A benchmark framework for CAN IDS

Dario Stabili[1], Francesco Pollicino[1] and Alessio Rota[1]

[1]University of Modena and Reggio Emilia, Department of Engineering "Enzo Ferrari", 41125 Modena, Italy

Abstract

This paper presents a benchmark framework for Controller Area Network (CAN) Intrusion Detection Systems (IDS). This framework ships with a dataset and currently supports 4 detection algorithms designed specifically for the CAN bus. The dataset is composed by a set of traces gathered from a licensed, unaltered passenger vehicle used for the training of the detection models and by a set of traces containing simulation of the most common attack scenarios affecting modern vehicle. The detection performance of the 4 supported detection algorithms allows to compare their behavior against different attack scenarios, highlighting the best detection algorithms against the different attack scenarios.

1. Introduction

Among the current trends of the automotive industry, we can witness a steady increase in the adoption of drive-by-wire technologies, Advanced Driving Assistance Systems (ADAS) and Internet connectivity, resulting in a proliferation of Electronic Control Units (ECUs) connected to heterogeneous sensors and actuators that monitor and control many features of the vehicle and its surroundings. These features are designed to increase safety, however software-controlled actuators introduce security vulnerabilities [1] that have already been documented in public white papers and technical reports [2, 3]. These works demonstrate that it is possible to obtain remote control of the vehicle dynamic, allowing attackers to interfere with braking, steering, and acceleration activities of the driver. To prevent unauthorized access to the in-vehicle networks, cyber security researchers already proposed Intrusion Detection Systems (IDS) designed for the Controller Area Network (CAN) [4], which is the most deployed internal network communication protocol within modern vehicles.

The current state-of-the-art already includes many research efforts related to intrusion detection for in-vehicle networks based on the CAN bus [5]. Some of these intrusion detection systems are designed to analyze the low-level characteristics of the ECUs, basing their detection methods on the analysis of the clock skew of the microcontroller [6] or by fingerprinting the voltage differentials of CAN transceivers [7, 8]. These solutions are able to detect any inconsistency by comparing the low-level characteristic during transmission

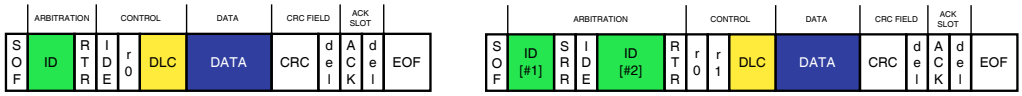
ITASEC21: Italian Conference on Cyber Security, April 07–09, 2021, Online
dario.stabili@unimore.it (D. Stabili); francesco.pollicino@unimore.it (F. Pollicino); alessio.rota@unimore.it (A. Rota)
https://weblab.ing.unimore.it/people/stabili (D. Stabili); https://weblab.ing.unimore.it/people/pollicino (F. Pollicino); https://weblab.ing.unimore.it/people/rota (A. Rota)
0000-0001-6850-334X (D. Stabili); 0000-0002-2421-1852 (F. Pollicino)

against the detection model. However, to implement these detectors it is necessary to use dedicated hardware platforms and it is not possible to use them on the whole CAN transmission. Other detection methodologies specifically designed for the whole CAN bus are based on the analysis of the inter-arrival times of the messages [9, 10, 11], on aggregated statistics of the bus communication [12, 13, 14], or on the analysis of the fields of CAN messages [15, 16, 17, 18]. These detection algorithm are designed to detect anomalies by inspecting the flow of the CAN messages, hence it is possible to deploy them for the identification of attacks targeting the whole CAN system. This paper presents a benchmark framework for Controller Area Network (CAN) Intrusion Detection Systems (IDS). The main contributions of this paper are (I) the proposal of a detailed threat model, allowing the test and the comparison of anomaly detection algorithms and (II) the analysis and the comparison of 4 detection systems for the CAN bus against a set of anomalies representing the considered threat model. The rest of the paper is organized as follows. Section 2 introduces the background knowledge required for the understanding of this paper, while Section 3 presents the benchmark framework. In Section 4 the dataset used for the evaluation of multiple intrusion detection systems is described, and Section 5 presents the detection performance of the selected algorithms. Finally, conclusions and future works are outlined in Section 6.

2. A primer on CAN

The Controller Area Network is a vehicle bus standard designed to allow the nodes of the network to exchange data without requiring a host computer [4]. CAN is one of the most deployed networking protocol for internal vehicular communications due to its high resilience to electromagnetic interference and its cheap implementation. The ECUs on the same CAN segment can communicate using CAN data frames. The CAN data frame is composed by 3 main fields: the identifier (ID), the data length code (DLC) and the payload (data). The ID is used to distinguish among different types of CAN data frame. Data frames characterized by a given ID are produced by only one ECU, while receiver ECUs use the value of the ID to select data frames that are relevant for their functioning. The ID is also used for arbitration of the CAN messages, where lower values of this field denote messages with higher priority. The size of the ID field depends on the type of CAN message, as show in Figure 1. Figure 1a depicts a CAN data frame in the standard format, which have IDs with a size of 11 bit, while the extended format depicted in Figure 1b defines the ID field with a size of 29 bits. The extra 18 bits of the extended format are encoded separately from the 11 bits of the standard format for backward compatibility. The DLC field encodes the number of bytes composing the data field. The data field encapsulates the information that the sender ECU transmits to other ECUs on the network. The data field has a variable size (from 1 to 8 bytes) and usually packs several different signals. The CAN standard leaves complete freedom to the car manufacturers about the structure, number, encoding and semantic of these signals. Hence, without having access to the formal specifications of the CAN messages for a particular vehicle model, the signals encoded in the data field can only be interpreted as

an opaque binary blob.



(a) Base format (b) Extended format

Figure 1: Data frame types comparison

Data transmission on the CAN bus uses a loss-less bit-wise arbitration method for contention resolution. The CAN specification defines each bit sent on the network as either "dominant" (logical value 0, actively driven to voltage level by the transmitter) or "recessive" (logical value 1, passively driven to ground level by the transmitter). The idle state of the network is represented by the recessive value. During data transmission, if one node sends a dominant value on the network and another node sends a recessive value, the node sending the dominant value of the network will win arbitration and any collision on the network is avoided, thus allowing to send high-priority messages on the network without any delay due to transmission errors. To avoid collisions between nodes, each transmitting node reads the logical value of the bus after writing each bit of the message. If any transmitting node reads a bus value different from the one it has written on the bus it stops transmission and attempts re-transmission of the message after the current transmission is concluded by the sender.

3. The benchmark framework

In this section the threat model considered for the definition of the attacks analyzed by the framework is presented in Section 3.1, while Section 3.2 describes the anomaly detectors currently supported by the framework.

3.1. Threat model

The threat model considered in this paper is based on different attack scenarios already published by security researchers on technical reports and white papers [19, 2, 20, 21, 22, 23].

3.1.1. Replay Attack

The replay attack is used for injecting messages on the CAN bus to subvert its normal behavior, by exploiting modern drive-by-wires capabilities such as brake-by-wire or steer-by-wire. The replay attack sequence is usually selected after an initial phase of reverse engineering of the values encoded in the payload, allowing the attacker to inject messages which content is expected to command part of the vehicle dynamics. For the purposes of this paper, we consider two different typologies of replay attack based on the length of the injected message sequence:

1. Single ID Replay: A Single Message ID is injected on the normal flow of the CAN bus with a particular frequency;
2. Sequence Replay: A sequence of messages is injected on the normal flow of the CAN bus with a particular frequency. The length of the injected sequence is a parameter that will be inspected in the detection process.

3.1.2. Fuzzing Attack

The fuzz-testing (also known as fuzzing) attack describes the process of automatically generating and sending malformed input to the software under test, while monitoring its behavior [24]. Fuzzing attacks to automotive internal networks can be described as the injection of CAN messages with malformed values in the fields composing the data frame. For the purposes of this paper, two different fuzzing attacks are considered:

1. ID Fuzzing: CAN data frames with the ID and data field randomly generated. The ID field is chosen to be different from the values of IDs observed in the dataset;
2. Payload Fuzzing: CAN data frames with valid ID field and randomly generated data field.

3.1.3. Disruption Attack

The disruption attack comprises all the other attacks focusing on the interruption of the normal operation of either the network or its components [25]. We remark that in the automotive scenario, a disruption attack can either target the network or its microcontrollers. For the purposes of this paper, we consider the following disruption attacks:

1. Denial-of-Service: CAN data frames with the highest priority are injected with high frequency, thus preventing any other node to start transmission of other messages;
2. ECU inhibition: messages generated by a target ECU are removed by sending the ECU in a bus-off state, thus preventing the target ECU to participate in the normal CAN communication.

3.2. Anomaly Detectors

The anomaly detectors supported by the framework are chosen as representative examples of different detection algorithms designed for the analysis of the different fields of CAN messages.

3.2.1. Message sequence algorithm

The message sequence algorithm [15] is based on the analysis of the message IDs. This algorithm analyzes the possible transitions between pairs of IDs and later uses the list of all the possible transitions for its detection purposes. There are no configuration parameter in this algorithm, hence it is implemented as-is.

3.2.2. Bus entropy algorithm

The bus entropy algorithm [13] is based on the analysis of the information entropy of the bus for detection purposes. This algorithm uses the entropy evaluated over a time window t to define the normal entropy range $[\mu_e - k\sigma_e, \mu_e + k\sigma_e,]$, where μ_e is the mean entropy of the time windows, σ_e is its standard deviation, and k is a tuning parameter. In our experimental evaluation we used a time window $t = 0.01$ seconds, resulting in $\mu_e = 2.7225$, $\sigma_e = 0.3176$, and with a value of $k = 3$ to achieve 0 false positives in the validation process.

3.2.3. Hamming distance algorithm

The Hamming distance algorithm [18] is based on the analysis of the Hamming distance of consecutive payloads of messages with the same ID for detection purposes. This algorithm evaluates the Hamming distance between consecutive payloads of message with the same ID for the definition of the normal model, which is composed by the pair $[min, max]$ values for each message ID, representing the minimum and maximum Hamming distance found in the between consecutive payloads of the same message ID. This normal model is later used for detection purposes, and if the Hamming distance evaluated between two consecutive payload of messages with the same ID is outside the $[min, max]$ range, than an anomaly is raised.

3.2.4. Missing message algorithm

The missing message algorithm [11] is designed to detect missing messages from the normal CAN communication. This algorithm evaluates the cycle time ct of each message ID and later uses it for the definition of the valid waiting time ($ct^{ID} * k^{ID}$) that normally occurs between two consecutive messages with the same ID, where k^{ID} is a configuration parameter. In the validation process on the clean dataset, we achieved zero false positives with a maximum value of $k^{ID} = 2$.

4. Dataset Description

This Section presents the dataset used for the evaluation of the detection algorithms. The clean dataset used for the training of the detection algorithms is described in Section 4.1, while Section 4.2 describes the infected dataset, that contains the attacks described in the threat model (see Section 3.1).

4.1. Clean dataset description

The dataset used for the definition of the detection models of the selected algorithms is collected from the high-speed CAN buses of an unmodified, licensed 2016 Volvo V40 by physically connecting a laptop to the OBD-II port with a PCAN-USB adapter by Peak System and a D-Sub to OBD-II cable. According to international standards, the high-speed CAN bus exposed on the OBD-II port is the powertrain segment, which is composed by ECUs that control different subsystems of the vehicle dynamic, such as the

cruise control system, the anti-braking system, the electronic stability control, and many optional Advanced Driver Assistance Systems (ADAS). The CAN recording process is configured to save metadata information about the CAN traffic (such as the timestamp and the type of the message) and the fields composing the messages (CAN ID, the DLC value, and the bytes composing the data field). These data are gathered during several driving sessions performed on different road types (urban, suburban and highways), traffic conditions, weather conditions and on different geographical areas (plain, hill and mountain). The whole dataset includes an aggregated amount of more than 10 hours of driving over 7 different CAN traffic traces, including more than 8 million messages belonging to 50 unique message IDs. The clean dataset is publicly available at [26].

4.2. Infected dataset description

The infected dataset is created by simulating the attacks described in the considered threat model (see Section 3.1) on the clean dataset. To avoid any possible bias toward unrealistic detection results due to the different frequencies of messages composing the clean dataset, IDs with different probability distribution are used for the simulation of attacks on the clean dataset. For the simulation of the different attacks, we selected 4 different message IDs to represent the most frequent message (CAN ID `0x10`, cycle time of 10 milliseconds, labeled as `top`), a medium frequency message (CAN ID `0x145`, cycle time of 20 milliseconds, labeled as `mid`), a low frequency message (ID `0x210`, cycle time of 25 milliseconds, labeled as `low`), and a non-periodic message (ID `0x1`, undefined cycle time, labeled as `not`).

The infected dataset is composed by the following attack scenarios:

- Replay attack - single ID: Set of traces in which a single valid ID is injected. This set is composed by 4 different traces, each one corresponding to the injection of one of the 4 selected IDs.
- Replay attack - valid sequence: Set of traces in which a sequence of IDs observed in the traces is injected. Different traces are generated by injecting sequences with different length, ranging from 2 to 10.
- Replay attack - invalid sequence: Set of traces in which a random generated sequence of valid IDs is injected. Different traces are generated by injecting sequences with different length, ranging from 2 to 10.
- Fuzzing attack - random ID: Set of traces in which a single invalid ID is injected.
- Fuzzing attack - random payload: Set of traces where a single valid ID is injected with randomly generated payloads. The valid ID used for the injection of a random payload is selected using the same criteria used for the single ID replay attack.
- Disruption - Bus DoS: Set of traces where 100 messages with the same ID are injected each second.
- Disruption - ECU inhibition: Set of traces where CAN messages with a particular ID are completely removed. This attack scenario represents an attacker that is able to permanently disable a target ECU. This set is composed by 4 different traces, each one corresponding to the removal of one of the 4 selected IDs.

5. Performance evaluation

The detection performance of the algorithms supported by the framework are evaluated by means of $\mathcal{F}-$measure, which is a statistical index representing the accuracy of a test. The $\mathcal{F}-$measure is harmonic mean between the precision (i.e. the number of correctly identified anomalies on the total of detected anomalies, hence including also the false positives) and the recall (i.e. the number of correctly identified anomalies on the total of actual anomalies, hence including the false negatives). Its value ranges from 0 to 1, where 0 denotes the inability of the detector to identify anomalies in the data, while 1 denotes the ability to identify all the anomalies and only the anomalies in the data. All the detection algorithms are implemented using the Python programming language on a server equipped with an Intel® Core™ i7-7700HQ CPU @3.8 GHz and with 16 GB of RAM running Fedora 33 x64.

5.1. Performance against the replay attacks

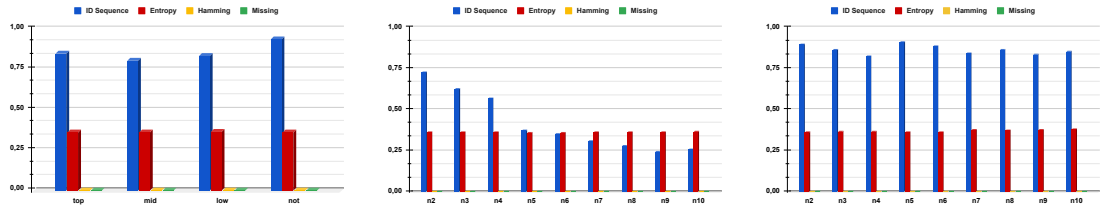
The detection results evaluated with the framework against the replay attacks are depicted in Figure 2, showing the $\mathcal{F}-$measure of the supported algorithms against the single ID replay attack (Figure 2a), valid sequence replay attack (Figure 2b), and invalid sequence replay attack (Figure 2c).

5.1.1. Single ID replay detection results

Figure 2a shows the detection results of the detection algorithm against the single ID replay attack. The x-axis of Figure 2a shows the labels representing the different replayed IDs, while the y-axis shows the $\mathcal{F}-$measure evaluated using the algorithms. The four vertical bars represents the four detection algorithms: message sequence (blue bar), bus entropy (red bar), Hamming distance distance (yellow bar), missing message (green bar). From the analysis of the results presented in Figure 2a it is possible to notice that the message sequence detection algorithm is able to constantly identify the injection of a single message ID, while the bus entropy detection algorithm only detects a limited subset of the injected messages. Moreover, we remark also that these results are independent of the injected message (despite being higher in case of injection of the not message), while the Hamming distance and the missing message algorithms are not able to detect any anomaly.

5.1.2. Valid sequence replay detection results

Figure 2b shows the detection results of the detection algorithm against the valid sequence replay attack. The x-axis of Figure 2b shows the length of the injected sequence, while the y-axis represents the $\mathcal{F}-$measure achieved by the different detection methods against the attack scenarios. The vertical bars represents the results of the detection algorithm, as already described for the previous attack scenario. From the analysis of the results presented in Figure 2b it is possible to notice the same trend already observed in the previous case, despite in this attack scenario the message sequence detection algorithm



(a) Detection performance against the single ID replay attack

(b) Detection performance against the valid sequence replay attack

(c) Detection performance against the invalid sequence replay attack

Figure 2: Detection results of the supported algorithms against the message replay attacks (left to right: single ID replay, valid sequence replay, invalid sequence).

performance decreases against the injection of longer valid sequences. This trend is explained by considering that with the injection of valid sequences of messages, the message sequence detection algorithm is able to detect an anomaly only in the transition from the normal CAN communication to the first message of the sequence or from the last message of the sequence to the normal CAN communication. Hence, by increasing the overall length of the attack, the percentage of anomalies that this detection method is able to detect decreases (since the internal transitions are considered valid).

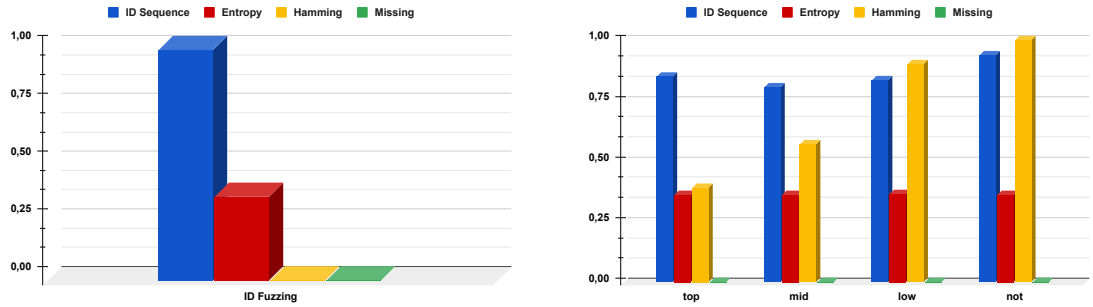
5.1.3. Invalid sequence replay detection results

Figure 2c shows the detection results of the detection algorithm against the invalid sequence replay attack. The results of Figure 2c are shown using the same structure already described for Figure 2b. The results presented in Figure 2c shows that the entropy detection algorithm is still unable to detect anomalies consistently, while the performance of the message sequence algorithm are higher compared to the previous attack scenario and being constantly higher than $\mathscr{F}-$measure = 0.8.

As a final comment of the detection results achieved with the framework against the replay attack scenario, we highlight that the detection algorithm based on the analysis of the sequence of messages is able to achieve higher detection results compared to the others, while the bus entropy anomaly detector struggles to achieve detection results higher than $\mathscr{F}-$measure ≥ 0.5. Moreover, we remark that the Hamming distance and the missing message detection algorithm are not able to detect any anomaly against this attack scenario.

5.2. Performance against the Fuzzing attack

The detection results achieved with the framework against the replay attack are depicted in Figure 3, showing the $\mathscr{F}-$measure of the algorithms against the random ID fuzzing attack (Figure 3a), and the random payload fuzzing attack (Figure 3b).



(a) Detection performance against the random ID fuzzing attack

(b) Detection performance against the random payload fuzzing attack

Figure 3: Detection results of the supported algorithms against the fuzzing attacks (left to right: random ID fuzzing, random payload fuzzing).

5.2.1. Random ID fuzzing detection results

The results achieved by the detection algorithms against the random ID fuzzing attack are shown in Figure 3a. The results presented in Figure 3a show that the detection performance in this attack scenario are similar to the previous scenario, with the only difference being that the message sequence detection algorithm is able to achieve a fixed $\mathscr{F}-$measure $= 1$ in all the attack simulation. This results is easily explainable by considering that the attack is simulated using a message ID never observed in the clean dataset, while the detection model of the message sequence algorithm is composed by all the message IDs found in the clean dataset. With this analysis it is clear that the message ID sequence is able to detect all the instances of this attack scenario, with absolute precision and recall.

5.2.2. Random payload fuzzing detection results

The results achieved by the detection algorithms against the random payload fuzzing attack are depicted in Figure 3b, showing the $\mathscr{F}-$measures (y-axis) achieved with the algorithms (vertical bars) in case of fuzzing of a payload with different IDs (x-axis). The results presented in Figure 3b are equal to the ones presented in case of the single ID replay attack (Figure 2a) for the message sequence, bus entropy, and missing message detection algorithms since the attack is simulated using the same message IDs. However, it is necessary to remark that the Hamming distance detection algorithm is able to detect anomalies, and that its detection performance increase by decreasing the frequency of the message.

As a final comment on the detection performance of the algorithms against the fuzzing attacks we highlight that the message sequence detection algorithm is still reaching high $\mathscr{F}-$measures in both attack scenario, achieving the maximum $\mathscr{F}-$measure against the random ID fuzzing attack, while the bus entropy algorithm is not able to detect anomalies consistently. The missing message detection algorithm is still not able to detect any

anomaly, but the Hamming distance detection algorithm is able to detect anomalies consistently in half of the simulated scenario against the random payload fuzzing attack.

5.3. Performance against the disruption attacks

Figure 4 shows the detection performance of the algorithms against the two considered disruption attacks: Bus DoS (Figure 4a) and ECU inhibition (Figure 4b). The results against the bus DoS attack in Figure 4a shows the $\mathscr{F}-$measure (y-axis) of the detection algorithms (vertical bars), as already presented in previous scenarios, while the results against the ECU inhibition attack depicted in Figure 4b shows, for each message used for the attack simulation (x-axis), the percentage of the detected anomalies (y-axis) of the four different detection algorithms (vertical bars).

5.3.1. Denial-of-Service detection results

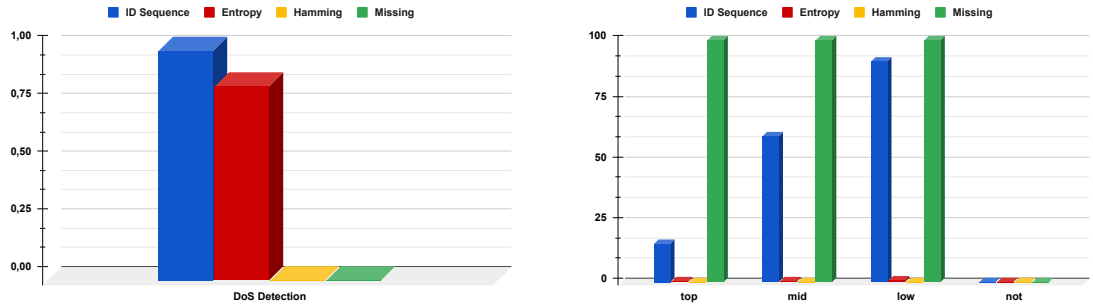
From the analysis of the results depicted in Figure 4a we highlight that the Hamming payload distance and the missing message algorithms are not able to detect any anomaly, while the message sequence and the bus entropy algorithms are both able to achieve consistent detection results. Moreover, we remark that the message sequence detection algorithm is able to achieve a perfect $\mathscr{F}-$measure $= 1$ also in this attack scenario, since the detection model does not have any transition between the same message ID and that the injection of a 100 consecutive messages with the same ID. On the other hand, the bus entropy algorithm is able to achieve $\mathscr{F}-$measure $= 0.8$ consistently, demonstrating the ability of this detection method to detect attacks composed by the injection of multiple messages within the same time window.

5.3.2. ECU inhibition detection results

In case of the ECU inhibition attack (Figure 4b we highlight that the best algorithm for the detection of the ECU inhibition attack is the missing message algorithm, that is able to detect almost 100% of the missing messages. The message sequence algorithm is also able to detect anomalies, despite its detection performance are not consistent throughout the different tests. The bus entropy and the Hamming distance payload detection algorithms however are not able to detect any anomaly. As a final remark, we highlight that in case of the removal of the `not` message, the tested algorithms are not able to detect any anomaly. This is easily explained by considering that the `not` message is a non-cyclic message, hence it is not possible to define the normal behavior of this message. This implies that it is not possible to identify the cycle time of the message, hence it is not possible to apply the missing message algorithm in this particular scenario.

6. Conclusions

This paper presents a framework for the comparison of anomaly detection algorithms designed for CAN communications. This framework is designed to compare of the



(a) Detection performance against the Denial-of-Service attack

(b) Detection performance against the ECU inhibition attack

Figure 4: Detection results of the supported algorithms against the fuzzing attacks (left to right: random ID fuzzing, random payload fuzzing).

detection performance of the algorithms against the same attack scenarios, representing known attacks to the CAN bus. The benchmark framework currently supports 4 detection algorithms designed to analyze different features of the CAN communications, and the experimental evaluation demonstrated that the message sequence algorithm is able to detect anomalies consistently in all the attack scenarios, while others methods have a more limited applicability. In particular, the Hamming payload distance algorithm and the missing message algorithm are proven effective in the detection of the only random payload fuzzing attack and the disruption ECU inhibition attack, respectively; while the bus entropy algorithm is only able to detect attacks carried out with the injection of a high-volume of messages. We are currently expanding the framework to include other detection algorithms and novel attack scenarios.

Acknowledgments

This research has received funding from COSCA (COnceptualising Secure CArs), a project supported by the European Unions Horizon 2020 research and innovation programme under the NGI_TRUST grant agreement no. 825618.

References

[1] P. Kleberger, T. Olovsson, E. Jonsson, Security aspects of the in-vehicle network in the connected car (2011).

[2] C. Miller, C. Valasek, Remote Exploitation of an Unaltered Passenger Vehicle, http://illmatics.com/RemoteCarHacking.pdf, 2015.

[3] Keen Security Lab of Tencent, New car hacking research: 2017, remote attack tesla motors again, 2017. URL: http://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/.

[4] R. B. GmbH, Can specification version 2.0, 1991. URL: http://esd.cs.ucr.edu/webres/can20.pdf.

[5] G. Dupont, J. den Hartog, S. Etalle, A. Lekidis, A survey of network intrusion detection systems for controller area network (2019).

[6] K.-T. Cho, K. G. Shin, Fingerprinting electronic control units for vehicle intrusion detection (2016) 911–927. URL: http://dl.acm.org/citation.cfm?id=3241094.3241165.

[7] K. Cho, K. G. Shin, Viden: Attacker identification on in-vehicle networks, ACM Conference on Computer and Communications Security abs/1708.08414 (2017). URL: http://arxiv.org/abs/1708.08414. arXiv:1708.08414.

[8] M. Kneib, C. Huth, Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks (2018) 787–800. URL: http://doi.acm.org/10.1145/3243734.3243751. doi:10.1145/3243734.3243751.

[9] M. Gmiden, M. H. Gmiden, H. Trabelsi, An intrusion detection method for securing in-vehicle CAN bus, 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA) (2016) 176–180. doi:10.1109/STA.2016.7952095.

[10] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, S. J. Prowell, Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks (2017).

[11] D. Stabili, M. Marchetti, Detection of missing can messages through inter-arrival time analysis (2019).

[12] M. Müter, N. Asaj, Entropy-based anomaly detection for in-vehicle networks, 2011 IEEE Intelligent Vehicles Symposium (IV) (2011) 1110–1115. doi:10.1109/IVS.2011.5940552.

[13] M. Marchetti, D. Stabili, A. Guido, M. Colajanni, Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms (2016).

[14] N. Nowdehi, W. Aoudi, M. Almgren, T. Olovsson, Casad: Can-aware stealthy-attack detection for in-vehicle networks, 2019.

[15] M. Marchetti, D. Stabili, Anomaly detection of can bus messages through analysis of id sequences (2017).

[16] V. Chockalingam, I. Larson, D. Lin, S. Nofzinger, Detecting attacks on the can protocol with machine learning, Annual EECS 588 (2016).

[17] M. J. Kang, J. W. Kang, A novel intrusion detection method using deep neural network for in-vehicle network security (2016) 1–5. doi:10.1109/VTCSpring.2016.7504089.

[18] D. Stabili, M. Marchetti, M. Colajanni, Detecting attacks to internal vehicle networks through hamming distance (2017).

[19] C. Valasek, C. Miller, Car Hacking: For Poories, 2014. URL: http://illmatics.com/car_hacking_poories.pdf.

[20] C. Miller, C. Valasek, CAN Message Injection – OG Dynamite Edition, http://illmatics.com/canmessageinjection.pdf, 2016.

[21] Keen Security Lab. of Tencent, Car Hacking Research: Remote Attack Tesla Motors, Tech. Rep., 2016.

[22] K.-T. Cho, K. G. Shin, Error handling of in-vehicle networks makes them vulnerable (2016) 1044–1055. URL: http://doi.acm.org/10.1145/2976749.2978302. doi:10.1145/

2976749.2978302.
[23] A. Palanca, E. Evenchick, F. Maggi, S. Zanero, A stealth, selective, link-layer denial-of-service attack against automotive networks (2017).
[24] M. Sutton, A. Greene, P. Amini, Fuzzing: Brute Force Vulnerability Discovery, Addison-Wesley Professional, 2007.
[25] CISA - Cybersecurity and Infrastructure Security Agency, Understanding denial-of-service attacks, https://us-cert.cisa.gov/ncas/tips/ST04-015, 2019.
[26] D. Stabili, M. Marchetti, VTC2019 Dataset, 2021. URL: https://weblab.ing.unimore.it/people/stabili/resources/.