) and the second for payload values that do not match values previously observed.

NIDSs N3 [3], N4 [4], N5 [5], and N6 [6] leverage the regularity of CAN messages in order to detect CAN bus attacks. Then we also develop two entropy-based systems following the work of Marchetti et al. [7] which fills in details for an idea proposed by Muter and Asaj [26]: N7 computes the entropy for a window of CAN messages, while N8 calculates the entropy values of flows of CAN IDs individually.

We set up and, where needed, train the implemented NIDSs. Then we test both how many false positives they produce and their capability of detecting different types of attacks.

a) Setup and Training: We use the training datasets introduced in the previous section to train the NIDSs (if they have a training phase). Following the author's guidelines for NIDSs parameters as much as possible still leaves some choices to be made. N4 computes an anomaly score over a sequence of elements, but [4] does not specify the size of the sequences. As elements themselves are computed over a one-second window, we choose to set the sequence size to two. To interpret the resulting anomaly scores we consider the maximum and the minimum scores as thresholds. For N7, a parameter k helps to adjust the detection threshold and requires to be adjusted for each car's dataset. For the Opel Astra, the Renault Clio and the Simulator and the Kia Soul, we follow [7] and select the lowest value which gives no false positive; k=4, k=2, k=5 and k=3 respectively. REgarding the Kia, avoiding the 4 false positives that are produced at this setting would require a k so large that no attack would be detected. N8 needs a window size (s messages) tailored to each dataset [7]. We choose the value producing the least number of false positives, giving s=40 for Opel Astra and the simulator, and s=50 for the Renault Clio and the Kia Soul.

For NIDSs with CAN ID specific models (all but N1 and N7) one has to decide what to do with IDs not seen in training. We assume an unknown CAN ID alert is raised, enabling detection of attacks using unusual CAN IDs (Diagnostic, Fuzzing CAN ID and DoS). This assumption does not add any false positives as no such IDs appear in the testing datasets.

b) False positives testing: Table III shows the false positives generated by the NIDSs using the parameters above. Some authors motivate the use of frequency analysis for CAN NIDS by assuming that most (or all) traffic on the bus is regular: if each CAN ID is sent with a defined period, a change in message frequency could indicate an attack. However, our car data shows certain CAN IDs are sent irregularly. To ensure that it is not simply the presence of irregular CAN IDs that causes the false positives reported in Table III we repeat the experiment after removing the irregular CAN IDs, which we arbitrarily define as those having a variance of their inter-arrival time above 0.002. This leads to 21, 10 and 1 irregular CAN IDs accounting for approximately 10%, 6% and 0.5% of the traffic for the Opel Astra, the Renault Clio and the Kia Soul respectively. As results are very similar to those in Table III, we do not present them in detail and conclude that irregular CAN IDs do not influence the false positive rate much.

The assumption made by N1 and several papers that CAN IDs above 0x700 are reserved for diagnostic purposes and should never be seen on the bus while driving is apparently wrong. The Opel data shows CAN ID 0x772, 0x773, 0x778 and 0x788 are repeatedly sent every second and the Volkswagen instrument cluster in our prototype sends out CAN ID 0x727 among other.

TABLE III: False positives; alerts generated for the attack-free datasets.

	Opel Astra		Renault Clio		Prototype		Kia Soul	
NIDS	**# alerts**	**%**	**# alerts**	**%**	**# alerts**	**%**	**# alerts**	**%**
N1	1657	0.20%	0	0%	229	0.76%	0	0%
N2	19917	2.46%	35288	30.428%	2444	8.12%	110949	9.96%
N3	411	0.05%	384	0.331%	8770	29.15%	302	0.027%
N4	0	0%	0	0%	0	0%	0	0%
N5	253468	31.40%	19008	16.39%	12	0.04%	4200	0.377%
N6	165	0.02%	163	0.14%	1806	6%	58	0.005%
N7	0	0%	0	0%	0	0%	4	0.0003%
N8	8	0.0009%	1	0.0008 %	2	0.006	7	0.0006%

A detection module in N2 looks for payload bytes not observed during the training phase. Our implementation raises an alert for each byte value that is not in range. This means that up to 8 alerts may be generated by a single CAN frame. That multiple alerts may actually come from the same message is not accounted for in the reported rate.

Concluding we see some approaches have very high false positive rates. Note that even a seeminly low false positive rate of 0.001% still translates to a false positive every minute.

c) Attack detection: A tick in Table IV indicates the NIDS raises at least one alert when run on that attack dataset. We see that N1 by its nature is very limited in what it can detect. All others are able to detect the attacks on Kia Soul but on the other data sets they each miss several (types of) attacks. Attacks performed somewhat subtly may not be detected.

VI. Discussion

All NIDSs evaluated in the previous section show significant weaknesses; they are not able to detect many of the attacks and often create too many false positives. Approaches that detect when an ECU is put in diagnostic mode like N1 are clearly very limited in their reach. Approaches N3-N8 rely on detecting flooding of ECUs. They do this because, as mentioned, spoofing and similar attacks have to circumvent ECU's message confliction resolution, which is typically achieved through flooding. However, it is hard to draw a clear line above which the amount of messages is malicious. Together this means the approaches use narrow indicator of compromise and/or are prone to false positives.

A CAN frame's payload provides little information without knowing how to interpret it and this interpretation can vary a lot between cars or even between CAN IDs. N2 and N8 consider the payload, but not its meaning, implying they still use rather limited indicators of compromise.

We believe that meaning-aware approaches are needed. Such an approach would interpret a message beyond its representation; instead of treating it like a collection of bytes, it would understand and use its semantics to build better indicators of compromise. This is essential for creating detectors that are hard to circumvent and can give "actionable" alerts; ones that provide enough information to be able to respond to them. Such approaches have been applied in other domains such as Building Automation [41] and Industrial Control System [42].

A possible approach to creating a meaning-aware NIDS is to use a combination of white box and specification-based detection approaches [43], [44]. These approaches require significant information about the system being monitored. However, car manufacturers are reluctant to share specifications (considered intellectual property), and without collaboration with industry actors it would require painstaking reverse engineering efforts for each and every model.

VII. Conclusion

This paper presents an evaluation of various CAN NIDS present in the literature and several conclusions regarding intrusion detection on CAN. This setting is a challenging one for NIDS; CAN messages provide very little information and their meaning varies wildly. As a consequence, existing approaches employ indicators of compromise that offer little guarantees of detecting an actual attack let alone give much information about the attack if one is detected. As such we consider that none of these approaches can be regarded as a meaning-aware intrusion detection system.

Having the manufacturer's specification might enable creation of meaning-aware NIDS for CAN bus. Feasibility of such an approach, which needs the cooperation of the manufacturers, can be investigated by seeing how much (partial) information about the meaning of CAN messages can be used to improve CAN NIDS results.

Acknowledgements

This work was supported by ITEA3 project APPSTACLE (15017) and ECSEL joint undertaking SECREDAS (783119-2).

References

[1] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, 2014.

[2] Y. Ujiie, T. Kishikawa, T. Haga, H. Matsushima, T. Wakabayashi, M. Tanabe, Y. Kitamura, and J. Anzai, "A Method for Disabling Malicious CAN Messages by Using a CMI-ECU," 2016.

[3] M. Gmiden, M. H. Gmiden, and H. Trabelsi, "An intrusion detection method for securing in-vehicle can bus," in *STA*, 2016, pp. 176–180.

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

[5] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks," in *CISRC*, 2017.

[6] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *ICOIN 2016*. IEEE, 2016, pp. 63–68.

[7] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," *IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow*, 2016.

TABLE IV: CAN NIDS testing results for various datasets

	Opel Astra						Renault Clio						Prototype						Kia Soul				
NIDS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Replay	Suspension	DoS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Replay	Suspension	DoS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Spoof speedo.	Suspension	DoS	Fuzzing	Impersonation	Spoof RPM	Spoof drive gear	DoS
N1	✔	-	-	-	-	-	✔	-	-	-	-	-	✔	✔	-	-	-	-	-	-	-	-	-
N2	✔	✔	✔	-	-	✔	✔	✔	✔	-	-	✔	✔	✔	✔	✔	-	✔	✔	✔	✔	✔	✔
N3	✔	✔	-	✔	✔	✔	✔	✔	-	✔	✔	✔	✔	✔	-	✔	-	✔	✔	✔	✔	✔	✔
N4	✔	✔	-	-	-	✔	✔	✔	-	-	-	✔	✔	✔	-	✔	-	✔	✔	✔	✔	✔	✔
N5	✔	✔	-	-	-	-	✔	✔	-	-	-	-	-	-	-	-	-	✔	✔	✔	✔	✔	✔
N6	✔	✔	-	✔	-	✔	✔	✔	-	✔	-	✔	✔	✔	-	✔	✔	✔	✔	✔	✔	✔	✔
N7	-	-	-	-	✔	✔	-	-	-	✔	✔	✔	-	-	-	-	✔	✔	✔	✔	✔	✔	✔
N8	✔	✔	-	✔	✔	✔	✔	✔	-	-	-	✔	✔	✔	-	-	-	✔	✔	✔	✔	✔	✔

✔ Attack detected - Attack not detected

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[9] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.

[10] C. Valasek and C. Miller, "CAN Message Injection - OG Dynamite Edition," http://illmatics.com/canmessageinjection.pdf, 2016.

[11] S. Nie, L. Liu, and Y. Du, "Free-Fall : Hacking Tesla From Wireless To Can Bus," *BlackHat USA 2017*, pp. 1–16, 2017.

[12] S. Corrigan, "Introduction to the Controller Area Network (CAN)," Internet Requests for Comments, Texas Instruments, Tech. Rep., 2002.

[13] "CAN Protocol - Understanding the Controller Area Network Protocol," www.engineersgarage.com/article/what-is-controller-area-network, accessed: 2018-07-25.

[14] "CAN overview," www.ni.com/white-paper/2732/, accessed:2018-07-25.

[15] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Grutese, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study." *USENIX Security Symposium*, vol. 39, no. 4, pp. 11–13, 2010.

[16] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," *Hacktivity 2015*, 2013. [Online]. Available: http://illmatics.com/car_hacking.pdf

[17] S. Froschle and A. Stuhring, "Analyzing the Capabilities of the CAN Attacker," *ESORICS 2017*, vol. 10492, pp. 464–482, 2017.

[18] K.-t. Cho and K. G. Shin, "Error Handling of In-vehicle Networks Makes Them Vulnerable," *CCS*, pp. 1044–1055, 2016.

[19] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *DIMVA 2017*. Springer, 2017, pp. 185–206.

[20] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ECU of the CAN bus," in *15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2018, pp. 1–4.

[21] K.-t. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," *Proc. of the 25th USENIX Security Symposium*, pp. 911–927, 2016.

[22] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," *DSAA 2016*, pp. 130–139, 2016.

[23] S. Abbott-Mccune and L. A. Shay, "Intrusion prevention system of automotive network CAN bus," *ICCST 2017*, 2017.

[24] S. Otsuka and T. Ishigooka, "CAN Security : Cost-Effective Intrusion Detection for Real-Time Control Systems Overview of In-Vehicle Networks," *SAE Technical Paper*, 2014.

[25] P. Waszecki, P. Mundhenk, S. Steinhorst, M. Lukasiewycz, R. Karri, and S. Chakraborty, "Automotive E/E Architecture Security via Distributed In-Vehicle Traffic Monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[26] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *IV 2011, IEEE*, 2011, pp. 1110–1115.

[27] D. Stabili, M. Marchetti, and M. Colajanni, "Detecting attacks to internal vehicle networks through Hamming distance," *AEIT 2017, IEEE*, 2017.

[28] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS : A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame," *PST 2017*, 2017.

[29] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," *IV 2017, IEEE*, pp. 1577–1583, 2017.

[30] M. J. Kang and J. W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," *VTC 2016*, 2016.

[31] B. Groza and P. S. Murvay, "Efficient Intrusion Detection with Bloom Filtering in Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, 2019.

[32] Q. Wang, Z. Lu, and G. Qu, "An Entropy Analysis based Intrusion Detection System for Controller Area Network in Vehicles," 2018.

[33] W. Wu, Y. Huang, R. Kurachi, G. Zeng, G. Xie, R. Li, and K. Li, "Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks," *IEEE Access*, vol. 6, pp. 45 233–45 245, 2018.

[34] D. Tian, Y. Li, Y. Wang, X. Duan, C. Wang, W. Wang, R. Hui, and P. Guo, "An intrusion detection system based on machine learning for CAN-bus," in *INISCOM 2018*. Springer, 2018, pp. 285–294.

[35] I. Studnia, Y. Laarouchi, M. Kaâniche, V. Nicomette, and E. Alata, "A language-based intrusion detection approach for automotive embedded networks," *International Journal of Embedded Systems*, vol. 10, 2018.

[36] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, "Car hacking identification through fuzzy logic algorithms," *FUZZ-IEEE 2017*, 2017.

[37] S. N. Narayanan, S. Mittal, and A. Joshi, "OBD SecureAlert : An Anomaly Detection System for Vehicles," *SMARTCOMP 2016, IEEE*, 2016.

[38] C. Ling and D. Feng, "An Algorithm for Detection of Malicious Messages on CAN Buses," *CITCS 2012*, 2012.

[39] G. Dupont, J. den Hartog, S. Etalle, and A. Lekidis, "Network intrusion detection systems for in-vehicle networks - technical report," https://security1.win.tue.nl/~gdupont/.

[40] S. Woo, H. J. Jo, and D. H. Lee, "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[41] D. Fauri, M. Kapsalakis, D. Ricardo, J. D. Hartog, and S. Etalle, "Leveraging semantics for actionable intrusion detection in building automation systems," in *CRITIS*, 2018, pp. 113–125.

[42] D. Fauri, D. R. dos Santos, E. Costante, J. den Hartog, S. Etalle, and S. Tonetta, "From system specification to anomaly detection (and back)," in *Proc. of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy, Dallas, TX, USA*, 2017, pp. 13–24.

[43] S. Etalle, "From Intrusion Detection to Software Design," *ESORICS 2017*, pp. 1–10, 2017.

[44] P. Uppuluri and R. Sekar, "Experiences with specification-based intrusion detection," *Recent advances in intrusion detection (RAID) '00*, pp. 1–18, 2000.