# Detection of Injection Attacks in Compressed CAN Traffic Logs

András Gazdag[1], Dóra Neubrandt[1], Levente Buttyán[1], and Zsolt Szalay[2]

[1] Laboratory of Cryptography and System Security
Department of Networked Systems and Services
Budapest University of Technology and Economics
{agazdag, dneubrandt, buttyan}@crysys.hu

[2] Department of Automotive Technologies
Faculty of Transportation Engineering and Vehicle Engineering
Budapest University of Technology and Economics
zsolt.szalay@gjt.bme.hu

**Abstract.** Prior research has demonstrated that modern cars are vulnerable to cyber attacks. As such attacks may cause physical accidents, forensic investigations must be extended into the cyber domain. In order to support this, CAN traffic in vehicles must be logged continuously, stored efficiently, and analyzed later to detect signs of cyber attacks. Efficient storage of CAN logs requires compressing them. Usually, this compressed logs must be decompressed for analysis purposes, leading to waste of time due to the decompression operation itself and most importantly due to the fact that the analysis must be carried out on a much larger amount of decompressed data. In this paper, we propose an anomaly detection method that works on the compressed CAN log itself. For compression, we use a lossless semantic compression algorithm that we proposed earlier. This compression algorithm achieves a higher compression ratio than traditional syntactic compression methods do such as gzip. Besides this advantage, in this paper, we show that it also supports the detection of injection attacks without decompression. Moreover, with this approach we can detect attacks with low injection frequency that were not detected reliably in previous works.

**Keywords:** CAN · Anomaly Detection · CAN traffic compression

## 1 Introduction

These days vehicles can be the target of various cyber attacks. In a modern automobile, there are numerous ECUs (Electronic Control Units) which are responsible for different functionalities. These ECUs are connected together with the Control Area Network (CAN) bus and they communicate with each other using the CAN protocol. There are several external interfaces which can be used to gain access to this inner network of the vehicle such as wireless interfaces (Bluetooth, WiFi, wireless TPMS) and the on-board diagnostic port (OBD).

The design of the inner network and its protocol and interfaces makes the CAN vulnerable against several attacks. We will elaborate more on these attacks in Section 3. While in the past, such attacks were considered low risk and not a real concern, recently, researches have demonstrated [1] [2] [3] that they are not so difficult to carry out and, hence, the risk is indeed considerable.

Cyber attacks on vehicles can cause physical accidents. This means that when an accident happens, forensic analysis must be extended into the cyber domain, and investigators must analyze whether the accident was caused or made possible by a cyber attack. Imagine, for example, that a compromised ECU provides false data and as a consequence, misleading information is displayed to the driver on the dashboard, or the airbag is disabled silently before a crash, or some autonomous driving function is enabled and the driver loses control over the vehicle. All these can either lead to an accident or increase its fatality. As the cyber attack on the vehicle may occur well before the accident that it causes, forensic analysis can be successful only if detailed logs are recorded for an extended period of time, not just for a few seconds before the accident[3].

In our view, in the future, especially with the increased penetration of autonomous vehicles, it will be indispensable to continuously record CAN traffic in vehicles and efficiently store these logs for later forensic analysis. Efficient storage of CAN logs requires compressing them. Compression not only saves storage space, but it also makes it easier to off-load logs from the vehicle. Usually, the compressed log must be decompressed for analysis purposes, and the analysis is carried out on large amount of decompressed data. This increases the inefficiency of the analysis itself. In this paper, we study the problem of detecting anomalies that may indicate cyber attacks on the compressed CAN traffic log, hence making analysis faster by not requiring decompression and most importantly by reducing the amount of data on which the analysis must be performed.

Anomaly detection cannot be performed on any kind of compressed CAN log, but the compression method must support the analysis of the compressed data. Hence, for compression, we use a lossless semantic compression algorithm that we proposed earlier [4]. This compression algorithm achieves a higher compression ratio than traditional syntactic compression methods such as gzip. Besides this advantage, in this paper, we show that it also supports the detection of certain types of attacks in the CAN log without decompression. More specifically, we can easily detect flooding attacks, where the attacker (e.g., a compromised ECU) injects a given type of periodic CAN message with a smaller repetition time (higher frequency) than its normal repetition time. Most of the attacks demonstrated in prior work were of this kind[1][2][3]. The increased frequency of injected false messages usually results in "overriding" the information carried in the legitimate messages. We show that such an attack causes a well-identifiable anomaly pattern in the compressed log even when the frequency of the fake messages is just slightly larger than the normal frequency.

The remainder of the paper is organized as follows: In Section 2, we give an overview on the existing anomaly detection works on CAN traffic. In Section

[3] https://www.nhtsa.gov/research-data/event-data-recorder

3, we describe how the CAN protocol and the CAN compression algorithm we use work. In Section 4, we discuss the attack scenario and the possible attacks against the CAN protocol that we and recent works take into consideration. We present our anomaly detection approach and its evaluation in Section 5. Finally, we conclude in Section 6.

## 2 Related Work

Anomaly detection on the CAN bus has been an actively researched field recently. Multiple approaches have been proposed varying in the interpretation of the CAN traffic. If the interpretation of the CAN messages are accessible it is possible to collect the actual vehicle parameters. Approaches using this knowledge usually perform anomaly detection on this high level data. The researches not using a CAN matrix are mainly focused on the communications properties such as repetition times of the messages.

A. Taylor et al. proposed a method from the first approach in [8]. They interpreted the CAN massages to build a current state of the vehicle. Then with a Long Short-Term Memory Network (LSTM) predicted the next state of the car. If the actual state, based on the following messages, is diverging from the predicted state they detect it as an anomaly.

S. N. Narayanan et al. proposed a hidden Markov models based approach to anomaly detection[11]. They used the OBD port available in every modern car to access the CAN bus. Packets captured through this interface are interpreted then and used to build the Markov model. They also understand states of the vehicle and define the possible state transitions. If an unexpected state transition is detected that means an anomaly in their model.

In [10] M. Marchetti et al. showed that anomaly detection can be efficiently performed based on CAN ID sequences. From the CAN traffic they only use the ID field of the messages. They build a transition matrix to understand the connection between messages. If during normal traffic an ID follows another then this transition is marked as normal in the matrix. Their anomaly detection method analyzes whether a not allowed transition appears in the traffic.

In another paper Taylor et al. [5] presented an anomaly detection approach that is based on repetition times of the messages on the CAN bus. They first splitted the traffic into flows. For every flow various measures are calculated such as the number of packets in the flow, the average Hamming distance between successive packet data fields and the average time difference between successive packets. During their analysis they show that the only reliable parameter for anomaly detection is the average time difference between successive packets. They use a one-class support vector machine (OCSVM) to classify the benign traffic and to detect anomalies. They measure the efficiency of their work only on syntactically generated traffic.

Although anomaly detection on compressed traffic has several advantages, this idea was not researched so far. We aim to close this gap by analyzing normal

and attacked compressed CAN traffic to determine what kind of anomalies could be detected with this approach.

## 3 Technical Background

### 3.1 CAN protocol

In modern cars the ECUs are controlling several processes. They measure their surroundings and according to the available information they perform operations. They are connected together with the CAN bus and communicate by its protocol, the CAN protocol, which uses CAN messages. In the protocol there is no authentication, and broadcast is used, so every ECU gets every message and selects which interests it. That is, all traffic is visible to everyone and any controller can send any type of message. The above mentioned attributes make the CAN vulnerable against several attacks. For example an attacker can easily send arbitrary messages once he gained access to the inner network.

A CAN message has the following format. Every message has an ID which can be 11 or 29 bits long. The meaning and the range of the IDs are manufacturer specific. The lower the value of the identifier field the more prior is the message. After the ID comes the data length field then comes the data.

```
1481492674.734327  0x260  8  00 00 00 00 00 00 00 6a
1481492674.736055  0x2c4  8  05 c8 00 0f 00 00 92 3c
1481492674.738092  0x2c1  8  08 03 35 01 6a d9 00 4f
1481492674.754306  0x260  8  00 00 00 00 00 00 00 6a
1481492674.759605  0x2c4  8  05 c8 00 0f 00 00 92 3c
1481492674.769823  0x2c1  8  08 03 39 01 70 d9 00 59
1481492674.774302  0x260  8  00 00 00 00 00 00 00 6a
1481492674.783129  0x2c4  8  05 c2 00 0f 00 00 92 36
1481492674.794246  0x260  8  00 00 00 00 00 00 00 6a
1481492674.801541  0x2c1  8  08 03 3b 01 74 d9 00 5f
```

**Example 1.1.** Simplified CAN traffic log

In Example 1.1 a CAN traffic log is shown. Each row corresponds to a message. The first column is the arrival time of the message (Unix time), the second column is the message ID, in the third column there are specific flags (which in our captured data are not used), the fourth column shows the length of the data in the message, and the last column is the data.

### 3.2 CAN compression algorithm

We used a recently published semantic compression algorithm [4] made specifically for CAN traffic compression. It achieves a good compression ratio on CAN traffic, exceeding other state of the art syntactic compression algorithms, such

as gzip. It is a lossless compression method which is a necessary requirement for being able to use it in forensic investigations after an incident.

```
0x260
start_time:1481492674.734327
period:19984
00 00 00 00 00 00 00 6a:  0#0,1#-5,1#12,1#-40,
                              1#-3,1#105,1#-87,
                              1#-16

0x2c4
start_time:1481492674736055
period:23540
05 c8 00 0f 00 00 92 3c:   0#0,1#10
05 c2 00 0f 00 00 92 36:   1#-16, 1#20
05 c5 00 0f 00 00 92 39:   1#111
05 c8 00 0f 00 00 92 3c:   1#-113, 1#-22

0x2c1
start_time:1481492674738092
period:31728
08 03 35 01 6a d9 00 4f:  0#0
08 03 39 01 70 d9 00 59:  1#3
08 03 3b 01 74 d9 00 5f:  1#440, 1#14
08 03 38 01 72 d9 00 5a:  1#-5
```

**Example 1.2.** Compressed CAN traffic log

The compression algorithm works as follows.

1. First, it separates the messages according to their message ID-s.
2. For each message ID, it records the time of the first message in the recorded traffic. (See 'start_time' in Example 1.2.)
3. It calculates the average time between messages with the same ID-s.
4. For a given ID, it sorts the messages into groups according to their data, so messages with the same data go into the same group.
5. Store the number of elapsed periods and the difference between the period based and the actual time stamp.

It is unnecessary to store the time for each message in every group, because the periodic nature of the bus arbitration can be exploited. Thus, it is enough to store how many periods elapsed since the previous message (in the group) and the difference from it in microseconds. For example 1#440 means that, after the previous message 1 period and 440 microseconds elapsed.

## 4 CAN attacks

An attacker could try to interfere with the normal operation of the CAN bus in multiple ways depending on the malicious intent. It is possible to achieve an

anomaly with just a few messages but in most cases to make an attack reliable a large number of messages are necessary. In the following paragraphs we describe the various possibilities of an attacker organized by the number of messages required. In the second part of this section we describe how we recorded infected CAN logs for this research.

### 4.1 Taxonomy of CAN attacks

**DOS against the CAN bus:** In this scenario the goal of the attacker is to completely disable the communication on the CAN bus. This can be achieved at least with two extreme approaches.

An attacker could disturb the transmission of every CAN packet by starting its own dummy transmission in the middle of every other packet. This way an error will occur during the reception of every packet. This attack does not need a full packet to be sent by the attacker just a few bits with the correct timing.

Similar effect can be achieved with the transmission of packets with the ID 0. The ID field of the CAN packet is also determines the priority of the message. The value of the ID decides which packet can be transmitted in case of multiple colliding packets. The smaller the ID of a packet is the higher its priority is. If an attacker sends continuously packets with the ID 0 then there won't be any resource left for the normal traffic.

Both of these attacks are operation critical for a vehicle. A complete DOS against the CAN bus isolates the ECUs from each other disabling most of their operations. These scenarios are trivial to detect but very difficult to handle.

**Messages with new IDs:** It is common in car manufacturing that the same hardware parts are used in various car models. This practice makes it possible for an attacker to try to trigger functionality in a car that would not be used otherwise. On the CAN level this means that messages could appear with previously unseen IDs.

Some attacks are realized with the usage of debug packets[1]. These scenarios also introduce packets with new IDs on the bus.

If all benign IDs are known in advance, identifying these attacks is simple. Messages with IDs not seen before can effectively be found with basic white-listing or simple anomaly detection.

**Irregular messages with known IDs:** Some CAN messages are only transmitted as a response to certain events. These messages are encountered rarely because they are responses to environmental changes and are not part of the regular operation of a vehicle. An attacker could inject any of these messages at random times to force an inconsistency in the operation.

Without an external source of information the only way to detect these messages is to correlate information from other packets. This is a challenging task in most cases if even possible.

**Messages with regular repetition times:** To interfere with the normal operation of a vehicle the regular communication of the ECUs should be altered. It is hard to remove a messages from the CAN bus (if en error occurs during transmission usually a re-transmission logic is triggered at the sender) thus the best possible option for an attacker is to send malicious packets additionally. A packet with fake content could force the vehicle into a compromised state until the next packet with correct content arrives. Most attacks require to keep the vehicle in a compromised state for the majority of the time. This means that the attacker is required to send a lot of malicious packets to minimize the effect of the original benign traffic.

Based on the goal of the attacker the frequency of the malicious packets could be anything between 1x and 10x of the original traffic. Our measurements and previous research[1] results also showed that a malicious traffic with ∼10x the frequency of the original traffic forces the vehicle to stay in a compromised state almost constantly.

### 4.2 Realized attacks

We used a test vehicle to demonstrate some of the attacks described previously. It allowed us to test our anomaly detection approach in a real life scenario as well.

**Speed indicator modification:** In this attack we were able to change the displayed speed of the vehicle. We achieved that even when the car was standing still without the engine running.

We performed this test with different attack frequencies. In the first attempt the frequency of the forged packets was the same as the original one effectively doubling the number of packets with the given ID. This caused the speed indicator to oscillate between the real speed (0 km/h) and the forged speed (30 km/h).

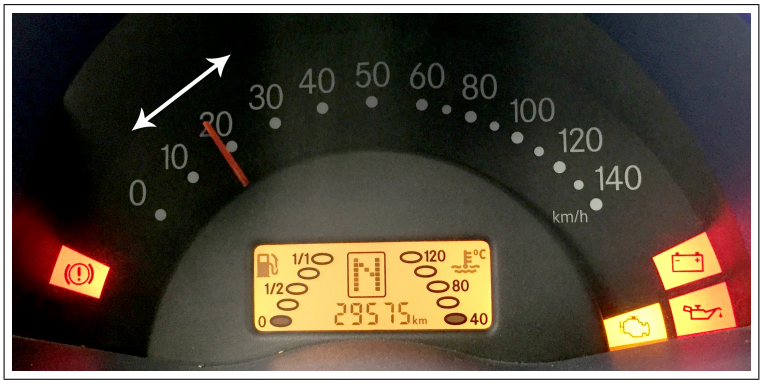


**Fig. 1.** Speed indicator attack with 1x frequency caused oscillation of the indicator needle.

In our second attempt we increased the frequency of the malicious packets to 10x the frequency of the normal traffic. This created a stable attack where the indicator showed continuously the speed defined by our attack.

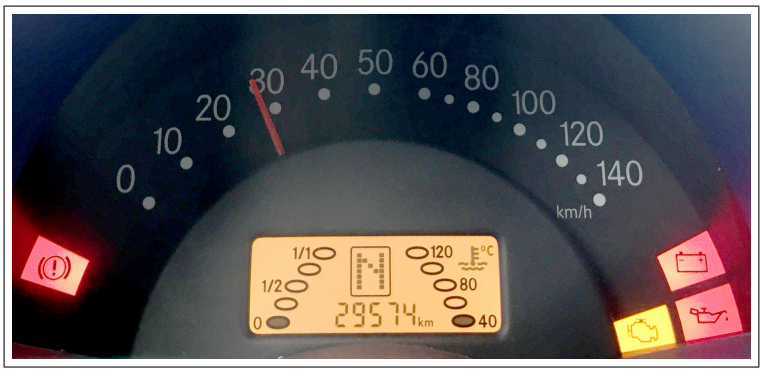


**Fig. 2.** Speed indicator attack with 10x frequency. The indicator shows 28 km/h while the real speed was 0 km/h.

**Transmission dashboard modification:** We also attacked the transmission signal for the dashboard (Figure 3). The engine was still not running but we were able to force the display to show that the vehicle was in gear 1.

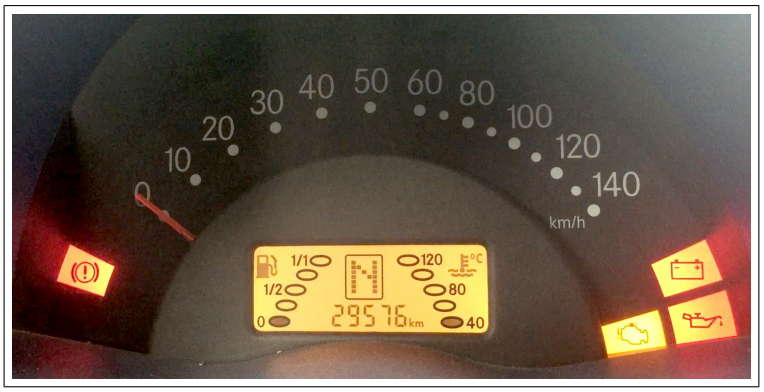


**Fig. 3.** Original state of the transmission display.

To achieve this goal we used a packet observed during previous test drives. The malicious packet injection frequency was also 10x of the original rate. As a side-effect we also modified the fuel level indicator and some control lights from the engine. In the original state the fuel level was low whereas during the attack

it showed that the tank is half full (Figure 4). This indicates that the fuel level and some of the control signals are transmitted in the same packet as the current gear.


**Fig. 4.** Attack on the transmission display. The engine was not running but the indicator showed gear 1. The control lights were switched off and the fuel level was increased.

## 5 Anomaly detection

### 5.1 Data sets

During the research we created two data sets. First, we created a synthetic data set where the attacks were manually injected into a clean CAN traffic log. Then we also performed some attacks against a real vehicle that gave us real life infected traffic logs.

**Synthetic data set:** We have captured a few hours of benign traffic from a mid class vehicle. With reverse engineering we found the signal used to display the RPM of the engine on the dashboard. We used this signal during our attacks to simulate an attack where false information is displayed to the driver. The RPM value is sent by an ECU in a message with the ID 110. Normally this message is send in every 10 millisecond. This attack belongs to the "Messages with regular repetition times" category described in Section 4.

We created a packet with a malicious content to insert into the traffic. The packet contained a higher RPM value than found in normal traffic.

We generated the malicious traffic with multiple steps. First we splitted the normal traffic into smaller chunks. Each chunk contained approximately 1 minute of traffic. As a base rule we decided that every attack should be at least 5 seconds long because a shorter attack on the dashboard would probably not disturb the driver thus it would not achieve any goal. We also generated longer attacks. For each attack scenario we increased the attack length with 5 seconds. This resulted

in attacks with random length in these intervals: 5-10; 10-15; 15-20; 20-25 and 25-30 seconds.

For every attack scenario we generated 100 malicious samples. They were each tested in our algorithm together with 100 benign samples.

We generated the malicious traffic simulating the normal operation of the CAN bus (including the bus arbitration). First, we generated 10000 of the malicious packets. The time stamp of the first packet was randomly chosen from the first half of the benign sample, the rest of the time stamps were calculated based on the chosen attack frequency. Then, we merged the benign and the malicious packets according to the time stamps. If two messages overlapped than the one with the higher time stamp was shifted after the other. If there was enough time until the next message then the shifted message was simply inserted. Otherwise the same logic was repeated again to resolve further conflicts in the time stamps. Generally, the bus load was relatively low in our test vehicle resulting a low number of those conflicts.

Once we had the 100 malicious and 100 benign samples for every scenario we compressed all of the logs with the chosen compression algorithm.

Furthermore we examined how the detectability of such an attack changes with the modification of the message injection frequency. We generated attacks where the injection frequency was 10 times, 5 times and 2 times higher than the original frequency of the given ID. We considered the 10 times higher frequency the default frequency for a flooding attack as our real life tests and other researchers also demonstrated it is an adequate frequency for an attack to have a stable effect.

In our captured traffic there are 18 different IDs. There are IDs with regular (14) and irregular (4) repetition times. We only focused on the regular IDs.

**Real life data set:** We implemented CAN attacks on a vehicle with real impact. We targeted both the speed and the transmission indicator. For the speed indicator we used 3 different attack frequencies: $\sim$10 times higher, 2 times higher and the exact same frequency as the original messages have. For the transmission indicator we also used a frequency 10 times of the original. We also collected benign traffic from the vehicle to compare it to the malicious logs.

### 5.2 Our anomaly detection algorithm

Our detection algorithm has the goal to decide whether a given message in the compressed CAN traffic log belongs to an attack or not. To address this, first we split the compressed log into separate ID files, where each file contains messages of a given ID. These files are analyzed separately.

We calculated different features of the malicious and benign logs to find the ones that distinguish them the most efficiently. Although, the changes of the repetition times had a significant impact on the structure of the compressed traffic log, the simplest and most powerful feature turned out to be the number of messages during a constant time window.

**Number of messages per minute:** In a time window of 1 minute we count the number of messages for each ID. Thus we get a feature for each ID: the number of messages in a minute. This will be different in a normal and an attacked traffic log. This approach is also intuitive. If we inject additional messages of an ID that has an approximately constant message rate per minute, the increase in the message rate per minute will indicate an attack.

This feature proved to be reliable for attacks both with higher and lower frequencies.

**Attack detection:** Based on the previously suggested feature, attacks can be detected efficiently. As can be seen in section 5.3, this approach separates malicious traffic logs from benign logs even visually making the decision easy.

### 5.3 Results

We evaluated our method on both synthetic and real life data with different attack frequencies.

On synthetic data we used the above mentioned 100-100 normal and attacked samples for attacks with different frequency. The histogram of the distribution of the attacks can be seen in Figure 5 and Figure 6. They demonstrate that the attacked traffic is efficiently distinguishable from the normal traffic even when the attack frequency is as low as 2 times of the original.

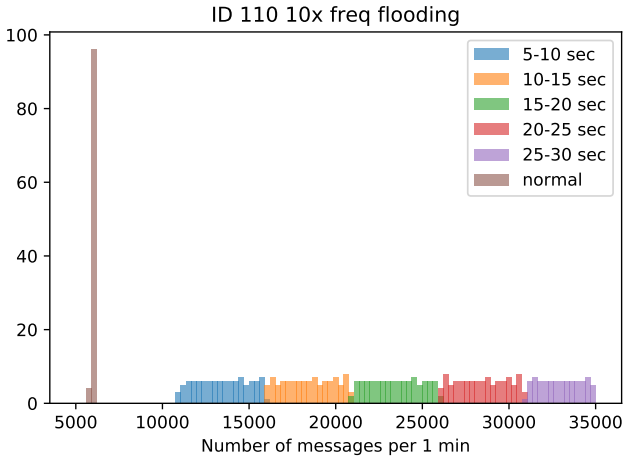


**Fig. 5.** The deviation of the number of messages per minute feature for 100 - 100 samples at 10x frequency (synthetic attack).

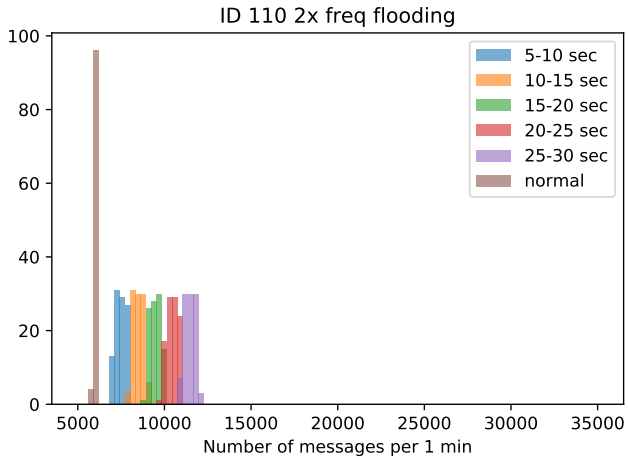


**Fig. 6.** The deviation of the number of messages per minute feature for 100 - 100 samples at 2x frequency (synthetic attack).

On the data from the real world attacks we performed the same calculations. Figure 7 and Figure 8 show that our algorithm achieves the same reliable results in the real life scenarios as well.

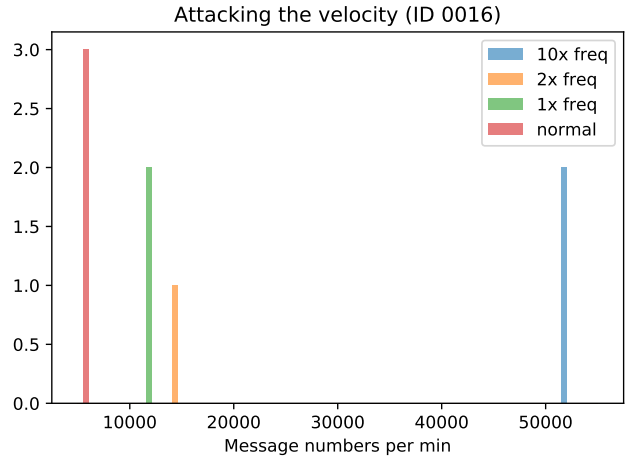


**Fig. 7.** Real attacks on the speed indicator. Comparison of the number of messages in normal and attacked scenarios.

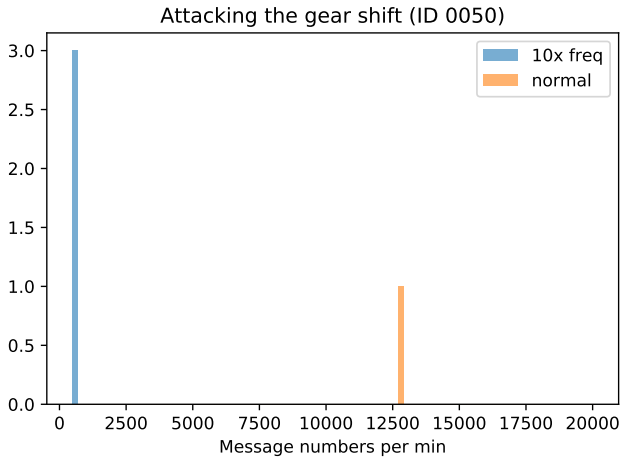


**Fig. 8.** Real attacks on the transmission indicator. Comparison of the number of messages in normal and attacked scenarios.

These results show that with this approach it is possible to achieve correct classification in every case. For the stable attacks, where a high message frequency is used, the proposed method produces a reliable result with 0 false positive and false negative rates. As the message injection rate decreases the confidentiality is also reduced but even for attacks with 1x injection frequency it remains high enough for a correct decision.

## 6 Conclusion

In this paper, we argued that cyber attacks on vehicles may cause physical accidents, therefore, forensic investigations must be extended into the cyber domain. In order to support this, CAN traffic in vehicles must be logged continuously and stored efficiently for later analysis. Our main contribution in this paper was a novel anomaly detection method that works on compressed CAN traffic logs. The advantage of running anomaly detection on the compressed logs is that less amount of data needs to be analyzed, hence, the efficiency of forensic investigations can be increased.

Our anomaly detection algorithm is based on analyzing the average frequencies of messages with given CAN IDs. The compression algorithm that we use preserves the number of messages per unit time in an easily extractable form in the compressed CAN log, which makes it possible to use our anomaly detection algorithm on the compressed logs. We demonstrated that this approach works reliably in a range of scenarios, including using data sets captured in real vehicles and modified with synthetically generated attacks as well as data sets captured

in real vehicles under real attacks. Our algorithm was capable to identify attacks is both cases.

Observing the average frequencies of messages with given CAN IDs may appear to be a simplistic approach for anomaly detection; nevertheless, it works reliably for detecting injection attacks. In addition, many prior works suggested that injection attacks are easy to carry out and they have noticeable effects, hence, this type of attack is one of the most important attacks to consider. Whether our method of analyzing the compressed logs can be adapted to other types of attacks, where message frequencies are not changed, is an open question and subject of our future work.

## 7 Acknowledgement

The work presented in this paper was partially supported from the grant GINOP-2.1.1-15. The project has been supported by the European Union, co-financed by the European Social Fund. EFOP-3.6.2-16-2017-00002.

## References


1. C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units", IOActive Labs Research, Tech. Rep., Aug. 2013. [Online]
2. K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachm, and S. Savage, Experimental security analysis of a modern automobile, 2010, pp. 447462.
3. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, Comprehensive experimental analyses of automotive attack surfaces, in Proceedings of the 20th USENIX Conference on Security, ser. SEC11. Berkeley, CA, USA: USENIX Association, 2011.
4. A. Gazdag, L. Buttyan and Z. Szalay, "Efficient lossless compression of CAN traffic logs," 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2017.
5. A. Taylor, N. Japkowicz and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," 2015 World Congress on Industrial Control Systems Security (WCICSS), London, 2015, pp. 45-49.
6. H. M. Song, H. R. Kim and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network" 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, 2016, pp. 63-68.
7. C. Miller and C. Valasek, Remote exploitation of an unaltered passenger vehicle. Black Hat USA, 2015
8. A. Taylor, S. Leblanc and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 130-139.
9. E. Evenchick, "Hopping On the CAN Bus", Black Hat Asia, 2015
10. M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 1577-1583.
11. S. N. Narayanan, S. Mittal and A. Joshi, "OBD_SecureAlert: An Anomaly Detection System for Vehicles," 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, 2016