FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza bezpečnosti elektronických jednotek vozu Tesla Model 3
Student:	Bc. Jan Michal
Vedoucí:	Ing. Jiří Dostál, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Tesla Model 3 je elektromobil nové generace s novou architekturou a infrastrukturou řídicích jednotek. Jelikož se jedná o "connected" vozidlo, nabízí se otázka ohledně problematiky počítačové bezpečnosti. V rámci závěrečné práce se seznamte s problematikou počítačové bezpečnosti v automobilovém průmyslu. Na testovacím voze proveďte analýzu zabezpečení vnitřních elektronických jednotek (ECU). Zaměřte se zejména na jednotku autopilota (ACU) a jednotku multimediálního obsahu (MCU). Popište SW architekturu a vnitřní zapojení. Za pomoci reverzního inženýrství odhalte co nejvíc informací o operačních systémech, souborových systémech a službách. Zmapujte povrch zranitelnosti (attack surface) a vytvořte threat model. Identifikujte vybrané zranitelnosti a implementujte "proof of concept" exploity. V případě objevení "zero day" zranitelnosti zahajte proces zodpovědného odhalení (responsible disclosure). Vše zdokumentujte a vyhodnoťte dopady na bezpečnost.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 13. února 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Tesla Model 3 Control Units Security Analysis

Bc. Jan Michal

Department of Information Security
Supervisor: Ing. Jiří Dostál, Ph.D.

January 7, 2021

Acknowledgements

I want to thank my supervisor Ing. Jiří Dostál, Ph.D. and his colleagues for their advice, support, patience and that they allowed me to work on this thesis in their company.

I am also grateful to my parents and friends for their support during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on January 7, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Michal, Jan. *Tesla Model 3 Control Units Security Analysis.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Tato práce se zabývá bezpečnostní analýzou osobního automobilu nové generace od firmy Tesla, Model 3. Hlavní cíl práce je úvod do bezpečnostního testování vozu, který by měl sloužit jako prvotní náhled na to, co a jak se děje pod kapotou. Provedeným výzkumem jsem zjistil, že vozidlo je alespoň na první pohled velmi dobře chráněno. Výsledky této práce umožňují dalším studentům začít s jejich vlastním výzkumem, který může jednoduše navazovat na tuto práci.

Klíčová slova Tesla Model 3, Počítačová bezpečnost, Model hrozby, Bezpečnostní analýza, Webový server

Abstract

The thesis is focused on the vulnerability analysis of a new generation electric car Tesla Model 3. The main goal of the thesis is an introduction to security testing of the car, which should serve as an initial insight into what is happening under the hood. I found that the vehicle is at the initial preview very well protected. The results of this work allow other students to begin their own research, which can easily follow this work.

Keywords Tesla Model 3, Cybersecurity, Threat Model, Vulnerability Analysis, Web Server

Contents

List of Figures

List of Tables

Introduction

We live in a world where everything changes really fast. Just take a look at the progress of mobile phones. In the last few years they conquered the world, they can do almost everything and still can fit into your pocket.

But we have other devices which are not so much changing in our lives and that are our cars. Cars have in comparison with phones several disadvantages. One of them is their safety, cars can easily kill a person inside the car in a car crash or hit someone outside by accident. Second big disadvantage is their price which makes them not as easily replaceable as mobile phones. Most people have their cars for almost a decade or even longer [4]. Vehicles have longer lifespan and thanks to that their software needs more attention, updating and reviewing.

Modern vehicles have more and more electronic systems. Some of them are required by law and the others are there because of safety, better user experience or to make driving and service easier for customers. All these systems are running some kind of software and as we all know software has bugs even the most secure ones have vulnerabilities or threats. So I have chosen to study cyber security of Tesla Model 3. It is shown as a new generation car, it does not have a combustion engine but instead an electric engine and has autopilot ability.

Chapter 1

Thesis Overview

This chapter serves as a longer and more extensive introduction.

1.1 Guidelines

If I do not want to skip or forget about some really important part of security research I will have to follow up already existing guidelines. There are many of them: PTEST, OWASP, etc. Sadly none of these above is ready for automotive security testing. From these guidelines I had chosen the Pen Test Standard with slight modifications that are described below.

1.2 Thesis structure

I will follow slightly modified steps listed in Pentest Standard to make sure that I will not skip something important. Each step in Pentest Standard will mean one chapter in this thesis.

The section Pre-engagement Interactions is contained in this chapter. The information needed is very little so it does not deserve a whole new chapter.

1.2.1 Background

I added chapter Background. It covers the theoretical background of information needed to understand this thesis and guides the reader to better understand concepts used later in the thesis.

1.2.2 Intelligence gathering

This chapter contains all information that was gathered from public sources mostly online and also already written other theses that was focused on cyber security of vehicles. The second part is the introduction of the vehicle Tesla

Model 3 from the user's perspective. The car is dismantled a little so the reader can see how some of the ECUs look like.

1.2.3 Threat modeling

As the name of the chapter suggests you can find here the threat model that was created during this thesis. It contains explanation and introduction into threat modeling tools and frameworks and ends with producing an own threat model.

1.2.4 Vulnerability analysis

This chapter is narrowly focused on only one part of the vehicle, the diagnostics web server. It describes how to connect to the web server and then its security analysis.

1.2.5 Exploiting

In the last chapter is demonstrated how to exploit vulnerabilities that were found during the vulnerability analysis.

1.3 Rules

1.3.1 Scope

Scope of this thesis is the vehicle only. However my work is in cooperation with a professional car mechanic. So we are able to dismantle and analyze everything in the vehicle including all electronic control units (ECUs).

The scope is more narrowed in chapter 4 and since this point the whole thesis is not general anymore.

1.3.1.1 Out of scope

Tesla's backend services or anything else except the car is out of scope.

1.4 Thesis goals

The reason that I chose this thesis and my main goal is to find new vulnerabilities which will lead to more secure vehicle systems or confirm that the car is well protected. However in order to complete and evaluate the thesis there are some smaller goals as well that have to be measurable.

- Get acquainted with the issue of computer security in the automotive industry.

- Perform an Internal Electronic Unit (ECU) security analysis on the test car.
- Describe SW architecture and internal connection.
- Reveal as much information as possible about operating systems, file systems, and services.
- Map the attack surface and create a threat model.
- Identify selected vulnerabilities and implement proof of concept exploits.
- If a ”zero day” vulnerability is discovered, initiate a responsible disclosure process.
- Document everything and evaluate the security impacts.

Chapter 2

Background

In this chapter are defined basic terms that are needed for understanding of the following chapters. The reader will be informed on concepts, terms and technology used in the automotive industry. The chapter is finished with software that I used during my work on this thesis.

2.1 Basic concepts

2.1.1 Connected vehicle (CV)

The concept of the connected vehicle is here for many years. It represents a vehicle which is capable of receiving and sending messages with its surroundings especially other vehicles. Navigation systems nowadays have built in connected vehicle technology such as dynamic route planning during the trip. The GPS system receives information via cellular network on the road ahead and considering the density of traffic or weather forecast can suggest to the driver different path due to safety reasons or saved up time. The connected vehicle does not make any decisions on received data but only passes relevant data to the driver. So he can use them to make smart and more informed decisions. This technology will also help traffic infrastructure gather data about problematic passages on roads. Then the government or road planners could use them to better design roads in the country [5].

2.1.2 Autonomous vehicle (AV)

Autonomous vehicle concept has many stages. The last stage is a fully autonomous vehicle which does not need any driver at all and is capable of driving itself into any desired destination. Some vehicles these days have some kind of this technology such as self-parking, collision detection, lane keep assist or car summon. Fully autonomous vehicle does not need to be a connected vehicle as well. Since it must be functional without any other dependency to

be able to navigate correctly. But if the autonomous vehicle has connected vehicle technology it can profit from it. With connected vehicle technology the car will be more efficient, safer and faster due to rightly chosen paths. The autonomous vehicle can not be without connectivity all the time because there will always be new roads or the old roads could be damaged and not safe or even destroyed. So the car will from time to time receive some software updates to function properly [5].

2.1.3 Vehicle to infrastructure (V2I)

As the name suggests this communication is happening between cars and infrastructure such as smart traffic lights or any other objects that you pass on the roads. The infrastructures gather data of weather forecasts, traffic density and other things that have an impact on safety on roads. Then the data is sended to the vehicle and the driver should be informed if there is some problem on his planned route [6].

2.1.4 Vehicle to vehicle (V2V)

Similar to V2I but this type of communication occurs between vehicles. They could exchange a lot of information. V2V communication enables vehicles to exchange speed and position of other vehicles in their area. This communication has coverage around 300 metres. Vehicles can detect and inform others of accidents, dangerous drivers or terrain issues. Main vision of V2V technology is to make driving predictable and safe for everyone on the road [6].

2.1.5 Vehicle-to-everything communication (V2X)

V2X technology includes both V2I and V2V and other not so famous communications such as Vehicle-to-pedestrian (V2P), Vehicle-to-cloud (V2C),Vehicle-to-home (V2H) and Vehicle-to-building (V2B). All these examples are shown in figure 2.1. Communication is based on dedicated short-range communications (DSRC). DSRC is wireless communication technology developed for usage in the automotive industry mainly for V2I and V2V. It allows vehicles and infrastructure to communicate efficiently with each other [6].

2.1.6 Software-over-the-air

When a vehicle updates over-the-air it downloads files from the server via Wi-Fi or cellular network. It can be downloaded directly to the car or some other device (mobile phone or PC) and then transferred into the vehicle with user's interaction. It is crucial for the update to be as small as possible so the manufacturer should only send the new parts of the software that need to be changed and not send the whole image which could be extremely large. In ideal conditions the manufacturer should have some sort of security signing

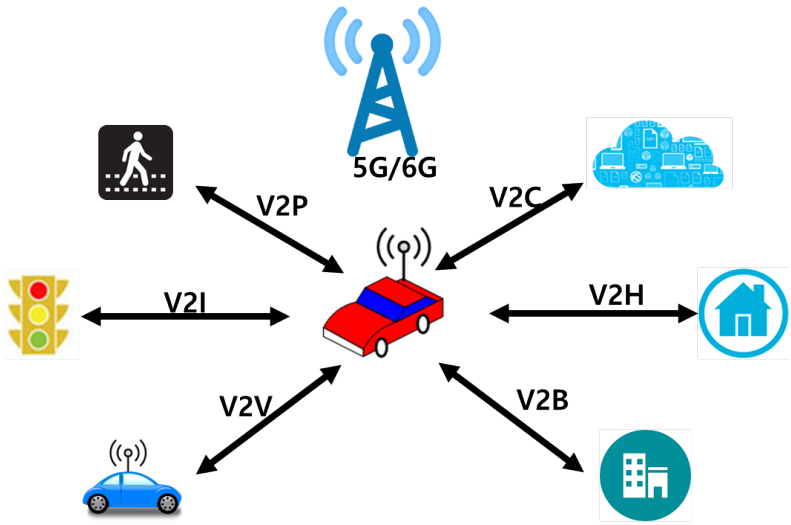

Figure 2.1: Usages of V2X technology [7]

that disables users to update their vehicles with non-original updates. When users can update to whatever software they want in their car it could be potentially dangerous. For users this type of update is comfortable because they do not need to come every time when the vehicle is in need for software update into the car dealership and their cars can update during night when the user is not using the car. Many car manufacturers sold some software features to the car. If you need some application running on your infotainment system you do not need to visit the dealership and only pay for it and then you receive that piece of software over-the-air [8].

2.2 Electronic Control Unit

The whole car consists of many different parts. We will primary focus on ECU - Electronic Control Unit. They are small embedded systems which are responsible for controlling electrical systems or subsystems in the vehicle [9]. Examples of ECUs:

- Battery management system
- Brake Control Module
- Door control unit
- Electric Power Steering Control Unit
- Engine control unit

- Human-machine interface
- Powertrain control module
- Seat Control Unit
- Speed control unit
- Telematic control unit
- Tire pressure monitoring system
- Transmission control unit
- Transmission control module

2.2.1 Automotive networks

If the car has more than one ECU it is needed to have some sort of communication within them. Computers communicate with each other via computer networks. Similar to computers the Electric Control Units in the car communicate with each other via automotive networks. The most common are:

2.2.1.1 Controller Area Network

CAN is one of the oldest of vehicle network protocols. It was released in 1986 and developed by BOSCH [10]. Nowadays CAN is the most widely used protocol which makes it extensively tested and very reliable. The whole network works as a multi-master, message broadcast system and has speed up to 1 Mb/s. Unlike the traditional networks such as USB or Ethernet which sends large blocks of data from one node to another the CAN works differently, it broadcasts a lot of short messages like current temperature or speed to the whole network. So all the nodes in the network receive all the messages and it is up to them if they react on them. This broadcasting system is also good for data consistency because all nodes in the network receive the same message. All this makes CAN ideal in all networks where there is a need for a large number of short messages because CAN is message based and not address based. It provides high reliability mainly in networks where messages need to be obtained by more than one device [11].

2.2.1.2 Local Interconnect Network

The first version of LIN was released in 1999 to supplement CAN network via dramatically lower cost due to no license fee and cheap nodes but on the other hand offers lower reliability and performance. It was developed by LIN Consortium (BMW, VW, Audi, Volvo, Mercedes-Benz, Volcano Automotive and Motorola). In 2010 was defined the most up to date LIN version as ISO

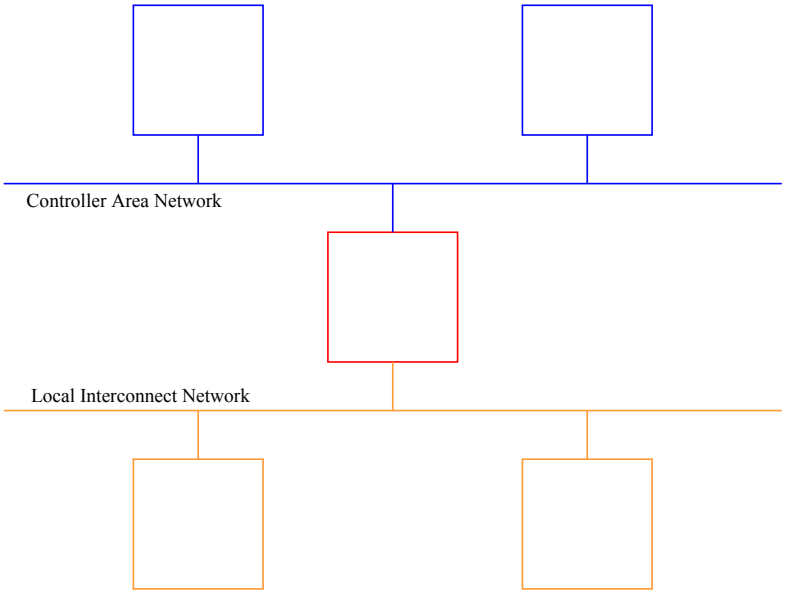

Figure 2.2: A LIN master typically serves as gateway to the CAN

17897 and was officially released in 2016. Unlike CAN the LIN does not work as a multi-master broadcast and instead has one master and up to 16 slaves. The maximum speed for LIN is only 20 kbit/s. Is almost in all new vehicles. Because of low reliability is LIN used for non critical vehicle components such as windows, air condition or wipers [12, 13]. In many cases one ECU from CAN works as a master node of LIN shown in 2.2.

2.2.1.3 FlexRay

FlexRay was developed in 2000 by FlexRay consortium (BMW, Daimler, Chrysler, Philips, Motorola and Freescale with Bosch joined them later) [14]. The main purpose of FlexRay was to eliminate drive-by-wire technology. Learn more about drive-by-wire technology at 2.2.2. FlexRay has two independent cable paths, called ”Channel A” and ”Channel B”. Two channels can either be used for data redundancy or to transfer different data. Maximum speed is 10 Mb/s which is ten times more than CAN. FlexRay can be both time-triggered and event-triggered at the same time. That is different from CAN which can be time-triggered or event-triggered but not both of them. Due to FlexRays reliability it is mainly used for critical components of a vehicle for example braking or steering. FlexRay is also more expensive than CAN and because of that it is used only in expensive vehicles. Tesla model 3 does not have this network system.

	CAN	LIN	FlexRay	MOST
Application	Soft real-time systems	Low-level communication systems	Hard real-time systems (X-by-wire)	Multimedia, telemetrics
Control	Multi-master	Single-master	Multi-master	Timing-master
Bandwidth	500 kb/s	19.6 kb/s	10 Mb/s	24.8 Mb/s
Data Bytes per Frame	0 to 8	0 to 8	0 to 254	0 to 60
Redundant Channel	Not supported	Not supported	Two channels	Not supported
Physical Layer	Electrical (twisted pair)	Electrical (single wire)	Optical, electrical	Mainly optical

Table 2.1: Automotive Networks Specifications [1]

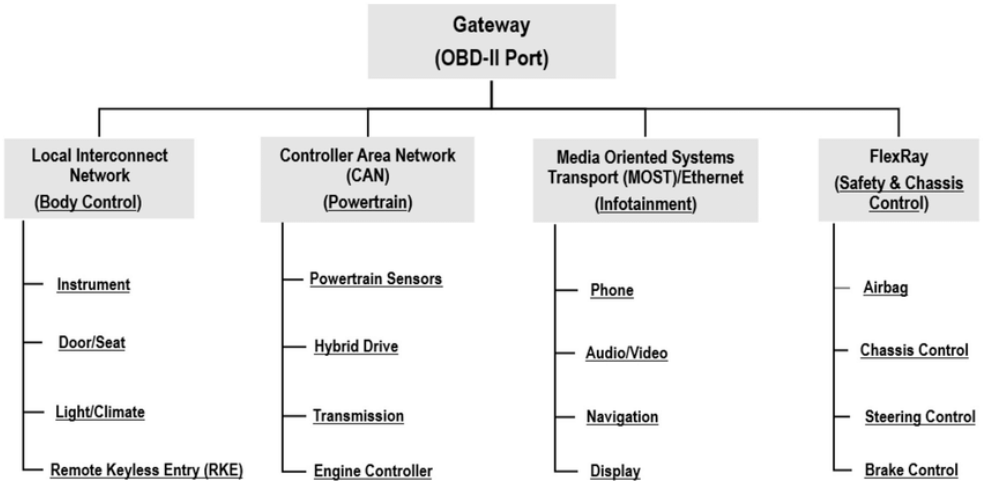

Figure 2.3: Each of the networks have different purpose in the vehicle [15]

2.2.1.4 Media Oriented Systems Transport

From the name itself it is clear that this type of network is primary for multimedia in the vehicle. It has the biggest maximum speed 24.8 Mb/s and thanks to that this type of network is best for video, audio and navigation systems.

All these types of networks (CAN, LIN, FlexRay and MOST) have use in almost every vehicle as is shown in 2.3. The specification of each and differences between them can be found in table 2.1.

2.2.1.5 Automotive ethernet

Automotive ethernet is a physical network that connects components inside of the car using a wired network. It meets the automotive, electrical, bandwidth, latency, synchronization and network management requirements. It has greater bandwidth than CAN or FlexRay which makes automotive ethernet better for in-vehicle systems that require transmits of large data. With comparison with MOST automotive ethernet does not need proprietary licensing, does not rely on heavy coax cables or easily damaged optical fiber [16].

2.2.1.6 BroadR-Reach

Automotive network protocol developed by Broadcom. It uses a simple twisted pair of wires at speed 100 Mb/s. On the both ends are BroadR-Reach PHY chips which are able to send and receive data at the same time. The main purpose for using BroadR-Reach is its small price and light weight with comparison to other automotive networks [17].

2.2.1.7 Other Automotive Networks

There are many other automotive networks but they are not so popular in the cars right now. Here are some examples just in case if you want to learn more: MI Bus, DSI Bus, BST Bus, MML Bus, byteflight, SMARTwireX, IEBus, Intellibus, Vehicle Area Network (VAN) and Planet.

2.2.2 Drive by wire technology

This type of technology is also known as ”x-by-wire” or just simply ”by-wire”. Conventional cars mainly use mechanical or hydraulic technology for basic vehicle operations (braking, steering or acceleration). But with Drive-by-wire these mechanical or hydraulic components are replaced with mainly electronics parts. This concept came from planes which use ”fly-by-wire” technology in order to control normal operation of planes since the 1990s [18]. On the other hand, drive-by-wire systems need to be very complex in order to work properly which can lead to possible software failure, electronic malfunctions in sensors and in same cases it can even result in car accidents and possibly passenger injury.

2.3 Attacks

Here are listed only the attacks that are used later in this thesis. Make sure you understand them before reading the next chapters. If you are familiar with them feel free to skip this section.

2.3.1 Cross-site scripting

Cross-site scripting (XSS) is an attack that is targeting end users of malicious websites. The website is trusted by the users but an attacker inserted there a malicious script that is intended to perform an attack on other visitors of the website. This attack is possible on websites where input from one user is shown to other users. This input can contain malicious script that is intended to access cookies, session tokens and any sensitive information from the users [19].

2.3.2 Clickjacking

Clickjacking or sometimes named UI redress attack is another attack that aims on end users of websites. The victim is tricked into clicking on a button that is invisible so the victim can not see it but is located on the top level of the page. The user typically performs a non harmful action such as receiving a free mobile phone but clicks on a button that could delete all his emails or transfer his money to the attacker [20].

2.4 Used programs

In this section are briefly described software programs that were used during this thesis.

2.4.1 Nmap

Network Mapper is a port scanner used for network exploration. It can scan for open ports with TCP and UDP connection and provide further information of targeted devices.

2.4.2 Nikto

Nikto is a web server scanner that is used to find problems, threats and any other security vulnerabilities.

2.4.3 Ghidra

Ghidra is a tool for software reverse engineering from the National Security Agency. The main purpose is analysis of compiled code on every major platform.

2.4.4 IDA

Interactive Disassembler is another tool for software reverse engineering. It comes in two variants one is free and with limited usage the second has paid and comes with more features.

2.4.5 Binwalk

Command line tool that searches, analyzes and extracts binary firmware images.

2.4.6 Sparta

Application with its own GUI used for network infrastructure penetration testing. It consists of many smaller programs such as nmap, nikto that together produce a complete picture of targeted infrastructure.

2.4.7 OWASP ZAP

Zed Attack Proxy from OWASP community is a web application security scanner.

2.4.8 Wireshark

Tool with a GUI that is used for analyzing and dumping network traffic.

Chapter 3

Intelligence Gathering

Pentest standard is mainly for applications or network testing. In the first section this chapter I will go through intelligence which can be mainly searched online. Information gained from the vehicle itself will follow up in the next section.

3.1 Known cyber attacks on vehicles

In this section is described how attackers performed cyber attacks on vehicles in recent years. It is good to know if car manufacturers are now more careful than in previous years of car development. Also we can look at what Tesla had already done for more secure systems in their vehicles and if there is still some more space for improvement. They are sorted from the oldest to the most recent not by their impact. Their impact is summarized at the end of this section in table 3.1.

3.1.1 General Motors Chevy Impala

Group of security researchers from University of California at San Diego and the University of Washington found vulnerability on General Motors' 2009 Chevy Impala. The group discreetly informed the GM of the found exploit and did not go to publicly show that information in media or public conferences. It was the reason that this attack is not so well known in public as well as the famous one from 2015 described in 3.1.3. General Motors took almost 5 years to fix that bug. The group reverse-engineered audio protocol and found buffer overflow vulnerability that was started by an mp3 file and granted them access into the car's whole system with all of its critical functions except the steering wheel [21].

3.1.2 Ford Escape and Toyota Prius

Two security researchers Charlie Miller and Chris Valasek managed to exploit vulnerabilities of Ford Escape and Toyota Prius. Exploitation required physical access to the OBD-II connector to work. They used car self-parking functions to hijack the steering wheel from the drivers control. They were also able to kill power steering, spoof the GPS and set speedometers and odometers to any value. Toyota was not impressed by those hacks and said that their systems are well secured against wireless attacks. Toyota also stated that for real car hacking you do not need to have physical access [22].

3.1.3 Jeep Cherokee

Those same two hackers (Charlie Miller and Chris Valasek) managed to wirelessly hack Jeep Cherokee. They would be able to take control over the car's functions such as steering wheel and brakes. This attack is in my opinion the most famous one and attracted a lot of attention into automotive cyber security. They exploited a vulnerability in a car infotainment system called Uconnect that relied on cellular connection to be accessible from the internet. Uconnect had open port 6667, there was running the Diagnostic Bus which should not be accessible from the external network but only from the internal network. Because Jeep used Sprint (American telecommunications company) cellular connection they scanned via Sprint sim card IP addresses 21.0.0.0/8 and 25.0.0.0/8 for open port 6667. This resulted in 1.4 million vehicles being affected by this vulnerability not only Jeep's cars but also Dodge and Chrysler cars manufactured from 2013 to 2015. So the discovery of this vulnerability was really significant. They did not however test if all those cars were in reality vulnerable to take control over all of them but they only validated that they had opened that D-Bus port [23].

3.1.4 Tesla Model S

Two security researchers Kevin Mahaffey and Marc Rogers had a presentation at DEF CON. They demonstrated how to remotely unlock the Tesla Model S. Researchers also showed video of how they could kill the car via their phone. They installed Trojan Virus into that specific car. To be able to do that they needed physical access into the vehicle. After the installation of Trojan Virus they could remotely control the vehicle. Tesla fixed this issue with their over-the-air update within one week. With these types of updates Tesla can fix all their cars at once [24].

3.1.5 Nissan LEAF

Troy Hunt researcher and also owner of Have I Been Pwned? found out in 2016 just one year after the famous Jeep Cherokee attack that he can via Nissan's

official mobile application access other Nissan LEAF. Nissan LEAF is one of the most selling electric cars in Europe it is also a connected vehicle and because of that Troy was able to send commands over the globe to different cars than his own. However he could not control critical components of the vehicle but only non-critical functions. He was able to remotely turn on or off air conditioning and seat heating and could get the history of car GPS tracking. This type of attack is not really dangerous because he can not turn the car on/off or take control over the moving vehicle but if someone took advantage of this vulnerability he could drain the car battery for example [25].

3.1.6 Tesla Model S and X

Chinese security researchers from a Tencent's KeenLab security team found a flaw in the Tesla Model S. The bug was in Tesla's internet browser which was based on the open source browser framework WebKit. The browser was able to unintentionally run code from malicious web sites if a user visited this site. To demonstrate this attack hackers set up a Wi-Fi hotspot named "Tesla Guest" (it is the name of the common Wi-Fi used in Tesla dealerships). They had to set up the exact same password as the original Wi-Fi hotspot used that allowed all Tesla cars to connect to this network (it was a shared password that could be found online). They configured their network that after successful connection with the car was established the car would immediately load the malicious site. Tesla did not agree that this type of attack would be successful and Elon Musk (Tesla founder) argued that the user needed to connect this malicious network manually and then navigate to the infected website manually. After the page was loaded it used a bug in the vehicle's linux operating system and gained full privileges on it's computer. The hacker's group was now in control of the vehicle's computer which was not directly connected to the CAN bus. Between them was a gateway which made it impossible to control the car's critical components such as steering and braking. The hackers simply replaced gateway firmware with their own. Tesla responded in 10 days with a security update that used new features that made it impossible to rewrite firmwares in ECUs without a cryptographic key. Now only tesla with their key can update to newer firmware. Tesla wanted to have code-signing for a long time and was working on it but this exploit accelerated this process [26].

3.1.7 Tesla's key fob

Security researchers from the KU Leuven university in Belgium explored Tesla's keyless entry system. Team discovered that the key fob is made by Pektron and is using only a 40-bit cipher for encryption. If they found out that they needed to get only 2 codes from any key fob. After that they could simply try out every possible key until they would find the right one. So they

computed all possible keys with all possible codes. This output was a table with size of 6 terabytes and they can simply find with two codes the specified key within 2 seconds. This attack did not work with Tesla's PIN code. You first need to input the PIN code and until then you can not start the car. Attack also relied on using passive entry key fobs; it did not work with key fobs that had disabled or without this feature. Team suspects that this vulnerability is in all key fobs made by Pektron. Pektron made key fobs for McLaren, Karma and Triumph motorcycles. Pektron simply upgraded from 40-bit encryption into 80-bit encryption. Because this attack was a hardware problem and not a software vulnerability the key fob needed replacement and no update was possible [27].

3.1.8 Key fob again

The same security researchers from the KU Leuven university that did the key fob attack above managed to exploit the upgraded key fob again. New attacks needed more time to perform and they did not perform it on the vehicle itself. Hackers only describe how the attack should succeed. The new key fob was not although using 80-bit encryption but it was implemented badly. The bug allowed the attackers to crack the 2 keys with 40-bit encryption so the fix only slowed time needed to crack the key 2 times. This time Tesla did over-the-air update into their whole fleet and the problem was solved. From the car they could update firmware even in their key fobs [28].

3.1.9 Honda's backend

There are not only attacks on the vehicles itself but also on the car manufacturer's backend services. Honda reported a cyber attack on their servers and their services were temporarily unavailable due to Ekans ransomware which is designed to attack industrial control systems networks [29].

Because this issue is not related to the car itself it is not listed in table 3.1.

3.2 Related work

Online can be found a lot of thesis, research papers or books that are dealing with automotive security. I will list a few of them which I found helpful, interesting and/or even used in this thesis.

3.2.1 Related books

A lot of books have been written on this subject but I will list only one which I found most helpful.

The Car Hacker's Handbook: A Guide for the Penetration Tester [9] written by Craig Smith is really great for starting with car hacking. It has a lot of

Vehicle	Attack vector	Year	Impact
GM Chevy Impala	Physical access - Infotainment system	2010	Low
Ford Escape Toyota Prius	Physical access - OBD-II connector	2013	Low
Jeep Cherokee	Telematics - Infotainment system	2015	High
Tesla Model S	Physical access - Trojan	2015	Low
Nissan LEAF	Telematics - Mobile application	2016	Medium
Tesla Model S and X	Wireless - Wi-Fi	2015	High
Tesla's key fob	Wireless access	2018-2019	Medium

Table 3.1: Vehicle cyber-attacks summary

different chapters that together create a complete starting guide for beginning car hackers. All of the chapters are written simply so almost anyone can read the book and find more information about this field of study.

3.2.2 Previous theses

As with the books there are many theses [30, 31, 32] focusing on security in the automotive industry. I only found one which is focusing directly on Tesla's car, the other theses listed here deal in general automotive cyber security not directly on specific cars.

3.2.2.1 Tesla Model 3 Internal Network Security Analysis

My colleague Filip Machala had written a similar thesis [30] on the same car but he focused on mapping the automotive's networks in the vehicle. He found out that car is via LTE connection visible from the internet on opened port 1720 and that BroadR-Reach network is without protection. In his thesis are also many non important findings which I am sure that will help me in the beginning. Even if it does not seem to be a big deal all this information is helpful for me because the analysis was performed on the same car with the same software package. So I can be sure that what is stated in his thesis is a reliable source.

3.2.2.2 Threat modeling of the AUTOSAR standard

Adi Karahasanovic in his thesis [31] made a threat model of AUTOSAR standard. It does not seem at first glance that our theses have much in common

but the background which is described in his thesis is really helpful for understanding problems in this field.

3.3 Information acquired from the vehicle

In this section you can find what can be revealed from the car just by looking and observing. In the last subsection 3.3.3 with little help from professional auto mechanic disassembling.

3.3.1 Outside of the vehicle

When I took a walk around the car it showed me only a few valuable pieces of information. The VIN (Vehicle identification number) and license plate number. We can search those numbers in online databases to find more info about the car. I can't list here any of the numbers due to privacy concerns. At the back of the car is a charging port but you can not unlock it without access into the car's touchscreen or the key fob assigned to this car. Between the left front and rear doors is a NFC reader for locking and unlocking the car. When the panel was disassembled the reader was on the right side as well but there isn't any antenna so you can't lock/unlock the car from the passenger's side. One of my colleagues tried to unlock another Tesla with the NFC card. It of course did not unlock the Tesla but the car started recording his behaviour which is great if you want to know what other people are doing with your car.

3.3.2 Inside of the vehicle

If you have access to the vehicle interior which some attackers could be able to. For example people who want to buy a car and they visit a car dealership then they can get hands on the interior. In american movies I saw many times that in luxury restaurants or parties you borrow your car to the service staff and they park it for you. Another example is staff in car dealerships where you need to go in case of regular service or if something in the car does not work properly. So if an attacker needed physical access to the inside of the vehicle for installing the exploit it is possible to be done. When Toyota stated that they only focus on securing the car from wireless attacks they did not obviously expect this opportunity. If you do not allow anyone to access your car interior you should only be scared of the wireless attacks.

The Tesla's interior is really minimalistic; there are only pedals, steering wheel with two rotating buttons, two levers and a big touch screen 3.1. The control of the vehicle's movement such as steering, braking, accelerating or gear shifting is not interesting for us. From the touchscreen we can control air conditioning, heating, navigation, charging port opening/closing and all what you expect to control on a car.

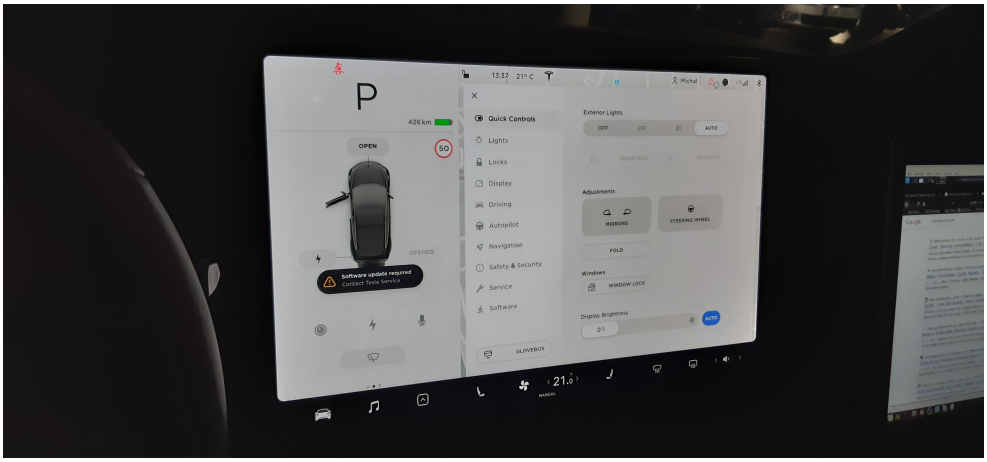

Figure 3.1: Touch screen in front panel

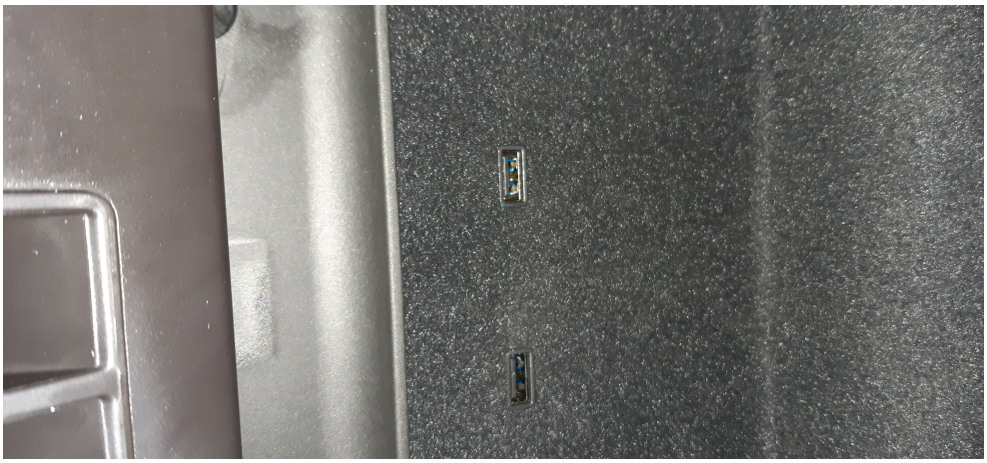

Figure 3.2: Front USB ports

The touchscreen has no connectors. But there are two USB ports below the touchscreen in the front panel 3.2. In the middle of the front seat is also a panel with another two USB ports and an OBD-II connector which are easily used from the back seats 3.3.

If you want to drive the car you need to place either a key fob or a NFC card on the middle panel of the car.

Figure 3.3: Rear USB ports and OBD-II connector

3.3.3 Disassembled vehicle

By disassembled vehicle I do not mean dismantled into smallest pieces possible but mainly removing coverage of units or sticking nose under the surface. From the outside of the car we can disassemble both of the mirrors. It shows us 3 antennas. One of them has not a label and is situated in the right mirror 3.5. Two remaining antennas are in the left mirror and both of them have labels with their brief description 3.4. One antenna is for LTE connection and the second one is for Wi-Fi connection. It is a smart solution from the manufacturer to put these antennas outside of the vehicle for better and more distant connection. The interior of the car is hiding treasures as well. One of them is located in front of the front passenger. If we disassemble the panel we will see the MCU 3.6. We almost immediately see that it has a lot of cable attached to it.

3.4 Summary

In the first section 3.1 of this chapter is described how known cyber attacks were done in recent years and also how every single manufacturer handled the threat. It was shown that Tesla takes cyber security in its vehicles very seriously and their responses are quick over-the-air fixes that even with those complicated changes arrive within days. After the attack on Tesla Model S described in section 3.1.6 manufacturer secured all their ECU against non-original firmware updates with cryptographic signing which means that we are not able to modify software on our car.

The next section 3.2 is about related work with having something in common with my thesis or was inspired by them.

Figure 3.4: Disassembled left mirror with Wi-Fi and LTE antenna

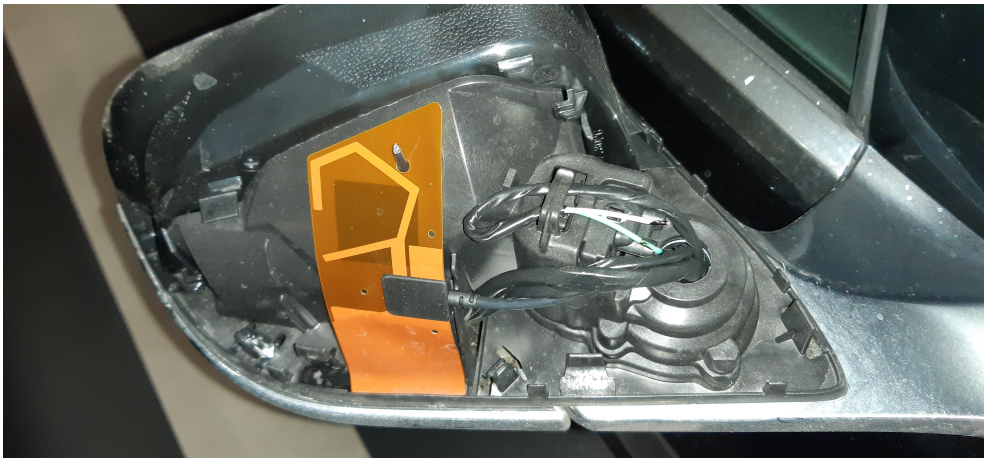

Figure 3.5: Disassembled right mirror

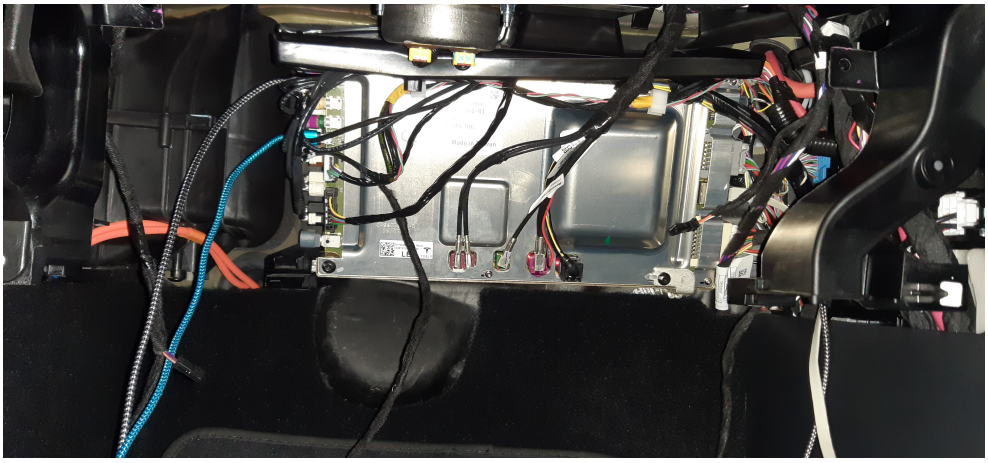

Figure 3.6: Media Control Unit

The last section 3.3 described the first look on the car from the outside view then the inside view and finished with the car disassembling for better representation of how the car's ECUs work together and what is where situated.

Chapter 4

Threat Modeling

In this chapter the reader will be introduced to the threat modeling. How is it done, what is it and why we do it.

Threat model is structured representation of information that affects security of the examined application, network, product or company.

Threat modeling is a process of collecting, gathering, organizing, analyzing and studying all available information. It enables doing smart, informed decisions. After producing a threat model we can make a list of security improvements sorted by their priority [33].

There are three main factors in threat modeling first of them are assets and the second are processes and the last are threat agents or sometimes called attackers.

4.1 Introduction to Threat Modeling

4.1.1 Assets and Processes

The assets represent all that we need to protect against the threat agents. It could be physical items such as the whole car or just digital assets such as data logs or other resources.

The processes are the next things that we want to protect. The attacker can disrupt the process in business then it could stop producing money which is highly important for the company. So in threat modeling we also want to know where processes have vulnerabilities in order to protect them from attackers. However in this case I am making a threat model of a vehicle so I will not take processes into consideration.

4.1.2 Threat agents

Sometimes also called just simply attackers. When we take into consideration the attackers we can divide them into groups by their skill, motivation and

Internal	**External**
Employees	Business Partners
Management (executive, middle)	Competitors
Administrators (network, system, server)	Contractors
Developers	Suppliers
Engineers	Nation States
Technicians	Organized Crime
Contractors (with their external users)	Hacktivists
General user community	Script Kiddies (recreational/random hacking)
Remote Support	

Table 4.1: Threat Agents [2]

also by their location to the organization either internal or external. In table 4.1 are listed main threat agents.

4.1.3 Attack surface

Attack surfaces are points from which an attacker can launch an attack onto the vehicle. They divide into categories which are described below.

Can be physical or wireless. If the attack is physical the attacker needs to have physical access to the vehicle itself. In some cases he needs even physical access to the port inside of the vehicle! The wireless attack on the other hand does not require physical access and can be done from distance depending on the chosen channel.

4.1.4 Attack tree

The final goal of an attacker has many smaller subgoals which can be visualized by the attack tree. The root of the tree is the main goal of the attacker and each childs of the nodes represents one possible path to it.

Figure 4.1 shows the attack tree of opening a safe. The main goal is to open the safe which can be done by many different paths. Each node is either possible, impossible or has childs that describe varying methods to complete the goal [34]

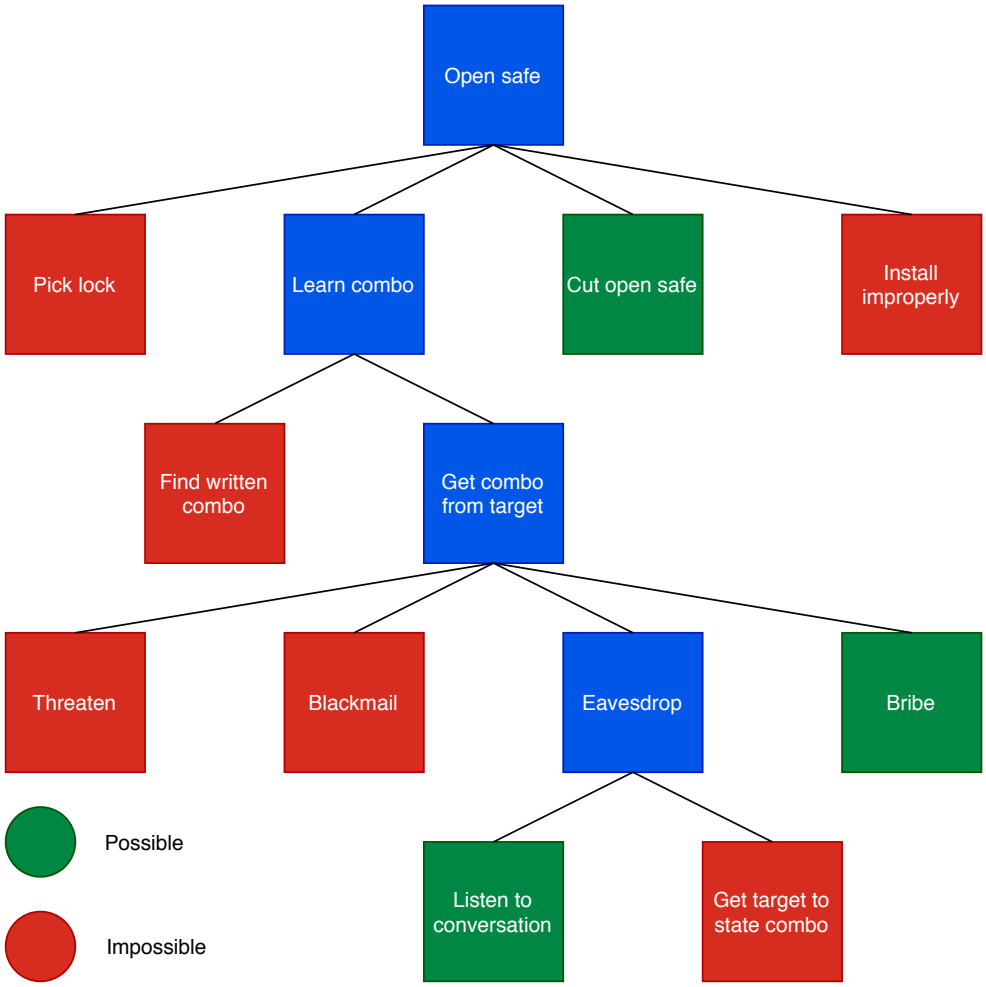

Figure 4.1: Attack Tree [34]

4.2 Threat modeling approaches

When making a threat model we can look at the whole process with different approaches. Here are the main four approaches: Software-centric, Asset-centric, Attacker-centric, Threat-centric.

4.2.1 Software-centric

With this type of approach we focus on software design of the system. It can be visualized with software architecture diagrams or use case diagrams. Software centric methods are used mainly for networks and complicated systems [35].

4.2.2 Asset-centric

This method is trying to identify the assets or data of an organization that entrusted them into some system or software. These assets are classified by their sensitivity and value to an attacker. Asset centric approach is using attack trees, attack graphs or visualized patterns that could potentially attack the asset [35].

4.2.3 Attacker-centric

In this approach as the name itself suggests we are focusing on the attacker. This method requires analysis of the attacker's skill and motivation to exploit vulnerabilities. We build the attacker's profile and then use it to understand which of the potential attackers is the most dangerous to us. After the organization understands potential attackers the organization should implement strategy to mitigate threats. With an attacker centric approach we focus on goals that the attacker wants to accomplish and how the attack could be carried out and in the end how to deflect and mitigate the attack [35].

4.2.4 Threat-centric

All of the approaches mentioned above are great but together they make an incomplete threat picture. That is the reason why another approach had to be developed. The threat-centric approach has three points of interest:

1. In a threat-centric approach we are again focusing on assets just as in asset-centric approach but this time not only on data. We also take into consideration the system's capabilities (ability to transfer data between accounts, physical systems).
2. If the IT system has assets there will be attackers who will try to get those assets. Threat-centric approach does not focus on the attackers but focuses on defenses against them.
3. The last point is that an attacker needs a starting point from which he could launch an attack (attack surface). We have to make an analysis of our attack surface which could possibly attackers use to start an attack.

This approach is basically trying to step ahead of the three traditional approaches listed above. The output of it is used to qualify the organization ecosystem [35].

4.2.5 Threat modeling methodologies

When it comes to threat modeling we can select from a wide range of threat modeling methodologies or sometimes called frameworks. Each of them belongs into one or some combination of the approaches listed above. Here

are listed the most used threat modeling methodologies: STRIDE, DREAD, PASTA, VAST, Trike, OCTAVE, NIST, TARA. For each I will make a brief description.

4.2.5.1 STRIDE

Stands for:

- Spoofing of a person or a computer - violates authenticity
- Tampering of data - violates data integrity
- Repudiation of an action - violates non-repudiability
- Information disclosure - violates confidentiality
- Denial of service - violates availability
- Elevation of privilege - violates authorization

STRIDE was developed by Microsoft in the late nineties. It is mainly used by developers during the design phase. It helps them to think about security of the product in the early phase of development. The goal of STRIDE is to make sure that the application meets the security properties namely confidentiality, integrity, availability (these three are also called CIA), authorization, authentication and non-repudiation [36].

4.2.5.2 DREAD

Stands for:

- Damage potential - How great is the damage if someone exploits the vulnerability?
- Reproducibility - How easily can someone reproduce the attack?
- Exploitability - How easily can someone launch this attack?
- Affected users - How many users are affected (percentage)?
- Discoverability - How easy is it to find the vulnerability?

Each of these questions has to be answered on a scale between one and three.

DREAD was developed as an add-on to the STRIDE model. DREAD allows us to rank threats when they are identified [36].

4.2.5.3 P.A.S.T.A.

Process for Attack Simulation and Threat Analysis (PASTA) is a process with seven steps that are focusing on technical security requirements and business objectives. Namely:

- Define objectives
- Define technical scope
- Application decomposition
- Threat analysis
- Vulnerability and weaknesses analysis
- Attack modeling
- Risk and impact analysis.

Each of these steps have several substeps [36].

4.2.5.4 Trike

Trike is an open source framework used for threat modeling and risk assessment. It does not operate from the attacker's side but rather tries to view the defensive part of the problem. Trike is modeling the system that you are defending and every system component is considered from CRUD (Create, Read, Update, Delete) view meaning who has the ability to update this particular component. Threats are then identified from a data flow diagram and could be either denial of service or escalation privilege [36].

4.2.5.5 VAST

Visual, Agile Threat Modeling (VAST) is designed to integrate into devops workflow. VAST was developed later from other methodologies with the vision to be scaled across the whole infrastructure. Thanks to late development VAST does not have shortcomings as the older methodologies. It is focusing on the development team and infrastructure team as well. Both of these teams have different threats and thanks to VAST they can see all the threats at one place [36].

4.2.5.6 OCTAVE

Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) focuses on organization risks and was developed at Carnegie Mellon University. The risks have three stages:

- Build asset-base threat profiles
- Identify infrastructure vulnerability
- Develop a security strategy and plans [36].

4.2.5.7 NIST

The National Institute of Standards and Technology (NIST) from United States has their own threat modeling framework consisting of four steps:

- Identify and characterize the system and data of interest
- Identify and select the attack vectors to be included in the model
- Characterize the security controls for mitigating the attack vectors
- Analyze the threat model [36].

4.2.5.8 Intel's TARA

Threat Assessment & Remediation Analysis [37] Consists of three main components: Threat Agent Library (TAL), Common Exposure Library (CEL), Methods and Objectives Library (MOL).

There are six main steps in TARA that need to be done.

1. Measure current threat agent risks to Intel.
2. Distinguish threat agents that exceed baseline acceptable risks.
3. Derive primary objectives of those threat agents.
4. Identify methods likely to manifest.
5. Determine the most important collective exposures.
6. Align strategy to target the most significant exposures [3].

4.3 Threat modeling

In this section is described how I defined assets of my threat model, which framework I chose as most suitable and the results of my threat modeling.

4.3.1 Assets

4.3.1.1 The car itself

I do not want the car to be stolen or severely damaged. However this asset can be easily protected with a safety garage and fence over the garage property. The people who do not own a garage could buy a place for their car in a guarded object in the vicinity of their residence.

4.3.1.2 The safety of the passengers in the car

This is a very important asset. I want all the safety systems to work properly such as airbags, ABS and so on. As well as the asset above this can be easily mitigated too. Simply not allowing anyone to come close to your vehicle is sufficient.

4.3.1.3 Personal information

The software in the car saves my personal information. I do not want this information to be accessible to anyone who walks by.

4.3.2 Threat agents

In my threat model I chose the most relevant attackers as following:

4.3.2.1 Tesla employees

Tesla internal employees, their contractors and suppliers have detailed knowledge of every car component as well as how are connected together all those components. Tesla software developers are able to acquire source codes of components that are developed in the house. The manufacturer workers in the factories could be potentially able to swap some of the components to which they have access for the same component but with modified and malicious code running on it. They could be later able to bypass the security modules in the vehicles and steal or break the car itself. Also some of the developers or managers from Tesla have access to Tesla's private keys which are needed to sign new versions of systems that are Tesla distributing as over the air updates. With knowledge of this key they are able to update software on the car with custom and potential malicious software.

4.3.2.2 Tesla retail workers

When the customer goes into the car dealership there are workers who help him with choosing the right car. These workers have available cars that are ready for the test drives with the potential customer. Later these cars are sold as demonstration vehicles with discounts. The workers there are able to swap components of the car with non original pieces which could lead to more frequent visits into the car dealership with problem solving and handling them more money. Another motivation is pairing another key with the vehicle and then they are able to steal the vehicle.

4.3.2.3 Car thieves

Those people work in groups or sometimes as individuals. Their main motivation is money or the car itself. They often buy or receive products that are

able to exploit the car's security systems and steal it just to sell it later.

4.3.2.4 Script kiddies

Are mainly individuals who are not familiar with cybersecurity details but only try to run the script that they download from somewhere on the internet and see what happens. They do not know what the script is doing and can not modify it to better suit their circumstances. Main motivations of script kiddies are fun, curiosity or harm to other people they do not like.

4.3.2.5 Rioters

Rioters or hooligans can damage or ruin the car standing on the street. They are unpredictable and the harm that they can cause is expensive and in some cases even unrepairable.

4.3.2.6 Envious neighbors

They are just neighbors and they do not like that someone has better items than them.

4.3.3 Threat agent's attributes

Threat agents vary in their attributes. Below are described the most obvious attributes with a brief explanation.

4.3.3.1 Access

Can be either internal or external. The internals are mainly workers in the company which is the object of the threat modeling or they are working on a product that is being analysed. The external is all remaining people that do not have access to the resources inside a company.

4.3.3.2 Goal

Goal is the final primary target of an attacker. If a hacker is trying to hack someone's banking account his goal is to steal money from the victim.

4.3.3.3 Intent

Intent can be either harmful or not. Sometimes the company workers just simply make a mistake or do not know that they are making something wrong.

4.3.3.4 Resources

Resources are all the things or humans that a threat agent can use for his attack.

4.3.3.5 Skills

Defines the level of attacker's knowledge. His ability in hacking or just the knowledge of the development process that the targeted company uses.

4.3.3.6 Motivation

Motivation is why the threat agent is attacking. Down below are listed the most popular.

4.3.3.6.1 Profit Can be direct or indirect. When someone steals a car and sells it he makes a direct profit. If a company casts a slur upon another company and starts selling more products that is indirect profit.

4.3.3.6.2 Fun/Entertainment A lot of enthusiasts are trying to hack their cars just for fun. If they manage to root their car's system they are able to play games on the vehicle system for example.

4.3.3.6.3 Cause trouble to other people Most of the script kiddies and hooligans want to simply harm other people they envy. They often do not care about anything else.

4.3.3.6.4 Curiosity/Learning new skills Many hackers only want to know what is inside their vehicle and how the components interact with each other.

4.3.4 Intel's TARA again and more detailed

I chose Intel's TARA framework to make my threat model. It suits best my purpose of the connected car technology. First I will describe more details of this framework that are needed to understand the creation of the threat model. The basic description of the framework can be found at 4.2.5.8. Now I will go into greater details.

4.3.4.1 Terms

- Objective - What the threat agent hopes to accomplish by the attack
- Method - Process by which a threat agent attempts to exploit a vulnerability achieve an objective
- Attack - Action of a threat agent to exploit a vulnerability
- Control - Tools, processes and measures put in place to reduce the risk of loss due to a vulnerability
- Exposure - Vulnerability without a control [3].

4.3.4.2 TARA's components

- Threat Agent Library

  This is the library that contains a simplified set of all possible threat agents. It defines eight most common attributes of the threat agents. From these attributes comes 22 unique combinations that represent the majority of the all possible threat agents.

- Common Exposure Library

  In CEL can be found known security vulnerabilities. Into Intel's Common Exposure Library are added other publicly available CELs for additional data information. The library is focusing on exposures that are relevant; it does not consider non relevant threats such as known viruses when the computer has up to date anti virus installed.

- Methods and Objectives Library

  This library contains known threat agent objectives (what want the threat agent to accomplish) and the methods how to reach the objectives. When MOL is merged with TAL it shows us the types of possible attacks that are based on many factors.

4.3.4.3 Threat Agent Library

I started with Intel's TAL [38] that is publicly available and added some new threat agents from section 4.3.2 that are not already listed in the original TAL. This creates a whole new TAL that describes the main threat agents who are the most dangerous for my case. I also deleted all the threat agents that are not relevant for my case. I do not think that someone from the government will try to attack this specific car. All agent's attributes that are listed in the table 4.2 are well described in [38]. From Tesla's internal employees and Tesla's service staff I created threat agents that I think are the most relevant. All new threat agents are:

4.3.4.3.1 Reckless Employee Is simply someone who makes a mistake and can damage what is in his radius.

4.3.4.3.2 Untrained Employee Newcomer in his job. He is in the beginning of his training and does not know what he can and can not do.

4.3.4.3.3 Disgruntled Employee Unhappy employee who does not like his employer and his company. He wants to cause damage and havoc inside the company and still remain hidden.

4.3.4.3.4 Internal Spy Could be an employee or someone who has access inside the company and his motivation is to steal information and give or sell them to other companies or individuals or just to the one who pays the most.

	INTENT	NON-HOSTILE		HOSTILE				NEW		
		Reckless Employee	Untrained Employee	Rioter	Employee Disgruntled	Internal Spy	Thief	Script Kiddies	Envious Neighbor	Car thief
Access	Internal	X	X		X	X	X			
	External			X				X	X	X
Outcome	Acquisition/ Theft					X	X			X
	Business Advantage				X					
	Damage	X	X	X				X	X	
	Embarrassment	X	X		X			X	X	
	Tech Advantage					X				
Limits	Code of Conduct		X							
	Legal	X								
	Extra-legal, minor					X	X	X	X	
	Extra-legal, major			X	X					X
Resources	Individual	X	X		X		X	X		
	Club			X					X	X
	Contest									
	Team									
	Organization					X				
	Government									
Skills	None			X			X	X		
	Minimal		X						X	
	Operational				X					X
	Adept	X				X				
Objective	Copy					X				
	Deny									
	Destroy			X	X				X	
	Damage				X			X	X	
	Take						X			X
	All of the Above/ Don't Care	X	X							
Visibility	Overt		X	X						
	Covert	X							X	
	Clandestine					X	X	X		X
	Multiple/ Don't Care				X					

Table 4.2: Threat Agent Library

4.3.4.4 Methods and Objectives Library

My MOL will be set up from Intel's MOL [3]. As well as in the section 4.3.4.3 I will modify the threat agents. The objectives will remain the same as Intel has them in [3].

4.3.4.5 Common Exposure Library

The CEL from Intel is confidential, it means that Intel is protecting it because they do not want to share their vulnerabilities with the whole world and I have to accept that. I will come up with attack surfaces that I think are the most important and then complete my hypothesis with another information from public sources. Below are listed all attack surfaces that I think are important to mention.

Threat agent	Attacker					Objective		Method									Impact				
	Access	Trust				Motivation	Goal	Acts					Limits								
		None	Partial Trust	Employee	Administrator			Copy, Expose	Deny, Withhold, Ransom	Destroy, Delete, Render Unavailable	Damage, Alter	Take, Remove	Code of Conduct	Legal	Crimes Against Property	Crimes Against People	Loss of Financial Assets	Business Operations Impact	Loss Of Competitive Advantage, Market Share	Legal of Regulatory Exposure	Degradation of Reputation, Image, or Brand
Reckless Employee	Internal		X	X	X	Accidental/ Mistake	No malicious intent, accidental	X		X	X			X			X	X	X	X	X
Untrained Employee	Internal		X	X	X	Accidental/ Mistake	No malicious intent, accidental	X		X	X		X				X	X	X	X	X
Rioter	External	X				Social/ Moral Gain	Satisfaction or Change Public Opinion			X	X					X	X				X
Employee Disgruntled	Internal		X	X	X	Personal Gain (Emotional)	Damage or Destroy Organization		X	X	X				X			X	X		X
Internal Spy	Internal		X	X	X	Personal Gain (Financial or Emotional)	Business or Technical Advantage	X		X	X	X			X		X	X	X	X	X
Script Kiddies	External	X				Social/ Personal Gain (Emotional)	No malicious intent or Satisfaction				X			X			X				X
Envious Neighbor	External		X			Moral/ Personal Gain (Emotional)	Satisfaction			X	X				X		X				X
Car thief	External	X				Personal Gain (Financial)	Obtain Financial Assets or Satisfaction					X			X		X				X

Table 4.3: Methods and Objectives Library [3]

4.3.4.5.1 Charging port The vehicle has a charging port on the rear left side. The Tesla does not have a combustion engine and uses electrical energy for driving. When the charger connects with the car there has to be some kind of communication in between which could be abused.

4.3.4.5.2 Controller area network What this type of network is and what is its purpose I discussed at 2.2.1.1. Any message on this network is broadcasted on all devices on the network and is trusted so if someone manages to infiltrate into this network he can send messages directly to all units. This could lead to unlocking and stealing the car if there is part of the CAN easily accessible from the outside.

4.3.4.5.3 Access Card The owner of the car needs to protect this card. If he loses it or allows someone to steal it. The attacker can unlock and steal his car.

4.3.4.5.4 Sensors The cameras, radar and all other vehicle's outside sensors are in my opinion another weak point. The sensors can be tricked or abused into thinking that there is no obstacle on the road when there is one or vice versa.

4.3.4.5.5 OBD-II On-board diagnostics port is mandatory on commercial cars. Many fraudsters are abusing OBD-II into illegally decreasing the number of travel kilometres on their cars which leads to increasing the price of the car [39]. However the port can be definitely abused in other ways as well.

4.3.4.5.6 USB There are a lot of USB attacks [40] and some of them can be for sure used on the car as well on the personal computer.

4.3.4.5.7 Ethernet port Ethernet port can be found at passenger's footwell in the MCU. This is the primary scope of this thesis in the following chapters.

4.3.4.5.8 Pin Code The car can be driven only after the user enters pin code if this security feature is enabled in the settings.

4.3.4.5.9 Infotainment The infotainment system is the computer with a touch screen between front passengers. It is used for media management, radio, navigation and also for controlling secondary functions of the car such as heating or air cooling.

4.3.4.5.10 Wi-Fi The vehicle can connect to Wi-Fi which is then used as the main network for downloading the updates for software. It is important to have a secured network to which the car is connected.

4.3.4.5.11 Cellular If the car is not connected to the Wi-Fi or is on the road where there is no other option it is using a cellular connection. Same as all the other technologies it can be abused in the way the authors of the technology did not think of. The vehicle can be tricked into connecting to a fake access point that could lead to a man in the middle attack.

4.3.4.5.12 Bluetooth Bluetooth is used mainly for communicating with the smartphone of the user. Nevertheless it is another surface that has to be secured.

4.3.4.5.13 Key Fob Is used for locking and unlocking the car and also is needed for driving. The key fob can be stolen or copied. The owner of the car has to guard the key fob very closely.

4.3.4.5.14 Tire Pressure Monitoring System Modern cars often have this type of technology and since 2014 it is even mandatory [41]. Modern in vehicle systems are often dependent on TPMS and when one wheel is reporting that it is out of air the car will probably stop or warn the driver and try to force him to stop for his safety. When this happens on the highway it can lead to serious damage to the car and the passengers as well.

4.3.4.5.15 Smartphone Almost all car manufacturers have their own application for smartphones. Users can see information about their cars and control the car with limited functionality. If the owners of the car can control it from their smartphone they have to make sure that their smartphone is using the latest updates of the operating system and application and is well secured otherwise attackers could use his smartphone as entry point to the vehicle.

For all these attack surfaces are classified their impact and how hard it is for attackers to abuse them in terms of access. The output of this classification is listed in table 4.4.

	Access			Impact		
	Physical Inside	Physical Outside	Wireless	Data Theft	Passenger's Safety	Vehicle Theft
Charging Port		X		X		
Controller Area Network	X	X		X	X	X
Access Card		X		X		X
Sensors		X			X	
OBD-II	X			X		X
USB	X			X		
Ethernet Port	X			X		
Pin Code	X			X		X
Infotainment	X			X		X
Wi-Fi			X	X		
Cellular			X	X	X	
Bluetooth			X	X		
Key Fob		X		X		X
Tire Pressure Monitoring System			X		X	
Smartphone			X	X		X

Table 4.4: Common Exposure Library

4.3.4.6 TARA's steps

1. One or more experts reviews and ranks threat level with usage of TAL. It is qualitative as well as quantitative work that will establish a foundation.

2. With TAL it is needed to measure threat levels for new projects or create new acceptable risk for the existing project. At the end of this step we discovered new threats that either have exceeded new risk or just popped up.

3. Objectives are a combination of threat agent motivation and his capabilities. From the threat agents identified in step 1 and 2 we conclude their primary motivation. This step uses MOL.

4. Here is MOL used to identify the probable attack methods. A method is defined as a combination of threat agent objective and his operating method.

5. Here takes place finding attack vectors using the CEL. They are then compared with the methods from step 4 and attack vectors. Then these exposures are ranked with regard to their severity. The outcome of this is a list of the most important exposures.

6. The results from this analysis are the most important areas of concern of information security. They should ideally be used to create a better defense system of an application.

4.3.4.7 Threat Modeling Outcome

The whole car and most of the physical attack surfaces can be easily protected by parking the car in the garage or over the fence on a protected property. There will be safer than on the street. The key fob and access card has to be also in a safe place that is also entirely into the hands of the owner of the car. He must be sure that his smartphone is well protected and updated if he uses it to access the car. That is everything the user can do to protect his car. The manufacturer of the car is almost fully responsible for the other attack surfaces. The owner of the car must only ensure that the software is up to date. This way the software will be protected against newest threats.

The car manufacturer, Tesla in this case, must ensure that the car is well secured and working properly. And if some new vulnerability is discovered Tesla has to fix the issue so that vulnerability can no longer be exploited.

The most important attack surfaces are the wireless ones because they have an effect on safety of the passengers. If the attacker needs physical access to the interior of the car and the outcome is to steal personal data it has the smallest priority.

Top priority attack surfaces:

1. Tire Pressure Monitoring System,

2. Cellular Connection,

3. Sensors,

4. CAN.

All of these are in complete control of the manufacturer.
Average priority attack surfaces:

1. Smartphone,
2. Key Fob,
3. Access Card.

On the other hand all of the middle priority attack surfaces are in complete control of the user of the car. He needs to make sure that no one will abuse them.

4.4 Summary

In the beginning of this chapter is described that assets are the things we are trying to protect. The assets can be splitted into many categories, can be physical or non physical such as a good name of a company.

Then there is described who is a threat agent. It could be a person or a process that can potentially destroy our assets intentionally or even without knowing that he has done something wrong.

At the end are described how the threat model can be evaluated and to be looked at. This has many methods of approaches and also there are a lot of frameworks that can help us with threat modeling. A good start is knowing what for is the framework constructed and how it works, so that I can choose the right.

Chapter 5

Vulnerability Analysis

5.1 Analyzing firmware

5.1.1 Obtaining firmware

Firmware can be obtained via various paths. Here are few examples:

- CAN sniffing
- Catching over-the-air update as man in the middle
- Read from SD card, hard drive or eMMC
- Some vendors simply place firmware on their web so you can download it and upgrade the device yourself.

I received mine firmware from my supervisor and his colleagues. They got it via reading from a hard drive. I will not describe details about it here because it was not my work. I downloaded one file that will be further analyzed.

5.1.2 Firmware Analysis

I started examining the file with file and binwalk commands. From the output shown at listing 1 can be found out that the file contains Squashfs filesystem and compressed with gzip.

Command unsquashfs can be easily used on the file to be uncompressed from a single file into the filesystem hierarchy. The whole command and its output can be at listing 2. The command unsquashfs creates a directory called squashfs-root also shown at listing 2. In the directory could be found a directory structure known as FHS (Filesystem Hierarchy Standard). FHS defines directory structure and their purpose in Linux distributions. It was specified,

```
$ file 2019.20.4.2.model3
2019.20.4.2.model3: Squashfs filesystem, little endian,
version 4.0, 1210323351 bytes, 23894 inodes, blocksize:
131072 bytes, created: Tue Jun 25 02:20:12 2019

$ binwalk 2019.20.4.2.model3

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Squashfs filesystem,
                              little endian, version 4.0,
                              compression:gzip,
                              size: 1210323351 bytes,
                              23894 inodes,
                              blocksize: 131072 bytes,
                              created: 2019-06-25
                              02:20:12
```

Listing 1: Output from file and binwalk command

developed and now maintained by The Linux Foundation [42]. In those directories is the whole filesystem which runs in the car on the infotainment unit.

I found out that the infotainment unit is named cid or ice by Tesla and in its local network has probably other units. File /etc/hosts serves the purpose of static network address translation. I searched the filesystem only slightly and listed all relevant information that I found. The whole filesystem has size 2.2 GB. All this information is shown at listing 3.

5.2 Examination of the vehicle

Tesla Model 3 has an ethernet port on the passenger's site under plastic covers. I plugged my computer into it and used information from the vehicle network listed in file /etc/hosts. I set my IP address to 192.168.90.101 which is not used by any of the relevant nodes. Then I used wireshark to capture all traffic that was happening on that port.

In wireshark we can sniff and capture packets that are running in the vehicle network. After a brief analysis I did not find anything important so I moved to the next step.

I used Nmap to discover what services are running on the target IP address. I also tried to discover all UDP services but none was found. There is a HTTP service running on port 8080. I simply tried to connect to it via web browser and a web interface popped up that is shown on figure 5.1. It has some buttons

```
$ sudo unsquashfs 2019.20.4.2.model3
Parallel unsquashfs: Using 4 processors
21882 inodes (35897 blocks) to write

[=============================================/] 35897/35897 100%

created 20289 files
created 2012 directories
created 1592 symlinks
created 1 devices
created 0 fifos

$ cd squashfs-root/
$ ls
bin      etc    lib64  opt   run      sys  var
deploy   home   media  proc  sbin     tmp
dev      lib    mnt    root  service  usr
```

Listing 2: Extracting filesystem and content of extracted firmware

```
$ cat etc/hostname
cid

$ cat etc/hosts
127.0.0.1 localhost
192.168.90.100 cid ice
192.168.90.100 ic
192.168.90.102 gw
192.168.90.103 ap ape
192.168.90.104 lb
192.168.90.105 ap-b ape-b
192.168.90.30 tuner
192.168.90.60 modem
127.0.0.1 mothership.vn.teslamotors.com firmware.vn.tesla
motors.com firmware-ota.vn.teslamotors.com firmware-bund
les.vn.teslamotors.com
10.33.168.223 hermes-prd1.teslamotors.com

$ du -sh squashfs-root/
2.2G    squashfs-root/
```

Listing 3: Hostname of the unit, content of etc/localhost and size of uncompressed firmware

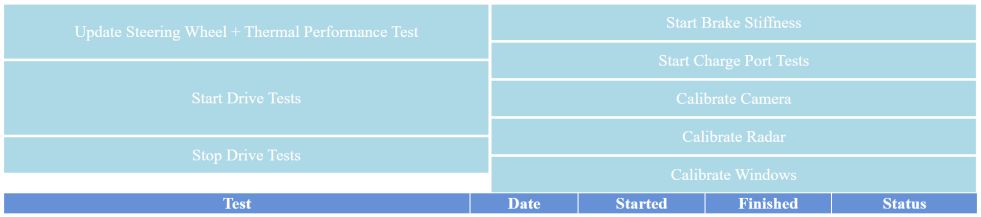

Figure 5.1: Web Interface on 192.168.90.101:8080

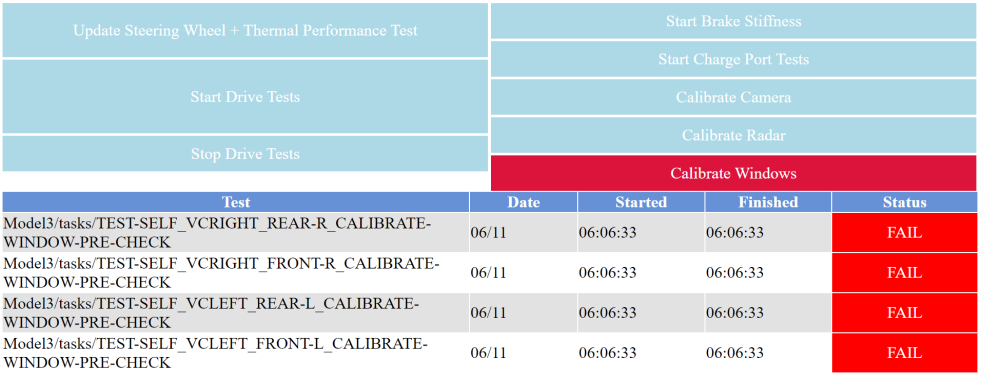

Figure 5.2: Web Interface with Failure on 192.168.90.101:8080

which are clickable and when I clicked them there just turned up a text with failure 5.2.

Now I know how the interface of the service looks like and I also know some texts that can be searched in the firmware for identification of that particular service.

5.3 Reverse engineering

"Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction. During this process we document and try to understand how the studied system works." [43]

Right now I do not have much information about the diagnostics service so it occured to me that I can search for the string I saw in the HTML file on 192.168.90.100:8080 which could possibly lead into discovery of the previously mentioned service. I selected string "Calibrate Windows" and searched it via command grep in the firmware file system and string was found in directory /opt/odin/core/engine/assets/onboard with filename index.html on line 27. Whole command and output is shown at listing 4.

```
$ grep -rnw . -e "Calibrate Windows"

./opt/odin/core/engine/assets/onboard/index.html:27:
<div class="button" id="window-btn" onclick="start_windo
w_calibration()">Calibrate Windows</div>
```

Listing 4: Successful searching for string "Calibrate Windows"

This gave me enough information where to start with reverse engineering the service as well as the name of the service. With this information I searched for all files with this name. Because the word "odin" is also contained in the word "encoding" I had to remove false positive results from the search.

```
$ find . -name "*odin*" | grep -v "ncoding"

./usr/bin/odin-trigger-trampoline
./service/odin-engine
./opt/odin
./opt/odin/odin.id0
./opt/odin/odin
./opt/odin/odin_bundle.zip
./opt/odin/odin.til
./opt/odin/odin.nam
./opt/odin/odin.id1
./opt/odin/odin.id2
./etc/sv/odin-engine
./home/odin
```

Listing 5: All findings of "odin" in filenames

Then I started investigating files and content of folders from the output shown at listing 5.

Only folder opt/odin and file etc/sv/odin-engine/run are interesting. File etc/sv/odin-engine/run looks like being responsible for starting diagnostics service on the vehicle. His shortened and interesting version is shown at listing 6.

Folder opt/odin contains a lot of files with .so endings. Those are files that are probably compiled libraries that are used by odin.

5.4 Running odin

If I succeed in starting up Odin I can test this application while it runs. This step is really important because when it runs in a safe environment it can not

```
$ cat etc/sv/odin-engine/run
#!/bin/sh
...
if [ -x /usr/bin/is-in-factory ] && /usr/bin/is-in-factory; \
then
    CONFIG_FILE=factory
else
    CONFIG_FILE=onboard
fi
...
if [ -f /opt/odin/main.pyc ]; then
    exec /usr/bin/python3 /opt/odin/main.pyc \
    --config-file=/opt/odin/config/$CONFIG_FILE.yaml start \
    engine
else
    exec /opt/odin/odin \
    --config-file=/opt/odin/config/$CONFIG_FILE.yaml
    start engine
fi
```

Listing 6: Shortened version of etc/sv/odin-engine/run

harm other applications in the vehicle or crash which could result in a trip to a Tesla's service shop.

5.4.1 Preparation

Before I will try to execute the program it is necessary to make fingerprint hashes of the binary files. We will store them and they will be lately used for comparison if the binary files have not changed. This step is really important and often ignored. Hashes of the files before running can be easily done by the "sha1sum" command in bash. It is convenient to store the hashes in file so that it can be easily compared later if some part of the program will change its data. I took a snapshot of my virtual machine and also turned off internet connection to make sure the program will not try to connect into Tesla's backend services. If a file is executed it shows help.

Odin requires a configuration file to be able to start. In the file etc/sv/odin-engine/run shown at listing 6 is the command where a specified configuration file is required by odin. Now we can take a look at it or even hopefully start the odin. I runned the command

```
exec opt/odin/odin --config-file=opt/odin/config/onboard.yaml start engine
```

```
$ ./odin

Usage: odin [OPTIONS] COMMAND [ARGS]...

  Main entry point for the ODIN cli.

  Args:     config_file: <str>

Options:
  --platform TEXT         vehicle platform name, e.g.
                          model_x. Overrides env var
                          and config file
  --fw-version TEXT       vehicle firmware version,
                          e.g. 17.12.4. Overrides env
                          var and config file
  --log-level TEXT        set log level, e.g. DEBUG.
                          Overrides config file
  --metadata TEXT         set the path to the metadata
                          required for ODIN per
                          Firmware
  --networks TEXT         set the path to the networks
                          required for ODIN per
                          Firmware
  --network-module TEXT   set the network module when
                          loading compiled networks
  --config-file TEXT      file that defines the
                          configuration for Odin, e.g.
                          configurations/config.yaml
  --help                  Show this message and exit.

Commands:
  isotp    ISO-TP debugging utilities.
  listen   Kickoff point for ODIN listen processes.
  odx      Runs ODX commands.
  start    Kickoff point for ODIN processes.
  tasks    Prints all task definitions for the
           current...
  uds      Runs UDS commands.
  version  Returns the current odin version.
```

Listing 7: Help contained in odin

without success. Program had issues with detecting on which platform it was running. So I added the option '–platform="model_3"' and it resulted in a different error. This time the program could not find file /opt/odin/odin_bundle.zip.

I later discovered that the program was not finding odin_bundle.zip relatively but absolutely at path /opt/odin/odin_bundle.zip. So I moved the whole odin folder into /opt/odin and made another snapshot of the virtual machine. After running the command again but this time in the right folder, odin started successfully and now we can see the odin interface at address 0.0.0.0:8080 which looks exactly like the one we saw when we connected into the real vehicle. There is only one small imperfection: odin could not find file /opt/odin/data/Model3/power_map.json which resulted in error but odin was still able to run. Later I will have to find why and if this file is important for odin but now it runs successfully and that is more important right now. The output is shown at listing 8.

```
$ sudo ./odin --config-file=config/onboard.yaml
  --platform="model_3" start engine
2020-07-16 09:51:23,982 -
  odin.platforms.platform_metadata_adapters- ERROR -
  Could not find power map json:
  /opt/odin/data/Model3/power_map.json
Traceback (most recent call last):
  File "odin/platforms/platform_metadata_adapters.py",
  line 187, in init_power_map
FileNotFoundError: [Errno 2] No such file or directory:
  '/opt/odin/data/Model3/power_map.json'
2020-07-16 09:51:23,985 - odin.core.cid.interface - INFO
  - Running subprocess: /usr/bin/is-in-factory (timeout:
  None)
2020-07-16 09:51:24,005 - odin_server.hermes - ERROR -
  Failed to connect to Hermes. Waiting 1 second(s) to
  reconnect.
======== Running on http://0.0.0.0:8080 ========
(Press CTRL+C to quit)
```

Listing 8: Successful start of odin

Now I can in a safe environment perform analysis without being afraid to potentially harm the vehicle. This is a very important step because when testing on the vehicle itself the odin service could theoretically be damaged when handled irresponsibly and then we could not start it again. Of course I tried again to click on some button but it failed again the same way that it failed on the real vehicle and also as I expected.

5.4.2 Examination

In the folder /opt/odin/core/engine/assets/onboard is located the server root. The source code of the web page is identical to the file located in this folder called index.html. With this knowledge we can search the server root for more information. The server root with all subfolders is shown at listing 9. There are many images of in vehicle connectors so I skipped them in the listing. There is one css file, two html files and two javascript files.

```
./index.html
./index_sx.html
./static
./static/js
./static/js/index.js
./static/js/index_sx.js
./static/img
./static/img/connectors
./static/img/connectors/X082.jpg
...
./static/img/connectors/G018.1.jpg
./static/css
./static/css/index.css
```

Listing 9: Files in server root

5.4.2.1 Examination of static files

5.4.2.1.1 HTML files Html file index.html is the file that can be seen when the web server is accessed via browser. The second html file index_sx.html can not be accessed on the web server. When I tried to access it the server always redirects me to index.html. After a brief examination I found out that index_sx.html contains two buttons for starting and stopping tests. After clicking on the stop test button a form for passcode prompts. However I am not sure how and when I can get access to this file via odin web server.

5.4.2.1.2 Javascript files Those two javascript files contain functions that support the logic and responses for the html files. There are two important parts the first is

```
xhttp.open("GET", "/api/v1/task\_history", true);
```

This command is used for getting history of tasks that were performed on the web server. They are refreshing at the bottom of the page. The next one is:

```
xhttp.open("POST", "/api/v1/products/current/commands", true);
```

This command is sending a post request to the web server with a command that the user wants to be executed such as "Calibrate Windows". I want to modify javascript file index.js and with that try to get all information from the odin.

5.4.2.2 OWASP ZAP

I tried to run OWASP Zap on the web server. Just to make sure I did not miss something important. Output of this program was successfully reported and can be found on the attached medium. The program reported 4 alerts, 1 with medium risk level, 2 with low risk level and 1 with informational risk level. I will describe all of the found alerts and examine their real risk level in my opinion sorted from the most dangerous to the least dangerous.

5.4.2.2.1 X-Frame-Options Header Not Set X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. ClickJacking attack is targeting the client who is browsing the malicious site. It can not be used to attack the web server thus I can not use it as an entry point of my exploit. How this vulnerability can be exploited is described at 6.3.

5.4.2.2.2 X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

MIME Sniffing could be used to send a file to the web server and potentially perform a XSS attack. But I do not see any option on how to send a file to the web server. However I will leave this type of attack on the alert if there will be some chance of performing this attack.

5.4.2.2.3 Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server

This alert warns that there is a possibility of attacking via XSS. If I will find some option to send data to the server I will definitely try to use this attack.

How this vulnerability can be exploited is described at 6.2.

5.4.2.2.4 Timestamp Disclosure - Unix A timestamp was disclosed by the application/web server - Unix

The fourth alert is only informational. Timestamp Disclosure can be abused by the attacker if the current timestamp is used by the web server in sensitive data such as key generating. If the web server somehow is making hand shakes with the connected computer and is using the key generated from this timestamp then there exists a possibility of solving this key without the need of brute force attack.

5.5 Summary

In this chapter was discovered that extracted firmware contains linux FHS and thanks to that is in an easily readable format for humans. The extracted file system is relatively large and surely contains a lot of interesting information. This could be used as the follow up thesis for other students.

Then there is a section with a real service that runs on the vehicle and is accessible via ethernet port. From the strings that were seen on that particular service can be found corresponding service in the extracted firmware.

The discovered files were examined and information from them used to start the service in the virtual machine. The chapter ends with examination of the HTML and javascript files that the service is using and the running service with OWASP ZAP.

Chapter 6

Exploiting

In the previous chapter were presented some vulnerabilities that could be potentially exploited. For each I will theoretically explain how I plan to exploit them and how significant will the attack be. Also for each I will make at least proof of concept exploits.

6.1 Proof of concept exploits

In section 5.4 is described how one can connect to a python aiohttp web server that is running on MCU. The web server is in my opinion used for other car diagnostics as substitution for OBD-II port. However OWASP ZAP reported two security vulnerabilities that could be more analyzed. Both of these vulnerabilities can not be used to directly attack the web server however I could try to insert some malicious code into the web server that could trigger when staff from the service would connect to the diagnostic server.

6.2 XSS attack

Cross site scripting can be used to attack someone else on the same site. Description can be found at 2.3.1 and where this attack can be performed is described at 5.4.2.2.3. This section contains designing of the attack.

To perform a XSS I have to insert malicious code into the page and then force browser or whatever the staff in the service is using to run the code. The main goal of this attack is to steal secret data from the staff's computer. But the page is very simple, it has only a few buttons and does not contain any forms to insert data. I have an extracted version of the html page and javascript script that I can modify but there is no option for moving the new files into the car's unit because the new packages that are updating the file system in the car have to be signed by Tesla's certificate. The javascript code contains on line 126 command

```
xhttp.open("GET", "/api/v1/task\_history", true);
```

and it could be an entry point for malicious code to be executed. I do not see any other option. My plan is this: put malicious code into task history and then the page will process it and perhaps try to execute it. The other plan if the code will not be executable could be to display some instructions for the worker to force him to hand over the secret data.

6.3 Clickjacking attack

How this attack works is described in 2.3.2 and where I found the flaw for this attack to be possible can be found at 5.4.2.2.1. To perform a clickjacking attack I would have to modify the UI elements of the page that is being displayed. I do not see any option here that would have the desired results. However one of the goals of this thesis is to suggest at least proof of concept exploits so I will outline what I could do if this step would be successfully solved. Again as in the previous section I would attack the staff's computer which I would try to force into sending secret data into my prepared server.

6.3.1 Attack outline

The UI of the page would still be the same and the buttons would also do the same action as before but right before doing that action that button will have an asynchronous request in JS code that will send secret data from the staff's computer into my server that will be prepared to receive this data.

6.4 Summary

In this chapter are outlined two types of attacks that cover vulnerabilities found in the previous chapter. Because they both need changes in the source code of the javascript files in the ECU's firmware I do not see importance in executing any of these attacks.

Conclusion

In the beginning of this thesis is described all information needed to fully understand this thesis. Then there is part with already known and demonstrated cyber attacks on vehicles and related works. Thereafter is the introduction of the vehicle of new generation Tesla Model 3 with several photos of slightly dismantled parts. The next chapter is about making a threat model. It discusses various threat modeling frameworks and ends with a creation of an own one. The chapter 5 contains security analysis of extracted firmware from the vehicle and focuses mainly on security of the web server. In the last chapter are outlined proof of concepts exploits that focuses on the vulnerabilities found in the previous chapter.

Since the beginning of working on this thesis I can see how the automotive industry is merging with information technology. The cars that will be produced in the future years will probably just be computers with wheels. That brings a lot of new threats and there are needed many people who will have to perform in depth analysis of every component that is the vehicle using to make sure that the cars are safe enough.

The result of this thesis is not only this text that you are reading but also the new information, experience and all other things that I learned during the work on the thesis.

There exist several ways of working on another theses that could build from information that was written or discovered during this thesis. In chapter 4 are described attack surfaces that could be further analyzed or there is also an option to continue analysis of the discovered web server.

Bibliography

[1] Schmid, M. Automotive bus systems. *Atmel Applications Journal*, 2006. Available from: https://user.eng.umd.edu/~austin/enes489p/project-resources/SchmidAutoBusSystems.pdf

[2] Threat Modeling. [Last Accessed on 13.10.2020]. Available from: http://www.pentest-standard.org/index.php/Threat_Modeling

[3] Rosenquist, M. Prioritizing Information Security Risks with Threat Agent Risk Assessment. [online], 2009, [Last Accessed on 21.11.2020]. Available from: https://media10.connectedsocialmedia.com/intel/10/5725/Intel_IT_Business_Value_Prioritizing_Info_Security_Risks_with_TARA.pdf

[4] Blackley, J. How Long Do People Keep Their Cars? [Last Accessed on 5.1.2020]. Available from: https://www.iseecars.com/how-long-people-keep-cars-study

[5] Murtha, S. Connected & Autonomous Vehicles Introducing the Future of Mobility. [online], [Last Accessed on 23.7.2020]. Available from: https://www.atkinsglobal.com/~/media/Files/A/Atkins-Corporate/north-america/sectors-documents/highways-and-bridges/library-docs/brochures/CAV_Report-NorthAmerica_v2.pdf

[6] Aries, K. Connected Vehicle Technology (V2V, V2I, V2X). [online], 2020, [Last Accessed on 20.7.2020]. Available from: https://www.verizonconnect.com/resources/article/connected-vehicle-technology-v2v-v2i-v2x/

[7] V2X. [Last Accessed on 24.7.2020]. Available from: http://168.131.150.78/xe/index.php?mid=V2X&document_srl=514

[8] Rumer, J.; Burnett, M. Software-Over-The-Air: An Automotive Accelerator. [online], [Last Accessed on 23.7.2020]. Available

from: https://www.bearingpoint.com/files/BEI008-07-ICL_SOTA-Software-over-the-air.pdf?download=0&itemId=473686

[9] Smith, C. *The Car Hacker's Handbook: A Guide for the Penetration Tester*. San Francisco, CA: No Starch Press, 2016, ISBN 1-59327-703-2.

[10] History of CAN technology. [online], [Last Accessed on 19.5.2020]. Available from: https://www.can-cia.org/can-knowledge/can/can-history/

[11] Corrigan, S. Introduction to the Controller Area Network. *Texas Instruments*, 2002. Available from: http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[12] Hackett, E. LIN Protocol and Physical Layer Requirements. *Texas Instruments*, 2018. Available from: http://www.ti.com/lit/an/slla383/slla383.pdf

[13] LIN Bus Explained - A Simple Intro. [online], [Last Accessed on 20.5.2020]. Available from: https://www.csselectronics.com/screen/page/lin-bus-protocol-intro-basics/language/en

[14] Liaw, D.-C.; Liu, I.-C.; et al. The FlexRay Implementation of By-Wire System for Electric Vehicle. *World Electric Vehicle Journal*, volume 2, 06 2012: pp. 1032–1038. Available from: https://www.researchgate.net/publication/290602220_The_FlexRay_Implementation_of_By-Wire_System_for_Electric_Vehicle

[15] El-Rewini, Z.; Sadatsharan, K.; et al. Cybersecurity challenges in vehicular communications. *Vehicular Communications*, volume 23, 12 2019. Available from: https://www.researchgate.net/publication/337725825_Cybersecurity_challenges_in_vehicular_communications

[16] Levi, Z. Automotive Ethernet: The Future of In-Car Networking? [online], 04 2018, [Last Accessed on 23.7.2020]. Available from: https://www.electronicdesign.com/markets/automotive/article/21806349/automotive-ethernet-the-future-of-incar-networking

[17] Maliniak, D. What's the Difference Between BroadR-Reach and 100Base-T1? [online], 05 2018, [Last Accessed on 23.7.2020]. Available from: https://www.electronicdesign.com/markets/automotive/article/21806576/whats-the-difference-between-broadrreach-and-100baset1

[18] Parsania, P.; Saradava, K. Drive-By-Wire Systems In Automobiles. 12 2012. Available from: https://www.researchgate.net/publication/287993938_Drive-By-Wire_Systems_In_Automobiles

[19] Cross Site Scripting (XSS). [online], [Last Accessed on 29.11.2020]. Available from: https://owasp.org/www-community/attacks/xss/

[20] Clickjacking. [online], [Last Accessed on 29.11.2020]. Available from: https://owasp.org/www-community/attacks/Clickjacking

[21] Greenberg, A. GM Took 5 Years to Fix a Full-Takeover Hack in Millions of OnStar Cars. [online], 9 2015, [Last Accessed on 18.7.2020]. Available from: https://www.wired.com/2015/09/gm-took-5-years-fix-full-takeover-hack-millions-onstar-cars/

[22] Greenberg, A. Hackers Reveal Nasty New Car Attacks–With Me Behind The Wheel. [online], 7 2013, [Last Accessed on 18.7.2020]. Available from: https://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video

[23] Charlie Miller, C. V. Remote Exploitation of an Unaltered Passenger Vehicle. [online], 8 2015, [Last Accessed on 18.7.2020]. Available from: http://illmatics.com/Remote%20Car%20Hacking.pdf

[24] Goodwin, A. Tesla hackers explain how they did it at Defcon. [online], 8 2015, [Last Accessed on 18.7.2020]. Available from: https://www.cnet.com/roadshow/news/tesla-hackers-explain-how-they-did-it-at-def-con-23/

[25] Hunt, T. Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs. [online], 2 2016, [Last Accessed on 18.7.2020]. Available from: https://www.troyhunt.com/controlling-vehicle-features-of-nissan/

[26] Greenberg, A. Tesla Responds to Chinese Hack With a Major Security Upgrade. [online], 9 2016, [Last Accessed on 18.7.2020]. Available from: https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/

[27] Greenberg, A. Hackers Can Steal a Tesla Model S in Seconds by Cloning Its Key Fob. [online], 9 2018, [Last Accessed on 18.7.2020]. Available from: https://www.wired.com/story/hackers-steal-tesla-model-s-seconds-key-fob/

[28] Greenberg, A. Hackers Could Steal a Tesla Model S by Cloning Its Key Fob Again. [online], 8 2019, [Last Accessed on 18.7.2020]. Available from: https://www.wired.com/story/hackers-steal-tesla-model-s-key-fob-encryption/

[29] Tidy, J. Honda's global operations hit by cyber-attack. [online], 6 2020, [Last Accessed on 18.7.2020]. Available from: https://www.bbc.com/news/technology-52982427

[30] Machala, F. Tesla Model 3 Internal Network Security Analysis. 2020.

[31] Karahasanovic, A. Automotive Cyber Security Threat modeling of the AUTOSAR standard. 2016.

[32] Spaan, R. Secure updates in automotive systems. 2016.

[33] Threat Modeling. [Last Accessed on 13.10.2020]. Available from: https://owasp.org/www-community/Threat_Modeling

[34] Schneier, B. Attack Trees. [online], 1999, [Last Accessed on 13.10.2020]. Available from: https://www.schneier.com/academic/archives/1999/12/attack_trees.html

[35] Approaches to Software Threat Modeling. [Last Accessed on 13.10.2020]. Available from: https://threatmodeler.com/approaches-to-threat-modeling/

[36] Fruhlinger, J. Threat modeling explained: A process for anticipating cyber attacks. [online], 2020, [Last Accessed on 13.10.2020]. Available from: https://www.csoonline.com/article/3537370/threat-modeling-explained-a-process-for-anticipating-cyber-attacks.html

[37] Wynn, J.; Whitmore, J.; et al. Threat Assessment & Remediation Analysis (TARA). [online], 2011, [Last Accessed on 13.10.2020]. Available from: https://www.mitre.org/sites/default/files/pdf/11_4982.pdf

[38] Casey, T. Threat Agent Library Helps Identify Information Security Risks. 09 2007, doi:10.13140/RG.2.2.30094.46406, [Last Accessed on 29.11.2020]. Available from: https://www.researchgate.net/publication/324091298_Threat_Agent_Library_Helps_Identify_Information_Security_Risks

[39] Bednář, M. Stočili jsme tachometr ojetého auta. Zabralo to jen pár minut. 6 2018, [Last Accessed on 11.12.2020]. Available from: https://www.novinky.cz/auto/clanek/stocili-jsme-tachometr-ojeteho-auta-zabralo-to-jen-par-minut-40265554

[40] Cimpanu, C. Here's a List of 29 Different Types of USB Attacks. 3 2018, [Last Accessed on 11.12.2020]. Available from: https://www.bleepingcomputer.com/news/security/heres-a-list-of-29-different-types-of-usb-attacks/

[41] Vaculík, M. Měření tlaku v pneumatikách: Zatraceně drahé bezpečí. 12 2015, [Last Accessed on 12.12.2020]. Available from: https://www.auto.cz/mereni-tlaku-v-pneumatikach-zatracene-drahe-bezpeci-91220

[42] Filesystem Hierarchy Standard. [Last Accessed on 24.9.2020]. Available from: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

[43] Zahradnický, T.; Kokeš, J. 1. Introduction to Reverse Engineering, Stack Frame Analysis. [online], 2019, [Last Accessed on 3.7.2020]. Available from: https://courses.fit.cvut.cz/MI-REV/media/lectures/rev01en.pdf

Appendix A

Acronyms

AV Autonomous Vehicle

CAN Controller Area Network

CV Connected Vehicle

DSRC dedicated short-range communication

ECU Electronic Control Unit

FHS Filesystem Hierarchy Standard

GM General Motors

GPS Global Positioning System

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IDA Interactive Disassembler

IP Internet Protocol

LIN Local Interconnect Network

LTE Long Term Evolution

MCU Media Control Unit

MOST Media Oriented Systems Transport

NFC Near Field Communication

OBD-II On-Board Diagnostics

OTA Over-the-air

OWASP Open Web Application Security Project

PC Personal computer

PHY Physical Layer

SW Software

TCP Transmission Control Protocol

UDP User Datagram Protocol

UI User Interface

USB Universal Serial Bus

V2I Vehicle-to-infrastructure

V2V Vehicle-to-vehicle

V2X Vehicle-to-everything

VAN Vehicle Area Network

VIN Vehicle identification number

XSS Cross-site scripting

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `output` ..... the directory with outputs from programs
- `src` ..... the directory of source codes
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format