2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Anomaly detection of CAN bus messages through analysis of ID sequences

Mirco Marchetti[1] and Dario Stabili[2]

Abstract— **This paper proposes a novel intrusion detection algorithm that aims to identify malicious CAN messages injected by attackers in the CAN bus of modern vehicles. The proposed algorithm identifies anomalies in the sequence of messages that flow in the CAN bus and is characterized by small memory and computational footprints, that make it applicable to current ECUs. Its detection performance are demonstrated through experiments carried out on real CAN traffic gathered from an unmodified licensed vehicle.**

I. INTRODUCTION

As the automotive industry pushes toward the adoption of more advanced infotainment systems and self-driving capabilities, modern vehicles become mobile networks of computing devices, possibly connected to the Internet. This paradigm shift paves the way for groundbreaking innovations, but also opens new avenues for cyber-attackers that are now able to approach the automotive domain and exploit software vulnerabilities in cars, ultimately leading to security and safety hazards for drivers and nearby people.

Several proof-of-concepts of attacks to modern unmodified and licensed vehicles have already been demonstrated by security researchers [1], [2], [3], [4]. In all cases, attackers were able to take limited control over safety relevant functions, such as throttle, brake and steering. These results spawned novel efforts from both the industry and research institutions towards novel approaches for securing modern vehicles against cyber-attacks. In particular, since all the known attacks that pose relevant safety risks involve the injection of malicious messages within the CAN bus of the attacked vehicles, an interesting and still wide open research field is the identification of methods and algorithms for analyzing messages transmitted over the CAN bus, with the aim of identifying possible evidences of illicit activities.

Within this field of research, this paper proposes a novel anomaly detection algorithm for the CAN bus of modern vehicles. To the best of our knowledge, this is the first algorithm based on the analysis of the sequences of messages that flow on the CAN bus. This feature can be extracted even without knowing the message specifications, hence our algorithm does not require knowledge of syntax and semantic of the CAN messages, that are kept highly confidential by car makers and by their suppliers. Moreover, computational requirements of the proposed algorithm are low enough to be compatible with the very limited hardware constraints of microcontrollers used to develop the ECUs (Electronic Control Units) embedded in modern vehicles.

Through an experimental evaluation carried out over real CAN traffic traces gathered from a licensed vehicle we demonstrate that the proposed algorithm is able to reach very high detection rates for several different attack classes involving the injection of very few CAN messages. We also highlight that our detector did not generate any false positive during our experimental evaluation.

The rest of the paper is organized as follows. Section II describes the proposed approach for intrusion detection. Section III discusses different attack scenarios that are used to evaluate the effectiveness of the proposed approach. Experimental results are then presented in Section IV. Section V compares the proposed algorithm with respect to related works, and outlines its main contributions and novelties. Finally, Section VI concludes the paper and proposes future works.

II. DETECTING ANOMALIES IN SEQUENCES OF CAN IDS

This section presents the intrusion detection algorithm proposed in this paper. To better describe its internals, we first present the structure of a generic CAN Data Frame in Section II-A. An overview of the proposed algorithm is presented in Section II-B, while details about training and detection phases are provided in Sections II-C and II-D, respectively.

A. Structure of CAN data frames

There are two different formats of CAN Data Frame: the base frame format and the extended frame format. The main difference between the two formats is the number of bits reserved for the message identifier, with an 11-bit ID in the base frame format and a maximum of 29-bit ID in the extended frame format. Apart from this distinction, the structure of any CAN message can be modelled as a tuple composed by ID, payload and checksum, as shown in Figure 1.

The Message ID is a unique identifier of the message that is used by every ECU of the in-vehicle network in order to identify if a message has to be processed or can be ignored. Every identifier is produced by only one ECU and can be of interest for one or more ECUs. The message payload contains different values produced by different sensors managed by a specific ECU and are encoded according to the specifications contained in the DBC file, a database-like file in a proprietary format that contains all the specifications

[1]Mirco Marchetti is with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125 Modena, Italy mirco.marchetti@unimore.it

[2]Dario Stabili is with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125 Modena, Italy dario.stabili@unimore.it

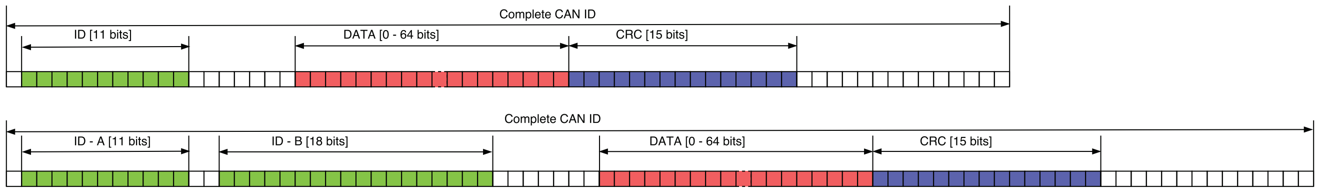

Fig. 1. Base vs Extended format

of every ECU of a specific vehicle configuration. This file is extremely precious for car manufacturers, and it is kept strictly confidential. In particular, the DBC file lists all the IDs that should be generated by all ECUs, and prescribes the message format associated to each ID. Format specification includes the number of fields within the message body, their length and boundaries, and syntactical rules for their correct interpretation. Moreover, the DBC specifies whether an ID is event-driven or periodic, and its cycle-time for periodic messages. The message checksum is a field containing the result of a CRC function computed over the Data Frame. This field is used by the receiving ECUs in order to identify possible message modification due to transmission errors. It should be noted that this field can only provide some form of protection against random modification of a CAN data frame. Since it is not built upon strong crypto and authentication primitives, attackers willing to inject data over the CAN bus or introduce arbitrary modifications in legit CAN data frames can easily compute a new valid CRC.

B. Overview of the proposed algorithm

The main idea behind the proposed algorithm is to build a model of the normal behavior of a CAN network based on a particular feature: recurring patterns within the sequence of message IDs observed in the CAN Bus. While other proposals evaluate different features (as described in Section V) our design choice derives from two main real-world constraints. The former is the lack of a DBC file containing formal message specifications and correct information about the message payload. The latter is the need for intrusion detection algorithms that can analyze all CAN messages in real time and execute over micro controllers characterized by very low computational and memory capabilities. Hence all detection algorithms based on complex statistical modeling are not applicable to the current automotive landscape.

An overview of the proposed algorithm is shown in Figure 2.

From the analyses of different traces recorded from our test car, we identified some recurring patterns on the ID sequences. Every IDs of our test model is followed only by a subset of all the available IDs, thus limiting the admissible transitions between all IDs. From this result, we developed an algorithm that is able to detect anomalies in the ID sequences. In order to create the model and the detection algorithm, we divided the work flow in two different phases. In the Training Phase (described in Section II-C) we used CAN bus logs to create a data structure that contains all the possible transitions between consecutive IDs observed

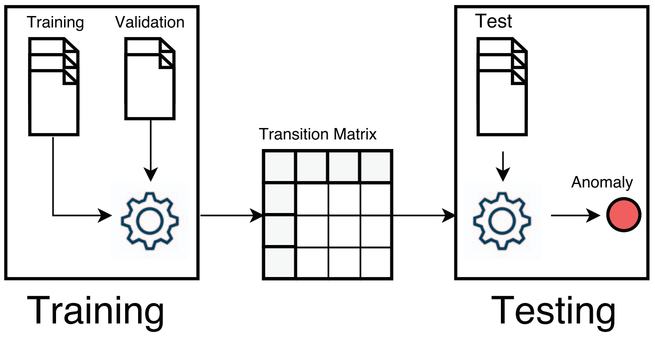

Fig. 2. Algorithm Overview

in legit CAN traffic traces that did not contain any attack. The end of this phase outputs the *transition matrix*. The Detection Phase (described in Section II-D) takes as inputs the transition matrix and the traffic traces to analyze in order to identify possible attacks.

C. Training Phase

In the training phase we analyze traces of real CAN bus traffic gathered from an unmodified licensed vehicle in normal conditions, without any attack. The aim of this phase is to model the normal behavior of the CAN bus in the form of its transition matrix, a data structure that identifies all the legit transitions between the message IDs of two consecutive CAN messages. This model will then be used as a reference to identify anomalies in the CAN traffic.

The transition matrix is a square matrix of order n, where n is the number of unique IDs available in the trace. At the beginning of the training phase, all values of the transition matrix are initialized to *false*.

The first step for the creation of the transition matrix is to inspect the sequence of the IDs on a legit trace and marking as *true* all the observed transitions between IDs. As an example, if a message having ID equal to i is followed by a message with ID j, than a value of *true* is stored in row i and column j of the transition matrix. The algorithm for the population of the transition matrix is represented in Figure 3.

The resulting data structure is a matrix composed by either *true* or *false* values. Such data structure allows the final algorithm to have very low latency in accessing the transition value. After the creation of the transition matrix, we used other legit traffic traces in order to validate our model. Starting from all the traces at our disposal (collected over 10 hours of driving in different traffic conditions, for

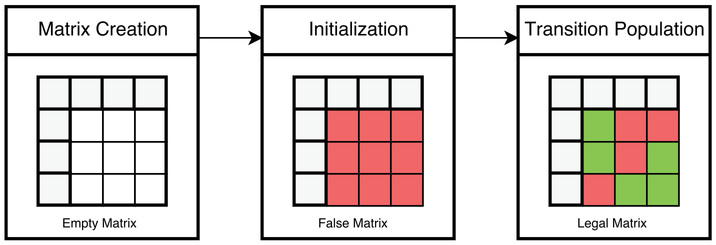

Fig. 3. Transition Matrix Population

a total of about 120 millions CAN messages) we isolated approximately 50 millions messages (corresponding to about 4 hours of driving) and used 20% of them for the training phase, while keeping the other 80% for validation purposes.

The validation process takes as input the *Transition Matrix*, created in the previous step, and the sequences of IDs extracted from the set of messages used for validation purposes. The steps of the algorithm are explained below:

1) Two consecutive IDs are read from the validation trace. If one or both IDs are missing in the transition matrix (meaning that these IDs have never been observed before in the training traces), validation fails. If both IDs are already present in the transition matrix, the validation process continues to the next step.
2) The transition condition from the first ID to the second is checked. If the transition value is *true* in the transition matrix, the algorithm continues to the next step. On the other hand, if the transition value is *false*, the validation process raises a *false positive* error, updates the transition matrix by setting the transition value to *true* and continues to the next step.
3) The algorithm extracts the next ID from the sequence and repeats the previous steps.

The algorithm for the validation process follows the flow chart represented in Figure 4.

The validation process is extremely important for evaluating the effectiveness of the proposed detection algorithm. In particular, it is important to minimize the number of false positives, since even a very low false positive rate may lead to the generation on unacceptable false alarms during normal driving conditions.

At the end of the validation process, the algorithm raised 0 false positives and the resulting transition matrix was equal to the one obtained at the end of the first phase of the training process, meaning that the transition patterns between IDs were strongly consistent between different traces collected in different driving conditions and that all the different transitions between IDs have already been observed in the message sequence used for training.

D. Detection phase

The last phase of the detection algorithm uses the previously created and validated transition matrix to detect anomalies on other traffic traces. The transition matrix is used to test the possible transitions between two different IDs: if the transition is marked as $false$ in the matrix,

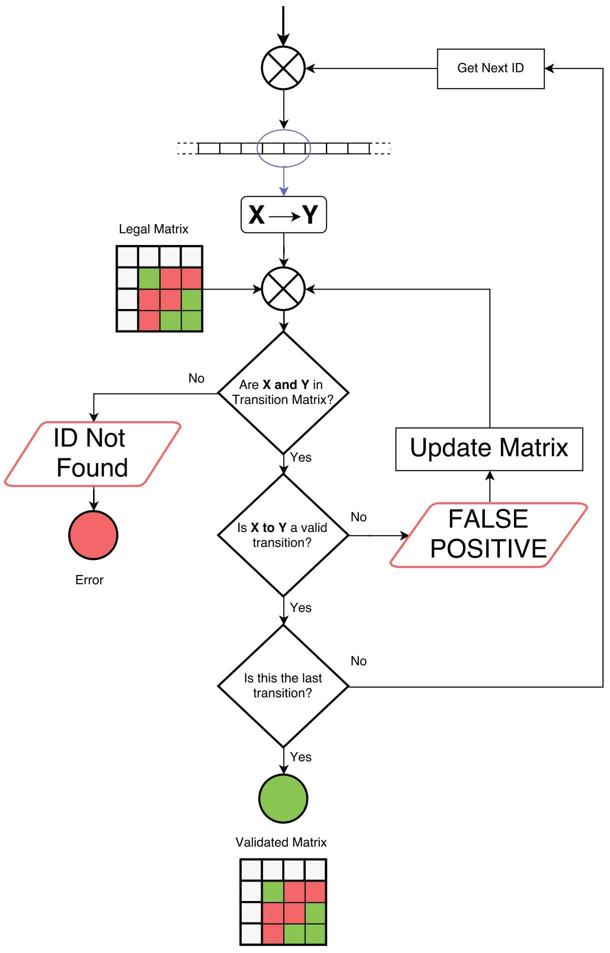

Fig. 4. Model Validation

such transition is considered as an anomaly, otherwise the transition is legit.

A thorough experimental evaluation that assesses the effectiveness of the proposed algorithm in identifying different classes of attacks (described in Section III) is proposed in Section IV).

E. Memory and computational requirements

We now discuss the memory and computational requirements of the proposed detection algorithm.

Memory Requirements: The memory requirement for the algorithm are strongly dependent on the vehicle model and configuration, being tied to the number of different IDs that flow on the analyzed CAN bus. In order to minimize the memory requirement, we introduce a data structure that ties each CAN ID to an index that ranges from 1 to the number of IDs, and that is used to identify the row and the column of the transition matrix that correspond to a given ID. This data structure is the *ID-position* array.

The detection algorithm then requires only n^2 bits of memory for storing the transition matrix, plus $n*11$ to $n*29$ bits for the ID-position array (depending on the length of the IDs), where n represents the number of different IDs. The memory requirements of the transition matrix in our test case (with 45 different IDs of 11 bits each) are 2025 bits (254 bytes) for the transition matrix and 495 bits (62 bytes) for the ID-position array, for a total memory requirement of 316 bytes.

Computational Costs: The detection phase of the proposed algorithm based on simple lookup and comparison operations. The extraction of the ID from all CAN messages is a basic operation that is already performed by all ECUs connected to the CAN bus, hence this activity does not impose any additional computational burden. The lookup of the *true* or *false* value within the transition matrix is an extremely fast operation, thanks to the direct access through indexes retrieved from the ID-position array. The computational cost of this operation is O(1), and does not depend on the number of distinct IDs. The lookup of an index corresponding to a given ID within the ID-position array is the only operation that could increase the computational cost of the proposed solution. Its computational complexity is $O(n)$, where n is the number of unique IDs in the vehicle. This operation can be optimized by ordering the IDs according to the ID frequency: IDs that appear more frequently on the CAN bus can be placed in the first places of the ID-position array, so that they will be encountered earlier in a sequential scan of the array. This optimization allows us to further reduce the average computational cost of the lookup phase.

Centralized vs distributed solution: In the CAN networks there are a particular set of ECUs known as *Gateways* that are placed on the edge of different sub networks and that allow CAN data frames to flow among different branches of the CAN bus. These ECUs have full visibility of all the traffic that flows in different branches of the CAN bus, hence by placing the detection algorithm in one of these ECUs it is possible implement a centralized detection system.

The proposed detection algorithm can also be distributed across the different branches of the CAN bus. This strategy requires the deployment of the detection software in at least one ECU for each branch, but has the benefit of reducing both the computational costs and memory requirements, because only a subset of IDs flow in each branch. In this scenario we need to create different transition matrices and ID-position arrays for each monitored branch.

III. Attack scenarios

To assess the effectiveness of the proposed detection algorithm we simulated attacker activities by injecting forged CAN messages within traffic traces gathered from an unmodified licensed vehicle. This process has already been adopted for the evaluation of several intrusion detection algorithms based on the CAN bus [1], [5], and allows us to simulate the effects over the CAN bus of real attack strategies [2], [6], [3].

In particular, starting from real CAN traffic traces, we create two different datasets. The former contains injections of single CAN messages, as described in Section III-A. The latter includes more complex attacks characterized by the injection of sets of CAN bus messages that mimic real attacks, discussed in Section III-B. The legit CAN traces that served as a basis for message injection were gathered from the OBD-II port of an unmodified 2011 Ford Fiesta during several hours of driving in real traffic conditions. This trace includes a total of 45 different message IDs.

A. Basic Injection

This dataset has been designed to test the ability of the proposed algorithm to detect an attack comprising one or more injections of the same legit CAN message.

The probability distribution of each message ID is a peculiar characteristic of every car, and can even change from model to model based on the optional functions implemented and activated. Figure 5 shows the distribution of message IDs within the legit can traces. the x-axis represents the different message IDs, labeled from 1 to 45. The y-axis represents the number of messages having a given ID within a legit trace of 48 million messages (about 4 hours of driving). From this figure it is clear that some IDs have a much higher probability of appearing on the CAN bus with respect to other IDs.

This aspect is interesting to study how the performance of the proposed anomaly detection algorithm are influenced by the popularity of the injected messages. To carry out a comprehensive evaluation for CAN messages characterized by different probabilities, we select four IDs representing messages at the 20th, 40th, 60th and 80th percentiles in the message distribution shown in Figure 5. We then use these representative IDs to create four different set of traces. Each set comprise ten different malicious traces obtained by injecting simulated attack sequences including from one to ten occurrences of the representative ID. Each attack sequence lasts one second, and injected messages are uniformly distributed within that second. Attacks sequences are injected in random parts of the legit traces, and are always separated by at least one second of attack-free CAN traffic. At the end of this process we obtain 40 different attack traces, including from 1900 to 19000 attacks depending on the number of injected messages.

B. Realistic attacks

For the second dataset we selected three different types of attacks, called *Replay*, *Bad Injection* and *Mixed Injection*.

A *Replay* happens when an attacker injects in the CAN bus a sequence of messages that have previously been read from the same CAN bus. To better simulate a real-world attack scenario, we extract sequences of normal messages from the training traces, and build the malicious traces by injecting these normal sequences within legit traces that have not been used for training. This scenario represents a worst-case for our algorithm, since all transitions between injected messages have already been observed during training and thus cannot be detected through the transition matrix. Hence the only chance of detection is related to the transitions at the beginning and at the end of the injected sequence.

In a *Bad Injection* attack, the attacker injects a sequence of messages that never appeared before over the same CAN bus. To simulate this attack, we leverage the transition matrix

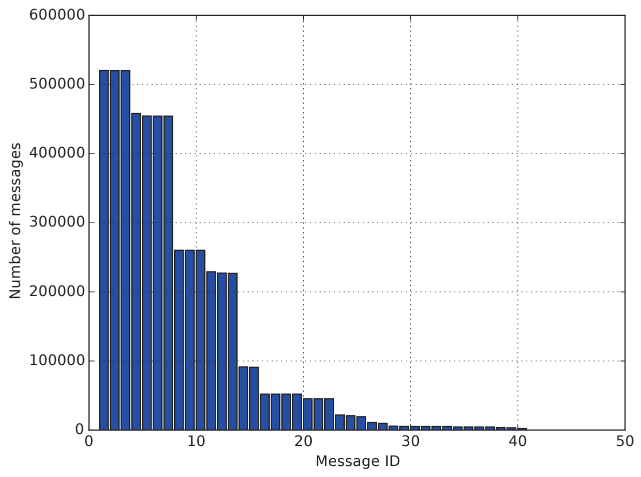

Fig. 5. Message ID distribution

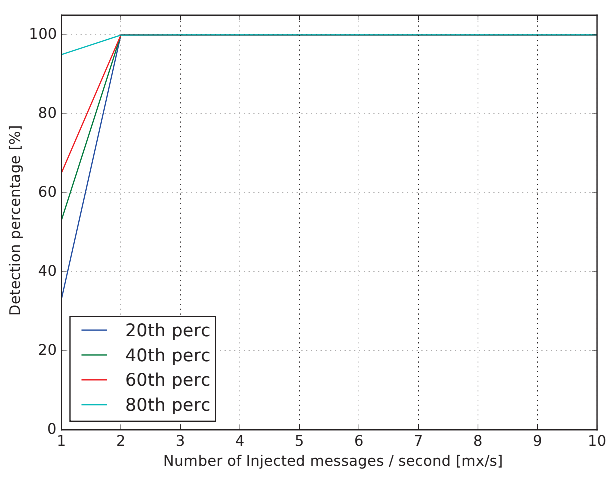

Fig. 6. Statistical Injection detection percentage

to build message sequences that were not observed during training. Finally, we build the malicious traces by injecting these sequences within legit traces that have not been used for training. This is a best-case scenario, since injected sequences include at least one transition that has not been included in the transition matrix.

Finally, *Mixed Injection* attacks are generated by injecting sequences that comprise a random mix of CAN messages. This scenario simulates a realistic attacker that builds an attack payload only by focusing on its goals, without taking into account normal message transitions and message probability distributions.

For each of these attacks we generate multiple malicious traces, changing the length of injected message sequence (from 1 to 10 messages) and with different sequences for each length, for a total of 375 different malicious traces (125 in each of the three attack scenarios).

IV. Experimental evaluation

To verify the ability of the proposed algorithm to detect different types of attacks that involve injection of CAN messages, we test it against traces containing malicious traffic that were described in Section III.

Before testing, we trained and validated an instance of the proposed anomaly detector by using approximately 40 million messages, extracted from about 4 hours of driving in different conditions. The training phase was made on about 8 million messages and the validation on the other 32 millions.

After training, we applied our detector to traces including both basic injections and realistic attacks. Experimental results are described in Sections IV-A and IV-B, respectively.

A. Detection of basic injections

We first test our detector against the 40 attack traces described in Section III-A. Detection results are shown in Figure 6.

The x-axis represents the number of packets included in each injected message sequence, while the y-axis represents the number of attacks identified by our detector, expressed as a percentage over the total number of attacks. The four different lines refer to traces including messages at the 20th, 40th, 60th and 80th percentile of the probability distribution, as described in Section III-A.

Percentile	Detection Rate
20th	32%
40th	45%
60th	66%
80th	95%

TABLE I
Percentile single injection detection rate

Results show that if the attacker injects only a single message, detection performance heavily depends on the probability distribution. In particular, messages that appear on the CAN bus with higher probability are less likely to be detected, while messages with a lower probability have higher detection rates.

The detection performance in case of single message injection for every percentile are shown in Table I.

Detection performance also improve for longer attacks, comprising more than one message. In particular, we highlight that sequences composed by at least two messages are always detected (detection probability equal to 100%), independently by the probability distribution of the injected messages.

We also highlight that the detector did not generate any false positive during these tests, thus demonstrating that transitions of message IDs represent a stable feature, that is preserved in attack-free conditions.

B. Detection of realistic attacks

We then test the detector against the three different attack classes described in Section III-B. Detection results are shown in Figure 7.

The x-axis represents the number of packets included in each injected message sequence, while the y-axis represents

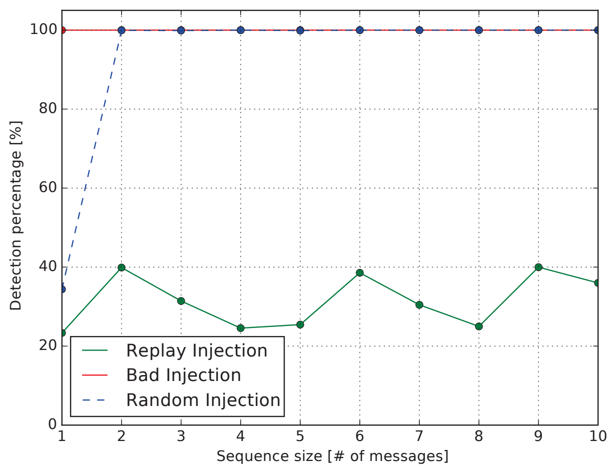

Fig. 7. Sequence Attack Detection

the number of attacks identified by our detector, expressed as a percentage over the total number of attacks. The three different lines refer to traces including replay, bad injection and random injection attacks, as defined in Section III-B.

In the case of a Replay attack, we can clearly see that, despite the length of the sequence of injected messages, the detection percentage does not follow any particular trend, and varies within a range of 20% to 40%.

We remark that this represents a worst case scenario for the proposed anomaly detection algorithm, because attacks are a replay of message sequences already seen during training. Hence, the algorithm can only detect anomalies in the transitions at the beginning and at the end of the injected sequence. This means that the length of the sequence does not influence the capacity of our algorithm to detect anomalies, and thus the detection percentage. A possible solution for increasing the detection rates in this attack scenario is to use a our algorithm alongside with different approaches. In [1] we proposed an algorithm that uses statistical inspection on the messages for anomaly detection that performs extremely good detecting injection of massive replay messages.

On the other hand, Bad Injection attacks represent the best case for our algorithm. Since injected sequences have never been observed during training, all of them contain at least one transition that is not included in the transition matrix. Hence our detection rate in this case is always 100%, independently on the sequence length.

The more realistic scenario is represented by Mixed Injection attacks, that simulates generic attack sequences. For this scenario, our algorithm is able to detect attacks that comprise only one message about 40% of the times. However, detection rates improve considerably as the length of the injected sequence increases. For sequences composed by only two messages, the detection rates goes up to 99.9%, and we achieve 100% detection rate for all sequences of three or more messages.

Finally, we highlight that the proposed algorithm generated 0 false positives for all the 375 attack traces belonging to this scenario. These results demonstrate that the sequences of the messages on the CAN bus is a useful feature for detecting possible intrusions. Our approach is able to detect efficiently an high percentage of attacks without generating any false positives.

V. Related work

Recent reports of attacks carried out by injecting malicious CAN data frames within the CAN bus of modern vehicles [3], [6] motivated several research efforts aiming at improving the security of in-vehicle networks. Since the CAN bus is an insecure communication channel that lacks authentication, confidentiality, integrity and availability guarantees [7] several research efforts focused on securing communications among ECUs by leveraging cryptographic algorithms [8], [9]. While this approach is theoretically sound, economic competition among car makers pushes towards the adoption of very simple ECUs whose computational limitations are not compatible with modern cryptography, thus limiting the applicability of similar proposals.

On the other hand, another research field focuses on detecting attacks carried out by injecting malicious messages over the CAN bus, rather than preventing them. Intrusion detection systems represent a popular technology for realizing and improving attack detection capabilities in modern information systems [10], [11], [12]. They can analyze network packets (Network IDS), activities executed within computers (Host IDS) or a combination of these kind of events (Hybrid IDS).

Many research efforts strive to apply the well known concept of Network IDS to the network architectures and protocols that connect ECUs in modern cars. In particular, several works aimed at developing robust anomaly detection approaches for the CAN bus. Anomaly detection is a promising approach, since in principle it allows to identify novel and unknown attacks. The main assumption that lies beneath anomaly detection is that it is possible to build a model that describes the normal behavior of the CAN bus, and that attacks can be detected because they introduce a measurable deviation from the normal profile [13]. Works within this field propose different models for the definition of the normal behavior of the CAN bus.

The simplest approach is to detect CAN bus messages having an invalid ID. In this case, the normal model is just a set of valid message IDs, either gathered from the formal specification of a given vehicle, or learned by sniffing correct messages from the can bus. This approach allows easy and precise identification of attacks that inject CAN messages with an invalid ID, but can be easily foiled by attackers clever enough to inject arbitrary messages with valid IDs. Hence this approach can be useful against basic fuzzing [14], [15] techniques, but becomes useless against more sophisticated or determined attacks.

Another popular approach is based on the analysis of the cycle time that characterizes many periodic CAN messages [4]. Unfortunately, this approach cannot be applied to non-periodic messages. Moreover, even periodic messages may be characterized by a variable cycle time, influenced by events (i.e. ECUs that change their behavior based on vehicle conditions) or load on the bus. This variability leads to the generation of many false positives.

Other works propose the creation of a model based on aggregated statistical features, such as the average bus load and message entropy [1]. Similar proposals are effective in the identification of massive attacks (such as message flooding), but cannot detect attacks based on the injection of just few malicious CAN messages, that are not enough to generate a measurable deviation in the aggregate statistical features of the whole CAN bus.

Besides detection performance, it is also imperative to design detection algorithms that are compatible with the tight computational constraints of automotive ECUs. As an example, while the work proposed in [5] represents an interesting detection approach based on Deep Neural Network, its high computational cost render it clearly unsuitable for modern vehicles.

In this paper we propose a novel intrusion detection algorithms that is based on the identification of recurrent patterns of IDs.

The proposed algorithm has several advantages with respect to the state of the art. First of all, it does not require the formal DBC specification, and can be trained over traces captured from licensed vehicles. Moreover, experimental evaluations demonstrate that the proposed algorithm is able to detect with high probability even stealth attacks, in which attackers only inject a very limited number of packets having legitimate ID and correct format. Moreover, it achieves a very low false positive rate (no false positives have been observed in our experimental evaluation based on real traffic traces that were not used for training). Finally, the proposed algorithm has been specifically designed to fit the very hard computational constraints that are typical of micro controllers used to implement vehicular ECUs. We also highlight that our proposal can be easily integrated with other complementary detection algorithms based on different features, thus contributing to improve the effectiveness of future vehicular intrusion detection systems based on heterogeneous data and analysis approaches.

VI. Conclusion

This paper presents a novel anomaly detection algorithm designed to identify attacks based on the injection of malicious messages over the CAN bus of modern vehicles. The proposed algorithm is based on the analysis of the sequence of CAN bus messages generated by a vehicle under normal conditions, and fits the tight memory and computational constraints of ECUs embedded in current vehicles. Experimental evaluations based on real CAN traces gathered from a licensed and unmodified vehicle demonstrate its high performance in terms of high detection and low false positive rates.

Future works include the integration of this algorithm with complementary intrusion detection approaches based on different features.

References

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

[2] C. Miller and C. Valasek. (2015) Remote exploitation of an unaltered passenger vehicle. Appeared in Blackhat US conference. [Online]. Available: https://www.blackhat.com/us-15/briefings.html#remote-exploitation-of-an-unaltered-passenger-vehicle

[3] Keen Security Lab of Tencent. (2016) Car hacking research: Remote attack tesla motors. [Online]. Available: http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/

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

[5] M. J. Kang and J. W. Kang, “A novel intrusion detection method using deep neural network for in-vehicle network security,” in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–5.

[6] C. Miller and C. Valasek. (2015) Remote exploitation of an unaltered passenger vehicle. White paper of Blackhat US conference. [Online]. Available: http://illmatics.com/Remote%20Car%20Hacking.pdf

[7] B. GmbH. (1995) Can specification version 2.0. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf

[8] M. Wolf and T. Gendrullis, *Design, Implementation, and Evaluation of a Vehicular Hardware Security Module*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 302–318. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31912-9_20

[9] B. Carnevale, F. Falaschi, L. Crocetti, H. Hunjan, S. Bisase, and L. Fanucci, “An implementation of the 802.1ae mac security standard for in-car networks,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 24–28.

[10] M. Andreolini, M. Colajanni, and M. Marchetti, “A collaborative framework for intrusion detection in mobile networks,” *Information Sciences*, vol. 321, pp. 179–192, 2015.

[11] M. Marchetti, M. Colajanni, and F. Manganiello, “Framework and models for multistep attack detection,” *International Journal of Security and Its Applications*, vol. 5, no. 4, pp. 73–90, 2011.

[12] M. Colajanni, D. Gozzi, and M. Marchetti, “Enhancing interoperability and stateful analysis of cooperative network intrusion detection systems,” in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*. ACM, 2007, pp. 165–174.

[13] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[14] S. Bayer and A. Ptok, “Dont fuss about fuzzing: Fuzzing controllers in vehicular networks.”

[15] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, “Fuzzing can packets into automobiles,” in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, March 2015, pp. 817–821.