Universidade Federal de Pernambuco
Centro de Informática
Departamento de Informática

Graduação em Engenharia da Computação

In-Vehicle Network Security: Attacks and Countermeasures

Caio Augusto Pererira Burgardt

Trabalho de Graduação

Recife
10 de dezembro, 2018

Universidade Federal de Pernambuco
Centro de Informática
Departamento de Informática

Caio Augusto Pererira Burgardt

In-Vehicle Network Security: Attacks and Countermeasures

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Departamento de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Divanilson Rodrigo Campelo*

Recife
10 de dezembro, 2018

To my family and my love.

Acknowledgements

I'd like to thank my mother, Cristiane, and my father, Otávio, for all these years looking after me and supporting my decision of coming to Recife to study in UFPE. I'd also like to thank my brother and best friend, Victor, for always providing me company and never letting me give up when times got tough. I'd like to thank my girlfriend and love, Camilla, for putting up with me for years and for being the best part of me. I'd like to thank Professor Divanilson for always pushing me harder to accomplish more and giving me the honor of being part of his research team. I'd like to thank Thomas Hogenmueller and Timo Lothspeich for showing me the automotive industry. I'd like to thank the people at Tempest Security Intelligence for the guidance and opportunity to work in the information security field.

If cryptography is outlawed, bayl bhgynjf jvyy unir cevinpl
—ANONYMOUS

Resumo

Redes automotivas estão crescendo em tamanho e complexidade. Essas redes utilizam várias tecnologias diferentes e agora a indústria está adicionando Ethernet aos veículos. Para proteger qualquer ambiente conectado, são necessárias soluções de segurança para lidar com as possíveis ameaças para cada tecnologia utilizada. Ataques recentes mostraram que as redes automotivas, apesar de parecerem isoladas, podem ser atacadas com uma certa facilidade; se o atacante tem acesso à rede interna, não existem soluções de segurança eficientes para impedi-lo de afetar funções de um automóvel. Neste trabalho de graduação, serão estudados ataques e medidas que a indústria automotiva vem adotando para se proteger. Também será proposto um modelo de segurança para a Ethernet automotiva baseado em MACsec.

Palavras-chave: Redes Automotivas, Segurança, Ethernet, CAN, MACsec

Abstract

Automotive networks are growing in size and complexity. These networks use several different technologies and now the industry is adding Ethernet to the vehicles. To protect any connected environment, security solutions are needed to deal with potential threats to each technology used. Recent attacks have shown that automotive networks, although isolated, can be attacked with some ease; if an attacker has access to the internal network, there are no effective security solutions to prevent him/her from affecting the functions of the automobile. In this graduation work, we study attack cases and counter measures that the automotive industry has been adopting to protect itself. We also propose a security model for automotive Ethernet based on MACsec.

Keywords: Automotive Networks, Security, Ethernet, CAN, MACsec

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Today's cars can be considered as computer networks on wheels . The nodes in these networks are typically Electronic Control Units (ECUs) responsible for various car features. Even though ECUs are often vulnerable devices with many security flaws, intra-vehicular networks used to be an isolated, hard-to-reach attack surface, which makes cars unattractive targets to hackers. This belief that cars would always be too simple and too isolated to be attacked resulted in automotive networks evolving in complexity and size without the further consideration of security flaws, demonstrated for example in the design of the Controller Area Networks (CAN) protocol [Spe91] . Today's vehicles have many different network technologies within them, but CAN is undeniably the main internal network technology in use and is an ISO standard. However, CAN is a message-oriented transmission protocol that, by itself, does not support authentication or encryption. There are attempts to correct these security features, such as CAN frame authentication in AUTOSAR Secure Onboard Communications [Muc16]. Recent work has shown that once an attacker has access to the vehicle's internal network, he can easily control it [MV15]. Nowadays, with the addition of Internet connectivity and other popular communication systems (eg, Bluetooth), it has been demonstrated that remote attacks can be possible even without physical access prior to the intraveicular network [4], and these new attack surfaces keep appearing more and more often, as shown in Figure 1.1.

The automotive industry is now adding Ethernet as a new in-vehicle network technology in cars, especially after standardizing the 100BASE-T1 Ethernet [MK17]. The flexibility, scalability and popularity of the Ethernet protocol coupled with the cost-effectiveness of the 100BASE-T1 solution make Ethernet an increasingly attractive choice for ECU manufacturers. However, the Ethernet protocol has a considerable amount of security failures present, as it does not have authentication, confidentiality and integrity [KSM13]. The absence of any type of authentication and encryption allows, for example, attacks that are still recurring in home and corporate networks, such as Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP). However, the IT community has over 30 years of experience in defending Ethernet-based networks from attackers, which makes the attacks in these networks much more familiar to security engineers and, in turn, makes our defenses much more efficient.

1.1 Motivation

The current security state of in-vehicle networks is this bad state because cars did not use to have so many surfaces of attack during most part of history. Any attacker that wanted to achieve full car control through CAN injection needed to physically compromise the vehicle first and

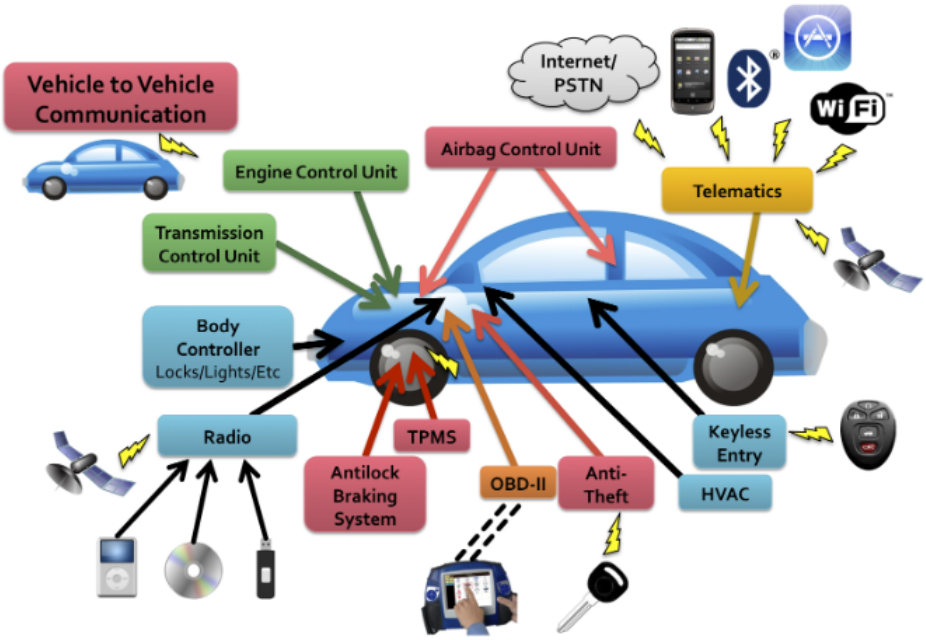

Figure 1.1: New attack surfaces have been appearing and allowing possible access from outside into inside critical systems. Source: [CMK+11]

perform very invasive sabotaging. But nowadays with the many different new surfaces and new application, remote exploits can happen from various different vectors.

While we need to try to protect all the new remote attack surfaces, it's unreasonable to expect that we can make every connectable device perfectly secure. The insecurity of the Internet of Things is like hacking in the 90s and this is leaking into other connected devices, like OBD-II dongles[FPKS15]. Even if we could protect the connectable devices, sometimes we cannot protect the user from himself. The user may download a malware into his/her smartphone that communicates to the bluetooth of the car or buy a maliciously modified hardware to connect to his car. The in-vehicle needs to assume that breaches are possible and could happen any instant.

1.2 Objectives

The objective of this work is to analyze the security of internal automotive networks, focusing on Controller Area Network (CAN) and automotive Ethernet. It will be verified whether the mitigations proposed by the industry and other works really are effective against possible attackers and we'll analyze MACsec as a security strategy in Automotive Ethernet.

CHAPTER 2

Basic Concepts

Before diving into the security side of the automotive network, we first need to understand how the internals of today's vehicle work. Some might say that cars are computer systems, but actually the modern car is a conglomeration of different systems tied together through different network technologies, like the Local Interconnect Network (LIN), Flexray and the Controller Area Network, also known as CAN. The nodes in those networks are the electronic control units (ECU). The ECUs are devices that control the systems present in the vehicle, sometimes dealing with simple sensors, other times dealing with complex calculations.

No network technology is the same. Each of them has been designed for specific applications and has its specific properties, as shown in Table 2.1.

Bus	LIN	CAN	Flexray
Made For	Low Level Subnets	Soft Real-Time	Hard Real-Time
Application Examples	Door Locking HVAC	Driving Assistant Engine Control	Emergency systems
Architecture	Single-Master	Multi-Master	Multi-Master
Transfer Mode	Synchronous	Asynchronous	Both
Data Rate	20 kBit/s	1 MBit/s	10 MBit/s

Table 2.1: Different Automotive Technologies Properties

The CAN network is especially reliable and popular, being used as the main network technology connecting ECU's in the automotive network. CAN has been made an ISO standard called ISO 11898[Sta93]. Cars nowadays ship with an On Board Diagnostics port (OBD-II), which connects itself directly to the vehicle CAN bus and allow for diagnostic communications with the ECUs.

2.1 Controller Area Network (CAN)

The Controller Area Network as a protocol is a message-oriented bus communication protocol. It's a standard since 1993 and almost every car nowadays has it as its main network technology.

2.1.1 Network Architecture

The CAN network is a bus, which has a broadcasting behaviour. It's message-oriented, which means there is no address for source or destination in the frames. The frames only have a message ID and devices connected to the bus choose which frames are important through the message ID. With these two characteristics, any device connected to the bus can read and inject frames freely[KCR+10].

With bus-based technologies, there is the need to avoid collision and desynchronization of communications. If two ECUs try to send frames at the same time, if there is no arbitration mechanism, one might overwrite the other and both of them can lose their frames. In the CAN bus, there is the concept of dominant and recessive bits, in which the bit zero (0) is the dominant, and the bit one (1) is the recessive bit. Whenever two bits are being sent at the same time, the dominant bit will be the one actually written in the CAN bus because of the physical properties of the bus. We can see the bus as an `AND` gate with the inputs as the bits two different ECUs send at the same time, like in Table 2.2.

ECU A	ECU B	CAN
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.2: The CAN bus behaviour during conflicting bit transmissions

2.1.2 Frame Format

In CAN there are four types of frames: the data frame, the remote frame, the error frame and overload frame. The data frame is the main frame as it carries data from transmitters to receivers and plays the most important role in a CAN network. The remote frame works as a request-like CAN frame, it is sent into the bus and the device which can respond to it sends a data frame with the data requested. CAN uses the method of bit stuffing to code bit streams. During the transmission, whenever a transmitter detects five consecutive equal bits, it transmits the opposite bit in the stream. In the data frame and remote frame, all data fields are bit stuffed, except for the ACK field, the end of frame and the delimiter in the CRC.

The error frame and overload frame are very similar and play a secondary role in the CAN ecosystem. The error frame is a sequence of dominant bits sent by a device that detects an error in the CAN bus, like a bit transmission error when a transmitter is sending a recessive bit out of the arbitration field and reads a dominant bit on the bus. The overload frame just provides extra delay between two frames.

2.1.2.1 Data Frame

In this work, we'll focus on the data frame, which is the only frame that actually carries data. As we can see in Figure 2.1, every bit in the frame has a meaning and understanding the frame formatting will be crucial to evaluate security characteristics of messages. The remote frame looks almost exactly like a data frame with no data field.

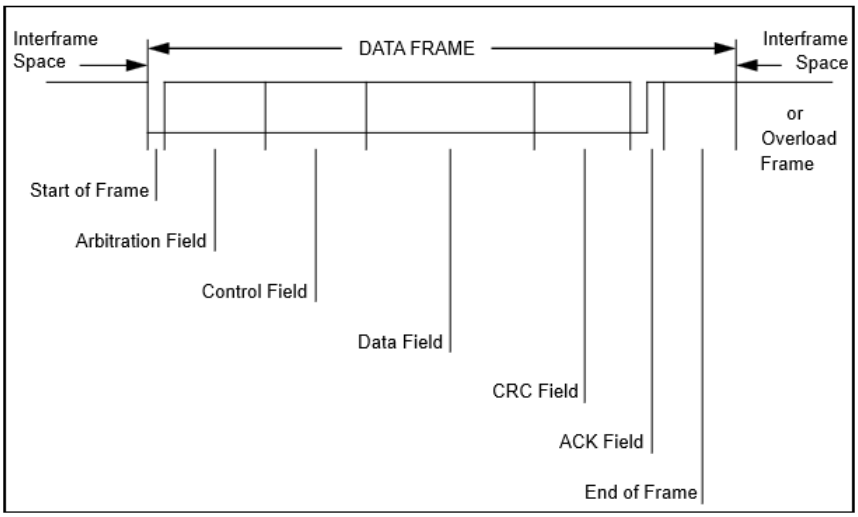

Figure 2.1: CAN's Data Frame Format. Source: [Spe91]

2.1.2.1.1 Start of Frame It's a single dominant bit which marks the beginning of the frame. It's used for synchronization among the devices connected to the bus.

2.1.2.1.2 Arbitration Field The arbitration field is composed of an Identifier or Message ID of 11 bits, which identifies the content of the message, and a single Remote Transmission Request bit, which if recessive (1), signifies that the frame is actually a remote frame and not a data frame. The arbitration field defines message priority, the messages with dominant bits in the more significant bits will have the priority and devices sending messages with less priority will cease transmission. CAN also allow for extended frames where the identifier is 29 bit long.

2.1.2.1.3 Control Field The control field is 6 bits long. Four of which determine the length of data in bytes, and two are reserved for future extensions in CAN.

2.1.2.1.4 Data Field The field which contains the data. Its length depends on the control field, ranging from 0 to 8 bytes. The remote frame type carries no data.

2.1.2.1.5 CRC Field A quick frame check sequence calculated over the frame when a device is receiving a frame transmission. The receiving device calculates the CRC value and checks with the received value to check for possible transmission errors. The CRC size has a fixed

value and a last recessive bit as a CRC delimiter

2.1.2.1.6 ACK Field Acknowledgment information with two bits. The first bit the sender sends as recessive. When a device receives the frame, it sends a dominant bit to overwrite the sender's recessive bit. This way the sender can check whether or not some device has actually received the transmission. The second bit is always recessive. It works as a single bit ACK delimiter.

2.1.2.1.7 End of Frame The end of frame is made of seven recessive bits sent at the end of the transmission of a frame, determining the end of transmission.

2.1.2.1.8 Interframe Space Data frames and remote frames are separated from preceding frames by at least three recessive bits which delimit the interframe space.

2.1.2.2 Other Frames

The error frame has two different fields. After an error is detected by a device, the device sends 6 dominant consecutive bits as an error flag to trigger a bit stuffing error in the other devices. The second part is 8 recessive bits after the error flag.

Overload frames have two bit fields and happen right after the end of another frame. It destroys part of the interframe space by injecting six dominant bits. After that the the devices send 8 recessive bits just like the error frame.

As mentioned above, the main type of frame is the data frame as it is the only frame with actual data.

2.1.3 CAN with Flexible Data-rate (CAN FD)

Standard CAN has the low data transmission efficiency of 8 bytes per frame. In order to achieve a higher throughput, CAN FD has been created, which allows for a more frequent bit sampling during the frame's data and CRC field. The CAN FD also defines a data field of up to 64 bytes in size and a bigger CRC field in result[H$^+$12].

A higher data rate allows for shorter bits in CAN and makes the new length of 64 bytes have a smaller overhead. There are no security impacts in this change, but the extra data rate and data field space can be helpful with the implementation of security protocols if we are deploying signatures or padding for encryption[WJKL16].

2.2 A Brief Security Analysis of CAN

In order to standardize the security development of systems, there have been proposed many models that enumerate requirements or objectives that should be achieved to determine if a system is secure or not during its design. The CIA model one of the oldest and most famous one, getting its name from the three basic security objectives: Confidentiality, Integrity and Availability [RKJ06]. We use this framework and some of its extensions to analyze the security

of CAN as a communication protocol.

2.2.1 Versus the CIA Model

2.2.1.1 Confidentiality

Achieving confidentiality means making sure that only authorized systems and users can read secret data. Normally, this requirement is achieved in the form of encryption with strict key management. This way even if data is intercepted in transit by an attacker, the attacker can't learn anything from it.

There is no support in CAN for encryption and no standard strategy for key management in the CAN bus network. Confidentiality is not a major concern in the in-vehicle automotive environment, especially inside CAN, where the data inside the frames mostly pertain to the vehicle state and not very sensitive information.

2.2.1.2 Integrity

Integrity is the requirement that defines that an attacker cannot change the content of a message without warning the destination with some indication of tampering. When integrity is maintained, the tampering becomes obvious to the original devices in the conversation, so they can just discard it and the tampering does not matter.

While there is a CRC which checks for possible mistakes and wrong bits of the data in the frame, it doesn't do much against a malicious attack. The cyclic redundancy check was designed to check for non-intentional damage in data, an attacker could easily bypass the calculation as the CRC uses no cryptographic primitives.

If an attacker does try to alter the frame in transit, an error would be easily detected by the sender as he would notice the bit change. But nothing stops the attacker from replaying modified messages when he wants.

2.2.1.3 Availability

Availability is when an attacker cannot generate a denial of service on the data access, or in this case, communications. By sending dominant bits consistently in his frame's message ID, the attacker can get his frame to always win the arbitration and stop every other frame to be sent. Attackers don't even need to send frames with high priority, they can just create errors by sending dominant bits during any recessive bits to stop the current frame in the bus to be sent.

2.2.2 Versus other security requirements

The CIA model has been criticized for being antiquated and very simplistic, not addressing requirements that modern application in the Internet may have. There have been several attempts of changing the requirements or adding a couple more.

2.2.2.1 Authentication

A system with authentication can prove the authenticity of the source of data or message. This is one of the gold standards of communication, normally achieved through signatures with a strict key management.

The CAN standard really struggles with this security requirement due to the fact that the protocol is message-oriented, and there is no information about the origin of the frame inside the frame.

Authentication is likely the most important security requirement due to the easiness of traffic injection in CAN. While CAN has no authentication support, there has been some techniques trying to achieve CAN authentication. We discuss them in the next chapters.

2.2.2.2 Others

There are many other security requirements left, such as privacy and non-repudiation. As mentioned before, CAN traffic is almost all vehicle related, which means it holds almost no sensitive information. Traffic from inside the car rarely leaks to outside the in-vehicle network, making privacy not a priority in the CAN security scenario. Which is not to say that privacy issues can't be exploited, there is research which shows that the fingerprinting of a driver from the CAN behaviour is entirely possible and attaching devices to your in-vehicle network could hurt your privacy [ETKK16].

Security requirements like non-repudiation or deniability aren't considered priority for CAN. Such requirements are much more application-related and the security of in-vehicle network traffic wouldn't benefit anything with adding technologies to cover these requirements. The lack of basic CIA model requirements is a much bigger issue.

2.2.3 Last Thoughts

The protocol is incredibly robust, but without any security features built-in. The architectural choices of a broadcast bus network with a message-oriented protocol makes it very hard for security engineers to adapt known technologies to the in-vehicle environment. Thanks to the lack of authentication, once an attacker gets access to the CAN bus, every frame the attacker injects will be read as a legitimate frame, possibly gaining full control of the vehicle's functions including cyber physical functions like steering, braking or accelerating[MV13].

The automotive engineers bet on the fact that CAN is just too isolated for an attacker to actually gain access to, but thanks to the growing trend of the Internet of Things, there are new attack surfaces appearing. In the next chapter, we show some vulnerability researches that resulted in remote attacks and demonstrate that CAN offers almost no post-exploitation defenses.

CHAPTER 3

Attacks on Automotive Networks

Before studying defense techniques, we should analyze and study real attack cases so we can have a better understanding of the threat model and security impacts of the vulnerabilities and flaws in automotive networks.

3.1 The Jeep Hack

Figure 3.1: The 2014 Jeep Cherokee. Source: [MV15]

This attack is unquestionably the most impressive one. The security researchers Charlie Miller and Chris Valasek have shown to the world that by exploiting vulnerabilities found in the Head Unit of a Jeep Cherokee, they could assume control of the car[MV15]. To the best of our knowledge, this was the first time researchers performed a public security analysis of an automobile which resulted in fully remote control without any prior physical tampering of the target.

The target in this attack was a 2014 Jeep Cherokee, as the one shown in Figure 3.1. As the Figure 3.2 show, the head unit (Radio) is connected to two CAN buses in the car. So the main objective of the researchers was developing an attack which gains control of that device, so they could inject to both networks and have their messages reach every ECU in those networks.

The Jeep Cherokee uses the Uconnect 8.4AN/RA4 radio manufactured by Harman Kardon as its head unit and only infotainment system. This system runs the QNX operating system and is available Automobiles. This head unit has some interesting surfaces of attack. It has a USB port, a WiFi interface and a cellular network interface with a globally reachable Sprint IP address. The researchers focused on the reverse engineering of the head unit and the different services running on it, finding enough vulnerabilities to create a remote exploit that results in the full control of the car.

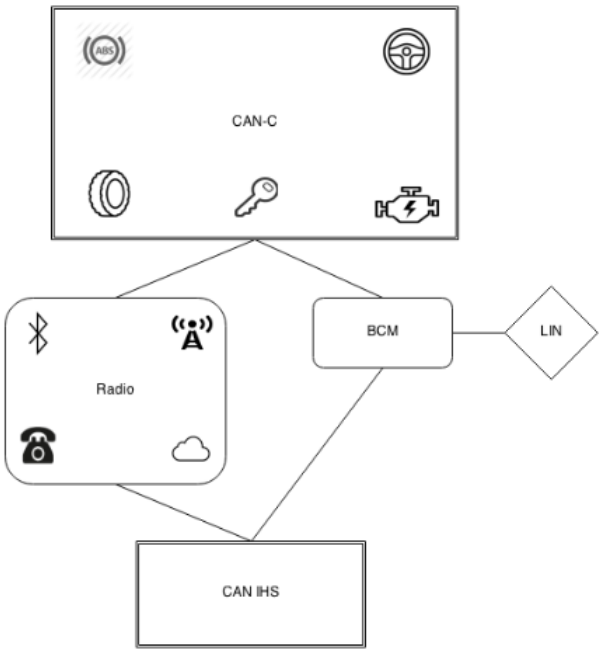

Figure 3.2: The 2014 Jeep Cherokee network architecture, with the Radio as the head unit connected to two different CAN buses. Source: [MV15]

The exploit works as follows: firstly, an attacker has to discover the IP address of a target vehicle. An attacker could just scan the Sprint network after a vulnerable vehicle. After that, the attacker can use the running unauthenticated D-Bus service on port `6667` to execute arbitrary shell commands. At this point of the exploit the attacker already has some control of the car, such as retrieving GPS coordinates, controlling the radio volume, the radio station, the air conditioning and even changing the display in the head unit. In order to control the cyber physical actions of the car, such as steering or braking, the attacker needs to remotely re-flash the head unit with a modified firmware that allows the injection of those CAN messages into the buses. This can be done through the command injection in D-Bus, putting the head unit into the update mode and restarting it with a new modified firmware. If all those steps were

```
# netstat -n | grep LISTEN
tcp        0      0  *.6010                 *.*                    LISTEN
tcp        0      0  *.2011                 *.*                    LISTEN
tcp        0      0  *.6020                 *.*                    LISTEN
tcp        0      0  *.2021                 *.*                    LISTEN
tcp        0      0  127.0.0.1.3128         *.*                    LISTEN
tcp        0      0  *.51500                *.*                    LISTEN
tcp        0      0  *.65200                *.*                    LISTEN
tcp        0      0  *.4400                 *.*                    LISTEN
tcp        0      0  *.6667                 *.*                    LISTEN
```

Figure 3.3: Open ports in Jeep's head unit. Source: [MV15]

successful, the attacker has gained full control of the car.

While this attack is very interesting by itself, the researchers didn't need to use all their findings to perform it. Among other remote surfaces in the jeep, the researchers also tested its Wi-Fi capabilities. The 2014 Jeep Cherokee has the option of an in-car Wi-fi, which the owner can have as a hotspot accessible after paying the service. The researchers found out that the Wi-Fi system allowed for the usage of WEP encryption and even no encrypion. The Wireless Encryption Protocol (WEP) has been known to be broken for years and there is no good reason to not use encryption in a car hotspot. Even though WPA2 is the default, the researchers found that, through disassembling the binary 'WifiSvc' from a head unit's chip, they found predictable WPA2 key generation for the WiFi.

They discovered the random key generation is actually just a function of the epoch time in seconds, they state the evidence says that the time used is the moment the head unit first boots up. Even though it's hard to discover the exact moment of the first head unit's booting, based on the year of the car it's easy to generate a password list to conduct a brute force attack against the WPA2 protocol. The researchers state that if you could guess what month the vehicle first started, you'd only have to try around 15 million passwords assuming they're never started in the middle of the night. They estimate an attacker could realistically crack the WPA2 password in 2 minutes knowing the month and less than half an hour knowing the year.

However, there is another, even easier, way to crack the WPA2 password. As soon as the head unit start for the first time, it needs to know what time it is since it has no GPS or cellular connections, but inside the head unit there is a Lua script that sets the time to Jan 1 2013 00:00:00 GMT if the head unit fails to get the time. Apparently, on the researchers Jeep, the clock still hadn't change during the WPA2 key generation. Their WPA2 key was generated by the clock Jan 01 2013 00:00:32, showing that it took 32 seconds from the head unit's first booting to the generation of the WPA2 key. If the researchers found their test car like this, there might be more vehicles with highly predictable Wi-Fi keys. The researchers state that a brute force attack exploiting this vulnerability only has to test a couple dozen keys, allowing the key to be cracked almost instantaneously.

There is also a vulnerability that allows for insecure updating through a USB port. If you connect a valid ISO file for the Uconnect system, the system will recognize it and begin the updating process. The system soon reboots, if you remove the USB stick before the boot, the update fails, but if you remove it when the system is powered down during the reboot process,

it asks you to insert back the USB stick and you can insert a different USB stick. While the researchers aren't sure about what verifications the unit runs on the new ISO, they discovered that it has to be similar to the original for it to work. However, it still allowed them to change the root password and the code that verifies the ISO, allowing for further modifications.

This research resulted in the recall of over 1.4 million vehicles. The authors also mention that the exploit is wormable as it requires no interaction and the attack vector is scannable in Sprint networks. This means that an attacker could develop a malware to automatically start infecting vulnerable cars and inject dangerous traffic in their systems.

3.2 Bosch Drivelog Connector System

Figure 3.4: Bosch Drivelog Connector's dongle and smart phone application. Source: https://americansecuritytoday.com/wp-content/uploads/2017/04/Bosch-Drivelog.jpg

Argus Cyber Security is an Israeli automotive cyber security company. In one of their researches, they performed the security analysis of a Bosch product, the Drivelog Connector shown in Figure 3.4 [ARGb]. The system has two parts, an OBD-II dongle and a smartphone application. The dongle is a small device to be plugged into the OBD-II port of the car. The device then sends diagnostic information to the user through an app on their smart phone.

Their research found two major vulnerabilities, which when exploited together, an attacker could gain some degree of CAN frame injection into the bus connected to the exposed OBD-II port. The first vulnerability is the low effectiveness of the message filter in the dongle. The second one was an information leak in the authentication process between the dongle and smart

phone application.

The smart phone connects to the Drivelog Connector dongle through Bluetooth. The researchers tried listening to the bluetooth communication only to discover that it is encrypted, so they reverse engineered the application protocol through patches in the smart phone application to replicate all communication to a UDP port. They discovered that the smart phone made requests to the dongle choosing the input parameters for injection of messages into the OBD-II port. There was a message filter for one parameter, namely the one that controls which OBD-II PID will be used. The OBD-II PID tells which values are being diagnosed. However, the CAN message ID was also controlled by the smart phone, so even though there was a message filter to stop invalid diagnostic messages for OBD-II, the attacker could inject OEM specific messages that pass the filter and affect the vehicle in different ways.

For an attacker to explore the previous vulnerability, he/she needs to make the users smart phone send these messages into the Bluetooth communication with the dongle somehow. This could be achieved by making the user download a fake Drivelog Connector app that is actually modded to exploit that vulnerability. Another option is for the attacker to inject the communication directly into the dongle. This would be unviable if it wasn't for the second vulnerability.

During the authentication process between the smart phone and dongle, there is a major information leak that allows any attacker that tries pairing to effectively retrieve the PIN used for authentication and allow them to authenticate to the dongle themselves. The authentication process is shown in Figure 3.5

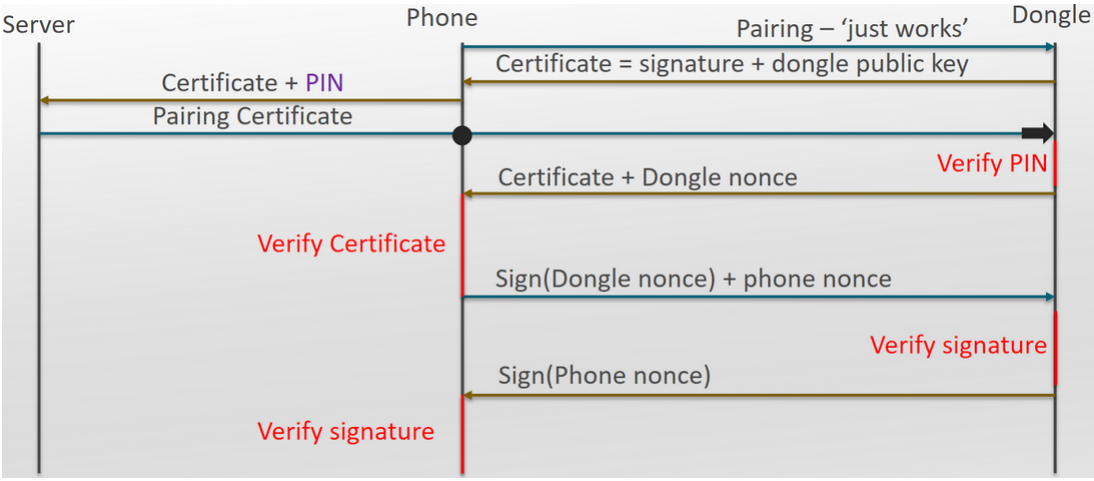

Figure 3.5: Bosch Drivelog Connector's authentication diagram between smart phone and dongle. Source: [ARGb]

The authentication requires communication with a back-end server that generates a pairing certificate which shows the dongle that the back-end server which verifies if the knows the correct PIN for the received dongle certificate. But this dongle certificate is just a SHA-256 calculated over the dongle's MAC address, the dongle's public key and the PIN. The attacker already knows MAC address of the dongle through the Bluetooth pairing process and already receives the certificate and public key as soon as the Bluetooth pairing is over. That leaves the

attacker with an entropy of 8 digits, which is only 100 million tries to guess the correct PIN to access the dongle.

Using both vulnerabilities, an attacker first pairs with the Drivelog dongle and receives the dongle certificate. Then the attacker can perform an offline brute-force to figure out the PIN and connect to the dongle. After that, the attacker can inject malicious CAN bus messages that are allowed through the message filter.

3.3 Progressive Snapshot driving tracking tool

Figure 3.6: Progressive Snapshot driving tracking tool. Source: https://www.repairerdrivennews.com/2017/05/17/progressive-usage-based-insurance-is-the-future-likely-assisted-by-oem-data/

This tracking tool is an OBD-II dongle used by Progressive, an insurance company, with the objective to monitor the driver's driving behaviour in order to adapt the prices of insurance. This way better drivers, i.e. drivers who drive more safely than others, would be eligible to receive discounts.

The researcher who analyzed this device, discovered that the dongle contained a cellular modem that sent vehicle data to Progressive's systems. In this communication between the dongle and the servers, there were no encryption or authentication mechanisms. Moreover, the dongle supports firmware updates from the server with no signature or security measures to assure integrity or authentication[Thu].

In order to use the Progressive Snapshot dongle as an attack vector, the attackers have two possible options: either simulate a base station and get in a man in the middle position to pretend to be the Progressive servers, or hack into the Progressive servers themselves. After that, the attackers can behave like in the Jeep hack and change the dongle's firmware into a malicious firmware that is able to perform specific attacks through messages injected into the OBD-II port.

3.4 Zubie

A very similar device to the Progresive Snapshot is the Zubie. It's a safety enhancing device that you plug into the OBD-II port of your car. It connects to the Zubie servers and shows

Figure 3.7: The Zubie dongle. Source: https://zubie.com/features/

the data collected in the users smart phone. Its main objective is improving the driver's safety through tracking of car's performance and location to offer advices on how to become a safer and more responsible driver.

Argus has found out that the device suffers from the same vulnerabilities that the Progressive Snapshot does: the device was not using proper encryption or authentication mechanisms [ARGa]. The system was using standard HTTP as the communication protocol between the servers and the dongle, with unsigned firmware updates.

An attacker could easily exploit this lack of server authentication through a DNS spoofing, a fake cellular base station and in many other scenarios that allows the possibility of impersonating the Zubie servers. After that, the attacker could exploit the vulnerability of not having signed firmware updates to upload his/her own malicious firmware to control the victims car.

CHAPTER 4

In-Vehicle Network Defenses

In this chapter, we present some techniques the automotive industry has been considering to stop the attacks and mitigate the vulnerabilities, like the ones mentioned in the last chapter.

4.1 Secure Boot

In the Jeep hack, a crucial step of the exploit chain required the upload of a maliciously customized firmware to the Head Unit. In order to forbid the running of this software, the OEM's could implement a mechanism called Secure Boot.

Secure Boot is not an automotive industry exclusive idea, in fact, it's supported by most BIOS in home PCs. It's a Unified Extensible Firmware Interface (UEFI) mechanism which allows only software with valid signatures to be booted in the machine it's working on. The manufacturer loads some databases of keys and signatures in the device in manufacturing time. The firmware would then be signed and have its signature checked before booting.

Secure booting is a very strong security mechanism that stops malicious firmware from being booted, but previously there have been vulnerabilities in specific implementations that allowed for bypasses in some occasions, which makes secure booting not a silver bullet.

4.2 Security Access Service

In Chris Valasek and Charlie Miller's research, they have encountered a diagnostics authentication mechanism when reverse engineering a 2010 Toyota Prius and a 2010 Ford Escape[MV13].

This mechanism is called Security Access. It's part of the Unified Diagnostics Service (UDS) defined in ISO 14229-1[Sta13]. The UDS is a standard for a diagnostics communication protocol for ECUs in the automotive environment. It defines a series of services to retrieve information about the car's functionality and state. One of the services defined is the Security Access.

The service works as a simple Challenge-Response protocol. Firstly, the tester sends a message requesting a "seed" to the ECU, which in return, randomly generates it and sends it to the tester. The tester and the ECU should both share a cryptographically secure function and a key, which when applying the key and the seed to the function it generates a response which is difficult for an attacker to guess. So the tester sends the generated response and the ECU checks if it's the right response, if it is, the tester can now perform the related diagnostic actions.

Sadly, the standard is very loose and has no information on what the function and keys should be or how the seed is generated. This leaves plenty of room for error in implementations

of the Security Access service. Different ECUs had different implementations which in turn had different issues.

For example, in the 2010 Ford Escape, the Parking Aid Module ECU generated the same seed every time. This behaviour with the fact that the response is only 24 bits long, means that an attacker could simply brute force the response or replay it from a legitimate Security Access session. For the other modules in the Ford Escape, the ECUs would always generate different seeds, so the researchers reverse engineered the Ford Integrated Diagnostic Tool software to extract the keys with success.

In the Toyota Prius, most diagnostic actions do not even require authorization through the Security Access. The ECUs that do, generate different seeds every 10 failed authentication tries or whenever the car starts, which makes it difficult to do an online brute force. However, through reverse engineering a Toyota tool called the Toyota Calibration Update Wizard, the researchers found out that the function used to generate the response uses only two of the four bytes in the seed to generate the response, severely cutting down the entropy of the entire system. In the end, the researchers successfully retrieved three keys.

4.3 Secure Onboard Communications

In order to tackle the lack of authentication in CAN, the AUTOSAR community proposed a strategy to stop malicious traffic injection directly in the CAN bus. The Secure Onboard Communications (SecOC) is a module which introduces CAN frame authentication in the automotive environment through the addition of signatures to the in-vehicle communications[SECb]. The module works with both symmetrical cryptography and asymmetrical.

To authenticate the frame, SecOC already assumes key management and exchange has been taken care of. The technique is simply appending a signature to the protected data unit (PDU) in the CAN frame. This strategy requires a freshness value to maintain a degree of uniqueness of signatures so replay attacks are unfeasible. For example, in symmetrical mode, in order for a device to send a message to another, the sender calculates a message authentication code (MAC) over the input data and the freshness value using a shared key, and appends the calculated signature with the freshness value. The receiver then checks if the freshness value is correct and recalculates the MAC from the message received, if it's correct, the receiver accepts it. The entire scheme is represented in Figure 4.1.

Even though it is a very simple strategy, when implementing this system in CAN there is the issue of fitting all this into the regular CAN frame. With the regular CAN frame we need to truncate the 128 bit MAC into a 27 bit value, which is vulnerable to brute force attacks.

There are two suggestions of possible ways to maintain the freshness of signatures, either by time stamps or monotonically increasing counters. When using counters, each device should store the counter values for each type of the message, in CAN, this means a different counter for every different message ID. Both key management and synchronization of freshness values are not specified in SecOC, which could be an issue since desynchronization could happen even without an adversary[SECa].

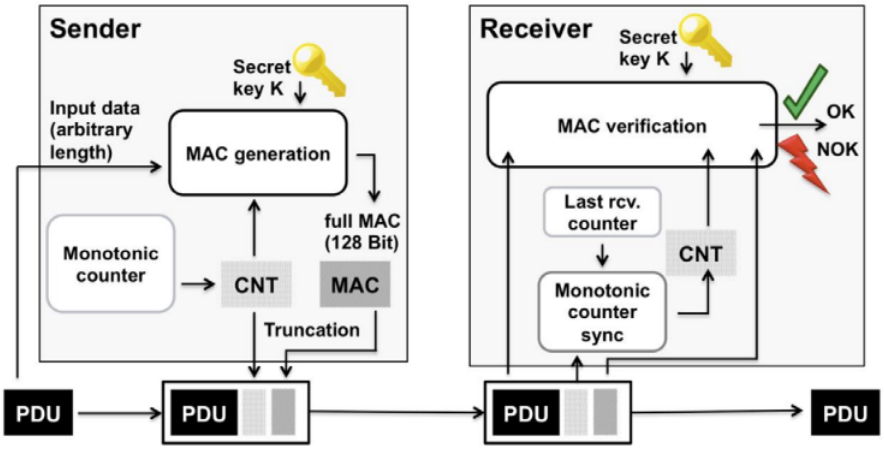

Figure 4.1: Symmetrical mode SecOC working with a counter as its freshness value. Source: [SECa]

4.4 Plug and Secure Key Establishment

Having two devices in a bus exchanging keys frequently is still a challenge in CAN. The Plug and Secure Key Establishment (PnS) is an elegant and low-cost theoretical symmetric key establishment protocol for CAN that exploits a physical property in the bus to define a key between two devices[ML15].

As mentioned in Chapter 2, the CAN bus works like an `AND` gate. Whenever two devices transmit a bit at the same time, the dominant bit (zero) overwrites the other and becomes the bit actually seen in the bus by the other nodes.

Suppose that two devices, Alice and Bob, are sharing the same CAN bus and want to share the same secret value. Following the PnS key establishment protocol, Alice and Bob generate a random string of bits each. Then for every bit in the string they add the inverse of that bit right after it, for example, the secret bit string `0 1 0` becomes `01 10 01`. After that, both parties transmit the generated strings at the same time. Thanks to the previously mentioned physical property, the bits are going to mix and overwrite each other. Alice and Bob are going to read which bit tuples have a recessive bit (1) in it, and discard them from the final secret. Then Alice and Bob just revert to the original secret but without the bits removed in the last step. This way, both devices have a secret bit string and know that the other has the same secret, but with opposite bits. In Figure 4.2, we can see an example of the PnS key establishment protocol step-by-step, but with small key-sizes.

A third, maybe malicious, device in the bus cannot learn much from the protocol alone. An `AND` gate has a bias toward the dominant bit (0), but that is removed with the step in the protocol which discards the tuples with a recessive bit. The only thing that a malicious device can tell in this level of abstraction is that the keys are the opposite of each other.

Even though this method of key establishment is amazingly simple and low cost, there is no implementation model of how it would even fit the CAN standard. There is much work to be

```
1) Generate Bit Strings:
a = 0 1 1 0 1 0 0 1 0 1
b = 1 0 1 1 0 1 0 1 1 0
2) Extend Bit Strings:
a_ext = 01 10 10 01 10 01 01 10 01 10
b_ext = 10 01 10 10 01 10 01 10 10 01
can   = 00 00 10 00 00 00 01 10 00 00
3) Define Shared Secret:
key_a = 0 1 0 1 0 0 1
key_b = 1 0 1 0 1 1 0
```

Figure 4.2: The steps of Plug and Secure Key Establishment.

done when considering how to use this technique on real CAN networks, including synchronization, key management, implementation and performance analysis.

The protocol is reliable, even with its randomness which may create some bad situations with keys smaller than others. But the protocol itself bets on a physical property to protect confidentiality of the initial values transmitted. This exposes the protocol to side channel attacks involving differences in bit timing or different voltages [JWAG18], which could possibly recovery entire keys. However, this is a very minor issue, as the protocol suggests a new random key in every instance and attacks like this require physical access directly to the CAN bus, which would mean physical security is already broken and most security mechanisms can't protect against a physical attacker.

4.5 Intrusion Detection Systems

If we cannot stop an attack, we could at least try detecting it. An intrusion detection systems (IDS) is a device or software which detects anomalies in a network, aiming to evaluate in real time if there is an ongoing attack in that same network.

In the automotive environment, we could use many different traffic features to detect attacks. In the early works of Chris Valasek and Charlie Miller, their attacks on the Toyota Prius and Ford Escape involved injecting frames directly in the CAN bus with a much higher frequency then the expected for those messages[MV13]. An IDS could detect drastic differences in frequency for the type of the message sent and possibly warn other systems in the car of the possible security breach. There are many other features that could be used for detecting ongoing attacks, such as unexpected message ID's or physical features like voltage difference or propagation delays.

In Figure 4.3, we can see the times the frame with the CAN ID `0210` had a appeared in one second, showing the variation in frequency. In the graph, we see that the message with the CAN ID `0210` has showed up over 90 times with frequency of 28 frames per second. However, in the attacks that the researchers used this frame, they had to inject with a frequency 10 or 20 times than that[MV13], making it very easy to detect their attacks through frequency analysis of the appearance of the CAN ID.

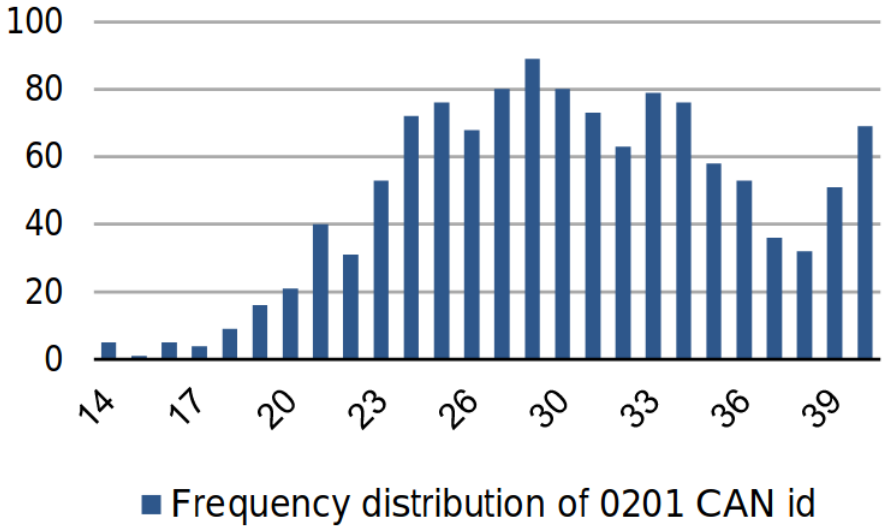

Figure 4.3: Ford CAN ID 0210 frequency distribution. Source: [MV13]

CHAPTER 5

Automotive Ethernet

The automotive Ethernet is still in its early stages there are no fully Ethernet cars. However that gives us the opportunity that the automotive industry did not have when designing CAN, which is to include security measures and model them after known vulnerabilities. We research security measures so that when automotive Ethernet takes off in the market we will already have counter-measures in place.

5.1 Ethernet

The Ethernet communication is by far the most common method of communication in corporative and domestic networks. It is defined over many different IEEE standards, but the main families of standards are the IEEE 802.3 which define the physical and link layer of the standard.

5.1.1 The Ethernet Network Architecture

Ethernet is a simple protocol, the first version of the Ethernet was a bus-based communication over a coaxial cable. Much like CAN, the sending device puts an identifier into the frame and the receiving device would check for that identifier and decide if it wants that frame or not. The difference, however, is that this identifier is actually a physical address identifying which device should receive the frame, and not like CAN, where it identifies the content of the frame. However, just like any other bus-based technology we have collisions and need a media access control scheme to mitigate this issues. In Ethernet, the most commonly used was the carrier sense multiple access with collision detection (CSMA/CD).

Because of the collisions and other problems that come with a bus-based communication, the normal standard use of the Ethernet changed to a switched model. In switched Ethernet, links are full-duplex and point-to-point. The connection between multiples devices is managed by a forwarding device called a switch. The switch can have many connections and keeps a table of physical addresses to forward frames through the correct ports.

5.1.2 The Ethernet Frame

The most common Ethernet frame formatting is the one shown in Figure 5.1. It uses the Medium Access Control (MAC) header to give meta data about the frame for other devices. At the very start of the frame there are two MAC address fields, one for the destination device and one for the source device, both 6 bytes long. Right after the address, there is a 2-byte field

called EtherType. The EtherType is a value that identifies the type of data inside the payload. For example, the value 0x0800 is the one which identifies that the following payload is an IP packet. The payload then is followed by a CRC checksum for integrity checks.

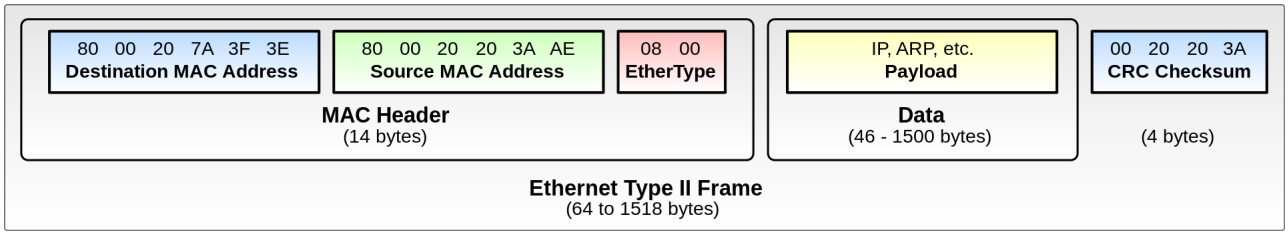

Figure 5.1: The Ethernet frame format. Source: https://en.wikipedia.org/wiki/Ethernet_frame

5.1.3 Standard Ethernet Security

Much like CAN, the Ethernet is a physical layer and link layer model without any security counter-measures built in [KSM13]. There are many attacks on the Local Area Networks (LAN) thanks to these issues. A very famous attack is the ARP cache poisoning, also known as ARP spoofing. The Address Resolution Protocol (ARP) is a mechanism which devices in an Ethernet-based network use to discover which MAC address has a route to a certain IP. This protocol works in a broadcasting behaviour since the device trying to send an IP packet needs to figure out which device in his network has a path to that destination IP address. The attacker takes advantage of the lack of authentication mechanisms coupled with broadcasting behaviour to pretend that it has a route to an IP and like that rewriting the inner ARP tables of devices in the network. A successful attack gives the attacker a man-in-the-middle position for further communication, allowing the attacker to perform an array of attacks ranging from denial of service to session hijacking. An attack like that is illustrated in Figure 5.2.

5.1.3.1 Security in Higher Layers

Instead of protecting Ethernet itself, we could use technologies in higher layers to secure communication. Thanks to the high bit rate and throughput of Ethernet, we have been using the higher layers inside the frame to deal with security issues for some time. The two most famous technologies used are the Transport Layer Security (TLS) and the Internet Protocol Security (IPsec).

The Transport Layer Security is a protocol for end-to-end security. It protects the transport layer communications with encryption and authentication through the use of a public key infrastructure with certificates [DR08]. TLS is a vital part of the Web, as it's a main part of HTTPS.

The IPsec is a very complex protocol used to protect IP networks. It works very similarly to a virtual private network and protects the network layer up to the application layer with encryption and authentication[IPS98].

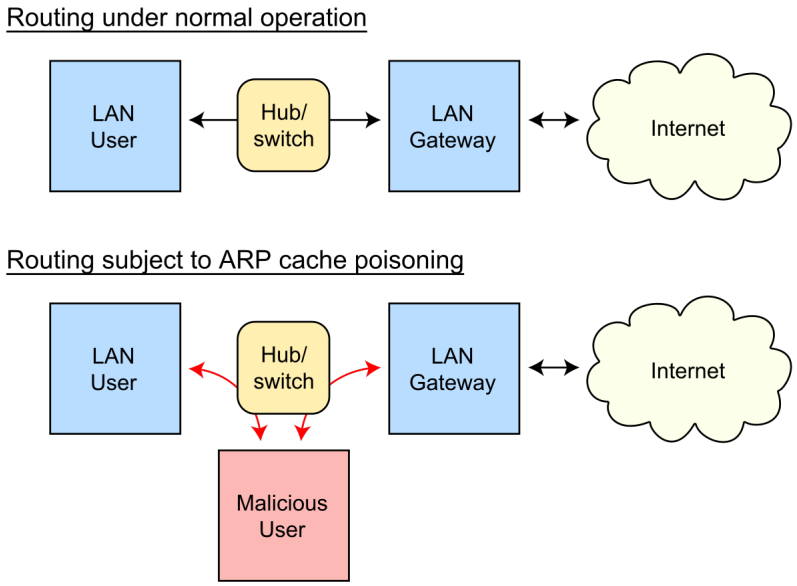

Figure 5.2: An ARP spoofing attack illustrated. Source: https://en.wikipedia.org/wiki/ARP_spoofing

5.1.3.2 Security in Lower Layers

Even though we can protect the higher layers, lower layers are still vulnerable to attacks such as the aforementioned ARP spoofing. There are a couple of lower layers security mechanisms for Ethernet-based networks defined in the standard family IEEE 802.1.

The IEEE 802.1X defines a port-based network access control. Using the authentication and authorization techniques, switches can control if devices connected to their ports were authorized to be part of the Ethernet network[80210].

The IEEE 802.1AR specifies credentials called DevIDs to be assigned by devices attached to the same LAN. It defines authentication techniques and enrollment protocols, working in conjuction with 802.1X[bor18].

The IEEE 802.1AE, also known as MACsec, is a standard that specifies the use of cryptographic Cipher Suites for authentication and optional encryption for hops in an Ethernet-based network[mac06].

5.2 An Automotive Ethernet Model

Automotive Ethernet is a new network technology option for the in-vehicle environment. In the automotive environment usually different functional domains are separated, e.g. Body Domain, Chassis Domain, Infotainment domain or Powertrain domain. Those domains are interconnected by so called gateways where additional security measures such as firewalling or

IDS deploying could be implemented in order to ensure that only harmless communication is taking place. In automotive Ethernet, this domain separation is probably going to be done with the help of VLAN tagging. Following the switched Ethernet topology, a diagram for a generic automotive Ethernet is shown in Figure 5.3

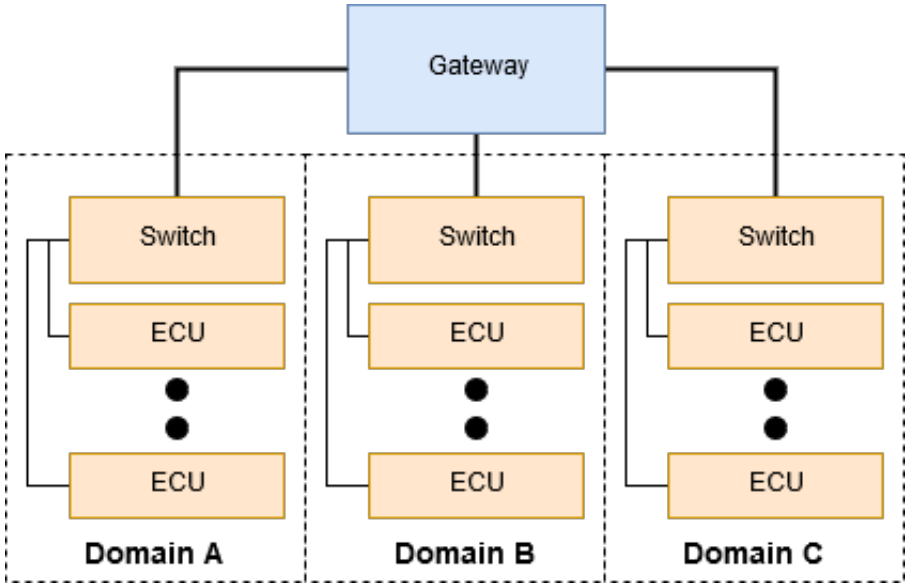

Figure 5.3: A generic automotive ethernet model

5.3 IEEE 802.1AE - MACsec

The IEEE 802.1AE MAC (Medium Access Control) security standard, known as MACsec, is a layer-2 solution that defines a secure communication by adding authentication and the possibility of encryption using AES-GCM (Advanced Encryption Standard in Galois/Counter Mode) to Ethernet nodes [mac06]. MACsec adds a header of 8 or 16 bytes called the SecTag which overwrite the EtherType. The SecTag has some meta data for the protocol's functionality, such as identifying if the frame is encrypted or not, which security association that frame belongs, anti-replay values and and other pieces of meta data. There is also the addition of a 128 or 256 bit Integrity Check Value (ICV) so that a receiver node may check for tampered frames.

A MACsec secure association works with a concept called a secure channel, which is a one way, one-to-many channel defined by the symmetric keys in place. One device may have different secure channels with other devices. Since these channels are one way, in order to have two devices communicating in a secure association with each other we need two secure channels, each one with their own set of keys. MACsec keys can be defined in many ways and there are no special key exchange in the MACsec standard. However in IEEE 802.1X it is defined a key agreement protocol called the MACsec Key Agreeement (MKA), which we won't get into details in this dissertation. The device figures out which key to use by checking a secure channel identification in the SecTag.

Figure 5.4: The MACsec Frame Format.

MACsec was designed to support hardware implementation to make security operations faster. However, since the solution is implemented in layer 2, the security associations are hop-based, which means that the trust only covers one Ethernet hop, not extending to the next one. Just like end-to-end secured communications, there are problems if gateways require changing any attribute in the frame without becoming a part of the security association and holding the proper keys.

5.4 An Automotive Ethernet model with MACsec

Consider an automotive Ethernet network that supports functionalities such as infotainment, ADAS (Advanced Driver Assistance Systems) and the exchange of real-time control data. These different automotive functions are served by different in-vehicle Ethernet domains, which are separated by VLANs, like in Figure 5.3. This separation allows the assignment of QoS requirements and security policies to each domain which can be enforced by firewalls and intrusion detection or prevention systems.

5.4.1 Threat Model

The main security threat to a car is an attacker controlling the vehicle's functions through network traffic injection. The worst case scenarios happen when the attack is feasible remotely as those kinds of attacks are potentially scalable and not limited to a single vehicle but could affect a complete fleet of vehicles.

In-vehicle attacks are those in which the attacker is able to inject frames into the automotive network by either physically installing an attacker controlled device on the network or by compromising an existing device and achieving some sort of control over the traffic generated by it.

Outside attacks are those that typically happen through a device connected to the outside somehow. The attacker's entry point possibilities could be a Bluetooth connection to the sound system, a V2V (vehicle-to-vehicle) communication with other nearby car or Cellular Network. These communications enter the vehicle through devices called connectivity units.

5.4.2 Deploying MACsec in the network

Our objective is to achieve something close to an end-to-end authentication scheme throughout the entire Ethernet network. Firstly, we assume all keys are randomly generated, have already been put into place and switches are purely working as frame forwarding devices. This way the switch won't be a part of the MACsec associations and won't need to hold any keys. Devices in the same domain can communicate with each other securely, with every frame authenticated by the ICV in MACsec frames.

In standard MACsec implementations, the VLAN tag integrity is protected by the ICV. So, when traversing from an automotive domain to another, the gateway needs to exchange the VLAN tag from the original domain to the destination one. This requires the recalculation of the ICV by the gateway. Turning the gateway into a single point of failure in the system and breaking end-to-end security in the network.

There are a couple ways to bypass this behaviour with the gateway, one of them could be simply removing the VLAN tag from the ICV calculations. This way the gateway can just blindly change the VLAN tag value based on the destination MAC address.

There is not much payoff in encrypting communications inside the in-vehicle's network domains. Using MACsec authentication properties, secure ECU software development and gateways that check for traffic behavior are enough security that far inside the network. When using only MACsec with simple gateways with switching capabilities, we get a key distribution similar to the end-to-end security TLS offers the Internet. By not using encryption in intravehicular network communications, gateways can use deep packet inspection to detect more complex security aspects too, such as invalid or abrupt changes in data fields inside the frames. There are no major payoffs in enabling encryption in MACsec for the automotive environment.

We could use the single point of failure in the gateway to our benefit by making the gateway keep track of the security associations it should be a part of. Like a firewall, we could use the gateway's position to drop frames in communications that should not be allowed. For example, an entertainment ECU like a radio unit does not need to communicate with safety-critical ECUs, like in the Jeep hack, so there shouldn't be any security associations defined in the gateway, therefore the frame is not forwarded.

In an article to be submitted to IEEE Communications Magazine in 2019, we further analyze the issue of the automotive gateway behaviour in MACsec networks, especially in the presence of VLANs as a way to separate and define automotive domains [CB19].

5.4.3 Versus the Threat Model

The attacker's main objective is the control of the car's safety-critical functions. MACsec is resistant to replay attacks so an attacker inside the network needs to be able to retrieve the keys of devices that send control messages to safety critical devices and spoof that message as the original device.

In an outside attack scenario, the attackers start with the keys of a device and all its MACsec security associations. If the network was correctly designed, the compromised ECU has no access through the gateway to the safety critical functions, and the impact is isolated.

With MACsec's authentication mechanisms and replay attack resistance we get a security

model similar to the Secure Onboard Communications mentioned in Chapter 4.

CHAPTER 6

Conclusion

Automotive networks are considerably new in the world of information warfare. But we've seen that even though, to the best of our knowledge, there have been no accidents caused by cyber attacks, researchers have shown that with time to reverse engineer a modern car, they came up with a remote exploitation with minimum user interaction that resulted in controlling the car physical functions. They also made the point that their exploit could be used in a worm malware and let it spread around the internet. Had they any ill-intent, they could create mass destruction and possibly many deaths.

With the introduction of Ethernet as a new automotive in-vehicle network technology, we gain the opportunity to develop the applications with security in mind. Not only that, many of the usual defenses used in corporative and home networks could possibly be translated to the automotive environment, just like we've shown with MACsec.

6.1 Contributions

In this undergraduate dissertation, we've presented a small survey on attacks and vulnerability researches conducted in current vehicles and attachable devices. We also analyzed the current counter-measures being proposed by the automotive industry. In the end, we presented and analyzed a security model based on MACsec for the upcoming automotive Ethernet.

6.2 Future Work

There are still questions pertaining to what applications the automotive Ethernet will hold. After these have been made public, we can start threat modelling after them and evaluating the security impacts of their design. There is still grounds to cover in research of Intrusion Detection Systems for the Automotive Ethernet environment.

Bibliography

[80210] Ieee standard for local and metropolitan area networks–port-based network access control. *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)*, pages 1–205, Feb 2010.

[ARGa] A remote attack on an aftermarket telematics service. `https://argus-sec.com/remote-attack-aftermarket-telematics-service/`. Accessed: 2018-11-20.

[ARGb] A remote attack on the bosch drivelog connector dongle. `https://argus-sec.com/remote-attack-bosch-drivelog-connector-dongle/`. Accessed: 2018-11-19.

[bor18] Ieee standard for local and metropolitan area networks - secure device identity. *IEEE Std 802.1AR-2018 (Revision of IEEE Std 802.1AR-2009)*, pages 1–73, Aug 2018.

[CB19] Divanilson R. Campelo Caio Burgardt, Timo Lothspeich. Securing automotive ethernet networks with macsec and vlans. *To be submitted in IEEE Communications Magazine*, 2019.

[CMK+11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*, pages 77–92. San Francisco, 2011.

[DR08] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. Technical report, 2008.

[ETKK16] Miro Enev, Alex Takakuwa, Karl Koscher, and Tadayoshi Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(1):34–50, 2016.

[FPKS15] Ian D Foster, Andrew Prudhomme, Karl Koscher, and Stefan Savage. Fast and vulnerable: A story of telematic failures. In *WOOT*, 2015.

[H+12] Florian Hartwich et al. Can with flexible data-rate. In *Proc. iCC*, pages 1–9. Citeseer, 2012.

[IPS98] RFC IPSEC. 2401,"security architecture for the internet protocol" s. *Kent, R. Atkinson*, 1998.

[JWAG18] Shalabh Jain, Qian Wang, Md Tanvir Arafin, and Jorge Guajardo. Probing attacks on physical layer key agreement for automotive controller area networks (extended version). *arXiv preprint arXiv:1810.07305*, 2018.

[KCR+10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[KSM13] Timo Kiravuo, Mikko Sarela, and Jukka Manner. A survey of ethernet lan security. *IEEE Communications Surveys & Tutorials*, 15(3):1477–1491, 2013.

[mac06] Ieee standard for local and metropolitan area networks: Media access control (mac) security. *IEEE Std 802.1AE-2006*, pages 1–150, Aug 2006.

[MK17] Kirsten Matheus and Thomas Königseder. *Automotive ethernet*. Cambridge University Press, 2017.

[ML15] Andreas Mueller and Timo Lothspeich. Plug-and-secure communication for can. *CAN Newsletter*, pages 10–14, 2015.

[Muc16] Alexander Much. Automotive security: challenges, standards, and solutions. *Softw. Qual. Prof*, 18(4), 2016.

[MV13] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. last accessed from h ttp. *illmatics. com/car hacking. pdf on*, 13, 2013.

[MV15] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015:91, 2015.

[RKJ06] Ronald S Ross, Stuart W Katzke, and L A Johnson. Minimum security requirements for federal information and information systems. Technical report, 2006.

[SECa] Is car hacking over? autosar secure onboard communication. `https://published-prd.lanyonevents.com/published/rsaus18/sessionsFiles/8996/SBX3-W1_Is%20Car%20Hacking%20Over-AUTOSAR%20Secure%20Onboard%20Communication.pdf`. Accessed: 2018-11-20.

[SECb] Specification of secure onboard communication. `https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf`. Accessed: 2018-11-22.

[Spe91] CAN Specification. Version 2.0. *Robert Bosch GmbH*, 1991.

[Sta93] ISO Standard. 11898: Road vehicles—interchange of digital information—controller area network (can) for high-speed communication. *International Standards Organization, Switzerland*, 1993.

[Sta13] ISO Standard. 14229: Road vehicles – unified diagnostic services (uds). *International Standards Organization, Switzerland*, 2013.

[Thu] Hacker says attacks on 'insecure' progressive insurance dongle in 2 million us cars could spawn road carnage. https://www.forbes.com/sites/thomasbrewster/2015/01/15/researcher-says-progressive-insurance-dongle-totally-insecure/#c1077f81772b. Accessed: 2018-11-21.

[WJKL16] Samuel Woo, Hyo Jin Jo, In Seok Kim, and Dong Hoon Lee. A practical security architecture for in-vehicle can-fd. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2248–2261, 2016.

[WWP04] Marko Wolf, André Weimerskirch, and Christof Paar. Security in automotive bus systems. In *Workshop on Embedded Security in Cars*, 2004.

[ZG13] Kai Zhao and Lina Ge. A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*, pages 663–667. IEEE, 2013.

This volume has been typeset in LaTeXwith the UFPEThesis class (www.cin.ufpe.br/~paguso/ufpethesis).