Real World Evaluation of a Novel Security Testing Environment for Vehicular Control Units via CAN Networks

Jürgen Wurzinger[1], Peter Priller[1], Aleš Kolar[1] and Markus Nager[1]

Abstract: Due to the introduction of networked entertainment and safety features in modern cars, vehicular communication systems are no closed networks anymore. As research demonstrated, based on these new attack surfaces existing safety and functional tests cannot fully satisfy security needs of modern cars. With this work we introduce an application called CAN Communication Tester (CAN-CT) which addresses this problem for the Controller Area Network (CAN).

CAN-CT is an application that makes use of hacking attacks to systematically inject, replay and invalidate messages. We show that we can successfully spoof messages, suppress all communication on the bus and transition electronic control units (ECUs) into error states. CAN-CT is capable of learning from the traffic on the bus and executes targeted attacks. By attacking an ECU the same way a hacker would, we are able to examine the implementation of protection mechanisms in an intuitive yet effective way. Furthermore this approach allows us to test the proper working of possible attack detection techniques as well.

Using Hardware-in-the-Loop Systems (HiLs) we tested CAN-CT with actual state of the art ECUs. We revealed vulnerabilities like lacks in the plausibility checking of messages or completely omitted examinations of the frame structure. Those flaws opened up serious threats to hackers. In exploiting those vulnerabilities we showed that we were able to take over message IDs owned by another ECU and hence achieve targeted manipulations.

Revealing weaknesses in real-world ECUs allowed us to demonstrate the applicability and impact of applications like CAN-CT for automotive systems. Our results highlight the need for security tests to complement traditional testing environments.

Keywords: CAN; Controller Area Network; Security Testing; Automotive; Car Hacking; Test Framework; Hardware in the Loop; ECU; Electronic Control Unit; Penetration Testing;

1 Introduction

Modern cars consist of up to 100 electronic control units (ECUs) communicating over various vehicular networks [Ch09]. The upcoming interconnection of state of the art vehicles with their environment introduced advances regarding safety and entertainment, while simultaneously opening up a large number of new attack surfaces as well. The prevalent security by isolation approach is broken and the internal communication systems are no closed networks anymore [Ko14].

Even though manufacturers use strong security measures to protect a vehicle's I/O channels from hackers, there is no 100 percent secure system. This fact was demonstrated by

[1] AVL LIST GmbH | Hans-List-Platz 1, A-8020 Graz

many researchers, who were able to exploit almost every interface of a car with its outside world [Ch11, VM15]. This circumstance stresses the need that besides the protection of those interfaces also the communication on internal vehicular networks has to be secured. This is especially true for the Controller Area Network (CAN), as it is the most widely used communication system for automotive applications [La13] and connects the most critical components in a vehicle.

For previous generations of vehicles, security measures and possible hacking attacks were hardly a concern. CAN communication was just tested for safety and functional requirements but not for malicious behavior. Nowadays, stronger emphasis is and will be put on more secure CAN communication [NSL13]. Yet due to the growing complexity of modern ECUs and in-vehicle communications it is very likely that flaws in the implementation of such safety and security measures exist. This implies that we need a testing environment to verify the proper working of these techniques. An important way to test implemented security measures is to simulate hacking attacks to complement traditional safety and functional tests.

With this work we introduce a tool named CAN Communication Tester (CAN-CT) that addresses this requirement for the CAN protocol in its current version 2.0. Making use of various hacking attack types allows us to detect implementation flaws of the CAN protocol's and partially also of the application layer's communication mechanisms. We are able to falsify messages, suppress all communication on the bus and transition ECUs into error states. Thus, the contribution of this work is two-fold. On the one hand we introduce an intuitive security testing approach with novel CAN specific attacks (see Section 4.2) and on the other hand we provide a detailed evaluation of the applicability of such a testing environment and its impact for real-world scenarios and commercial ECUs.

2 Related Work

In the last few years extensive academic as well as non-academic research was done to determine the vulnerability of vehicles against hacking attacks. Koscher et al. [Ko10] were one of the first to test this vulnerability comprehensively in a real-world scenario. Similar to Koscher et al. Hoppe, Kiltz, and Dittmann [HKD11] did a thorough examination of security threats to CAN buses. Besides a theoretical analysis of vulnerabilities and possible countermeasures, various attacks on the CAN bus were executed in a self-made testing environment. Yet, unlike our approach, no systematic bus off, spoofing of recurring messages, or Denial of Service (DoS) attacks (see Section 4.2) were carried out.

Other than these approaches we are not focusing on the exploitation of protection mechanisms or developing countermeasures; instead we are using CAN attacks as the basis for our systematical tests to detect possible vulnerabilities of the CAN communication.

Our attack-based testing approach differs from previously introduced functional or model-based testing frameworks by its *systematic and comprehensive execution of hacking attacks*. Even though model-based testing approaches like Marinescu's et al. [Ma14] validate internal functions in detail, they still cannot sufficiently simulate the malicious intelligence of a hacker. We believe that this is also true for other traditional testing methodologies.

Bayer et al. [Ba14] as well as Talebi [Ta14] outlined concepts similar to our's on a more theoretical level. While Bayer et al. discussed the topic of a systematical execution of automotive security evaluations and penetration tests, Talebi studied the field of security evaluations by looking at the CAN bus as a fully simulated environment using the simulation software CANoe[2]. However, to the best of our knowledge, a fully working prototype of such a testing environment has not been presented so far.

In October 2015, Jenik presented a black box testing approach based on a concept called exhaustive fuzzing [Je15]. Similar to this approach also CAN-CT can be used for black box testing; however, instead of fuzzing we learn from the traffic on the bus and execute sophisticated and targeted attacks. This allows us to examine not only the correct communication of ECUs but also to test potential attack detection mechanisms.

As a consequence, CAN-CT can be understood as a tool that helps to verify secure implementation of CAN communication. In a final version security testing environments like the CAN-CT shall be used in combination with security evaluation methodologies like the security framework for the automotive domain of Glas et al. [GGV14] or the "automotive security evaluation assurance levels" (ASEAL) introduced by Bayer et al. [Ba14].

3 Current CAN Communication Protection Mechanisms

Vehicular communication networks followed a security by isolation approach. As, however, this isolation is not existent anymore, new security techniques are needed. So far no generally accepted security measures for CAN communication exist. As car hacking research demonstrated, security through obscurity remains the prevailing approach to protect cars [VM15, MV13, Ko14].

Authentication mechanisms are only implemented for diagnostic purposes, like flashing the ECU [Ko14, MV13, VM15]. Koscher et al. state, in their examinations, that these mechanisms use weak keys, a fixed challenge (seed) and are both just 16 bits long [Ko10]. Miller and Valasek [MV13] showed that normal CAN communication is not protected at all. Besides the nondisclosure of the message IDs and the encoding of the content, some manufacturers use additional application layer cyclic redundancy checks (CRCs), alive counters and time out windows; however, no actual security measures are implemented.

Application layer CRCs use a simple calculation over data bytes and the CAN message ID to assure integrity of the message. *Alive counters* serve as a sequence number which increases with every sent message. This allows the recipient to determine whether the messages arrived in the right order or if a message was lost. Most vehicular ECUs rely on periodically incoming status messages with a fixed cycle. A higher bus load for instance may cause the transmitting ECU to not exactly adhere to this defined cycle time. This

[2] see `http://vector.com/vi_canoe_en.html`

circumstance makes it necessary for ECUs to also accept messages deviating from the expected time of receipt. This time tolerance to determine the validity of a cyclic message is called *time out window*.

As a consequence, the current ways to protect the CAN bus against unauthorized manipulation are the nondisclosure of message IDs and the proprietary encoding of data, the usage of mechanisms like application layer CRCs, alive counters and time out windows as well as a (weak) authentication for diagnostic applications [TAA14]. Besides those techniques, ECUs use plausibility checks to determine the correctness of a received message. These plausibility checks verify the validity of a received CAN signal by comparing it to associated signals generated by any sensor, sent by another ECU or calculated from related other values.

4 CAN Communication Tester (CAN-CT)

In this paper we introduce a software application prototype named CAN Communication Tester (CAN-CT) to efficiently test vehicular ECUs for security (and potentially safety) vulnerabilities.

CAN-CT executes hacking attacks on the CAN bus. Various kinds of attacks, like replay, spoofing or Denial of Service (DoS) attacks, are implemented, can be parameterized, combined and will then be sent on the bus. Our aim is to cause unexpected behavior of the targeted ECU and in further consequence to unearth flawed implementations of the CAN communication or of the application itself.

We use a PCAN-USB[3] as interface for CAN-CT to the CAN bus of the device under test. This adapter in combination with the *basic* library provided by PEAK-System[4] allowed us to read and write CAN messages from any PC.

For CAN-CT's architecture we made use of known software design patterns to avoid dependencies on the used hardware and to clearly separate logic from view and data. The resulted modular design allows for an easy extensibility regarding further attacking types, specific higher layer protocols and possible future protection techniques. We are able to achieve a timing accuracy of our attacks of less than 100 μs. Furthermore CAN-CT is capable of analyzing the CAN traffic in-depth and thus, provides the user with information about protection mechanisms in use and interesting targets on the bus.

4.1 Attack Scenario

The underlying attack scenario of CAN-CT is based on recent car hacks like of Valasek and Miller as well as of Checkoway et. al [VM15, Ch11]. It is best described by understanding CAN-CT as a compromised ECU. It assumes that we, as an attacker, already got full

[3] see `http://www.peak-system.com/PCAN-USB.199.0.html?&L=1`
[4] see `http://www.peak-system.com/PCAN-Basic.239.0.html?&L=1`

access to the CAN bus, by, for instance, exploiting a vulnerability in an ECU with both CAN access and an external interface (e.g. Bluetooth), thus acting as gateway. This means CAN-CT is capable of reading and writing arbitrary CAN messages. As the CAN bus is a broadcast medium, deleting messages between two nodes is not possible. Furthermore as we are not reprogramming the CAN controller of the compromised ECU, all attacks have to comply with the CAN protocol.

4.2 Attack Features

To provide a complete security testing environment, we focused on four main tasks: *monitoring the CAN bus, analyzing the traffic, attacking ECUs* and *simulating CAN nodes*.

Besides the possibility of monitoring the messages and events on the CAN bus, CAN-CT can also be used to simulate other ECUs. By connecting additional CAN controllers, we can specify details of the CAN communication of each node. Based on this description CAN-CT then sends the CAN messages accordingly. Thus, it is possible to build customized CAN networks, mainly needed for basic testing purposes.

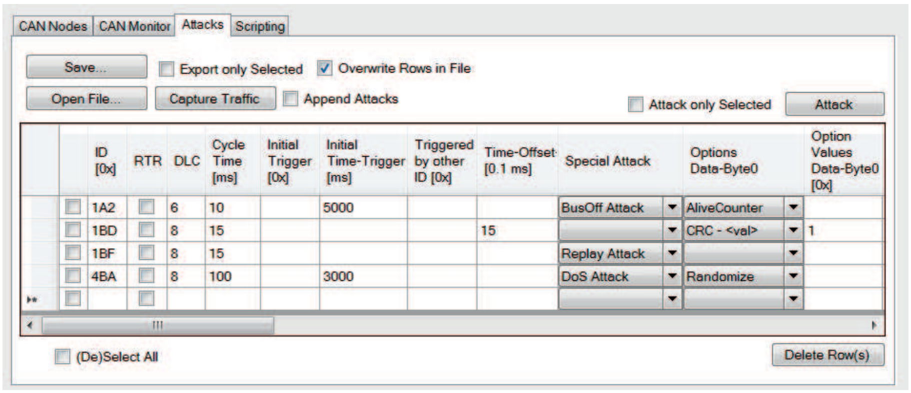
CAN-CT further offers an in-depth analysis of received messages. Each message will be examined for the presence of specific protection mechanisms like application layer CRCs or alive counters. By comparing the details of the currently analyzed message with its preceding messages, we can determine its cycle time and figure out if and in which data bytes such protection measures are used.

The gathered intelligence is of great importance for further attacks, as it offers a profound knowledge about possible targets. This information can then be used by CAN-CT to serve as basis for arbitrary attacks. Besides a C#-scripting engine CAN-CT offers attack templates for

- replay and spoofing attacks,
- bus off attacks, and
- suppression of CAN communication attacks.

While replay and spoofing attacks can be understood as masquerade attacks with the aim to manipulate an ECU by impersonating the original sender of a message, bus off attacks represent a special form of a Denial of Service (DoS) attack (see Section 5.3.2). With this attack we can make an ECU unavailable while preserving all other communication on the bus. In contrast to this targeted DoS attack, we further implemented another form of a DoS attack which suppresses all or part of the CAN communication (see Section 5.3.1).

By specifying different parameters and attack options, we can execute sophisticated attacks with CAN-CT without the need of writing any line of code. Furthermore CAN-CT supports the validation of protection mechanisms like sequence numbers, application layer checksums, time outs, plausibility checks and error or failure handling mechanisms. We can use triggers to determine the time or the event when an attack should be executed (e.g. after the successful execution of another attack, after a specified time, upon the receipt of a specific message etc.). All these attack definitions can be specified in CAN-CT's attack definition user interface depicted in Figure 1.



☐ Export only Selected ☑ Overwrite Rows in File ☐ Append Attacks ☐ Attack only Selected

		ID [0x]	RTR	DLC	Cycle Time [ms]	Initial Trigger [0x]	Initial Time-Trigger [ms]	Triggered by other ID [0x]	Time-Offset [0.1 ms]	Special Attack	Options Data-Byte0	Option Values Data-Byte0 [0x]
	☐	1A2	☐	6	10		5000			BusOff Attack	AliveCounter	
	☐	1BD	☐	8	15				15		CRC - <val>	1
	☐	1BF	☐	8	15					Replay Attack		
	☐	4BA	☐	8	100		3000			DoS Attack	Randomize	
▸*	☐		☐									

☐ (De)Select All

Fig. 1: CAN-CT's plan of attack definition

4.3 Limitations

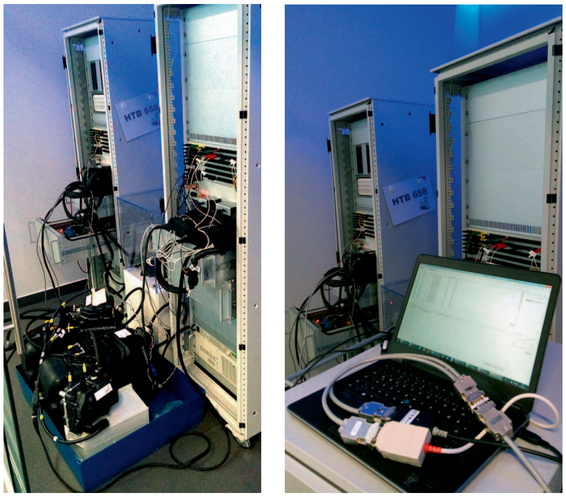
As we made use of a third party USB to CAN adapter (PCAN-USB) all our actions had to conform to the CAN protocol (layer 1 and 2). A delay of around 1.3 ms caused by the used hardware did not provide us with the accuracy needed for our tests. We circumvented this limitation by making use of time-triggered attacks. These attacks anticipate the next time of receipt of a target message and react accordingly. This way we are able to assure an accuracy of less than 100 μs

Due to the large number of different higher layer protocols we decided to not implement any of those protocols. Yet based on CAN-CT's generic design, it is possible to define attacks in a way to meet most requirements of higher layer protocols as well as to extend it, to support future protection mechanisms and attacks. However, complex bidirectional communications are not directly supported for now. Thus, the built-in support of attacks to verify the implementation of diagnostic protocols is subject of future work.

5 Evaluation of Real-World Applicability and Impact

To validate the general functioning of CAN-CT as well as to evaluate its real-world applicability and impact, we examined it in multiple setups.

After proving the general functioning of CAN-CT in a laboratory setup we tested it with actual vehicular ECUs. For this purpose we decided to make use of Hardware-in-the-Loop Systems (HiLs). A HiL can be understood as a test framework where parts of the system are simulated while other parts are actual hardware devices. This approach allows to test a hardware in a true-to-nature test condition without the need of the actual system to be built [RSW07]. In the automotive domain the real hardware is commonly an ECU. The HiL acts as an interface to the simulated environment (e.g. engine), which provides all necessary electrical inputs as well as measures back all ECU electrical outputs. Hence, the hardware ECU under test gets all the information from the HiL as if it was coming from actual vehicle components; providing a test environment which is nearly similar to a real-life vehicle test.



(a) Photo of a heavy duty vehicle Hardware-in-the-Loop System setup

(b) Connecting CAN-CT via the PCAN-USB adapter

Fig. 2: A heavy duty vehicle Hardware-in-the-Loop System setup

We used two different HiL setups at the *Virtual Test Field* (HiL laboratory) of the AVL LIST GmbH[5] to verify CAN-CT. One setup represented a heavy duty vehicle and the other one a passenger car. The passenger car setup consisted of one real hardware ECU: the engine control unit; whereas the heavy duty setup consisted of two ECUs: engine and aftertreatment control unit (see Figure 2). To the best of our knowledge these ECUs represent quite typical vehicular control units, used in current-generation vehicles.

5.1 Overview of Evaluation Results

Using the HiL environment we were able to test the impact and the applicability of CAN-CT in close to real-world scenarios. Here we revealed significant weaknesses and unexpected behaviors of the ECUs under test.

These flaws encompassed serious lacks in the plausibility checking of messages, a too loose time out window for incoming messages, a not strict enough alive counter checking

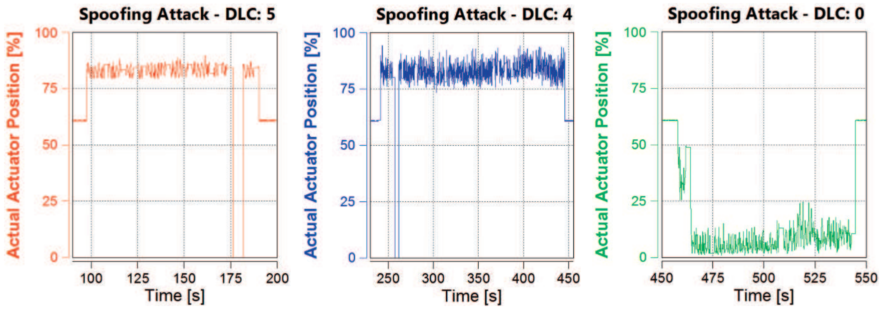
[5] see `https://www.avl.com/`

procedure and a completely omitted examination of the correct frame structure. Furthermore we demonstrated that all tested ECUs are susceptible to our DoS attacks. In the following sections we are going to detail selected parts of these results.

5.2 Spoofing Attack

Being able to spoof messages on the CAN bus serves as the key task in many hacking attacks. With CAN-CT we were able to successfully impersonate other ECUs. We used information gained by analyzing the CAN traffic. Based on the knowledge if, and which message parts contained additional protection mechanisms we were able to draft our spoofed messages to meet all validation criteria at the receiving ECU. By sending our message right after the original message of the sender, we were often able to overwrite the original message in the recipients receive buffer in both HiL setups. This way our message prevails and is able to manipulate an ECU. Besides spoofing recurring messages we can further send CAN commands to cause a requested behavior.

Based on this approach we carried out various attacks to examine the security level of the target's protection mechanisms. One test focused on the target's validation of the received message frame. We discovered that the targeted ECU of the heavy duty vehicle HiL setup did not carry out any validation checks of the frame structure. A message that was defined in its specification to have a fixed length of eight data bytes, was accepted by the receiving ECU even when bytes were missing or the message was sent without any data at all. Missing values were just treated as zeros.



(a) Spoofing attack with a data length of 5 bytes (b) Spoofing attack with a data length of 4 bytes (c) Spoofing attack with a data length of 0 bytes

Fig. 3: Overview of different frame structures with regard to their data length (DLC) and their impact on the acceptance by the targeted ECU in the heavy duty vehicle HiL system setup. The noise, especially in Figure 3b, is caused by a not perfectly successful spoofing attack. The original transmitter of the message was able to place some messages in between our spoofed messages. This caused a minor adaption of the actuator position by the manipulated ECU.

This behavior is demonstrated in Figure 3. The picture shows spoofing attacks with different payload lengths, indicated by the data length code (DLC). The actuator position, we aimed to manipulate to correspond with our spoofed target position, changed for each plot due to the shorter data length of the message. For the first plot (data length of five bytes

(DLC: 5)) it was set to 80 percent (hex 0320). The second attack with four bytes of data already missed the last part of the target position. Instead of the hex value 0320 we just sent 03 which equals 76.8 percent when completed with zeros to the value 0300. The last graphic shows the attack without any data (DLC: 0). Instead of detecting this message as faulty it is again completed with all zeros which led to a target position of zero percent.

As a message without any payload does not contain any information, it should under no circumstances be complemented with zeros and treated as valid by the receiver. This implies that the ECU under test lacks a tight enough verification of incoming messages.

Similar to the above described shortcoming we validated the correctness of the target's time out window. A message that was sent every 15 ms, was still accepted even with an additional delay of 5 ms. Although we observed a deviation between the supposed and the actual time of receipt of up to 1.5 ms in both HiL setups, we believe that a tolerance of more than 5 ms is larger than required.

We were further able to detect vulnerabilities with regard to the alive counter checking procedure and the plausibility checks in the passenger car HiL setup. We demonstrated that the tested ECUs did not check whether the received signals are plausible. For instance we could show that a rise in the signaled vehicle speed from 0 to 250 km/h within one message was accepted by the ECU.

All these shortcomings ease potential hacking attacks as no knowledge of the underlying system is needed. As the common security principle for normal CAN communication in a vehicle is still best described by the security through obscurity approach, these flaws weaken this protection mechanism and offer susceptibilities for a number of attacks.

5.3 Denial of Service Attacks

Besides the manipulation of an ECU the second key task of a CAN hacking tool is the suppression of communication as well as making resources unavailable. For this purpose we implemented two different attacking types. The ECUs under test of both HiL setups did not differ significantly from each other regarding their reaction and behavior during our attacks. Thus, the following paragraphs detail the results of the heavy duty vehicle HiL setup.

5.3.1 Suppressing Communication

The first attack aims to suppress all lower priority messages. Due to the arbitration process of the CAN protocol and the dominant state of a binary zero, only messages with a lower ID (higher priority) than the one, that is flooding the bus, can prevail.

This behavior is illustrated in Figure 4. Here we can see, that, when the attack started at around 90 seconds, no more messages were received (no fluctuations anymore). Just in the

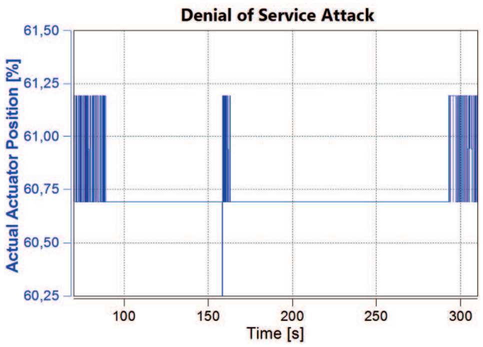


Fig. 4: DoS attack with ID 1BEh in the heavy duty vehicle HiL system setup. The fluctuations at the beginning and at the end of the figure show the original signal. The steady line indicates that no messages were received during this period of time.

middle of the attack the targeted ECU was able to prevail for a short period of time due to a collision on the bus.

During the attack we were not able to observe any error states or failures of the monitored devices. This implies that the ECUs continue their work with the lastly received values. However, when all communication on the bus is blocked, we would have expected the ECU to enter a ”safe” mode or signal a warning. In a real-world hacking attack this could cause serious threats to the driver as no communication over this CAN bus is possible anymore.

5.3.2 Bus Off Attack

The second DoS attack we implemented does not block the CAN communication per se, but it continuously causes collisions on the bus, whenever the targeted ECU tries to transmit a message. Sending a message at the exactly same time as the sender, where the attack message differs from the original message only after the arbitration field, allows us to cause collisions without the target having any way to prevent it. This way we create the impression that the target’s CAN controller is faulty.

Due to the CAN protocol’s error handling mechanisms, the target has to increase its internal error counter, every time a collision occurs. As soon as this error counter reaches a value of 255, it enters the bus off state. In this state the ECU is not allowed to send nor to receive messages anymore. However, as we observed, the targeted ECUs regularly reset their internal error counters, allowing them to participate on the bus again as soon as possible.

This behavior is depicted in Figure 5. Here we aimed to send a falsified message content along, to directly take over this message ID (original actuator value 35 percent, spoofed value 30 percent). Shortly after we started our attack (bold, red line), we were able to make the target unavailable for the first time. However, a bus error was detected - the signal fell to a value of zero - which caused our spoofed message to be declined by other recipients as well. At second 760 the targeted ECU reset its error counter, allowing it to send the actual message again. Soon afterwards we were able to successfully spoof our actuator position target value between seconds 780 to 810. At second 810 neither we nor the actual ECU could transmit the target value anymore. Due to the continuous reset of the target's error counter, we had to continue to cause collisions, not allowing us to manipulate other ECUs simultaneously. However, we were able to make the target unavailable while preserving all other CAN communication.

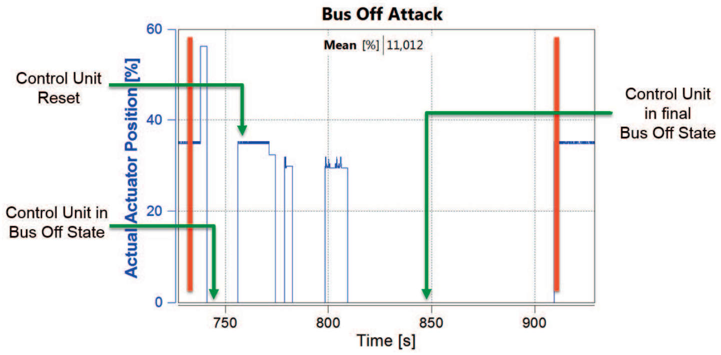


Fig. 5: Bus off attack in the heavy duty vehicle HiL system setup (the bold, red lines indicate the start and the end time of the attack)

6 Outlook and Conclusion

As car hacking becomes a serious threat, it is getting increasingly important to ensure a correct and secure behavior of every single ECU. Protection mechanisms are just as good as their implementation. We believe that a very efficient method to find security vulnerabilities in ECUs is, to attack them the same way as a hacker would.

Currently almost no security measures are implemented for normal communication on vehicular CAN buses. Thus, it is even more important to assure the proper working of existing protection techniques. Therefore, with CAN-CT we implemented an application that is capable of attacking ECUs while simultaneously tracking whether undesired or unspecified behavior was detected. This allows us to test the correct implementation of protection mechanisms on the one hand, while on the other hand attack detection techniques can be examined this way as well.

In this work we focused on the proof of concept of CAN-CT. The next step will be a successful integration into security evaluation methodologies and safety and security standards. Here we will have to develop a suitable concept for the systematic generation of test cases according to defined security goals.

The focus of CAN-CT in its current version is on the examination of normal CAN communication. Sophisticated bidirectional communications or diagnostic protocols are not implemented so far and are subject of future work. Using a specifically designed CAN controller would further allow for not having to conform to the CAN protocol. This way additional attack scenarios like the physical connection of a malicious device to the CAN bus can be supported more realistically.

With the use of Hardware-in-the-Loop Systems we were able to prove CAN-CT working properly in a real-world scenario. We revealed significant shortcomings in commercial ECUs and thus, demonstrated its applicability and impact for current automotive systems. For that reason we want to highlight the importance of a security testing framework to complement existing safety and functional testing environments.

References

[Ba14] Bayer, Stephanie; Enderle, Thomas; Oka, Dennis-Kengo; Wolf, Marko: Automotive - Safety & Security 2014 (2015), Sicherheit und Zuverlässigkeit für automobile Informationstechnik, Tagung, 21.-22.04.2015, Stuttgart, Germany. volume 240 of LNI. GI, 2014.

[Ch09] Charette, Robert N.: This car runs on code. In: IEEE Spectrum. volume 46, p. 3, 2009.

[Ch11] Checkoway, Stephen; McCoy, Damon; Kantor, Brian; Anderson, Danny; Shacham, Hovav; Savage, Stefan; Koscher, Karl; Czeskis, Alexei; Roesner, Franziska; Kohno, Tadayoshi: Comprehensive Experimental Analyses of Automotive Attack Surfaces. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings. 2011.

[GGV14] Glas, Benjamin; Gramm, Jens; Vembar, Priyamvadha: Automotive - Safety & Security 2014 (2015), Sicherheit und Zuverlässigkeit für automobile Informationstechnik, Tagung, 21.-22.04.2015, Stuttgart, Germany. volume 240 of LNI. GI, 2014.

[HKD11] Hoppe, Tobias; Kiltz, Stefan; Dittmann, Jana: Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures. Rel. Eng. & Sys. Safety, 96(1):11–25, 2011.

[Je15] Jenik, Aviram: Increasing resilience by finding unknown vulnerabilities. In: iCC 2015 Proceedings - 15th international CAN Conference, Vienna, Austria, October 27-28, 2015. pp. 06–1–06–5, 2015.

[Ko10] Koscher, Karl; Czeskis, Alexei; Roesner, Franziska; Patel, Shwetak; Kohno, Tadayoshi; Checkoway, Stephen; McCoy, Damon; Kantor, Brian; Anderson, Danny; Shacham, Hovav; Savage, Stefan: Experimental Security Analysis of a Modern Automobile. In: 31st IEEE Symposium on Security and Privacy, S and P 2010, 16-19 May 2010, Berleley/Oakland, California, USA. pp. 447–462, 2010.

[Ko14] Koscher, Karl: Securing Embedded Systems: Analyses of Modern Automotive Systems and Enabling Near-Real Time Dynamic Analysis. PhD thesis, University of Washington, 2014.

[La13] Lawrenz, Wolfhard: CAN System Engineering. Springer-Verlag London, 2013.

[Ma14] Marinescu, Raluca; Saadatmand, Mehrdad; Bucaioni, Alessio; Seceleanu, Cristina; Pettersson, Paul: A Model-Based Testing Framework for Automotive Embedded Systems. In: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE, pp. 38–47, August 2014.

[MV13] Miller, Charlie; Valasek, Chris: , Adventures in Automotive Networks and Control Units, 2013.

[NSL13] Navet, Nicolas; Simonot-Lion, Françoise: In-vehicle communication networks - a historical perspective and review. In (Zurawski, Richard, ed.): Industrial Communication Technology Handbook, Second Edition. CRC Press Taylor&Francis, 2013.

[RSW07] Ren, Wei; Steurer, Michael; Woodruff, Steve: Accuracy evaluation in power hardware-in-the-loop (PHIL) simulation center for advanced power systems. In: Proceedings of the 2007 Summer Computer Simulation Conference, SCSC 2007, San Diego, California, USA, July 16-19, 2007. pp. 489–493, 2007.

[Ta14] Talebi, Sareh: A Security Evaluation and Internal Penetration Testing Of the CAN-bus. Master of science thesis, Chalmers University of Technology, 2014.

[TAA14] Tao, Zhang; Antunes, Helder; Aggarwal, Siddhartha: Defending Connected Vehicles Against Malware: Challenges and a Solution Framework. IEEE Internet of Things Journal, 1(1):10–21, 2014.

[VM15] Valasek, Chris; Miller, Charlie: , Remote Exploitation of an Unaltered Passenger Vehicle, 2015.