Literary Review

Michiel Willems

April 18, 2018

Abstract

Software is an integrated part of todays cars and has gained a lot of attention in recent years. This paper will focus on the security of the various software components that make up a modern car, focusing on a vehicle's diagnostic capabilities. First the necessary technical background (e.g. ECU's, CAN, OBDII, etc) will be explored. The rest of the paper will give a detailed analysis of the security issues that have arisen in recent years, as well as some protocols and methods that were proposed to counteract potential exploits. This paper will conclude with a proposal to secure the OBDII connector inside modern cars.

1 Introduction

The modern vehicle is now called a 'Computer on wheels' [2] . it has undergone a lot of change since it was introduced in the late 19th century. The automobile was a purely mechanical device, designed to transport people from one place to another in relative speed and comfort. The modern automobile contains up to 100 millions lines of code that's spread out over as much as 70 individual ECU's (Electronic Control Units) [1]. Each ECU is an embedded device that is designed to perform a specific function (e.g. braking, opening the door, speed control, etc). In addition to having this wide variety of embedded devices a modern vehicle will also employ a data bus that allows all ECU's inside a vehicle to communicate with each other. There are multiple standards that are employed even within a single vehicle, but the CAN (Controller Area Network) protocol is probably the most common and will be the focus of this paper. Alongside internal communications, a lot of newer models will also incorporate some kind of external communications. This can range from vehicle to infrastructure (V2I) communications (e.g. wireless gas payment at a gas station, wireless diagnostics at a repairshop or even virtual traffic lights) or vehicle to vehicle (V2V) communications (e.g. automatically following another vehicle) [3] [4].

All of the extended functionality introduced into cars greatly improves their flexibility and comfort, but also makes them increasingly vulnerable. As with any interconnected computer system it is susceptible to attack from malicious agents if the right level of security [14] is not attained. This is made even worse by the introduction of V2I and V2V communications since this allows for remote attacks, eliminating the need for physical access to the network (which of course is hard to attain). A lot of the ECU's that facilitate external communications can now be seen as potential attack vectors. Take Bluetooth for example: A lot of cars include Bluetooth functionality to allow users to connect their phones and play music. The Bluetooth protocol has a large protocol-stack and has been been known to have problems in the past [5]. Another problem is that it's easy for a phone to get compromised (visiting a malicious website) [6].

There are numerous points of entry to the internal vehicle network both physical (Breaking into the vehicle) or wireless (Bluetooth, TPMS, Radio system) [6]. One of these (and also one of the most vulnerable) is the OBD-II interface. The OBD-II protocol is widely deployed in auto mobiles as a way of getting diagnostics information from the vehicle. The interface it introduces allows full access to the internal network, and it has been shown [6][5][10][11] that a set of messages and signals could be injected on a car's CAN bus to control key components (e.g. lights, locks, brakes, and engine) as well as injecting code into key ECUs. The focus of this thesis will be to try and mitigate this kind of illegal access by introducing some type of access control to the OBD-II interface.

2 Background: OBD-II and CAN

2.1 OBD-II

The OBD-II (On Board Diagnostics Protocol) was introduced to allow for self diagnostic and reporting functionality for ECU's inside a vehicle, and has been mandatory in every car produced in the united states since 1996. [7]. It allows users (testers, developers, repairmen, etc) to query ECU's about diagnostics information in order to perform a detailed analysis of the vehicles internal systems. In order to allow for a user to communicate with the vehicle's network the protocol introduces a data link connector (DLC). The DLC is a 16-pin hardware interface (although only 9 pins are enforced by the standard) that is found close to the steering wheel (by law it is required to be installed within 0.61 m of the steering wheel) [7]. It introduces parameter ID's which are codes used to identify and query specific data. The protocol is designed to work with multiple signalling protocols (the messaging protocol that is used to request and receive data from the network) but the CAN protocol is mostly implemented (Since 2008 all vehicles sold in the us must implement this signalling protocol).

There are multiple ways for a user to interact with this interface:

- Standard Diagnostic scanning tool is a dedicated device that consists of a small hand-held module (equipped with a small screen and some buttons) connected to a male DLC-connector (The DLC inside the vehicle is always a female).
- An advanced Diagnostic scanning tool that includes a DLC-connector with wifi/Bluetooth compatibility, allowing for remote diagnostics using a smartphone.
- A DLC-connector with a usb adapter allowing access via dedicated software on a pc.

Typically the ODB-II is used like this (CAN as signalling protocol): First the user enters the PID of the data he/she wants to query into a diagnostic tool. Secondly this data is packaged in a CAN frame and sent on the CAN-bus. Thirdly the ECU that is responsible for the data identified by the PID in the message recognizes it as it's own, and transmits a CAN frame containing the requested data. Fourthly the diagnostic tool recognizes the response and displays the data to the user. [8]

2.2 CAN

The Controller Area Network (CAN) protocol is a high-speed, promiscuous protocol, broadcasting all net-work traffic to all nodes on a given bus. Although new technologies are slowly gaining more attention, CAN is still one of the most common ways for Electronic Control Units (ECUs) to communicate [10]. A typical CAN frame (a CAN message is called a frame) has a 64 bit payload field, which means each message contains a maximum of 8 bytes of data. Each CAN frame includes a destination address (11 bits) and nodes are expected to ignore traffic not addressed to them. However a CAN frame does not include a source address [2]. An interesting thing about CAN is what happens when multiple nodes are trying to transmit a message at the same time. The protocol will give priority to the message with the lowest destination ID. This is easily facilitated since a transmitted 0-bit is dominant while a 1-bit is recessive (which means that of both are transmitted at the same time the 0-bit will 'win' and be broadcasted on the bus). Any ECU sensing a change in the bit it sent will suspend it's communication, thereby ensuring that the message with the lowest destination ID will be sent first.[9]

CAN messages have multiple purposes: First there's informative messages that are designed to transmit data from and to ECU's (e.g. the Anti-Lock System (ABS) broadcasting the speed of each wheel). Secondly there's action messages that are designed to request another ECU to perform an action (e.g. daptive Cruise Control (ACC) module requesting the brakes to be applied). Lastly there's the diagnostic messages defined by the OBD-II protocol. [10]

3 Security Issues

It has been stipulated that 'CAN, by design, offers no protection from manipulation' [2]. This conclusion can be ascribed to a number of protocol limitations. First a CAN frame does not include a source address, so the ECU receiving a messages has no guarantee about it's origin. This allows malicious agents to easily construct and insert CAN messages to alter the vehicle's behaviour, as has been frequently demonstrated by Charlie Miller and Chris Valasek's expoits [5][10][11]. Secondly because of CAN's broadcast nature all data that is transmitted will be readable by all nodes on the network. This shortcoming allows for compromised ECU's to read and interfere with data that is not intended for it. Thirdly the fact that messages are prioritized solely based on destination ID makes it trivial to perform a denial of service attack. Lastly There is no inherent support for addressing, encryption and authentication [10].

The lack of security of the CAN protocol combined with the access that the OBD-II allows makes for a very vulnerable system that is open to many forms of attack. The impact of these attacks vary. Some of these could even be detrimental to the safety of any people inside the vehicle. If a malicious agent were to gain remote access to the vehicle system (For example via a Wifi compatible ODB-II data tool covertly inserted into the vehicle), He would be able to inject CAN frames to distract the driver (Steering changes, Turn on wipers, etc) [12]. Other attacks could have a less safety critical impact. Examples of this type of attacks are changing emissions information (similar to the dieselgate Volkswagen schandal), Changing the odometer value (The odometer records the distance the vehicle has driven so far in it's lifetime, changing this value would increase the cars value when it's being sold), Changing recorded crash data, etc. [13]

4 Proposed Solution: OBD-II Role Based Access Control

The problem this thesis will attempt to tackle is unauthorised access to the vehicle CAN network via the OBD-II DLC. Since the OBD-II protocol was designed solely for diagnostic purposes our goal will be to limit access accordingly. The system we will attempt to design and implement is basically a role based access control protocol that limits the capabilities that someone connected via the DLC has, according to which role this person belongs. These roles range from the owner of the car, A technician in a repair shop, a policeman, a technician at an official dealership. Each of these roles should facilitate different capabilities within the system (e.g. a technician at an official dealership should be able to update an ECU and perform various alterations, while a technician at a basic repair shop should only be able to diagnose a mechanical problem).

One of the first problems we had to solve was where we were going to implement our solution. The simplest solution would be to introduce a small microcontroller into the vehicle that serves as a sort of firewall between the OBD-II DLC and the rest of the vehicle network. However shipping this solution to existing vehicles would imply introducing hardware and substantial rewiring of the existing subnetworks. A better solution would be the implement this solution on a microcontroller that is already present inside most vehicles. Luckily since the modern automobile contains multiple subnetworks, some even with different communication protocols (LIN, MOST, etc) a gateway ECU is needed to facilitate communications between different subnetworks. This ECU is a perfect place to implement our solution since it has a pivotal role in forwarding traffic around the vehicle.

Now we know where to implement our solution but we haven't explained how. The solution we propose is to associate each role with a distinct public/private key pair. The public key will be safely stored on the central gateway inside the vehicle network. While the private key will be distributed to the appropriate role holder (Technician, policeman, user, etc). The idea is that whenever someone wants access to the vehicle network, he/she will have to prove ownership of the corresponding private key, thereby proving they possess the appropriate role. Aside from the public keys the central gateway will also hold a permission table, specifying which messages are allowed to be transmitted by which role.

5 Key Challenges

Despite the relative simplicity of our solution there are a number of challenges we will have to overcome. First there's the Key infrastructure. At both ends the keys must be safely stored. The private keys must be safely distributed to the appropriate agents, while the public keys must be stored inside the vehicle in a way they can't be altered (except maybe by someone with a very high permission level). Secondly there's the authentication protocol, which proves the ownership of the appropriate private key. This protocol should prove the ownership of the private key while still maintaining it's secrecy. There should also be some king of session management. Moreover the CAN protocol only allows for 8 bytes of data per message so multiple messages should be involved in it's operation. Thirdly there's the permission table, which should also be stored in a way that it's impossible to alter from outside, since this would undermine the entire protocol's purpose (Anyone could just insert a self declared key and get the highest clearance level). Lastly there's a lot of technical issues that will become apparent as implementation begins. These are mostly relating to performance (e.g. the gateway should give priority to action and informative messages so as not to break the timeliness of the entire system).

References

[1] Lee Pike, Jamey Sharp, Mark Tullsen, Patrick C. Hickey and James Bielman, *Securing the automobile: A comprehensive approach*, 2015.

[2] Dan Klinedinst, Christopher King, *On Board Diagnostics: Risks and Vulnerabilities of the Connected Vehicle*, 2016.

[3] Pierre Kleberger, *On Securing the Connected Car*, 2015.

[4] Brian Russell, Aaron Guzman, Paul Lanois, Drew Van Duren, *Observations and Recommendations on Connected Vehicle Security*, Cloud Security Alliance Internet of Things Working group, 2017.

[5] Charlie Miller, Chris Valasek, *A Survey of Remote Automotive Attack Surfaces*, 2015.

[6] Aastha Yadav, Gaurav Bose, Radhika Bhange, Karan Kapoor, N.Ch.S.N Iyengar, Ronnie D. Caytiles, *Security, Vulnerability and Protection of Vehicular On-board Diagnostics*, 2016.

[7] https://en.wikipedia.org/wiki/On-board_diagnostics.

[8] https://en.wikipedia.org/wiki/OBD-II_PIDs.

[9] https://en.wikipedia.org/wiki/CAN_bus.

[10] Charlie Miller, Chris Valasek, *CAN Message Injection*, OG Dynamite Edition , 2016.

[11] Charlie Miller, Chris Valasek, /textitAdventures in Automotive Networks and Control Units.

[12] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage University of California, San Diego Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno, University of Washington *Comprehensive Experimental Analyses of Automotive Attack Surfaces.*

[13] Stephanie Bayer, Thomas Enderle, Dennis Kengo Oka , Marko Wolf. *Security Crash Test Practical Security Evaluations of Automotive Onboard IT Components*, 2015.

[14] European Union Agency for Network and Information Security (ENISA), *Cyber Security and Resilience of smart cars*, 2016.

[15] Allen Lyons, California Air Resources Board , *On-Board Diagnostics (OBD) Program Overview*, 2015 .

[16] Marien Saarinen, http://www.autoexpress.co.uk/car-news/consumer-news/92304/car-hacking-study-shows-over-100-models-at-risk, *Car hacking: study shows over 100 models at risk*, 2015.

[17] *ISO 14229-1:2013: Road vehicles Unified diagnostic services (UDS) Part 1: Specification and requirements.* ISO, 2013.

[18] *ISO 13400-1:2011: Road vehicles Diagnostic communication over Internet Protocol (DoIP) Part 1: General information and use case definition.* ISO, 2011.

[19] Alexei Sintsov, *(pen)testing vehicles with CANToolz*, 2016.

[20] Philip Koopman. *A case study of toyota unintended acceleration and software safety.* Public seminar, September 2014.

[21] *Guidelines for the Use of the C Language in Critical Systems.* MISRA, 2004.

[22] Koen Claessen and John Hughes. *QuickCheck: A lightweight tool for random testing of haskell programs.* In Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP 00, pages 268279. ACM, 2000.

[23] Bart Jacobs, Frank Piessens, *The VeriFast Program Verifier*, Department of Computer Science, Katholieke Universiteit Leuven, Belgium.

[24] Olaf Pfeiffer, Christian Keydel, *Scalable CAN security for CAN, CANopen and other protocols*, 2017.

[25] Job Noorman, Pieter Agten ,Wilfried Daniels ,Raoul Strackx Anthony Van Herrewege, Christophe Huygens, Bart Preneel, Frank Piessens ,Ingrid Verbauwhede, *Sancus: Low-cost trustworthy extensible networked devices with a zero-software Trusted Computing Base*, iMinds-DistriNet and iMinds-COSIC, KU Leuven.

[26] Jan Tobias Muhlberg, Frank Piessens, Jo Van Bulck, *VulCAN: E icient Component Authentication and So ware Isolation for Automotive Control Networks*, Imec-Distrinet KuLeuven, 2017.

[27] Andreea-Ina Radu, Flavio D. Garcia, *LeiA: A Lightweight Authentication Protocol forCAN*, School of Computer Science,University of Birmingham, UK.

[28] Stefan N urnberger, Christian Rossow, *vatiCAN, Vetted, Authenticated CAN Bus*, CISPA, Saarland University, Germany.