This is the peer reviewd version of the followng article:

Detecting attacks to internal vehicle networks through Hamming distance / Stabili, Dario; Marchetti, Mirco; Colajanni, Michele. - (2017), pp. 1-6. (Intervento presentato al convegno IEEE 2017 AEIT International Annual Conference - Infrastructures for Energy and ICT (AEIT 2017) tenutosi a Cagliari, Italy nel September 2017).

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

12/10/2024 04:40

(Article begins on next page)

Detecting attacks to internal vehicle networks through Hamming distance

Dario Stabili, Mirco Marchetti and Michele Colajanni
University of Modena and Reggio Emilia, Italy
Email: {dario.stabili, mirco.marchetti, michele.colajanni}@unimore.it

***Abstract*—Analysis of in-vehicle networks is an open research area that gained relevance after recent reports of cyber attacks against connected vehicles. After those attacks gained international media attention, many security researchers started to propose different algorithms that are capable to model the normal behaviour of the CAN bus to detect the injection of malicious messages. However, despite the automotive area has different constraint than classical IT security, many security research have been conducted by applying sophisticated algorithm used in IT anomaly detection, thus proposing solutions that are not applicable on current Electronic Control Units (ECUs). 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. Moreover, the proposed algorithm has been designed and implemented with the very strict constraint of low-end ECUs, having low computational complexity and small memory footprints. The proposed algorithm identifies anomalies in the sequence of the payloads of different classes of IDs by computing the Hamming distance between consecutive payloads. Its detection performance are evaluated through experiments carried out using real CAN traffic gathered from an unmodified licensed vehicle.**

I. Introduction

The increasing adoption of advanced infotainment systems and self-driving capabilities makes modern vehicles similar to mobile networks of computing devices, often connected to the public Internet. This trend opens different scenarios of groundbreaking innovation, but also exposes novel attack surfaces that cyber-attackers can exploit. Recent works shows that it is possible to remotely control a vehicle [1], [2], [3], with implicit safety hazards for drivers and nearby people.

In all cases, attacks were made possible by the lack of proper security countermeasures in current vehicle networks. Systems and algorithms for the detection of attacks injected over the CAN bus of modern vehicles is still an open research area, whose importance is motivated by the relevant safety risks for drivers, passengers and people nearby a vehicle under cyber-attack.

This paper proposes a novel anomaly detection algorithm for the CAN bus of modern vehicles that is based on the evaluation of the *Hamming distance* [4] between the payloads of consecutive CAN messages having the same ID [5]. We remark that both message ID and payload can be easily gathered and extracted from the CAN bus. The proposed analysis does not require full specifications of the payload syntax and semantic, that car makers and OEM consider as strictly confidential information. Hence the proposed approach is immediately applicable to any licensed vehicle.

Performance of the proposed algorithm have been tested against the injection of malicious messages on real CAN traffic traces gathered from the test vehicle at our disposal. Moreover, computational and memory requirements of the proposed algorithm have been evaluated to be low enough to be compatible with the hardware constraints of ECUs (Electronic Control Units) embedded in current vehicles.

To the best of our knowledge, this is the first algorithm that inspects the sequences of the payload values for different classes of IDs of the vehicle.

II. Related work

Attacks to modern vehicle executed by injecting malicious CAN messages in the CAN bus [1], [3], [2] motivated novel research efforts aimed at improving the security of the CAN bus. Many solutions for Anomaly Detection in classical IT networks have already been proposed [6], [7], [8], based on several different approaches: they can analyze network packets (Network IDS), activities executed within computers (Host IDS) or a combination of these kind of events (Hybrid IDS). Independently on the monitored resources, IDSes can identify signatures or anomalies. A signature approach require a database of known attacks in order to detect anomalies. It is possible to detect only anomalies that are included in such database, meaning that unknown vulnerabilities can be exploited in order to successfully perform attacks. In the automotive domain this approach is not applicable, because attacks are emerging and novel.

Related work in the automotive field focus on detecting attacks by analyzing the data available on the CAN bus, as mandated by the CAN specifications [5]. Many research efforts apply algorithms typical of anomaly-based [9] Network IDS to those data.

The simplest approach is to detect CAN bus messages having an invalid ID [10]. 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 [11] techniques, but becomes useless against more sophisticated or determined attacks.

Popular approaches are based on the analysis of the cycle time of periodic CAN messages [12], [13], [14]. However, cycle time variability caused by contention or by events happening while driving leads to the generation of many false positives. Moreover, this approach is inapplicable to all non-periodic messages.

Other works model the normal behavior of the CAN bus using several statistical features [15], [16]. Similar proposals can only detect massive attacks injecting hundreds or thousands of messages in a very short time frame, but fail against targeted attacks involving just a few messages.

A different approach has been proposed in [17], in which the normal behaviour of legit CAN traffic is created by inspecting the possible transitions between consecutive message IDs. Despite achieving good detection results, this approach is limited to the inspection of message IDs and behaves poorly against targeted injection of highly frequent messages.

The algorithm proposed in this paper models the normal behaviour of CAN traffic by evaluating the difference between consecutive payloads of the same ID by means of the *Hamming distance*. This design choice has several advantages with respect to the state of the art. First, it does not require full knowledge of the syntax and semantics of CAN messages. Normal features can be learned just by analyzing traffic traces sniffed from a licensed vehicle. Moreover, experimental evaluations demonstrate that the proposed algorithm is able to detect even stealth attacks that involve the injection of very few CAN messages.

We also remark that proposed solution for improving CAN bus security complies with the hardware constraints of a typical automotive ECUs, having very low memory and computational requirements. As an example, while the detection approach based on deep neural networks proposed in [18] is very interesting from a theoretical point of view, its high computational cost makes it inapplicable to modern vehicles. On the other hand, the detection approach proposed in this paper can be applied to modern ECUs.

III. Background

A. Controller Area Network

The Controller Area Network (CAN) is a vehicle bus standard designed to allow data exchange among microcontrollers without requiring a host computer. The CAN bus was developed at Robert Bosch GmbH and officially released in 1986. The last specification of the CAN Bus is the CAN 2.0, published in 1991 [5]. The CAN bus is a multi master serial bus with at least two different nodes, connected through a two-wire bus. Data are sent through the network using a particular type of frame, the *Data frame*. Data frames can have different structures depending on the formats, as shown in Figure 1. The main fields are the *Identifier* and *Data Field*. The *identifier* is used to distinguish among different types of CAN data frame. Data frame characterized by a given identifier are usually produced by a specific ECU, while every ECU reads the identifier field of every message in order to determine which messages are of interest for its processing logic. The identifier is also used for arbitration of the CAN messages: lower values of this field denote messages with higher priority. The identifier is the only field whose size varies depending on the type of CAN message. In particular, as shown in Figure 1a, the identifier field for the *basic format* has a fixed length of 11 bits, while for the *extended format*, shown in Figure 1b, the identifier has a maximum length of 29 bits. Despite the different number of bits used in the *identifier*, basic and extended format are compatible with each other The *Data field* is a sequence with a maximum length of 8 bytes (64 bits), used to represent the values transmitted over the CAN bus. A generic data frame usually packs several different signals within the same Data field, and the CAN standard leaves complete freedom to the car makers about the structure, number and meaning of signals. Hence, without having access to the formal specification of a CAN message, its Data field looks like a binary blob.

B. Hamming Distance

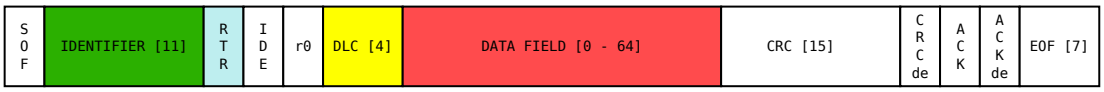
The *Hamming distance* is used to measures the minimum number of substitutions required to change one string into the other. The *Hamming distance* has been proposed by Richard Hamming in 1950 [4] and it is widely used in several disciplines including information theory, coding theory and cryptography. The generic formula for evaluating the *Hamming distance* between two words of equal length k can be found in equation 1:

$$\mathcal{H}_d(x, y) = \sum_{i=1}^{k} |x_i - y_i| \tag{1}$$

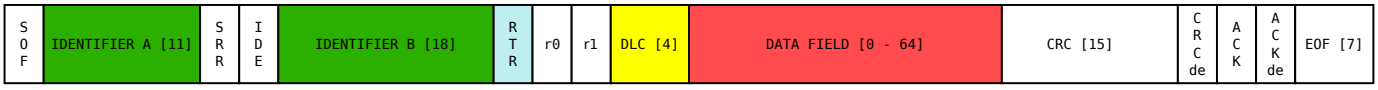
In the equation 1, the evaluated $x_i - y_i$ is equals to 0 when x_i is equals to y_i, 1 otherwise. In particular, in this paper the Hamming distance is evaluated on two binary strings. This implementation of the Hamming distance is known as *Hamming cube*, a variation of the classic Hamming distance applied on strings that are comparable to vertices of an hypercube graph. A generic binary string of length n can be evaluated as vector in $\mathbb{R}^{\mathrm{n}}$ by treating each symbol in the string as a real coordinate. This means that each string can be evaluated as a particular vertex of the n-dimensional hypercube, thus, the *Hamming distance* of two binary strings is equivalent to the *Manhattan distance* between the vertices of the generated hyperline. This paper evaluates the *Hamming distance* between two binary string, reducing the hypercube to a single hypervector. Having payloads composed by 64 bits of data, the hyperspace containing all the different possible vertices have a maximum size of 2^{64}. For the data frame payloads, composed by 64 bits, the *Hamming distance* is evaluated as shown in Equation 2, where p_k is a generic payload at time k and p_k^i is the i_{th} bit of that payload.

$$\mathcal{H}_d(p_t, p_{t+1}) = \sum_{i=1}^{64} p_t^i \otimes p_{t+1}^i \tag{2}$$

Fig. 1: Dataframe types comparison



(a) Base format



(b) Extended format

IV. Attack Scenario

To prove the effectiveness of the algorithm proposed in this paper, attacker activities have been simulated by injecting forged CAN messages within the legit CAN traffic generated by a licensed, unmodified vehicle. This process has been widely used for the evaluation of several intrusion detection algorithm [11], [12], [13], [15], [17], allowing security researchers to simulate effects over the CAN bus of real attack strategies [1], [3] without safety hazards. For the testing purposes of the proposed algorithm, two different datasets have been created. The former simulates *Fuzzing attack*, a reverse engineering techniques executed by injecting randomly forged values and studying the reaction of the whole system, as described in Section IV-A. The latter simulates a *Replay attack*, where legit payloads are injected into the CAN bus, as described in Section IV-B.

A. Fuzzing Attack

Fuzzing attack simulation traces have been created in order to test the proposed algorithm against the preliminary attack phases normally executed by attackers targeting an unknown vehicle. These activities are required [19], [20], [1], [11] for reverse engineering of the encoded signals transmitted over the CAN bus. This dataset has been created starting from different traffic traces gathered from the test vehicle under different driving and traffic conditions. For each ID found in the collected traces, a random payload is generated and injected, simulating a classical *fuzzing* approach.

B. Replay Attack

In order to prove the effectiveness of the proposed algorithm against a *replay attack*, a dataset have been created by injecting already seen message payloads in a randomly chosen position within the trace of normal traffic. This dataset has been created starting from different valid traffic traces recorded from a licensed unaltered vehicle in different driving and traffic conditions. For each ID of the test vehicle, one payload has been randomly chosen from the ones seen on the collected traces and injected in random positions. Different traces present different injected payloads, while in the same trace the injected payload is always the same.

The injection position of the payload for both *fuzzing* and *replay attack* is randomly chosen in a window of size N for each message ID. Different sizes of the injection windows have been tested in order to determine how that size could impact the detection of the injected messages. For the final generation of malicious datasets, the size N of the windows changes between 10, 25 and 50, generating a total of 30 different traffic traces containing different anomalies.

V. Algorithm Description

The algorithm proposed in this paper evaluates the *Hamming distance* between two consecutive payloads of the same ID in order to detect variations from the previously created and validated normal model. The *Hamming distance* is evaluated using the formula expressed in Equation 2.

The normal model has been created and validated on all the different IDs, gathered from the main CAN bus of a 2011 Ford Fiesta®. For the purposes of this paper, five different traces have been collected, each one recorded under different traffic situations, aiming to maximize the variability of the message payloads. The proposed algorithms is composed by two different phases: an initial *model creation and validation*, that generates all the references for each ID of the car model and validates the preliminary results against other traces; and the *live detection* phase, designed and implemented in order to test the created model against real attack scenarios and to demonstrate that the proposed solution is applicable to ECUs with limited hardware resources.

A. Model Creation and Validation

In the *Model Creation and Validation* phase the algorithm inspects the previously collected CAN log traces in order to compute the metrics used by the proposed algorithm. In particular, the minimum and maximum Hamming distance evaluated between sequences of payloads are extracted for each different message ID.

For model creation we selected 20% of the gathered CAN bus data. Can traffic has been split into subtraces, one for each different ID. For each subtrace, we computed the Hamming distance between the payloads of consecutive messages. We then selected the minimum and the maximum among these distances and associated these two value to the related ID. These values represent the *Hamming range* that characterize consecutive messages of the given ID. Our model is based on the assumption that the Hamming distance between two consecutive normal payloads will fall in the Hamming range learnt during training. On the other hand, messages injected by

an attacker will contain values that are in conflict with those generated by licit ECUs. If the Hamming distance between an injected message and the adjacent messages having the same ID are out of the Hamming range, our approach is able to detect the attack.

In the validation phase, the algorithm checks the remaining 80% of the gathered data, evaluating the Hamming distance between every pair of consecutive payloads belonging to the same message ID and comparing it with the Hamming range learnt during training. A *false positive* alert is raised if the distance evaluated for a particular message ID is out of the reference range, since these traffic traces contain no attack. During the validation process the algorithm raised 0 *false positives*, meaning that the minimum and maximum Hamming distance learnt in the training phase represent a stable feature that can be used to identify message injections.

B. Live Detection

The *live detection* phase is the part of the algorithm designed for the real time detection of anomalies on the CAN bus. This phase has been designed to adapt to the very limited resources of common ECUs that can be found on today's vehicles, characterized by only a few kilobytes of RAM and requirements and very low computational power. Using the previously created model, the algorithm in this phase is able to determine anomalies by evaluating the Hamming distance of two consecutive payloads of the same message ID. When the evaluated distance is outside the Hamming range associated to that particular ID, an anomaly is detected.

VI. Experimental Evaluation

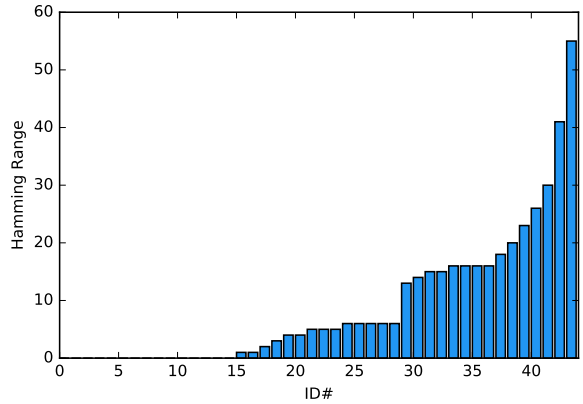
The proposed algorithm has been tested against the two different types of attacks commonly used to subvert vehicle security and reverse engineer the behaviour of ECU previously described in Section IV.

Preliminary analyses on the detection results highlighted that many IDs had similar detection rates for both *Fuzzing* and *Replay* attacks. After a more detailed investigation, we found out that IDs having very close detection results also have very similar Hamming ranges.

Statistical analysis of the Hamming ranges is shown in Figure 2, where the y-axis represent the Hamming range and the x-axis is an identifier of the 43 different IDs that have been found in the CAN traffic traces. IDs have been ordered with respect to the associated Hamming range. It is possible to observe that IDs can be naturally classified in three main categories:

- **NoRange**: IDs for which the Hamming distance between consecutive messages is always constant, hence the minimum and maximum Hamming distances are equal and the Hamming range is 0
- **SmallRange**: IDs for which the distance between the maximum and minimum *Hamming distances* (Hamming range) is always lower than a σ reference value

Fig. 2: Hamming range distribution



- **MidRange**: IDs for which the distance between the maximum and minimum *Hamming distances* (Hamming range) is always higher than σ

The value of σ has been empirically determined to be equal to 6. This choice allows to group together IDs with very similar detection rates, and is motivated by the relatively high gap that exists between IDs $28th$ and $29th$, as shown in Figure 2. According to this classification the *NoRange*, *SmallRange* and *MidRange* classes comprise 15, 13 and 15 message IDs, respectively.

A. Fuzzing Attack Detection

Figure 3 represents the detection results of the proposed algorithm against a fuzzing attack as described in Section IV.

For each class we show three different detection results, represented as sets of three vertical bars. The leftmost set of bars refers to a fuzzing message injected every 10 normal messages, the middle bars to an injection every 25 messages, and the rightmost bar refers to an injection every 50 messages. Within each set of bars, the first refer to the *NoRange* class, the second to the *SmallRange* class, and the third to the *MidRange* class. The x-axis represent the attack frequency, while the y-axis represents the detection rate.

Results in Figure 3 show that the proposed algorithm is able to detect the injected attack with percentages close to 100% injection of fuzzing messages in cases of both *NoRange* and *SmallRange* classes. In particular, the detection rate for the *NoRange* class is always higher than 98% and for the *SmallRange* class is always higher than 97%.

On the other hand, the detection rate is lower for attacks involving the IDs classified as *MidRange*. In this class the detection rate varies from 20% to 30% depending on the intensity of the attack. Attacks with a shorter period and a higher injection frequency present better detection results with respect to those characterized by a longer period and a lower injection frequency.

Poor detection results in case of the *MidRange* class are a direct consequence of the relatively high Hamming range that

Fig. 3: Fuzzing Results Overall Performance

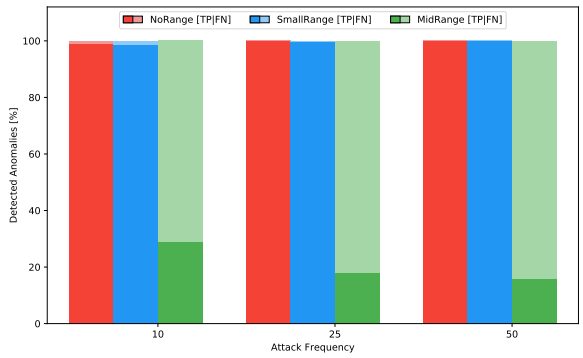
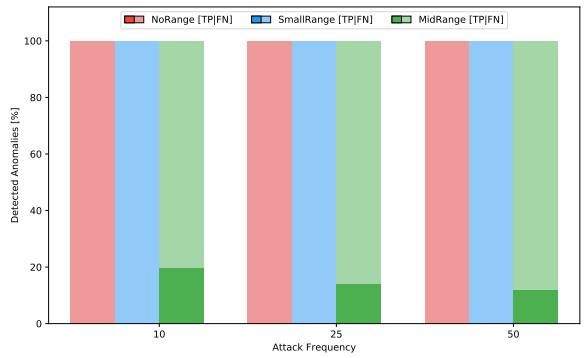


Fig. 4: Replay Results Overall Performance



characterize IDs belonging to this class. *MidRange* class is composed by IDs with Hamming range above the selected σ, meaning that the Hamming distance between payloads of IDs belonging to this class are more prone to change significantly during the vehicle dynamic. Thus, by injecting randomly generated payloads, there are higher probabilities that the injected payload is close enough in terms of Hamming distance to it neighbours, generating higher false negatives and keeping the detection rates smaller than the other cases.

We remark that the proposed detection algorithms never raised a false alert, hence the false positive rate is 0.

1) Replay Attack Detection: Figure 4 represents the detection results of the proposed algorithm against a fuzzing attack as described in Section IV.

Similarly to the previous attack scenario,for each class we show three different detection results, represented as sets of three vertical bars. The leftmost set of bars refers to a replay message injected every 10 normal messages, the middle bars to an injection every 25 messages, and the rightmost bars refers to an injection every 50 messages. Within each set of bars, the first refer to the *NoRange* class, the second to the *SmallRange* class, and the third to the *MidRange* class. The x-axis represent the attack frequency, while the y-axis represents the detection rate.

It is possible to observe that the proposed method is not suitable for the detection of replay attacks, especially for the *NoRange* and *SmallRange* classes. In particular, our method is never able to raise an alert for replay attacks on IDs belonging to the *NoRange* class, while detection rates for the *SmallRange* class are always below 2%. This result is caused by the very low Hamming range that characterize legit messages, that are very similar among themselves. Since a replay attack is executed by injecting a legit message out-of-place within the CAN traffic, the injected message will be very similar to the adjacent licit traffic.

Better results can be achieved for IDs belonging to the *MidRange* class, and characterized by higher Hamming ranges (and higher message variability). Detection rate for this class varies between 20% and 10%, depending on the attack frequency. While these detection rates are small, we remark that the proposed detection method raised no false alerts, hence the false positive rate is always 0.

B. Computational cost

An accurate inspection of computational complexity and memory requirements for the proposed algorithm is given in this section to prove its low computational requirements that make it applicable to common ECUs found in modern licensed vehicles.

Computational Complexity: The algorithm for live detection discussed in Section V-B requires to compare the current payload value of a specific ID with the previous recorded payload from the same ID. The evaluation of the Hamming distance of the two payloads takes place in a single loop, that compares all the N bits composing the payload and sums the result of a bit-wise *XOR* operation among the bits at the same index of the two different payloads, adding one in case the two bits are different from each other, 0 otherwise. At the end of this loop, the evaluated distance is compared to the reference values for the same ID, raising an anomaly when the value is outside the valid range. Computational complexity of the final implementation of the live detection has been evaluated as $O(N)$, where N is equal to the number of bits forming the payloads, with a maximum value of 64.

Memory Requirements: The proposed detector uses one indexed data structure to store the previous payload for each ID. The size of this structure is evaluated as $N_{id} * L_{id}$, where N_{id} represents the number of unique IDs flowing on the internal network and L_{id} denotes the number of bits composing the payload for that particular ID. Maximum memory requirements could be evaluated as $N_{id} * 64$, where 64 is the maximum allowed value for L_{id}.

For our vehicle requirements, the final implementation requires a maximum of 344 Bytes to store the previous messages, having 43 unique message IDs flowing on the CAN bus.

Common low-end ECUs are generally composed by microcontrollers with 1 computational core, having a working frequency in the orders of hundreds of Mega Hertz, and with

few hundreds of Kilo Bytes of RAM. For the tested vehicle the live detector only requires up to 344 Bytes of memory, and its operations can be carried out by a common micro-controller equipped with a single core. Hence, the proposed live-detection algorithm can be implemented on common low-end ECUs.

VII. Conclusion

This paper proposes a novel algorithm that aims to detect cyber-attacks that involve the injection of malicious forged CAN messages into modern vehicles networks. In particular, the proposed detection algorithm analyzes the sequences of payloads of all messages that are transmitted, and compares the Hamming distance between consecutive payloads of the same ID with respect to a reference range of valid Hamming distances that is built during an off-line training phase. The use of the Hamming distance is motivated by its very low computational complexity. As a result, the proposed live-detection algorithm have very small memory footprint (in the order of a few hundreds of bytes) and can be executed even on low-end microcontrollers that characterize the ECUs deployed within modern vehicles.

Experimental evaluation carried out over CAN traffic traces gathered from an unmodified 2011 Ford Fiesta show that the proposed model is able to detect almost 100% of fuzzing attacks characterized by an injection frequency higher than one injection over 25 messages.

Future work involve the design of other detectors characterized by low computational complexity that are able to achieve better detection performance for replay attacks, as well as the integration of the proposed detection algorithm with other approaches already proposed in the automotive security literature.

References

[1] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," *Blackhat USA*, vol. 2015, pp. 1–91, 2015. [Online]. Available: http://illmatics.com/RemoteCarHacking.pdf

[2] A. Greenberg. (2015) Hackers remotely kill a jeep on the highway - with me in it! [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[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/

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

[4] R. W. Hamming, "Error detecting and error correcting codes," *Bell Labs Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[5] R. B. GmbH. (1991) Can specification version 2.0. [Online]. Available: http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf

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

[7] 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.

[8] 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.

[10] C. Ling, "An Algorithm for Detection of Malicious Messages on CAN Buses," in *CITCS-12*, 2012.

[11] 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.

[12] 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.

[13] 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 '17 Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, 2017, pp. 2–5.

[14] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 911–927. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

[15] 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.

[16] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1110–1115, 2011.

[17] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *28th IEEE Intelligent Vehicle Symposium (IV2017)*, June 2017.

[18] 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.

[19] C. Valasek and C. Miller, "Car Hacking : For Poories," *Last Accessed from http://www.ioactive.com/pdfs/IOActive_Car_Hacking_Poories.pdf*, pp. 1–26, 2014. [Online]. Available: http://illmatics.com/car{_}hacking{_}poories.pdf

[20] C. Miller and C. Valasek. (2014) Adventures in automotive networks and control units. [Online]. Available: https://www.ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf