Extending Vehicle Attack Surface Through Smart Devices

Rudolf Hackenberg,
Nils Weiss, Sebastian Renner

University of Applied Sciences Regensburg, Germany
Email: rudolf.hackenberg@othr.de
{nils2.weiss, sebastian1.renner}@othr.de

Enrico Pozzobon

University of Padua, Italy
Email: enrico.pozzobon.1@studenti.unipd.it

***Abstract*—Modern cars include more and more features that first emerged from the consumer electronics industry. Technologies like Bluetooth and Internet-connected services found their way into the vehicle industry. The secure implementation of these functions presents a great challenge for the manufacturers because products originating from the consumer industry can often not be easily transferred to the safety-sensitive traffic environment due to security concerns. However, common automotive interfaces like the diagnostics port are now also used to implement new services into the car. With dongles designed to read out certain vehicle data and transfer it to the Internet via the cellular network, the owner can access information about gas consumption or vehicle location through a mobile phone app, even when he is away from the car. This paper wants to emphasize new threats that appear due to the ongoing interconnection in modern cars by discussing the security of the diagnostics interface in combination with the use of an Internet-connected dongle. Potential attack vectors, as well as proof-of-concept exploits will be shown and the implications of security breaches on the safe state of the vehicle will be investigated.**

Keywords*– *On-Board-Diagnostics; Cellular Network; Automotive Security.

I. INTRODUCTION

The term ”On-Board-Diagnostics (OBD)-II-dongle” refers to a group of aftermarket devices that can be connected through the OBD-II interface to upgrade the functionality of new and old cars, and can be installed by a customer without any technical knowledge [1]. These dongles are usually available at a low price and promise interesting features, like connecting the vehicle to a smartphone through the Internet and letting the owner monitor certain in-vehicle data like fuel consumption on different tracks and the Global Positioning System (GPS) data points to determine the cars's position over time. The OBD-II devices are available for every vehicle that implements an OBD-II diagnostic port, which applies to almost every vehicle which is participating in common traffic these days.

Even if the relatively easy improvement of cars' features through plugging in an OBD-II-dongle sounds tempting, the devices can bring along particular risks and alter the security of a vehicle in the long run. OBD-II-dongles use the same protocols as repair shop tester software to read data from the car's bus systems. [2] After reading the device conditions, the data is sent to a backend server on the Internet, which acts as a database for the frontend application that interfaces the user. If the device uses weak security measures, potential vulnerabilities in the dongle's firmware can open an insecure gateway to the electronic infrastructure of the whole car [3]. In further sections, this possible attack surface shall be described and a possible exploit will be introduced.

In Section II, related work to this paper will be shown; Section III will give a short overview over relevant automotive diagnostic protocols, while Section IV will explain discovered vulnerabilites of OBD-II-dongles, that have been investigated. Section V will cover security threats that can follow from installing an OBD-II-dongle, before Section VI will conclude the results of this paper and give a short outlook to possible future work in this field.

II. RELATED WORK

Investigating security vulnerabilities and introducing possible attacks is already being researched for a couple of years. Especially exploiting weak in-vehicle protocols like Control Area Network (CAN) is a pretty well-known topic [4]. Also, attacks using a pirate Base Transceiver Station (BTS) in cellular networks have already been introduced by Paget in 2010 [5]. The possibility to perform an over-the-air attack on a specific telematics dongle, has been shown by Szijj et al. in 2015 [6]. More recent work, especially including targeting the standard OBD-II-interface through wireless signals, has been conducted by Zhang et al. in 2016 [7]. This research team also proposed an attack on OBD-II-dongles, but – unlike this article – their investigation was focused on controlling an OBD-II-dongle through a paired phone's Bluetooth connection. Besides attacks on dongles and the cellular network, additionally, ways to exploit a repair shop tester, including the diagnostic protocols that are also mentioned in this article, have been published [8]. This paper will cover parts of the different research areas mentioned above and propose a way to analyze and exploit OBD-II-dongles and the interface's diagnostic protocols wirelessly, without the use of supplementary devices like smartphones.

III. AUTOMOTIVE DIAGNOSTICS PROTOCOLS

After the first efforts to implement and unify a diagnosis interface in passenger vehicles more than 20 years ago, some standards regarding the hardware interface and the used protocols have been developed. Even though early perceptions of the capabilities of a diagnostic interface focused on the possibility of gathering information on the cars' emissions only, with the ongoing progress in car manufacturing a lot more functionality was realized through the OBD-II-connector. Therefore, also the protocols that handle the diagnostic communication evolved over time and are nowadays used for transferring complex data

 ISBN: 978-1-61208-582-1

structures, for example during reprogramming an Electronic Control Unit (ECU). The following two subsections will introduce two important standards in the environment of automotive diagnostics.

A. Diagnostics on Control Area Networks

The ISO-15765-2 standard introduces the network and transport layer services of the Diagnostics over CAN protocol [9]. It describes the way data of different size can be transmitted in a reliable way. Besides the transferring of single frames – which are usually limited to a maximum length of 8 bytes in the CAN protocol – it especially specifies the handling of larger payloads. The standard, often also referred to the name ISO-TP, shows a dictate to enable the transmission of messages with a payload up to 4096 bytes. This rise of capacity is achieved by introducing a rule set for segmenting the data into multiple frames and implementing a specific frame type to indicate that a message is being segmented, the *Segmented Frame*.

B. Unified Diagnostic Services

The previously described ISO-TP standard is widely used for the transmission of data on the CAN bus and the Unified Diagnostic Service (UDS) protocol (also called ISO-14229) makes use of it [10]. The UDS protocol describes regulations to enable a standardized communication between a diagnostics tester and all ECUs present in the bus topology of a car. It implements a request/response message model on the ISO/OSI session layer and above. The model prescribes that every request has to be answered with a positive or negative frame according to the standard. Basically, a common request consists of a source and destination address, a service id that uniquely identifies the request and some request-specific parameters. To indicate if the request was successful, only the first byte of the response has to be examined. In the positive case, it has to contain the value of the service id added to *0x40*, if the response is negative the message starts with *0x7F*.

Besides the structure of the messages, the UDS protocol also describes a great amount of standard services. Some of them can be used to read out specific data from an ECU (*ReadDataByIdentifier*), but there also exist services that are designed to write certain bytes in the ECU's storage (*WriteDataByIdentifier*). Furthermore, routines to control specific functions inside the car are also defined by the standard. For example, the routine *ECUReset* sends a reboot request to the ECU with the address given in the destination address parameter. So, an individual that gains access to the OBD-II-interface under any circumstance can craft all standard messages by gathering information through reading the publicly available UDS-Standard. With this knowledge for example a reset of any ECU is possible. Another remarkable command is in charge of the control of the communication on the shared CAN-Bus (*CommunicationControl*). This command can completely turn of the reception and transmission functions of an ECU. This feature is usually used during the flash procedure of the ECU via the CAN-Bus. The whole traffic on the bus, except the traffic between a repair shop tester and the ECU which has to be flashed, gets disabled to speed up the flashing time by providing the full bus bandwidth. An attacker can easily shut down the communication of an ECU through this command.

IV. OBD-II Dongle Security

Multiple Internet-connected OBD-II-dongles have been tested for security vulnerabilities that could be exploited by an attacker to wirelessly inject malicious CAN frames into a vehicle over the OBD-II connector.

While the backend infrastructure and the user web interface for each of these dongles is made by the company responsible for the distribution of these dongles, the hardware and firmware are outsourced to different Original Equipment Manufactorers (OEMs). Local distributors do not have access to the source code of the firmware, and are unable to asses the security of their product. Due to inability to communicate with the OEMs, a blackbox security analysis has been conducted.

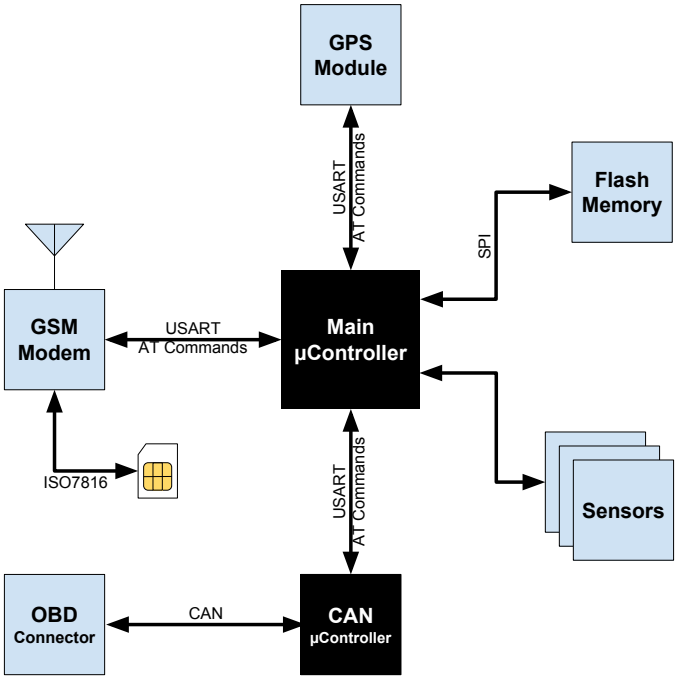

Figure 1. Example block diagram of dongle hardware

Figure 1 shows the general block diagram of the hardware found in the examined dongles. A primary microcontroller is responsible for power management, event logging, and firmware flashing. A secondary microcontroller communicates with the car via CAN bus and other protocols. A Global System for Mobile Communications (GSM) modem provides Internet connectivity and a GPS receiver allows for location tracking. Other sensors, like microphones, accelerometers and gyroscopes are also present in some of the examined hardware.

A. GSM Vulnerabilities

The cellular modem is the prime attack entry point for an attacker. These devices must be able to establish an Internet connection over long stretches of roads that might not be covered with 3G (or newer) cellular technology, which offers a good security model. Because of this, automotive Internet-connected hardware must support 2G cellular connectivity (GSM with General Packet Radio Service (GPRS)/Enhanced Data Rates for GSM Evolution (EDGE) Internet) which supports cryptographic authentication only of the Mobile Station (MS) and not the BTS.

 ISBN: 978-1-61208-582-1

Since the connection between the MS and the BTS is vulnerable to a Man-in-the-Middle (MITM) attack in GPRS and EDGE, authentication of the server must be done at the application layer by the MS. All the analyzed OBD-II-dongles fail to implement proper cryptographic authentication at the application layer, possibly because of insufficient resources on the embedded microcontroller used or disregard for security from the developers.

In our tests, both OsmoNITB and YateBTS were used to setup a pirate BTS and successfully hijack the connection from a dongle as Paget already demonstrated [5]. This allowed for the protocol to be reverse-engineered, which made it possible to write a pirate backend server to further exploit the dongle.

In some cases, the backend server would reject the hijacked connection, detecting that the dongle was not connected using the legitimate Access Point Name (APN) (the dongle provider would operate an APN themselves). In these situations, it is still possible to analyze the protocol either by probing the Universal Asynchronous Receiver Transmitter (UART) line to the GSM modem with a logic analyzer and decoding the Attention Commands (AT) and Point-to-Point-Protocol (PPP) frames coming from the microcontroller, or connecting the pirate BTS to the Internet using the Subscriber Identity module (SIM) card from another dongle of the same distributor.

It is notable that strong cryptographic authentication could have been achieved using a standard Hash Message Authentication Code (HMAC) with a different key for each dongle, which has low enough complexity to be implemented on the low power hardware used.

B. Over-the-Air Updates

All examined dongles support Over-the-Air (OTA) updates to replace the microcontroller firmware, fix bugs and add features. These updates are usually initialized by a command received from the backend server to which the dongle reacts by downloading a firmware image over Hypertext Printing Protocol (HTTP) from a simple web server. The downloaded binary is flashed either in place by the running firmware, or by a static bootloader which can't be updated and has the ability to revert the flashing process if something goes wrong.

Naturally, since no cryptographic authentication is implemented at the application layer, it is trivial to provide a customized firmware after the OTA update is triggered using the pirate backend server.

Some dongles try to verify the integrity of the downloaded binary by putting checksums and length fields in various positions inside the firmware. In order to pass this verification, a reverse-engineering of the firmware software has been performed.

Different techniques were used for different dongles in order to obtain the firmware for reverse-engineering it. When it was possible to manually trigger an OTA update, simply sniffing the connection as described earlier was sufficient to extract the unencrypted firmware out of the Transmission Control Protocol (TCP) stream, or to obtain the HTTP Uniform Resource Locator (URL) from which it was possible to download different firmware versions.

In all dongles, the downloaded firmware is cached before the actual flashing on a non-volatile memory outside the main microcontroller. These memory chips work using the Serial Peripherial Interface (SPI) protocol, the same used by Secure Digital Memory Cards (SD-Cards), which made it easy to read the content and find recently flashed firmwares and older rollback versions to use in case of boot failure.

In some dongles, an obstacle for the reverse-engineering was created by the presence of a static bootloader that handled the flashing procedure. This bootloader resides in a distinct location in the internal flash of the microcontroller, and can't be replaced via an OTA update. This means that it was not possible to intercept it as described before. Moreover, debug interfaces like Joint Test Action Group (JTAG) and Single Wire Debug (SWD) were disabled on these dongles. However, the bootloader could be dumped by getting the dongle to execute a small piece of custom assembly code (8 bytes) that used the original serial output routine. This exploit payload was small enough to be fitted in the known firmware without changing the length and only a checksum needed to be changed. The exploit simply calls the write function in the standard C library to dump the flash page containing the bootloader over an UART line.

C. Attack Procedures

Once the reverse-engineering of the software, protocols and hardware schematics was completed, a wide array of attacks became possible. The first step for all attacks was hijacking the victim's GPRS connection. This can usually be done by simply transmitting the pirate BTS signal with higher power than the legitimate BTS. Sometimes, jamming the legitimate BTS signal is also required (for example for devices supporting 3G connectivity).

After a temporary hijack of the Internet connection of the victim's dongle was achieved, a rogue Domain Name Service (DNS) server was used to trick the dongle into connecting to the pirate backend server. Now the pirate backend server could spoof the commands required to change the dongle configuration.

The dongles configuration includes the Internet Protocol (IP)-address of the backend server, which could be changed to the attacker's server IP-address. At this point, even when the GSM hijacking was interrupted, the dongle would still try to connect to the attacker's server.

The attacker's backend server could be used to trigger OTA updates which allow the attacker to flash exploited firmwares on both microcontrollers. This means the attacker had full access to the microcontroller responsible for interaction with the car, and could send any desired command on all the interfaces supported by the victim's dongle.

V. Security Threats

A. Surveillance

During the research on the investigated dongles, many possible ways to spy on an user were discovered. With the integrated sensors on the dongle, very accurate movement profiles can be created. An internal microphone of one investigated dongle could be used to eavesdrop on a driver.

B. Denial of Service

With the equipped CAN transceiver on the dongle, many sophisticated denial of service attacks on the car's internal network are possible. The simplest denial of service is a general

 ISBN: 978-1-61208-582-1

or broadcasted *ECUReset* UDS command. This will reset all ECUs of the vehicle because the internal gateway distributes a broadcast command. With the possibility of modifying the dongle's firmware, ECU-targeted, conditional or persistent attacks are also possible. The Diagnostic communication over CAN (DoCAN) protocol with extended addressing allows an attacker to reset one specific ECU. It is possible to trigger a reset when certain conditions are met, for example if the accelerometer of the dongle is detecting high centrifugal forces. Also the reset of an airbag ECU based on the detection of brake force is possible. A persistent denial of service can be achieved with *CommunicationControl* commands or with setting ECUs in special modes, like the programming mode. An attacker can advice an ECU to be completely silent on the bus. Some of this mode changes are persistent. At least the erasure of some program parts on an ECU, which is usually performed from a repair shop tester during the flash procedure, leads to a persistent denial of service. A so-called smart device can increase the attack impact by specific conditions in dangerous situations of a car.

C. Distributed Denial of Service

Through persistent modification of a dongle's firmware, it is possible to hijack the communication and hide the MITM-attack for the dongle's operator. In this way, attackers can infect many dongles and start a distributed denial of service attack at a specific time. A distributed attack will create a much higher public visibility for such an attack, and can easily harm the image of a car manufacturer or a dongle operator. More advanced firmware modifications allow an attacker also to collect specific information about the host vehicle of an attacked dongle. It is possible to read out the Vehicle Identification Number (VIN), the vehicle manufacturer and even information about installed equipment. This allows extremely fine-grained attacks.

D. Malicious ECU reconfiguration

Usually car manufactures use the same ECU design for multiple car variants and sometimes even for different car models. For this reason, the firmware of an ECU has to be highly configurable. In this research, multiple ways to change the configuration of an ECU were discovered. For example, the functions for releasing airbags can be reconfigured. Such configurations can be done trough repair shop testers. Any authentication secrets can be extracted from the binary of the firmware, but also a security session hijack is possible. With a custom firmware on a GSM-OBD-II-dongle, the challenge can be caught and passed over GSM to a control server. There, a second software part can simulate a car and receive the proper response from an original repair shop tester. In this way, a dongle can get security access through a MITM-attack on a remote simulated repair shop tester connection.

E. Malicious ECU reprogramming

The signature processes of investigated ECUs did not show any weaknesses so far, but if an attacker is able to sign it's own firmware or bypass the verification process, he can also ship this firmware through an infected OBD-II-dongle. The dongle can independently flash this firmware to a specific ECU. Without any further work, an attacker is always able to downgrade a firmware to a previous and correctly signed version. Sometimes car manufacturers release new firmware versions because of security patches. By flashing an obsolete firmware, an attacker can reopen a fixed security vulnerability, which could be exploited in a second step.

VI. Conclusion and Future Work

While the vulnerabilities of the OBD-II-connector have been known for a long time, car manufacturers only had to worry about illicit modifications made by the car owners themselves, since access to the OBD-II-interface required for the attacker to be physically inside the car. More recently, a wide array of OBD-II-dongles appeared on the market, and many of them implement wireless connectivity with uncertain security. Zhang et al. demonstrated how Bluetooth OBD-II-dongles can be exploited by an attacker who has access to the victim's smartphone [7]. This paper showed how GSM-OBD-II-dongles are vulnerable to attacks from a relatively long range, and allow the attacker to obtain a persistent access to the OBD-II-connector over the Internet.

As more and more OBD-II-enabled devices are presented to the public, it is impossible to trust that all of them will maintain a good security architecture. Instead, it would be advisable that car manufacturers start treating the OBD-II-connector as a highly dangerous attack surface. It was shown that since the CAN bus interface on the OBD-II-connector is used by repair shops to make modifications to the car configuration, it is also possible for a remote attacker to realize the same operations through an insecure Internet-connected OBD-II-device. In a more secure car architecture, the OBD-II-connector would be used only for the standard diagnostic OBD-II PIDs, which shouldn't include operations critical for security and safety.

In the future, one approach to extend the work conducted could be trying to automate parts of a security investigation. Even if the results of the research on different OBD-II-dongles delivered new insights on the security of the interface, it would save time and the outcome would be more comparable, when some steps of the security analysis could be done automatically. Therefore, knowledge about previously discussed vulnerabilities has to be taken into account and specific test scenarios have to be created. In the end, a custom-built framework for performing penetrations tests on OBD-II-dongles will be the major goal. Also if certain parts – like the reverse-engineering of the device's hardware – need to be realized manually, a partly-automated tool to guide the security researcher regarding the execution of prearranged test cases could possibly improve the investigation process by saving time. By automating chosen test procedures and therefore uniforming the structure of their output, the test results will also be more standardized, which helps with interpreting and evaluating the accomplished findings.

Besides the attempt of automating the present investigation process of OBD-II-dongles, also applying and extending the discoveries already made to other in-vehicle systems, that are connected to the Internet, could be a valid proceeding. For example infotainment systems can implement a WiFi-Access-Point, to which passengers can connect to. Because these systems usually provide Internet access through their own GSM connection, they are possibly vulnerable to similar attacks based on a pirate BTS, like the one shown in this paper. Basically every connected device that is present in a

 ISBN: 978-1-61208-582-1

modern car is worth analyzing in regards to security. As the number of such devices will grow and vehicles will get intra- and inter-networked, lots of different areas of research in this domain will need emphasized attention and could possibly be a follow-up for the presented investigations.

REFERENCES

[1] International Organization for Standardization, "ISO 15031-3: Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 3: Diagnostic connector and related electrical circuits: Specification and use," 2016. [Online]. Available: https://www.iso.org/standard/64636.html, visited 2017.07.13

[2] W. Yan, "A two-year survey on security challenges in automotive threat landscape," in 2015 International Conference on Connected Vehicles and Expo (ICCVE), Oct 2015, pp. 185–189.

[3] D. S. Fowler, M. Cheah, S. A. Shaikh, and J. Bryans, "Towards a testbed for automotive cybersecurity," in 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), March 2017, pp. 540–541.

[4] D. K. Nilsson and U. E. Larson, "Simulated attacks on can buses: Vehicle virus," in Proceedings of the Fifth IASTED International Conference on Communication Systems and Networks, ser. AsiaCSN '08, 2008, pp. 66–72.

[5] C. Paget, "Practical Cellphone Spying," DEFCON 18, 2010.

[6] A. Szijj, L. Buttyán, and Z. Szalay, "Hacking cars in the style of stuxnet," Oktober 2015. [Online]. Available: http://www.hit.bme.hu/ buttyan/publications/carhacking-Hacktivity-2015.pdf, visited 2017.07.13

[7] Y. Zhang, B. Ge, X. Li, B. Shi, and B. Li, "Controlling a car through obd injection," in 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), June 2016, pp. 26–29.

[8] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in 9th USENIX Workshop on Offensive Technologies (WOOT 15), Washington, D.C., Aug. 2015.

[9] International Organization for Standardization, "ISO 15765-2:2016: Road vehicles – Diagnostic communication over Controller Area Network (DoCAN) – Part 2: Transport protocol and network layer services." [Online]. Available: https://www.iso.org/standard/66574.html, visited 2017.07.13

[10] ——, "ISO 14229-1: Road vehicles – Unified diagnostic services – Part 1: Specificatoin and requirements," 2013. [Online]. Available: https://www.iso.org/standard/55283.html, visited 2017.07.13

 ISBN: 978-1-61208-582-1