Andras Gazdag - Levente Buttyan - Zsolt Szalay*

FORENSICS AWARE LOSSLESS COMPRESSION OF CAN TRAFFIC LOGS

In this paper, we propose a compression method that allows for the efficient storage of large amounts of CAN traffic data, which is needed for the forensic investigations of accidents caused by the cyber-attacks on vehicles. Compression of recorded CAN traffic also reduces the time (or bandwidth) needed to off-load that data from the vehicle. In addition, our compression method allows analysts to perform log analysis on the compressed data. It is shown that the proposed compression format is a powerful tool to find traces of a cyber-attack. We achieve this by performing semantic compression on the CAN traffic logs, rather than the simple syntactic compression. Our compression method is lossless, thus preserving all information for later analysis. Besides all the above advantages, the compression ratio that we achieve is better than the compression ratio of the state-of-the-art syntactic compression methods, such as zip.

Keywords: *CAN, network traffic capture, semantic compression, forensic analysis*

1. Introduction

Modern vehicles have multiple embedded controllers, called ECUs (Electronic Control Units), connected together by internal communication networks such as the CAN bus (Controller Area Network). The ECUs are programmable devices, and many of the vehicles' functions now rely on software running on them, as well as on protocols for exchanging information between the ECUs via the CAN buses. Often, vehicles also have interfaces, such as the OBD (On-board Diagnostic) port and various wireless interfaces that make it possible to access certain parts of the vehicle's internal network from the outside. Such access may be needed for diagnostic and maintenance purposes, for connecting the mobile consumer devices to the entertainment unit of the vehicle, or allowing for the transfer of various sensor data in and out of the vehicle. The concept of *connected cars* goes even further by introducing short range wireless connections between vehicles and to the Internet infrastructure, enabling new types of safety and infotainment applications.

All this development means that modern vehicles should be considered as the cyber-physical systems, in which special purpose computers control physical processes, and those computers are no longer isolated from the cyber space out there. This introduces an entirely new domain of problems for vehicles, and road safety in general: the domain of *cyber security*. Indeed, the ECUs in vehicles can be compromised in similar ways as computers are compromised on the Internet (e.g., exploiting a buffer overflow vulnerability in their software), and due to the increasing level of connectedness, such attacks are now possible to be carried out remotely (i.e., without requiring physical access to the vehicle). The feasibility of such remote attacks has been demonstrated by various research groups recently, showing also their potentially catastrophic consequences [1- 3].

The possibility of remote cyber-attacks on vehicles generated a lot of interest in developing protection measures that either prevent or detect such attacks. One of the proposed approaches is to perform the log analysis on recorded CAN traffic and identify intrusions either in real-time or in an off-line manner. While real-time intrusion detection seems to be the ultimate goal, off-line analysis of the logged CAN traffic would still be required for better understanding of how an attack worked and for forensic purposes in the case that the attack caused some physical damage.

Being able to detect and analyze cyber-attacks on vehicles requires continuous collection and recording of the CAN traffic. This can potentially lead to a large amount of data that need to be stored in the vehicle. In this paper, we propose a compression method that allows for the lossless, yet efficient storage of that large amount of data. Compression of the CAN traffic logs has

* [1]Andras Gazdag, [1]Levente Buttyan, [2]Zsolt Szalay
[1]Laboratory of Cryptography and System Security, Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics
[2]Department of Automotive Technologies, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics
E-mail: agazdag@crysys.hu

other notable advantages: it helps to shorten the time required to off-load the data from the vehicle. Moreover, the large amount of data is not only a problem for storage and communication; it also makes forensic analysis hard, resource intensive and time consuming. Our compression method allows analysts to perform the log analysis on the compressed data, therefore, it contributes to reduced analysis time and effort. We achieve this by performing *the semantic compression* on the CAN traffic logs, rather than simple syntactic compression. Besides all these advantages, the compression ratio that we achieve is better than the compression ratio of the state-of-the-art syntactic compression methods, such as zip.

The syntactic compression methods operate on the low level byte stream representation of the data. In contrast to this, the semantic compression methods interpret the data being compressed and take advantage of its semantic understanding. The semantic compression has generated considerable interest in the recent years. It has been successfully applied in different fields such as general database compression [4], video compression [5] and virtual machine memory compression [6]. What we propose in this paper is a new application area for it.

The rest of the paper is organized as follows: In Section 2, we give an overview on crash data recorders and show that they are not appropriate for supporting forensic analysis of cyber-attacks. In Section 3, we provide background information on the CAN technology. We describe our new semantic compression algorithm in Section 4, and we evaluate its performance in Section 5. In Section 6, we show that the proposed format is a useful approach to find traces of an attack. Finally, in Section 7, we conclude the paper.

2. Crash data recorder devices

Data recording devices that can capture information continuously or triggered by an event have existed in the transportation industry for decades. The best known such devices are probably the "black boxes" used in aviation to record data that can be used by investigators to reconstruct some of the circumstances of an airplane crash. Such recording devices now also exist in road vehicles: since September 2014, the so-called Event Data Recorder (EDR) is mandatory for every new passenger car and new light commercial vehicle (LCV) in the US. The purpose of the EDR devices is to collect data about the vehicle dynamics and the vehicle status that enable better accident reconstruction. It helps in validating insurance claims, encourages safer driving behavior and extends the scientific knowledge about real accidents, thus, resulting in safer vehicle design.

The European answer to EDR is the so called eCALL system that will be introduced in all new cars as of April 2018 in the EU. The major difference between the two systems is that EDR devices are offline systems, requiring a later data retrieval, while the eCALL is an online system that immediately calls the ambulance (dials 112) in the case of emergency.

The importance of an EDR-like "black box" increases with the deployment of highly automated functions in road vehicles, as there must be some objective evidence proving who was in charge of control in the vehicle in a critical situation. Due to the previously mentioned properties, the EDR devices in their current form are inappropriate for supporting forensic investigations related to cyber-attacks. The European eCALL system does not even record CAN data, so it clearly cannot be used in the case of cyber incidents.

There exist data recording devices, such as tachographs, that perform continuous data collection in vehicles. Tachographs are mainly used on heavy trucks, buses, and emergency vehicles to continuously record certain parameters of the vehicle such as its speed, its engine RPM and odometer values. Yet, the main purpose of tachographs is to monitor the duty status of the drivers of commercial vehicles and they are not designed to record the raw CAN traffic. They usually record only a few vehicle parameters with a certain recording frequency and they are not available on all kinds of road vehicles. Hence, similar to the EDRs, tachographs in their current form cannot really be used in investigations of cyber incidents affecting vehicles.

Hence, we can conclude that, although they have seemingly similar goals, existing data recording devices in road vehicles actually address a problem different from the one that we address in this paper and they are not appropriate for cyber incident investigations.

3. Background – the CAN protocol

The CAN protocol was designed to be simple, causing only a small overhead in the communication. None of the messages contain any authentication information. This nature of the protocol makes it very easy to spoof CAN messages, which, in turn, can lead to numerous other attacks based on the injection of arbitrary messages into the CAN traffic. Understanding the CAN messages, however, is a more complex problem, because it requires a priory knowledge of the network and the communicating parties.

In vehicles, the ECUs communicate with each other mostly periodically. While the communication is event based, regular repetition times enable a quite accurate prediction of the upcoming pattern of messages. Another main property of the traffic is that the content of messages is often repeating.

We confirmed these properties via capturing traffic in real vehicles. Our test vehicles, however, were not premium category vehicles. In premium category vehicles, there are more ECUs, which results in a higher variety of the CAN message types and message contents. Also, the same high variety is expected in autonomous vehicles. Yet, we believe that even in those cases,

1481492674.734327	0x260	8	000000000000006a
1481492674.736055	0x2c4	8	05c8000f0000923c
1481492674.738092	0x2c1	8	080335016ad9004f
1481492674.754306	0x260	8	000000000000006a
1481492674.759605	0x2c4	8	05c8000f0000923c
1481492674.769823	0x2c1	8	0803390170d90059
1481492674.774302	0x260	8	000000000000006a
1481492674.783129	0x2c4	8	05c2000f00009236
1481492674.794246	0x260	8	000000000000006a
1481492674.801541	0x2c1	8	08033b0174d9005f
1481492674.806689	0x2c4	8	05c2000f00009236
1481492674.814227	0x260	8	000000000000006a
1481492674.83034	0x2c4	8	05c5000f00009239
1481492674.833283	0x2c1	8	08033b0174d9005f
1481492674.834316	0x260	8	000000000000006a
1481492674.853767	0x2c4	8	05c8000f0000923c

Figure 1 Simplified CAN traffic log

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

Figure 2 A compressed message flow example

the CAN traffic is rather regular and exhibits features that can be exploited for its efficient compression.

4. The compression algorithm

The usage of the semantic compression and syntactic compression helps to achieve different goals. A clever combination of the two approaches could benefit from the advantages of both: the semantic compression reduces the file size while maintaining accessibility to the data, whereas the syntactic compression achieves the smallest possible file size.

To exploit the benefit of both approaches we propose to apply both methods at different operational phases. During the data collection, an on-the-fly semantic compression could reduce file sizes while keeping data available for immediate processing. The compressed data format allows a fast analysis of data flows because they are stored in blocks one after the other, whereas investigation of causality relations is more computing-intensive.

An optional long-term storage or cloud transfer of network logs requires the smallest file size, while the importance of immediate data accessibility is reduced. This implies the use of the syntactic compression at this phase.

We propose a compression algorithm that takes advantage of the largely periodic nature of the CAN traffic. The high-level approach of our algorithm is to separate the traffic into message flows, containing only messages that have the same ID, and then, compressing each message flow separately leveraging the previously identified properties.

The first step is done by filtering messages based on the ID field of the protocol. This step generates separate lists of messages where the only remaining information for each message to be stored is its timestamp in the log and the data content of the message. Storing a complete and separate timestamp for each message in a flow would be a waste of storage. Our more efficient approach takes advantage of the periodicity of messages. Theoretically, a new message with the same ID should come at an exactly predictable time point based on the inter-arrival time of this message type. However, this behavior can be changed by a higher priority message on the CAN bus. If the two messages are sent at the same time, then only the one with the higher priority will be sent, shifting the inter-arrival time of the messages with the lower priority. From this point on, this complete message flow will be shifted.

An efficient way to store the timestamp of a message is to store the number of periods (specific for that flow) passed since the last message of the same type and an additional offset value that is induced by either priority causes or measurement distortions. For each message flow, there are some additional metadata to be stored: the message ID, the first appearance of the flow in the log and the characteristic period length of the flow. These flow specific metadata should be followed by the message data and then the compressed timestamp for each message. An example of this compressed format can be seen in Figure 3, where the # sign separates the period number and the offset value in each compressed timestamp.

The operation of our semantic compression can be effectively demonstrated in Figure 1 that shows a simplified CAN traffic log. It has been truncated and reduced to only contain messages from three different ID types. Other than that, it is a real life traffic log.

In the first step the algorithm reads the messages separating them into groups with the same ID. In this case, it would result in 3 groups: 0x260, 0x2c4 and 0x2c1. The following step is the same for each group; that is compressing messages inside a group.

An efficient way to find repeated messages is to build a hash map of the messages using the message data as a key. At this point, the only remaining information to be stored is the arrival time of the message.

For a more efficient compression the timestamps are stored in a coded way taking advantage of the CAN traffic properties. The

Table 1 The semantic compression

Test case	Original trace file size	Text format	Binary format	Compression ratio for text	Compression ratio for binary
1	10095971	1710920	1090757	16.94 %	10.80 %
2	7040165	1334902	835539	18.96 %	11.86 %
3	19143383	3747229	2307146	19.57 %	12.05 %
4	21936245	4233994	2601354	19.30 %	11.85 %

Table 2 The semantic and Syntactic compressions combined

	Original trace file size	Original file compressed	Text format compressed	Binary format compressed	Compression ratio for text	Compression ratio for binary
1	10095971	1291315	546725	499998	5.41 %	4.95 %
2	7040165	937319	429234	390467	6.09 %	5.54 %
3	19143383	2569118	1194758	1092183	6.24 %	5.70 %
4	21936245	2895039	1332585	1223677	6.07 %	5.57 %

inter-arrival times of the messages can be calculated, based on the stamps, requiring only to store the small difference between the predicted and the actual arrival times.

It is possible, that a message data appears in the traffic from time to time. This also has an impact on the compression, i.e. we need to store the elapsed number of periods in every case, as well. This number usually has the value of 1 but for a recurring data this may vary.

The final result of the compression of this log can be seen on Figure 2. For storing the compressed timestamp the number of cycles and the arrival shifts are separated with a # sign.

We defined two equally lossless output formats for our algorithm. One is a text base (ASCII) representation of the traffic log, while the other is a binary format.

5. The compression Evaluation

We evaluated our algorithm in terms of performance and efficiency. As the most important performance metric, we calculated the compression ratio and as for efficiency, we also measured the speed of our implementation. We performed our measurements multiple times with different datasets originating from different vehicles. We used vehicles of three different brands all belonging to the low mid-level category built between 2005 and 2010.

We captured traffic with a Raspberry Pi based CAN interpreter. It allowed us to access the raw information on the CAN bus, and we saved every CAN message with a timestamp. We performed traffic captures through the OBD interface where the design of the vehicle allowed for an uninterrupted access to the powertrain CAN bus traffic through this connection. We were able to gather traffic logs of multiple hours in all three types of vehicles. The algorithm was capable of efficiently compressing data gathered during the test scenarios in every single case at least a magnitude faster that the incoming speed, as shown in Table on the Semantic compression. This speed makes our algorithm a good candidate for an on-board data compression for local usage of the information or as a preparation for a remote transmission.

The measured compression ratios show significant progress in the data sizes (Table 1 and Table 2). We were able to achieve the compression ratios of less than 20 % using an ASCII representation of the output of our algorithm. The binary representation shows an even more efficient compression with the results being around 10 % of the original file size.

If we applied the additional syntactic compression to our semantic compression it resulted in the smallest file sizes we were able to achieve. In the ASCII representation scenario, the combined result shows an approximate 6 % compression ratio while the binary case shows an approximate 5 % compression ratio.

This result can be considered as another proof that it is worth applying the semantic compression before the syntactic one, because with this combination the additional efficiency can be gained.

We validated the lossless property of our algorithm by checking that the SHA-256 hash of the original data matches that of the data that we restored after de-compression.

6. The forensic use of the compressed format

In the recent years several articles have been published about the vehicle security. Perhaps the largest impact was achieved by the papers of Koscher at al. [1] and Miller et al. [7]. These papers described a series of attacks against the vehicle CAN busses using different attack approaches. Their attacks on a network level can be divided into two categories: sending already known messages with a frequency different from the usual frequency and inserting

```
id:0110
start_time:1483093132166605
period:9994
0200000000270000:3#-392,1#426,
0200000000260000:1#-348,1#-47,1#-22,1#369,1#-37,1#-301,1#44,1#299,1#-209,1#-81,1#-97,
027d000000200000:2149#-3321,0#999,0#999,0#999,0#999,0#999,0#999,0#999,
    0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,
    0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,
    0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999,0#999
```

Figure 3 The compressed attack traffic log

messages with a previously unused message ID. Based on our research, our proposed CAN compression format can also be helpful to find attacks described in these works.

The first type of the CAN attack inserts new messages with a known message ID. The purpose of this attack is to flood the CAN bus with a modified message data suppressing the information sent in the original messages. This approach was used, for example, to modify information displayed to the driver. The appearance of the original message on the CAN bus cannot be prevented leaving the only option of sending the crafted messages with a much higher frequency (up to 10-20 times the original value).

The second type of the CAN attack inserts completely new, previously unseen messages into the traffic. In the presented works, those new messages were diagnostic messages. The goal of those messages was to trigger functionalities of the car that would otherwise be turned off. As an example, it is possible to use the park assistant feature of certain cars during the normal driving circumstances to change position of the driving wheel.

Both of these attacks produce a very specific pattern in the CAN traffic that can be easily identified in the compressed format proposed in this work.

One of the properties of the CAN traffic, also utilized by our compression, is the highly regular arrival times. This property is harmed when a higher frequency flooding attack is inserted into the CAN communication. The proposed format represents arrival times in an "elapsed cycles # offset" format. During the normal operation, the value of elapsed cycles is the most probably 1 and the offset is a relatively small number. This behavior changes notably when a log file that includes a flooding attack is compressed. In this case, the value of the elapsed cycles is always 0 because the time between attacking messages is around 1/10 - 1/20 of the normal inter-message time. This also results in an offset field with a greater value than the offset values in the normal case. An example of a compressed attack traffic log can be seen in Figure 3.

Using messages, as part of an attack, with completely new IDs generates an entirely new section in the compressed format. The first step of the compression is to separate messages into discrete groups of messages with the same ID. This step makes it very easy to find attacks using new message IDs. The easiest way to find this discrepancy is to compare multiple compressed traffic logs originating from the same vehicle. This allows the analyzer to significantly reduce analysis time.

Finding anomalies in the compressed format relies on the fact that the properties of the traffic are determined based on a benign traffic period. This can be rather easily achieved because only a very short time frame is required to calculate this information. This calculation can be repeated periodically and the result should be the same every time. If that was not the case, that could also be an indicator that probably an attack happened in that time frame.

7. Conclusion

In this paper, we presented an efficient way to perform lossless compression of the CAN traffic logs. Based on our observations of the periodic properties of the CAN traffic, we designed a semantic compression algorithm for the CAN traffic. With the use of our algorithm, storage efficiency and communication costs can be significantly improved, while keeping the possibility to perform analysis on the compressed data.

As part of our future work, we plan to further optimize the file formats. Our current byte level approach could still be refined if we could represent the number of cycles and offset information in a more condensed way. Another possible follow up is to define a loss compression algorithm where every information is dismissed that is not important for a later forensic analysis. This approach could result in a dramatic reduction in file size.

Acknowledgement

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] KOSCHER, K., CZESKIS, A., ROESNER, F., PATEL, S., KOHNO, T., CHECKOWAY, S., MCCOY, D., KANTOR, B., ANDERSON, D., SHACHAM, H., SAVAGE, S.: Experimental Security Analysis of a Modern Automobile. IEEE Symposium on Security and Privacy, 2010.

[2] CHECKOWAY, S., MCCOY, D., KANTOR, B., ANDERSON, D., SHACHAM, H., SAVAGE, S., KOSCHER, K., CZESKIS, A., ROESNER, F., KOHNO, T.: Comprehensive Experimental Analyses of Automotive Attack Surfaces. Usenix Security Symposium, 2011.

[3] GREENBERG, A.: Hackers Remotely Kill a Jeep on the Highway - With Me in It. Wired Magazin, July 21, 2015.

[4] JAGADISH, H. V., NG, R. T., OOI, B. CH, TUNG, A. K. H.: ItCompress: An Iterative Semantic Compression Algorithm. Proceedings of 20th International Conference on Data Engineering, 646-657, 2004.

[5] MEI, T., TANG, L., TANG, J., HUA, X.: Near-Lossless Semantic Video Summarization and its Applications to Video Analysis. ACM Transactions on Multimedia Computing, Communications, and Applications, 9(3), article 16, 2013.

[6] RAI, A., RAMJEE, R., ANAND, A., PADMANABHAN, V., VARGHESE, G.: MiG: Efficient Migration of Desktop VMs Using Semantic Compression. USENIX Annual Technical Conference, 2013.

[7] MILLER, C., VALASEK, C.: Adventures in Automotive Networks and Control Units [online]. 2014. Available: http://bit.ly/IOAresources.