CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Assessing the Security of Internal Automotive Networks

Bachelor's thesis in Computer science and engineering

Anas Alkoutli
Joakim Anderlind
Carl-Johan Björnson
Mathias Drage
Morgan Thowsen
Antonia Welzel

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Bachelor's thesis 2021

Assessing the Security of Internal Automotive Networks

Anas Alkoutli
Joakim Anderlind
Carl-Johan Björnson
Mathias Drage
Morgan Thowsen
Antonia Welzel

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Assessing the Security of Internal Automotive Networks
Anas Alkoutli Joakim Anderlind Car-Johan Björnson Mathias Drage Morgan Thowsen Antonia Welzel

Supervisor: Prof. Miroslaw Staron, Department of Computer Science and Engineering
Supervisor: Dr. Darko Durisic
Examiner: Prof. Christian Berger, Department of Computer Science and Engineering

Bachelor's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Abstract

Context In order to address the growing need for connectivity in today's cars, the in-vehicle network has increased in complexity, now consisting of over 100 electrical control units. Balancing the level of security with high performance is non-trivial, and current networks have shown to sacrifice security measures for performance, therefore leaving the networks sensitive to both manipulation and information retrieval.

Objective The first objective of this thesis was to assess the security of in-vehicle networks and identify potential security threats that may be exercised with commodity hardware and without expert knowledge in vehicular networking. Secondly, propose solutions to identified security vulnerabilities that act as a defence against the exercised attacks.

Method The project was executed with the Design Science Research methodology, where an artefact is developed and evaluated through iterations. The artefact consists of commodity hardware and open software as well as our approach to simulate an attack of an uninitiated tamperer. The applying artefact was evaluated by testing it on vehicle test beds. The evaluation was also the basis for the assessment and in extension the proposed solutions.

Results The in-vehicle network was susceptible to multiple attacks such as Man-in-the-middle and Replay attacks. For instance, multiple components in the network were successfully manipulated through replay attacks on the network. The replay attacks were conducted in combination with a minimization algorithm which enabled reverse engineering of specific functions with high precision. This made it possible to not only take full control but to block user input. Moreover, Man-in-the-middle attacks on the Ethernet traffic yielded raw data indicating a lack of encryption and also enabled us to map the network topology. To resolve the aforementioned issues, this thesis proposes solutions at varying security levels that would have prevented our attacks; such as CAN bus message counters, MACsec for low-level protection against sniffing and TLS encryption for the confidentiality of raw data.

Conclusions The project has shown that it is indeed possible to extract and manipulate data even with the limitations enforced in this project. In addition, it showed that the tested in-vehicle networks lack resilience against unauthorized access and manipulation. The proposed solutions protect against exercised attacks but are subject to future research in terms of implementation and overhead measurements.

Keywords: Vehicle, Network Security, CAN, Ethernet

Sammandrag

Kontext För att bemöta det växande behovet av uppkoppling i dagens bilar har fordonets interna nätverk ökat i komplexitet som nu består av över 100 uppkopplade styrenheter. Att balansera säkerhetsnivån med hög prestanda är inte enkelt, och nuvarande nätverk har påvisats offra säkerhet för prestanda. Detta lämnar nätverken sårbara för både manipulation och informationshämtning.

Mål Första målet i detta kandidatarbetet var att göra en bedömning av säkerheten hos fordons interna nätverk samt identifiera potentiella sårbarheter i nätverken. Bedömningen utfördes med kostnadseffektiv och allmäntillgänglig hårdvara samt utan specialistkunskap inom bilnätverk. Det andra målet var att föreslå säkerhetslösningar för funna sårbarheter.

Metod Projektet har utförts enligt Design Science Research metodiken, där en artefakt iterativt utvecklats och evalueras. Artefakten bestod av kostnadseffektiv och allmäntillgänglig hårdvara samt öppen mjukvara för att simulera en lekmannamässig attack. Evalueringen av artefakten baserades på resultatet från avlyssning och attacker på en testrigg. Detta utgjorde grunden till bedömningen av nätverkens motståndskraft gentemot sådana attacker samt förslag på lösningar.

Resultat Fordonsnätverket var mottagligt för flera attacker som Man-in-the-middle och Replay-attacker. Till exempel manipulerades flera komponenter i nätverket med framgång genom Replay-attacker på nätverket. Replay-attackerna genomfördes i kombination med en minimeringsalgoritm som möjliggjorde reverse-engineering av specifika funktioner med hög precision. Detta gjorde det möjligt att ta full kontroll och blockera användarinmatning. Dessutom gav Man-in-the-middle-attacker på Ethernet-trafiken rådata som indikerar brist på kryptering och möjliggjorde kartläggning av nätverkstopologin. Som lösning för de ovannämnda problemen föreslås skyddsmekanismer på olika säkerhetsnivåer som skulle förhindrat utförda attacker; CAN-räknare, MACsec för lågnivåskydd mot sniffing och TLS-kryptering för datasekretess.

Slutsatser Projektet har visat att extraktion och manipulation av data är möjligt trots de begränsningar som tillämpas. Dessutom påvisades det att de testade fordonsnätverken saknar motståndskraft mot obehörig åtkomst och manipulation. De föreslagna lösningarna skyddar mot utövade attacker men är föremål för framtida forskning vad gäller implementering och mätningar av overhead.

Nyckelord: Fordon, Nätverksäkerhet, CAN, Ethernet

Acknowledgements

First and foremost, we would like to thank our supervisors Prof. Miroslaw Staron and Dr. Darko Durisic whose expertise has been invaluable for guiding us throughout this project.

Thank you Anders Edvardsson and Zao Sun for tirelessly answering questions and logistically arranging the test bed. Last but not least, we would like to thank our industrial partner for making this thesis possible.

Anas Alkoutli, Joakim Anderlind, Carl-Johan Björnson, Mathias Drage, Morgan Thowsen, Antonia Welzel, Gothenburg, June 2021

List of Abbreviations

ACK	Acknowledge bit
ADAS	Advanced Driving Assistance System
ARP	Address Resolution Protocol
CAN	Controller Area Network
CRC	Cyclic Redundancy Checksum
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DDMin	Delta Debugging Algorithm
DLC	Data Length Code
DNS	Domain Name System
DoS	Denial of service
DSA	Digital Signature Algorithm
ECU	Electronic Control Unit
EOF	End of Frame
FCS	Frame Check Sequence
FTP	File Transfer Protocol
HMAC	Key-Hashed Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
ID	Identifier
IDE	Identifier extended
IFS	Interframe Space
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
LAN	Local Area Network
LIN	Local Interconnect Network
MAC	Media Access Control
MACsec	Media Access Control Security
MOST	Media Oriented Systems Transport
NFS	Network File System
OBD	On-Board Diagnostics
OSI	Open Systems Interconnection
SFD	Start Frame Delimiter
SOF	Start of Frame
SOME/IP	Scalable Service-oriented Middleware over IP
SRR	Substitute Remote Request
TCI	Tag Control Information
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol stack

TLS	Transport Layer Security
UDP	User Datagram Protocol

Contents

List of Figures

1 Introduction

The automation of the modern car has enabled new opportunities and features, and in many ways made cars more convenient and safer for the driver and passengers. This has led to an expansion in the cars' software and their internal communication system, at a cost of increasing complexity [1]. The increasing number of functions expected in a modern vehicle inevitably leads to an increasing number of sensors and devices, each with their own responsibility. In order to address the growing need for connectivity between components, the modern car has evolved into a network of over 100 independent computers called Electronic Control Units (ECUs). Each ECU controls one or more aspects of the electrical system. Together with the sensors and actuators, the ECUs communicate over the in-vehicle network to handle the majority of the functions, including brakes, lights, and engine controls. Two common in-vehicle network types are Ethernet and the Controller Area Network (CAN). Both of these networks have their strengths and weaknesses. CAN is the traditional network in vehicles and is cost-efficient, reliable, but is limited in terms of bandwidth and therefore security measures. Ethernet is a high bandwidth, general purpose protocol that will be predominantly used in vehicular networking in the future to support features such as autonomous driving [2]. With the adaptation of software-based technologies the vehicle also inherits many of the vulnerabilities common in the software industry such as Man-in-the-middle attacks. Such attacks could result in fatal consequences when directed towards critical functions. In order to minimize the threat of common attacks, the in-vehicle network has to be *resilient* against intrusion. In this report, a network is considered resilient against intrusion if it: (1) does not leak information upon breach, (2) is secured against unauthorized manipulation.

Prior assessments of resilience have proven the network to be fragile and possible to be manipulated [3]. In extension, showcasing that control of critical functions is practically possible. Studies on software-based security solutions have been conducted but have unveiled the dilemma of secure methods having too much overhead for real-time critical functionality [4, 5]. While a solution through added security hardware is feasible in terms of performance, it is expensive and unreasonable to retrofit.

Studies have been conducted analysing CAN security implementations [6, 3, 7]. For instance, Buttigieg et al. studied attacks on an individual component physically extracted from the vehicle [6]. However, the experiment was conducted in a synthetic environment where the connection to the CAN bus was simulated. In a live setting, direct access to individual components is limited. A comprehensive study had access to laboratory equipment and established the extent of the vulnerabilities [3]. Although, the exploitation of discovered vulnerabilities required total access to internal computers and stationary hardware, which is arguably unrealistic unless the car is already compromised. Moreover, CAN security measures in the form of encryption and authentication with medium to high security measures have been investigated. However, an investigation by Nowdehi et al. [4] concluded that it had too much of an overhead for a real-time system. Low security measures and synchronization-based security have been explored and utilized by manufacturers such as BMW [6]. However, the solution only applies to the debugging/service OBD-II port, leaving the internal CAN busses still vulnerable.

Due to its simplicity and versatility, Ethernet is nearly ubiquitous in modern computer networks [8]. Its widespread use and vulnerable design have resulted in a significant amount of research documenting and mitigating its vulnerabilities [8, 9]. However, Ethernet's use in automotive networks is not as well studied. Studies have been conducted on adapting existing mitigation solutions to meet the requirements of automotive Ethernet networks [10, 5, 11]. Nonetheless, there is a lack of investigatory research into vulnerabilities of automotive Ethernet networks as opposed to CAN [3, pp. 447–462]. A research project commissioned by the BMW Group [12] found vulnerabilities related to an Ethernet switch in a production car, yet the vulnerabilities were implementation specific and the report did not publicly give any recommendations for mitigating these issues.

While assessments of live vehicles have been carried out, they do not use the type of commodity hardware used by a hacker. And, while commodity hardware has been used for assessment in related work, it has exclusively been carried out on specific components extracted from a vehicle, thus not in a live setting. This paper studies vulnerabilities found with commodity hardware in a live setting and aims to discuss and propose software solutions to resolve discovered vulnerabilities. The research questions are as follows:
1. How resilient are in-vehicle networks against unauthorized access and manipulation, conducted with cost-efficient hardware and without expert knowledge?,
2. What software solutions could have prevented methods exercised in question 1?

To address these questions, automotive CAN and Ethernet have been analysed. The authors argue that substantial vulnerabilities exist with the aforementioned limitations, which enable data extraction and manipulation. For CAN, a counter based solution is proposed. While the solution is low security it has a low overhead and pose a first barrier of defence against replay attacks. In Ethernet, a list of recommendations is proposed, such as leveraging existing encryption models for application level encryption, randomized Media Access Control (MAC) addresses, and the use of existing authentication models for the Local Area Network (LAN).

The contribution of this thesis is an assessment of in-vehicle network resilience conducted without expert knowledge in vehicular networks and with commodity hardware. The thesis also proposes security measures that would have prevented the attacks exercised in the assessment.

The structure of this thesis is as follows: Related work is presented in chapter 2. Chapter 3 includes the theoretical background for context. The methodology and research design are presented in chapter 4. Results and proposed solutions are presented in chapter 5, and the discussion of the results in relation to related work is presented in chapter 6. Ethics is discussed in chapter 7, and the thesis is concluded in chapter 8.

2
Related Work

In 2010, Koscher et al. [3] conducted a comprehensive experimental security analysis on two modern automobiles. Similarly to our study, their purpose was to assess the level of resilience a conventional automobile has against digital attacks. Their attack methodology consisted of packet sniffing in combination with targeted probing, fuzzing and reverse engineering. As a result, they managed to manipulate every ECU that was tested, and therefore achieving complete control over components such as door locks and brakes. Koscher et al.'s strategies included advanced hardware and techniques that require expert knowledge about the components and electrical system in the vehicle. Our study on the other hand examines whether similar vulnerabilities are present in a production car from 10 years later restricted to more commodity hardware and limited access to the test bed.

Another experimental security analysis, but in a simulated vehicle environment, was conducted by Buttigieg et al. [6]. Their objective was to analyse the resilience of the CAN protocol specifically. By conducting a Man-in-the-middle attack with the use of commodity hardware, they managed to control the instrument cluster extracted from a BMW E90. Our study presents an alternative and less physically intrusive method for taking control over specific components by exploiting security issues in the CAN protocol. Furthermore, we present experimental results from an industrial automotive test bed that together with the results from Buttigieg et al. strengthen the need for security measures in the CAN protocol.

Lin and Sangiovanni-Vincentelli [4] proposed a software-only security mechanism for the CAN protocol to help prevent masquerade and replay attacks, which are also attack methods exercised in our study. Lin and Sangiovanni-Vincentelli focused on a security mechanism that keeps the bus utilization as low as possible to fit the CAN bus's limited bandwidth. Their solution consisted of three elements: ID tables, pair-wise secret keys between nodes, and message counters. The solution mechanism proposed by Lin and Sangiovanni-Vincentelli was used as a base for finding appropriate security solutions based on our experimental findings.

The security mechanism proposed by Lin and Sangiovanni-Vincentelli, along with nine other solutions, was evaluated by Nowdehi et al. [4] in 2017 in order to investigate the lack of adoption of suggested CAN security measures in the industry. Nowdehi et al. found that none of the examined solutions fulfilled all of the criteria set by industry experts, which highlighted the difficulty of developing security mechanisms appropriate for the industry. The measurements in cost-effectiveness, security level, and acceptable overhead provide a foundation for this thesis to reason about the viability of possible solutions.

Kiravuo et al. [8] conducted a survey of vulnerabilities present in LAN Ethernet networks. The survey states that Ethernet's prevalence in computer networks is partly due to its ease of configuration and simplicity. However, these attributes also make Ethernet networks vulnerable. Kiravuo et al. lists known vulnerabilities of Ethernet networks and presents possible mitigations of these vulnerabilities. This thesis verifies the existence of the vulnerabilities that were listed by Kiravuo et al. (*Unauthorized Join*, *Topology and Vulnerability Discovery*) in automotive Ethernet networks.

Corbett et al. [13] discuss the impact of the vulnerabilities documented by Kiravuo et al. [8] on automotive Ethernet networks. Since the safety of the vehicle depends on accurate and timely readings from its sensors, the authors decided to focus on protecting the integrity of the network. Their paper compares the attributes of the Ethernet with other automotive network standards and presents a list of security challenges and opportunities that differentiate Ethernet from the alternatives. Corbett et al. conclude with a list of recommendations to protect the automotive Ethernet network from misuse, such as fingerprinting ECUs based on physical layer attributes. This thesis provides alternative mitigation measures than presented by Corbett et al. based on our experimental evidence.

Lastly, Lang [5] studied existing security standards to find suitable ones to adapt to automotive Ethernet systems in order to mitigate the vulnerabilities documented by Kiravuo et al. [8]. Similar to Corbett et al. [13], Lang focused on protecting the integrity of Ethernet packets against modification. Using a simulated automotive network, Lang was able to find two standards (DSA and HMAC) that could protect the network's integrity while maintaining a low level of latency. The recommendations provided by Lang can be implemented in combination with the recommendations provided in our thesis.

3
Theoretical Background

In this section we describe terms and concepts that are necessary to understand the contents of this thesis.

3.1 The OSI Model

The communication layer of a vehicle or any network implement network protocols, which define the communication between hardware and software components [14]. These protocols can be organized into different layers, where each layer provides different services and is often dependent on the previous layer [14]. The International Organization for Standardization (ISO) developed the Open Systems Interconnection (OSI) model, which consists of seven layers as shown in Figure 3.1.

1. The physical layer consists of the network's physical transmission technologies, which control how signals are transmitted over the medium.
2. The data link layer establishes and terminates connections between other devices in the network. It divides binary sequences into frames and transmits them from source to destination.
3. The network layer decides the most suitable path through the network, and routes packages accordingly.
4. The transport layer divides and reassembles flows of data into packets that are transmitted through the lower layers.
5. The session layer creates and maintains communication channels between devices.
6. The presentation layer prepares data retrieved from the application layer for transmission. It defines how data is decoded, encrypted and compressed in the transmission between two devices.
7. The application layer consists of the protocols used by services and applications. Examples of such protocols are Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Domain Name System (DNS).

In order for the data to be sent to a receiver it uses a header, which contains the necessary information related to its protocol for routing the data between the different layers. Figure 3.1 shows the OSI model as well as the data and its header. While sending, the data is encapsulated at each layer and a new header is added. While receiving, each layer de-encapsulates the data and removes a header in order to move it to the next layer.

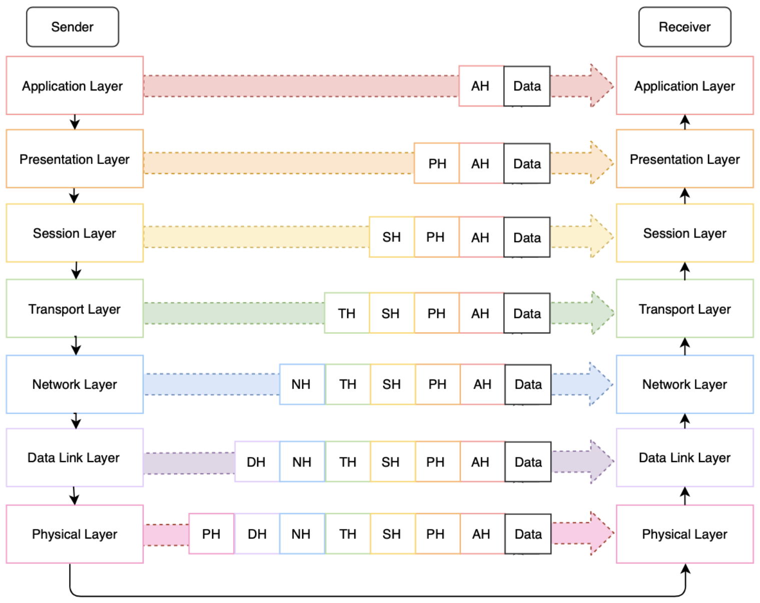


Figure 3.1: A visual representation of sending data from the sender to the receiver using the OSI model.

3.2 The In-vehicle Network

The in-vehicle network consists of interconnected ECUs. The software for the different vehicle functionalities is distributed over the ECUs [15]. Examples of network technologies that currently exist to connect the ECUs is CAN, the Local Interconnect Network (LIN), the Media Oriented Systems Transport (MOST), FlexRay, and the Ethernet [16]. Figure 3.2 illustrates an example model of the in-vehicle architecture with the different ECUs and the kinds of networks that are often used to connect them.

Figure 3.2 is a general model of what the in-vehicle network can look like, but the in-vehicle network's design can vary between different car models since they do not all follow the same standard.

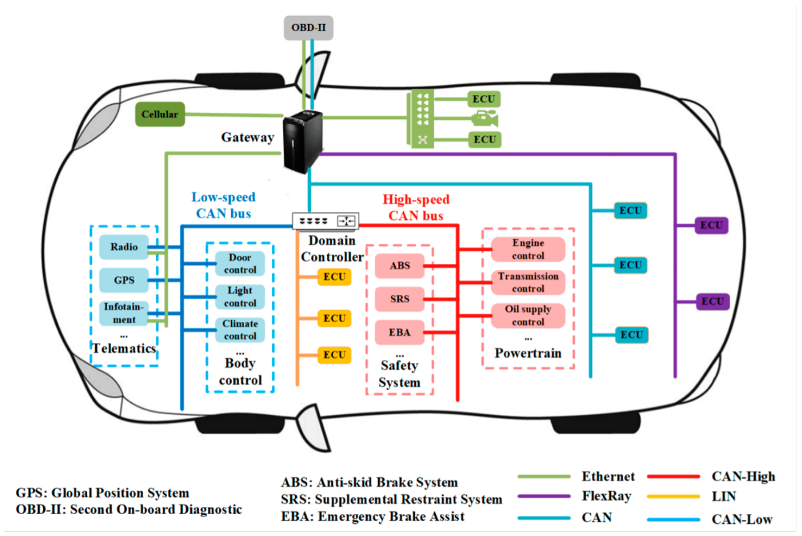


Figure 3.2: A model of the in-vehicle communication. From [17]. CC-BY.

3.3 The Controller Area Network (CAN)

CAN is a serial communications protocol used in automotives, aircrafts and industrial equipments. It is a standard that is designed to allow modules in a vehicle to communicate with each other without a host computer [18].

3.3.1 Physical Structure

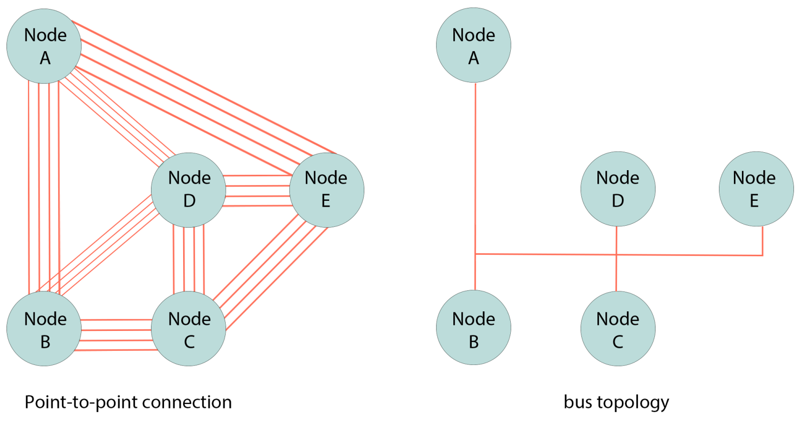


Figure 3.3: Point-to-point and Bus architecture

The CAN protocol utilizes a bus topology that allows for several modules to connect to a single bus opposed to point-to-point network topologies, see Figure 3.3 [18]. Components may request or send data or subscribe to certain messages while ignoring the rest. This greatly reduces the amount of wiring needed for all connected components compared to other network topologies [18]. However, point-to-point CAN connections are also common.

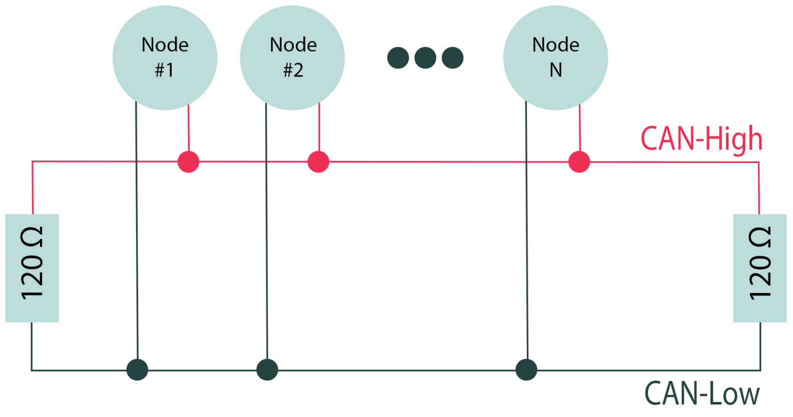


Figure 3.4: Visualization of the CAN bus topology which displays how each node is connected to both CAN-High and CAN-Low.

The components on the CAN bus in the vehicular domain are ECUs. The communication between ECUs on the bus is carried out through two CAN bus wires running the length of the entire network. The two wires are called CAN high and CAN low as seen in Figure 3.4. The connection of an ECU or Node n to the CAN bus is established by connecting it directly onto the CAN high and CAN.

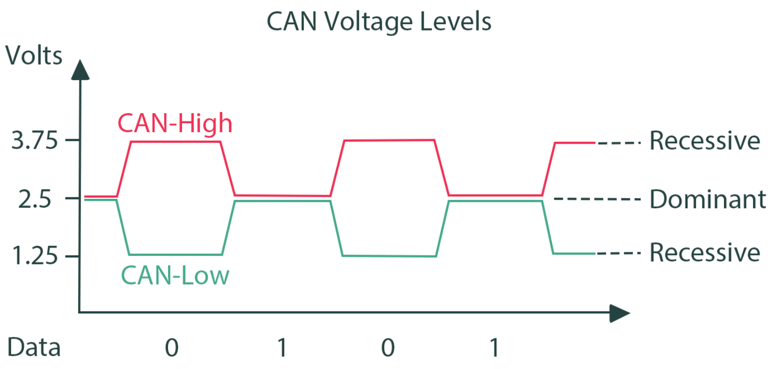


Figure 3.5: CAN high and CAN low voltage levels in data transmission.

Figure 3.5 depicts data transmission on the CAN bus. When the bus is in idle mode, both wires carry a voltage of $2.5V$ in a state called recessive which denotes a value of 1. For a value 0, or dominant state, the voltage of the CAN high line goes up to $3.75V$, and the CAN low drops to $1.25V$. The voltage difference between CAN high and CAN low is what represents the transmitted value. A power surge to the two wires preserves the difference between the two, making the bus resilient to inductive spikes and electrical fields and therefore appropriate for automotive usages [18].

3.3.2 Bus Performance

The CAN bus is used mainly for controlling various executive components in the vehicle, such as the engine control unit and the headlights. Larger transmissions such as those transmitted by a modern infotainment system are thus unsuitable for CAN [19]. Broadcasts onto the network are done in bit chunks called *frames* that have priorities as shown . The component broadcasting the highest priority frames gets immediate and uninterrupted access to the bus [18]. In the meantime, the other components can only receive data. A lower priority frame will have to be resent if blocked by a higher priority one. This makes the CAN bus a reliable option for sending urgent and high priority frames .

The CAN bus offers a bit rate of up to 1 Mbit/s at half duplex, with the most common bit rates being 125 and 500 kbit/s [18]. Since larger frames take a longer time to send, the introduction of large frames inevitably leads to higher latency. As explained earlier, the CAN bus queues the frames based on the priority of the identifier. Assuming a high priority frame is repeatedly sent onto the bus, then a lower priority frame will be blocked indefinitely. Real-time performance systems demand guarantees that the message will be delivered and on how much time the transmission will take until fulfilled. The CAN bus is not guaranteed to fulfil these terms, therefore it is unsuitable for real-time systems without clever use [20]. However, in a well-designed CAN implementation a study showed that at 30% busload, all frames may be transmitted and received within a realistic real-time application deadline [21]. Utilizing the priority inherent in CAN, a busload of up to 80% is possible while still receiving frames on time.

An analysis of the response time of the CAN bus showed that the shortest time for queuing and broadcasting a frame onto the bus is 5 ms for the messages with the highest priority identifiers. A frame with lower priority identification average 50-100 ms in response time. The CAN bus' efficiency, robustness, and cost-effectiveness make it a valid option to use in the industry, despite its real-time performance [22].

Authorization through software introduces more overhead for a given payload [4]. Several of the most promising authentication methods depend on a back and forth handshaking which inevitably increase the bus load. In order to uphold a realistic real-time application deadline, the amount of data sent on the bus would have to be cut down in proportion to the authentication overhead [21]. A benchmark from 2017 measures this authentication overhead on ten of the most promising solutions and shows that even weak authentications experience problems with scaling [4].

3.3.3 Frames

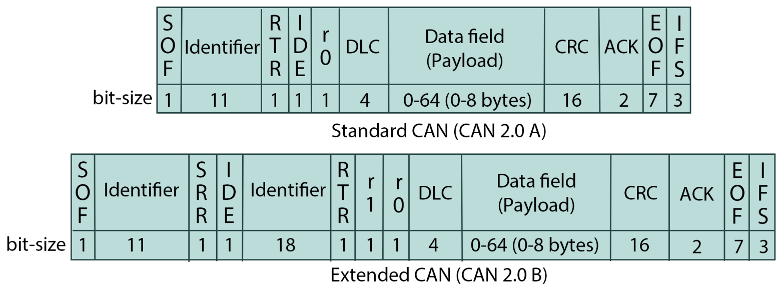


Figure 3.6: Visualization of standard- and extended CAN frame structure.

CAN frames are part of the OSI data link layer. Each frame is made up of subsections as can be seen in Figure 3.6 [18]. The standard CAN frame starts of with a SOF (Start of Frame) indicating the start of a new frame. Next is the identifier which subscribers may listen to. An identifier (ID) with a lower value has a higher priority and is therefore going to block other frames with a higher ID. The RTR indicates whether the frame is a request or message. The IDE, r0 and DLC fields together are called the control field. The IDE (Identifier Extended) indicates whether the frame is of standard or extended format, and it is first indicated after this bit has been read. DLC (Data Length Code) indicates the length of the payload.

The zero to eight bytes *payload* carry the frame data. The CRC (Cyclic redundancy checksum) is a 16 bit error detection checksum that is used to handshake whether the frame structure is intact. The ACK (Acknowledge bit) is transmitted by the client if the message has been correctly interpreted. The EOF (End of Frame) is used to detect bit stuffing. Lastly the IFS (Interframe Space) is a calculated delay such that the receiver may process the frame before the next message is sent.

One can see from Figure 3.6 that the extended CAN format differs from the standard CAN frame. For instance, the identifier is split into two fields. The SRR (Substitute Remote Request) is a placeholder to make sure that IDE is interpreted before the frame might differ. From the IDE which is now set to extended format and in further interpretation of the frame, the receiver knows that the frame is in extended CAN format. Apart from the extended identifier, SRR and a single bit r1, the extended CAN format is largely the same as standard CAN. Worth emphasizing is that the payload of the extended CAN is no larger than the standard CAN frame. The extended CAN format enables 2^{29} possible identifiers as opposed to the 2^{11} possible in the standard CAN frame but introduces more of an overhead by the two control fields as depicted in Figure 3.6.

3.3.4 Utility Software

There is software, both licensed and free, designed for working with CAN busses on a high level such as CANutils[1]. CANutils is a Linux specific set of utilities that enable high-level tools for the CAN bus such as transmitting, replaying, sniffing and dumping logs. Upon connecting to the CAN bus using CANutils, the bit rate must be specified to match the bit rate of the connected bus. Not matching the bit rate would not yield any results as CANutils would not be able to interpret frames correctly, discarding malformed frames where the CRC is violated.

3.4 Ethernet

The Ethernet is another type of communication protocol that connects the various devices in the in-vehicle network. The advantage of Ethernet is its high bandwidth, which is why it is often used in the infotainment system and the Advanced Driving Assistance System (ADAS) [23]. The infotainment system provides the driver with information and entertainment such as navigation often through an interactive panel. ADAS uses cameras and sometimes sensors to gauge its environment and therefore increases the safety and comfort while driving or parking, for example through the detection of pedestrians in the street [24].

In automotive Ethernet, the *100Base-T1*, and *1000Base-T1* Ethernet standards are used to support bandwidth speeds of 10, 100, and 1000 megabits per second respectively [25]. These standards consist of a single pair of twisted cables to transfer data between two components in the network. The single twisted pair of cables used in the *T1* standards saves about 80 % in cost and about 30 % in weight compared to the *TX* standards.

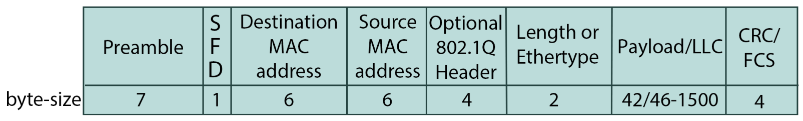
3.4.1 Network Architecture

There are different possible network architectures for Ethernet-based networks, such as CSMA/CD Ethernet [26]. CSMA/CD stands for carrier sense multiple access with collision detection and is a media access control scheme responsible for mitigating collisions in the network [27]. In the automotive context it is also referred to as a bus system, where the connection is half-duplex, since sending and receiving signals cannot happen simultaneously [26]. However, Ethernet is usually used in a switched model in order to avoid the issues that come with bus-based communication [27]. The switched model uses switches, which are forwarding devices, to control the communication between the hosts in the network [26]. The switches forward the frames to the corresponding ECUs based on their physical addresses [27]. The links in the switched network form point-to-point connections to each other, meaning there are only two units connected by the switch, as opposed to a bus system where signals are broadcasted. The links are also full-duplex, which means that signals can be sent and received simultaneously, and therefore there are no collisions.

[1]userspace utilities for Linux CAN subsystem (aka SocketCAN). www.github.com/linux-can/can-utils

3.4.2 Ethernet Frames

The data in the Ethernet network is sent in Ethernet frames, whose structure is shown below in Figure 3.7.



	Preamble	SFD	Destination MAC address	Source MAC address	Optional 802.1Q Header	Length or Ethertype	Payload/LLC	CRC/FCS
byte-size	7	1	6	6	4	2	42/46-1500	4

Figure 3.7: The Structure of a Ethernet frame.

The preamble and SFD are used to synchronize the data that is received from a host [8]. However, they are only relevant to the CSMA/CD system and not the switched network [26].

The destination and source MAC addresses are used to identify the sender and receiver, and the information is used by either the end nodes to conclude whether the packet is meant for them or the switches to forward the incoming packets [26].

The field for the MAC addresses is followed by the optional 802.1Q Header with the Tag Control Information (TCI), which contains information of the frame's priority [26]. In the next field, there is the information on the length or Ethertype of the payload [8]. Then comes the payload, which is the message that the frame is transporting and can have a minimum size of 42 or 46 bytes, depending on whether the 802.1Q Header is present [26]. Lastly, there is the checksum of the frame, also referred to as the CRC or FCS, which checks the integrity of the frame [8].

3.4.3 Protocols Related to Ethernet

Ethernet is the underlying network technology that corresponds to layer 1 and 2 in the OSI model. Therefore, it must be used in combination with a model that handles the functionality higher up in the OSI model, such as the TCP/IP protocol suite [24]. The TCP/IP model provides the functionality for layer 3 to 7 as well as the linking of the second and third layer in the OSI model [24].

One of the protocols that link layer 2 and 3 is the Address Resolution Protocol (ARP) protocol [24]. The protocol is responsible for finding the right MAC address for an IP address, which it does by broadcasting an ARP request [28].

The main protocol in the third layer is the IP (Internet Protocol), which is essential to the whole TCP/IP model since higher layer messages are dependent on this protocol, such as transport layer protocols [24]. The IP is responsible for addressing, i.e. finding the hosts in the network, and routing the messages to their destination host [24].

The protocols in layer 4 handle the transportation of information and are therefore very dependent on the IP for the data transmission [24]. The most important protocols in this layer are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP) [24]. The TCP is able to manage the traffic flow to avoid congestion and also provides relatively reliable connections, since messages that were transmitted unsuccessfully are detected and resent [24]. UDP on the other hand is one of the communication protocols that focuses on the transmission speed and is used to mitigate bandwidth issues, since it does not receive acknowledgements in the transmissions [29]. However, this also makes UDP not as reliable as other communication protocols since there is no response if messages could be sent [29].

The TCP/IP does not follow the OSI model exactly and layers 5, 6 and 7 in the OSI model amount to one top layer in the TCP/IP model, referred to as the Application layer. Some of the most known protocols in that layer are the DNS, which is responsible for matching the names of IP devices to their IP addresses, and the Network File System (NFS), which enables the transmission of files over the network [24].

In the automotive Ethernet, a middleware solution is needed in order to connect and enable data exchange between the different software components in a vehicle, such as SOME/IP [26]. SOME/IP is short for Scalable service-oriented Middleware over IP and is built upon TCP and UDP. As the name implies, SOME/IP is a service-oriented architecture where complex systems are organized into multiple loosely coupled "services" that communicate with each other using messages. A service-oriented architecture is particularly suitable for automotive use due to the distributed and specialized nature of ECUs.

3.5 On-Board Diagnostics

On-Board Diagnostics (OBD) is a diagnostics system built in a vehicle that is used for diagnosing the vehicle. Diagnostics include engine malfunctions, vehicle status, and rate of airflow.

The standard OBD-II specifies the OBD-II connector. The OBD-II connector exposes the CAN bus as part of the standard. As seen in Figure 3.8, CAN high and CAN low have pins on the connector for connecting it to the CAN bus.

the OBD-II port can be used to connect CAN loggers that can send requests over the bus. The targeted ECUs will receive the requests and send data back to the logger [30].

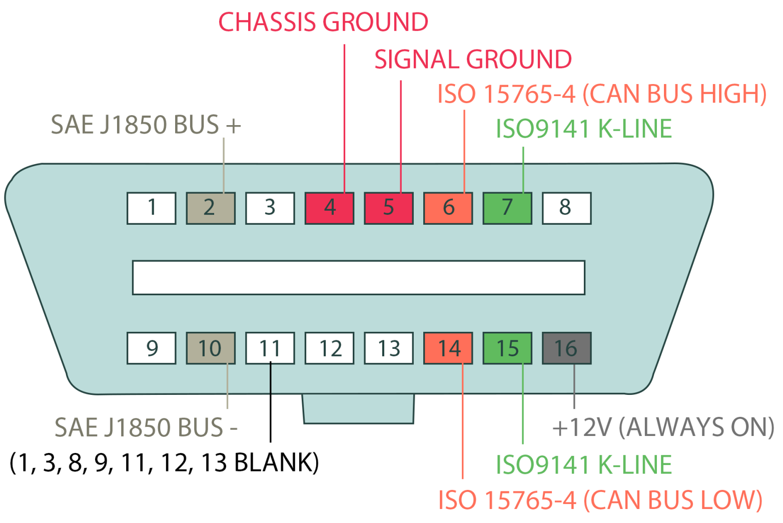


Figure 3.8: The structure of the OBD-II connector with standardized pinout.

The OBD-II port is also visible in Figure 3.2 as a physical access point to the in-vehicle network. It is considered a rather vulnerable point in a car since it is easily accessible which is a necessity for workshops and authorities in order to run and monitor vehicle diagnostics. Therefore, the port can easily provide the attacker with sensitive information [31].

3.6 The CIA Triad

Confidentiality, Integrity and Availability are all three key aspects of computer and information security, and together they form the CIA triad [32]. The CIA triad is used as an information security model to guide organizations' policies for securing data. Analysing the cybersecurity of a system in terms of these three aspects can help identify problems and by that improve the current security level.

Confidentiality refers to restricted information to only be available to authorized parties. Integrity, in terms of information security, means that data should be protected against improper modification and destruction. The Availability aspect implies that information and data should be available to authorized parties for access and modification when needed. A breach in any of the three aspects of the CIA triad compromises the security of the system.

3.7 Security Limitations of CAN and Ethernet

When the automotive industry began implementing CAN as an in-vehicle network, there was not much anticipation for the level of connectivity vehicles have today [33]. The initial intention was to connect only a small number of nodes and since the CAN bus then was not very accessible, security was not given the highest priority [33]. Subsequently, its design has led to different security issues, most of which originate from insufficient authentication and encryption [33]. One of these problems is the CAN protocol's broadcasting, where an attacker can listen to and reverse-engineer the CAN packets that are being broadcast and as a consequence also inject new packets to interfere with the network and in the end the vehicle [3]. There is also no authentication scheme to protect the bus nor is there usually any encryption to protect the data, which facilitates an attack on the bus [33]. Another limitation of CAN is therefore its susceptibility to denial of service attacks, for example by packet flooding [3].

Since CAN has a limited bandwidth, there are potential issues in how the network can be improved to solve certain security limitations. Ethernet on the other hand does not have this bandwidth limitation and can therefore more easily adapt to currently needed security measures. However, there are also multiple vulnerabilities in this in-vehicle network. For instance, a security problem in the Ethernet network is its accessibility. The communication system for Ethernet is made up of ECUs but also switches to which anyone can connect as long as there is a free port [8]. Moreover, a denial of service attack can also be performed on the Ethernet network by for example a resource exhaustion attack where frames are sent that "require additional processing and handling" [8]. Furthermore, there are also architectural issues since the security regarding the Ethernet network spans over multiple layers that trust each other, which can also be exploited by an attacker [8].

3.8 Security Threats to the In-vehicle Network

The in-vehicle network is complex and connected to many external services, which in turn can also increase the risk for different security threats, both through physical interfaces or remote attacks [34].

3.8.1 Denial of Service Attacks

One possible attack to the network is a denial of service (DoS) attack, which targets the vehicle's communication layer by disrupting the communication and rendering it unavailable [35]. For instance in the CAN bus, attackers can set the identifier segment of their messages to a low number and consequently their messages will be prioritized and block the other ones [31].

As mentioned above, DoS attacks in the Ethernet can be implemented through resource exhaustion attacks, where frames are constantly sent to exhaust the components in the network [31]. For example, Figure 3.9 illustrates a DoS attack targeting a network switch. The attacker repeatedly sends frames for the switch to log and will as a result overload it [8]. Consequently, the data traffic over this switch will either slow down or stop and block the communication to other components such as ECUs [8].

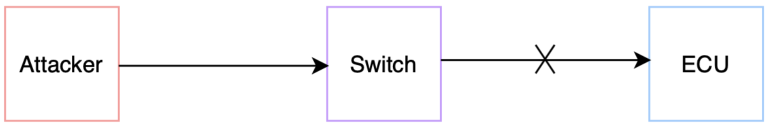


Figure 3.9: A Denial of service attack targeting a switch.

3.8.2 Man-in-the-middle Attacks

Another type of threat to the in-vehicle communication is the Man-in-the-middle attack. It involves an attacker intercepting the data traffic in the in-vehicle network and therefore being able to read the communication between different ECUs without the hosts knowing about it [36]. As a consequence, the attacker can collect sensitive information and also manipulate the data while it is being sent from one host to another [37].

The attack requires access to the network, which can be gained remotely or physically by for instance connecting to a network switch or exposed wires [38]. The attacker can then intercept the communication by exploiting the network protocols' weaknesses, for example through spoofing [38]. Figure 3.10 illustrates a Man-in-the-middle attack targeting a network switch.

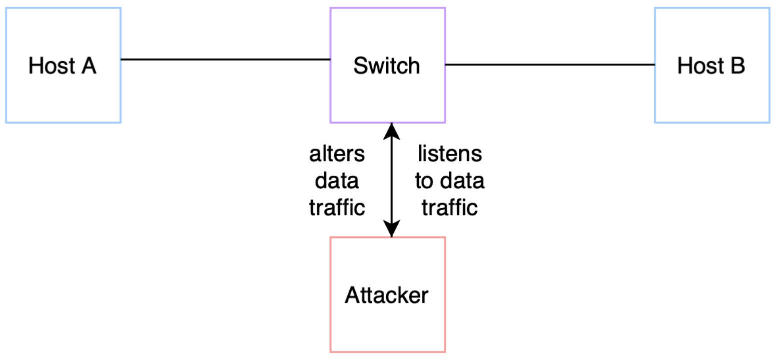


Figure 3.10: An example of a Man-in-the-middle attack. The attacker intercepts the communication from Host A to Host B and have the chance of altering the message before it reaches Host.

3.8.3 Injection Attacks

Any attack where the attacker sends fake messages through one of the entry points in the network, such as the OBD-II port or the infotainment system, is considered an injection attack [31]. In order to be able to send these fake frames when the structure of the network and therefore the messages are relatively unknown, the attacker can perform for example a Man-in-the-Middle attack or an eavesdropping attack, where they gain access to the data traffic and can then identify patterns to replicate later [31]. Examples of injection attacks are spoofing and replay attacks.

3.8.4 Replay Attacks

Another typical threat to the in-vehicle network is a replay attack, where the attacker captures signals sent over the network and then resends them without modification [7], as shown in Figure 3.11.

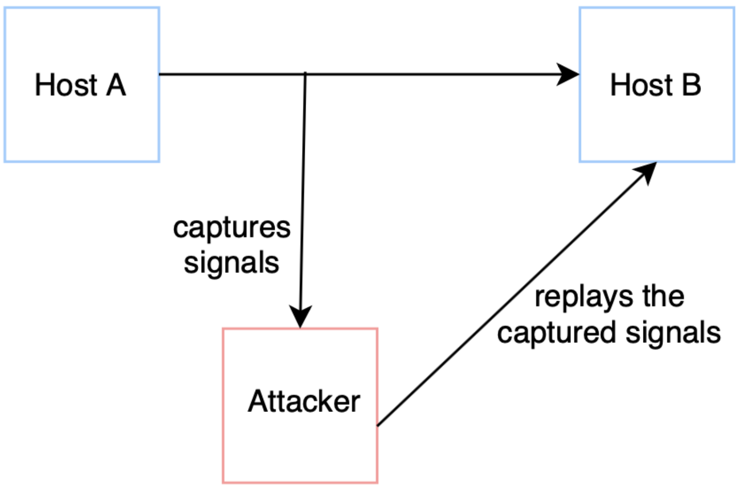


Figure 3.11: Illustration of a replay attack. Where host A's communication with Host B is captured by an attacker. The attacker may then *replay* the communication to Host B

As mentioned in the security limitations for CAN, the network has no authentication scheme which therefore enables a replay attack since the legitimacy of the source cannot be determined by the ECUs [39]. Even over the Ethernet where messages are often encrypted, replay attacks can still be implemented because the attacker does not have to be able to read the content of the message since the message is only reproduced from captured signals [8].

3.9 Security Solutions for CAN and Ethernet

The limited bandwidth of the CAN bus makes it difficult to implement security solutions such as encryption and authentication. The CAN bus has strict resource constraints and real-time requirements for critical functions that need to be fulfilled to ensure adequate performance, i.e. an acceptable overhead [4]. Two possible solutions for protecting the integrity of CAN frames while still passing these requirements are checksums and counters.

A checksum[2] represents a calculated sum of a message and is used to verify integrity of the transmitted data. The CAN frame payload is used to calculate the checksum by the sender. The checksum is then stored within the payload, called an in-message checksum. The receiver can by performing the same calculation as done by the sender, verify that the in-message checksum matches the receivers calculation in order to verify integrity. As a result, the receiver may detect and discard anomalies. Since the checksum is stored within the payload, a larger amount of frames may have to be sent to transmit the same amount of data, which in turn increases latency.

A counter on the CAN bus is used to invalidate previous frames. Both the sender and receiver have to share a synchronized counter. As with the checksum, the sender reserves part of the payload for the counter. The receiver checks whether the counter in the payload matches the receiver's internal counter and ignores the message upon mismatch. If a replay attack were to be conducted, the logged messages would be invalid upon injection due to a counter mismatch. As such, counters are an effective security measure against replay attacks.

VatiCAN is an authentication method that uses both a counter and checksum to secure the CAN bus. While it is an effective protection against common attacks, it quadruples the processing time for messages [4]. However, *VatiCAN* may still be considered to have an acceptable overhead for protecting a small number of critical messages.

Ethernet and the IP technologies built on it have by default no guarantees for confidentiality or authenticity. While this makes Ethernet adaptable and easy to configure, it also makes it vulnerable to malicious attacks. To grant these guarantees, security protocols have been developed to protect the different layers of the network.

In order to protect the network layer, the Internet Protocol Security (IPsec) was created to secure the IP protocols with encryption and a message authentication code contained in the frame [26]. Moreover, the Media Access Control Security (MACsec) was developed to prevent possible attacks on the MAC unit in the data link layer, such as ARP spoofing [26]. MACsec is implemented through point-to-point encryption at every hop, instead of end-to-end between two hosts as IPsec does, therefore providing authentication and encryption in LANs. This protects the connection between the devices from Man-in-the-middle attacks but does not protect from attacks targeting higher layers of the network stack. To protect the application layer a standard such as the Transport Layer Security (TLS) can be implemented. TLS is a cryptographic standard, frequently used by web-browsers to communicate sensitive information through untrusted networks.

[2]https://en.wikipedia.org/wiki/Checksum

4 Method

This section describes the methods used to answers our research questions. First, we describe the experiments conducted on the automotive test bed[1] followed by the methods used to analyse the results from the experiments. This section is split into CAN and Ethernet sections due to each protocol requiring different tools and techniques.

4.1 Research Design

The aim of this thesis was to assess the security of the in-vehicle network in a car with limited resources and without expert knowledge on the electrical system. Since we did not have any prior knowledge about the automotive software architecture and in-vehicle communication, it made it difficult to plan a singular experiment that would give us the information needed to answer our research questions. Subsequently, we chose to follow the *Design Science research* approach since it allowed us to progressively build knowledge about the system.

The approach is a flexible research method with the objective of gaining a better understanding of a problem through multiple iterations and finding possible solutions through the process of developing an artefact [40]. This makes it very suitable for a project with an explorative nature such as this one.

The test bed used during the project was only available at two separate occasions. This limited access to the test bed resulted in the project being formed as two iterations, where each visit contributed to the evaluation of our artefact as well as new information for developing our approach to attacking the in-vehicle security in the next iteration.

4.1.1 Problem Formulation

An important part of our research method was to define the relevant problem environment for developing a purposeful artefact that can lead to a better understanding of the issue and its solutions [40]. We initiated our research with a literature review to better understand the field of automotive networking. While our literature review consisted primarily of research papers, more applied and informal sources were used to better prepare our experiments.

[1]A semi functional automotive designed for rapid prototyping and developing

Moreover, we were able to gain more insight into the problem area by talking to experts working at our industry partner. This resulted in this project focusing on CAN and Ethernet for the assessment of the in-vehicle network security and influenced our approach for manipulating the networks. The discussions with these experts only related to our problem formulation in order to maintain a non-expert level of knowledge and not defeat the purpose of our thesis.

Lastly, the functionality of the test harness for manipulating the CAN bus was tested prior to our visit to the industry partner's office by utilizing the OBD-II port of a real car (2008 Ford Fiesta) to capture signals. This also gave us a better idea of how our artefact needs to be designed in order to ensure that it was operational.

4.1.2 Artefact

As mentioned above, the artefact in this project is made up of both the test harnesses, which consist of hardware and software and are described further in 4.3.2 and 4.4.2, as well as our general approach to making the assessment of the in-vehicle security. Our method to making the assessment of the in-vehicle network security consists of capturing signals and developed into injections by evaluating our results from the previous iteration.

4.1.3 Evaluation of the Artefact

The artefact was evaluated after each iteration in order to improve it and in the end be able to address our research questions and make an assessment of the security in the in-vehicle networks as well as to propose solutions to the found vulnerabilities. The method used for the evaluation was testing, specifically functional (black box) testing on specific components in the in-vehicle network security detailed as use cases in 4.3.1 and 4.4.1. The use cases were chosen based on the expected complexity of extracting information, and more importantly their relevance to the vehicle's security by evaluating the consequent security issues from retrieving the information and manipulating the component.

4.2 Test Bed

The test bed provided by our partner in the automotive industry consisted of an entire vehicular electronic system with all electronic components included, as seen in Figure 4.1. In other words, a stripped-down version of a real car without the mechanical parts such as chassis, engine, and wheels. This allowed for human interaction with the system which could be used for testing. The electronic system in the test rig was fully functional and it was possible to put it in active mode, which would mimic a started car. However, it was still a model that was produced in 2020 and therefore did not have all the security mechanisms that newer generations may have.

There were multiple reasons for the experiment to be conducted on a test bed instead of an actual production car. For one, testing on an actual car would be a costly process in terms of both resources and time. Moreover, whether the experiments are conducted on the test bed or a real car does not affect the outcome, since both networks are the same and the physical accessibility is not considered in the assessment of the security. Lastly, trying to manipulate the network on a live vehicle may be dangerous as it is difficult to predict the vehicle's behaviour through various manipulations.

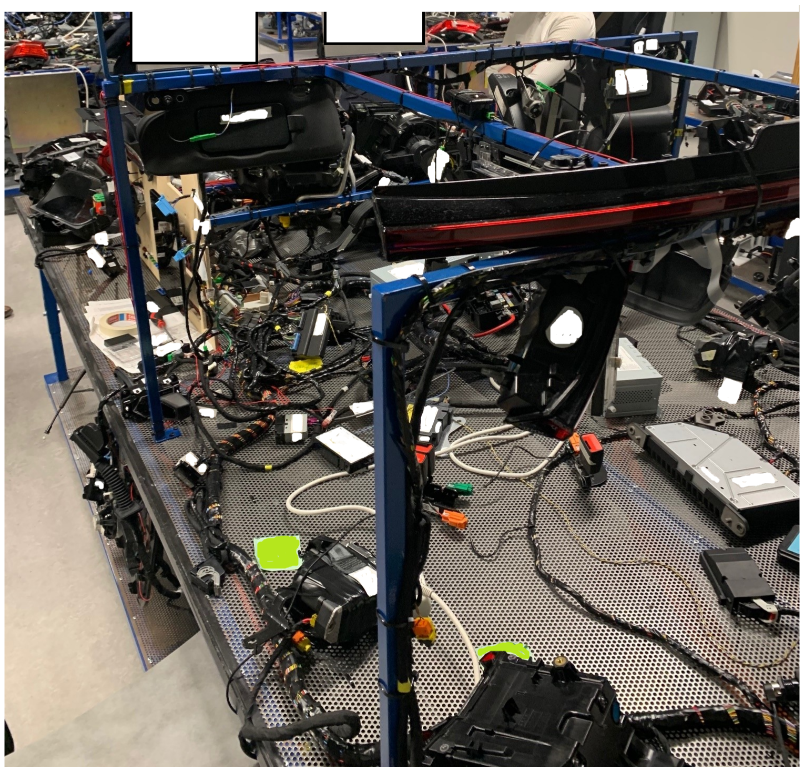


Figure 4.1: The test unit with the in-vehicle network used in this study

4.3 CAN

The approach for collecting and analyzing data related to CAN is described in 4.3.3 and 4.3.4. This information gathering relied on a ”Test Harness” (4.3.2) to record the frames being transmitted in the in-vehicle network, which was performed on the use cases explained in ”Use cases” (4.3.1). The injection based on the initial analysis of the collected data is detailed in 4.3.4.

4.3.1 Use Cases

Headlights
Controlling the headlights shows that the network is not resilient against unauthorized access and manipulation. The possible states of the headlights are small, for example, off, low-beam and high-beam. This eased the search process for identifying corresponding CAN frames since the payload should only contain a small set

of discrete values. Furthermore, headlights are an exterior component with easily accessible wiring making it a vulnerable entry point to the CAN bus. Control of the headlights is a proof of concept in the assessment of the CAN bus as a whole. This does not imply that the same vulnerabilities may be exploited for functions such as gas and brake. However, control of the headlights shows that the network does not provide a layer of security itself. Unauthorized manipulation was conducted by replay attacks as well as further reverse engineering.

Door Locks
Door locks are components that are expected to be secure due to the potential consequences of a vulnerability. While manipulation of headlights is a proof of concept, control of the door locks would imply low resilience even in security-critical components. The outcome of this together with the aforementioned use case will provide enough insight for the thesis to reason about the *level* of resilience of the CAN bus. Replay attacks were performed in order to assess the level of resilience.

4.3.2 Test Harness

The hardware harness was developed by studying similar research papers and their respective capture device. We also used official documents on the network standards and personal blogs describing similar experiments with CAN to decide what hardware and software to use for our experiments as well as how to develop it.

In order to interpret the binary data as readable CAN frames, this thesis utilized a Raspberry PI 4B running the Linux-based general-purpose operating system Raspian Pi OS. Sniffing the CAN bus was facilitated with the use of the Raspberry PI exposed IO pins to which interface cards are connected. The interface card was a Raspberry PiCAN2 which acts as a CAN transceiver. This card required no particular modification, which made it a suitable choice. The test harness can be seen in Figure 4.2.

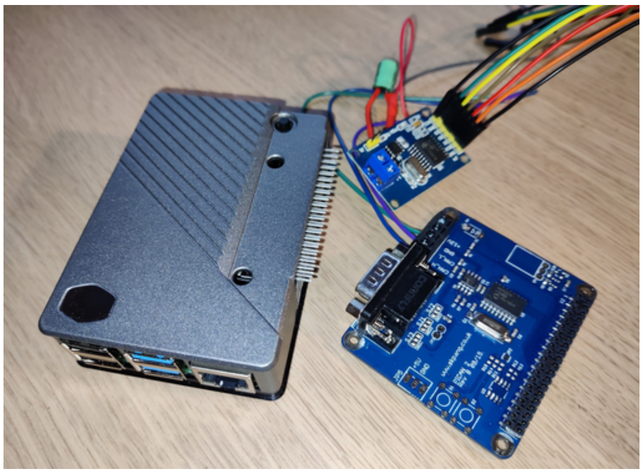


Figure 4.2: Raspberry PI 4B to the left. PiCAN2 to the right and MCP2515/TJA1050 at the top (not used).

Since manufacturers have experimented with encryption of the OBD-II port [6], we prepared a set of raw wires for connecting the device to the internal CAN, bypassing the OBD-II.

A goal for the thesis was to conduct the assessment with commodity and cost-effective hardware. The hardware harness was assembled with only commodity parts and the total cost amounts to $170. The harness supports injection and real-time sniffing and is cost-effective for the task in relation to industrial alternatives.

For the software in the test harness, the project utilized the CAN-Utils, which is a free and open source set of userspace tools and utilities for the Linux CAN subsystem socketcan[2]. The tools *CANdump*, *CANplay* and *CANsniffer* were used for capturing and analyzing CAN frames. CANdump is a tool for dumping data in real-time from a CAN interface. It also supports ASCII sign interpretation of the payload. CANplay transmits a frame or set of frames from a log file to the network. CANsniffer only displays CAN frames that have been changed within a given delta time frame, highlighting the changed bytes. Thus, CANsniffer simplifies real-time analysis by filtering *noise*[3].

4.3.3 Data Collection and Injection

A multimeter was used to identify CAN bus wires among the kilometres of wires present in the test bed. While oscilloscopes are well suited for this task, a multimeter is portable and emphasizes the uninitiated approach.
When a physical connection was established, the right bit rate was found by exhaustively testing the most common bit rates 125- and 500 kbit/s.

For collecting data, the CANutils application CANdump was utilized which captures all traffic on the bus. The captured traffic is then saved to a log file which can be reinjected onto the network with the application CANplay. Manually interacting with the use case specific component, while capturing the data traffic to a log file, may capture an executive frame or set of frames. In order to verify that executive frames were captured, a replay attack with the log file and CANplay was conducted. If the vehicle during the replay attack exhibited behaviour corresponding to the manual interaction then executive frames had indeed been captured.

For each use case component, a 3-4 second interaction was captured. In addition, a 2-minute log file was recorded without any manual interaction.

4.3.4 Analysis

The goal of the analysis was to identify which CAN frames corresponded to a functionality in the use case components, see section 4.3.1. In extension, this frame could be injected to perform an unauthorized manipulation of the system. Successfully

[2]SocketCAN. www.kernel.org/doc/Documentation/networking/can.txt
[3]data when the car is idle

conducting all Data Collection and Injection steps of a use case function implies that the function was susceptible to replay attacks (see section 4.3.3).

As described in the previous section, the analysis started with two collections of data A and B. A where the executive frames have been verified as present and B where they are not present, i.e. the large noise log (2-minute recording with no manual interaction). Since B did not contain the executive frames, all frames present in B were removed from A. The operation resulted in A being reduced in size while still containing the executive frames. During the test harness development, this method halved the number of frames when tested on a 2008 Ford Fiesta.

With hundreds or even thousands of frames sent each second a large amount of noise was still present in A. In order to further minimize the number of frames, an algorithm reminiscent of the Delta Debugging Algorithm (DDMin)[4] was used, as seen in Figure 4.3. The goal of using this algorithm was to find a minimal interval of frames that, when injected, still preserved the functionality. Each step as indicated by an arrow to a new subset was tested with a replay attack in order to check whether executive frames were still present. Red boxes represent that the executive frames were lost, green represents that the executive frames were present. In each step the interval is cut in half. If neither of the halves contains the executive behaviour, the splits were increased in size. When no further minimization was possible, a minimal interval of frames were found. However, redundant frames were still present within the interval.

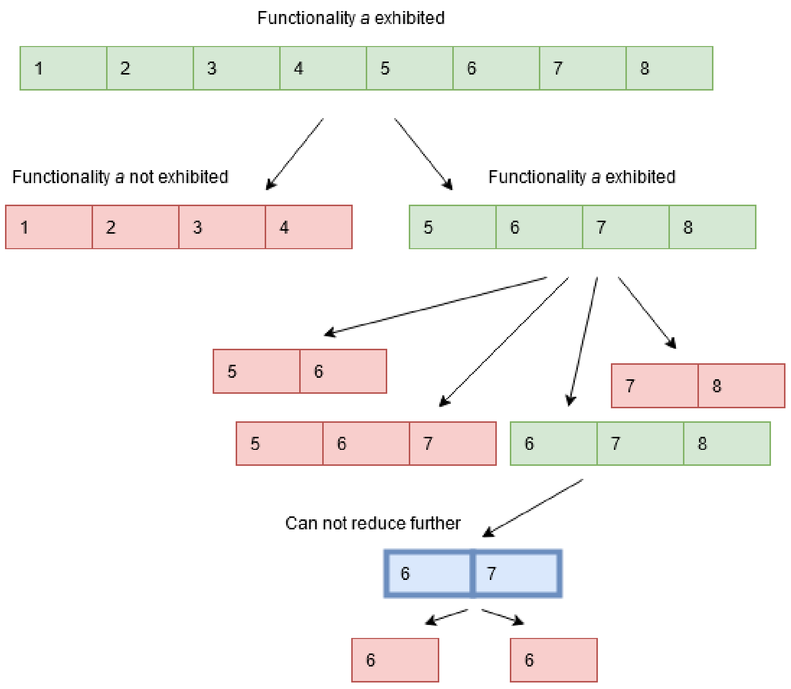
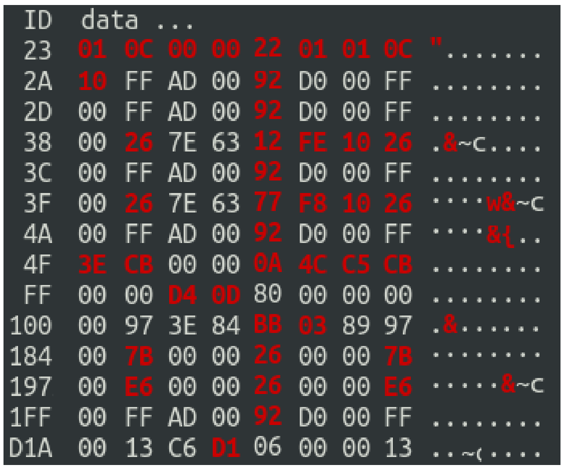

Functionality a exhibited
1 2 3 4 5 6 7 8
Functionality a not exhibited
1 2 3 4
Functionality a exhibited
5 6 7 8
5 6
7 8
5 6 7
6 7 8
Can not reduce further
6 7
6
6

Figure 4.3: Delta debugging minimization (DDMin) algorithm for finding function a in a log file interval.

[4]www.wikipedia.org/wiki/Delta_debugging

To remove redundant frames, the CANsniffer application was used. CANsniffer displayed frames present on the CAN bus where the data recently changed, as seen in Figure 4.4. Since the remaining unique frames from Delta Debugging Minimization were few, each frame was analyzed in CANsniffer while manually interacting with the use case function. Executive frames were found by studying the frames in CANsniffer which only upon interaction matched a frame in A. After this step, exact executive frames needed for manipulating the function were found. The complete flow of all steps are visualised in Figure 4.5.



```
 ID  data ...
 23  01 0C 00 00 22 01 01 0C ".......
 2A  10 FF AD 00 92 D0 00 FF ........
 2D  00 FF AD 00 92 D0 00 FF ........
 38  00 26 7E 63 12 FE 10 26 .&~c....
 3C  00 FF AD 00 92 D0 00 FF ........
 3F  00 26 7E 63 77 F8 10 26 ....w&~c
 4A  00 FF AD 00 92 D0 00 FF ....&{..
 4F  3E CB 00 00 0A 4C C5 CB ........
 FF  00 00 D4 0D 80 00 00 00 ........
100  00 97 3E 84 BB 03 89 97 .&......
184  00 7B 00 00 26 00 00 7B ........
197  00 E6 00 00 26 00 00 E6 .....&~c
1FF  00 FF AD 00 92 D0 00 FF ........
D1A  00 13 C6 D1 06 00 00 13 ..~(....
```

Figure 4.4: CANsniffer live analysis. Red text indicates a change of value while white indicates that it is static

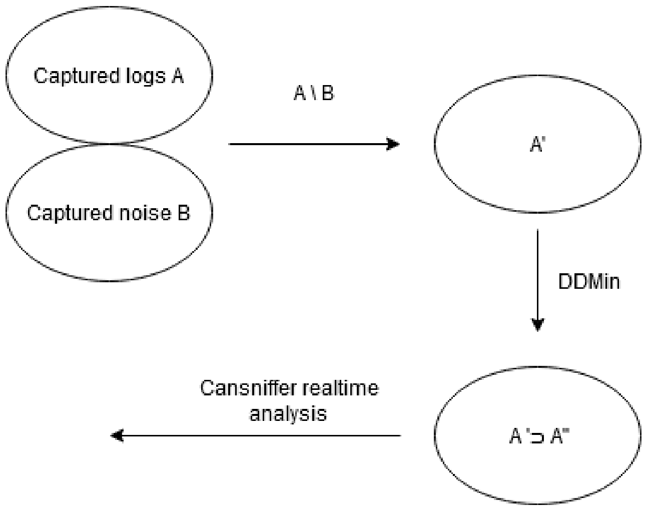


Figure 4.5: Reduction flow of captured CAN frames. Arrows represent the steps of CAN analysis.

With a set of only executive frames, each frame was injected individually into the CAN bus to determine each frame's responsibility. To further analyse a given frame, bitflips in the payload were performed and the frame reinjected. Single bitflipped frames reinjected with preserved function implied absence of an in-message checksum. In conjunction with the limited payload size of CAN frames, an absence of in-message checksum enabled exhaustive search for responsible bits within the payload.

The complete analysis was conducted by hand with the use of text editors supporting regular expressions and CANutils.

The OBD-II port exposed the internal CAN network as a part of the OBD-II standard but presented the internal CAN busses in a substantially different form and thus its data was not analysed.

4.4 Ethernet

Due to the nature of the IP protocol [41], data sent by devices is encoded by multiple layers of network headers. Our approach consisted of using our ”Test Harness” (4.4.2) to monitor and log traffic, and then perform our ”Use cases” (4.4.1), where we manipulate specific components in a predefined and well documented series of steps. All data gathered with our test harness is then put through our analysis process (4.4.4).

4.4.1 Use Cases

Control of the Infotainment System
The infotainment system in a vehicle is able to store different kinds of personal information through for example logs from connected mobile devices such as the contact list. Moreover, different vehicle functions are controlled via the infotainment system, such as the air-conditioning unit, settings of sensors and also the interface through which the user accesses the ADAS. An attack on the infotainment system would therefore be a serious threat regarding both safety and privacy.

The specific features handled by the infotainment system that we focused on were the air-conditioning unit, cameras used in the ADAS and the media player. The attack we conducted to take control of the infotainment system was a Man-in-the-middle attack. This enabled data collection without interfering with the connection in this point-to-point network.

4.4.2 Test Harness

The hardware used to capture signals was a capture module and an OBD-II to RJ45[5] Ethernet port adapter. The capture module was a *CM Ethernet Combo* from Technica Engineering, as seen in Figure 4.6, which is a device used for recording

[5]RJ45 is a type of cable connector used for Ethernet networks. The laptop used for storing the data in this project uses RJ45.

Ethernet traffic through *wiretapping*[6] between two devices. Since Ethernet networks are point-to-point connections, any interference would cause the communication to break. Moreover, it was crucial that the module was compatible with the 100-BASE-T1 Ethernet standard present in our test unit. This capture module was chosen since we find it to be a cost-efficient solution that is available to anyone. With the price at $1 200 at the time of writing, we conclude it to be a small investment considering the opportunity to manipulate vehicles worth more than tens of thousands of dollars.

Figure 4.6: Technica CM Ethernet Combo

The capture module was connected to the network by removing a cable from one of the ports on the ECU responsible for the infotainment system and connecting it to the capture module instead, and then using another cable to connect the module to the ECU's transmitting port. The data traffic was then recorded using the network protocol analyser tool *Wireshark* [7].

In order to discover any Ethernet interface exposed via the OBD-II port, a connector was needed to verify this. A custom OBD-II adapter was constructed by soldering an Ethernet cable to a male OBD-II connector, as seen in Figure 4.7.

[6]An act of using a listening device to listen to traffic over a connection
[7]www.wireshark.org

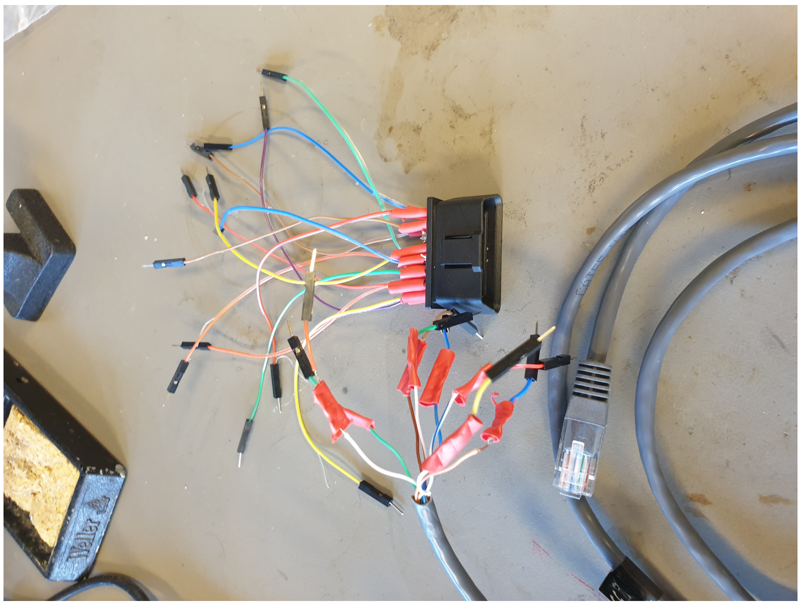


Figure 4.7: Soldered OBD-II and RJ45 connectors with breakout pins constructed.

4.4.3 Data Collection

The data from the test unit's Ethernet was collected through the capture module as well as the OBD-II port.

After the hardware was connected to the test bed's network, the network flow was recorded both without any user input as well as while testing the specific functions in the infotainment system Most recordings were done in time intervals of approximately 10 seconds, except for the ones regarding the media player which required more time in order to capture useful data. The steps for recording the data traffic for the three features are detailed below.

Air-conditioning unit

- Switching the air-conditioning between min/max at the interval of 1 second.
- Changing the air-conditioning temperature at the interval of 1 second.

Cameras used in the ADAS

- Turn on one of the vehicle cameras.
- Switching between the different Vehicle cameras at the interval of 1 second.

Media player

- Pair a phone over the infotainment system, and play music.
- Activating different functions over a longer period of time.

For the OBD-II port, data was gathered with the open source network scanning tool *nmap*[8] to try gather information about the systems running on the network.

After the data collection, all the data was stored locally on the laptop in use. Due to the potential sensitivity of the data, the data was encrypted before sharing with the thesis group for analysis.

4.4.4 Analysis

The main goals of the analysis of the Ethernet communication were to chart the network topology, document frequently used communication patterns between devices, and discover any trivially accessible payload data. Charting the network topology is a crucial step in understanding an unfamiliar network. This was done by creating a graph with network devices as vertices and inter-device communication as directed edges, where the direction of the edges was dictated by the flow of data, disregarding the two-way handshakes used by certain protocols (*ex TCP*). The capture module records network traffic in the form of *PCAPNG* [9] files and *Wireshark* was used to view and analyse them.

The identification of devices in the network was accomplished by filtering out unique IP addresses and MAC addresses from the Ethernet traffic dump. MAC addresses are allocated by IEEE [10], which we were able to cross-reference to identify the manufacturers of certain components. By connecting a manufacturer to an otherwise opaque device gave us insight into what possible use the device has and could even help decode the data received.

Identifying trivially accessible data was conducted by selecting possible packets based on packet size, and then analysing the packets by converting the binary data to ASCII lettering. Using this approach we were able to discover any payload sent on the network in plain text.

4.4.5 Injection Vulnerability

While the previous stages answer the ”unauthorized access” part of research question (1), this stage investigates unauthorized manipulation of the network.
We decided to test the Ethernet network's vulnerability to injection attacks on a component identified in the analysis (4.4.4) to prove that unauthorized manipulation of the network is possible. These tests were conducted through the network injection tool available in the capture module.
Two tests for manipulating the network were drafted:

1. Sending an ICMP echo package (”ping”) to one of the discovered devices. If the device sends a response then we have proven that manipulation of the network is possible.

[8]https://nmap.org/
[9]File format used for logging network communication (https://pcapng.github.io/pcapng/draft-tuexen-opsawg-pcapng.html)
[10]https://standards.ieee.org/faqs/regauth.html11

2. Send a SOME/IP request to a discovered SOME/IP service. If the service acknowledges the subscription, then we have successfully manipulated not only the network but a service in the network.

Test (1) was conducted with the GNU *ping* [11] tool and test (2) was conducted with a package injector written in Haskell.

[11]Ping is a network diagnostic tool and is part of the iputils package. www.github.com/iputils/iputils

5
Results

In this section, we document the results from the data capturing and data analysis. According to the ISO/OSI model (see Figure 3.1 in section 3.1), data from different layers was collected from the in-vehicle networks. For CAN the data was captured from the data link layer, while for Ethernet data was obtained from both the transport layer in the form of UDP/TCP packets and the application layer in the form of JSON objects. An overview of the attacks and their outcome is shown in Table 5.1.

Attacks on the in-vehicle networks			
Attack	Ethernet	CAN	
	Infotainment	Headlights	Door locks
Sniffing	✓	✓	✓
Injection	X	✓	-
Replay	-	✓	X

Table 5.1: The results of the conducted attacks for the use cases for both CAN and Ethernet. The green checkmark indicates that the attack was successful, the red X indicates that the attack was unsuccessful, and the hyphen indicates that the attack was not performed.

5.1 CAN

The results related to CAN are organised in the following way: First, we present the results from establishing a connection to the CAN bus. Secondly, results associated with the specific use cases are presented. Lastly, we provide recommendations on how to increase security based on the outcome of our experiments.

5.1.1 Establishing Connection

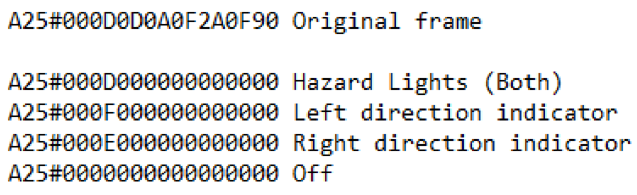
CAN bus wiring was identified by tracking cables from a headlight which only had two wires operating on the CAN standardised $2.5V$. With the cables identified and with the use of inexpensive cable tap products, the connection was established without interfering or breaking the original one. A successful connection was made evident by attempting to sniff data with CANdump in which no data would be shown if there was a connection fault. This was due to CANdump being set up to not display frames not validated by the CAN frame checksum. A connection fault could be either the wires being set up wrong or a mismatch in bit rate between the CAN bus and capture device. The correct bit rate was found by testing the most common bit rates 125kbit/s and 500kbit/s, see section 3.3.3. A connection to the internal CAN bus was established within the first hour of experimenting with the test bed without previous knowledge of the vehicle model. Therefore, CAN cables should be better protected from physical intrusion.

5.1.2 Use Case Headlights

Connected to the same CAN bus as the headlights, the hazard lights were turned on while capturing frames with CANdump. The log file was used in a replay attack while the hazard lights were turned off. This resulted in the test bed flashing the front hazard lights which implied a replay attack vulnerability. Since only the front hazard lights flashed, it led us to believe that a second CAN bus was present. This hypothesis was confirmed by conducting the same replay attack while connected to a CAN bus adjacent to the rear lights, which flashed the rear hazard lights but did not affect the front hazard lights. This is positive in terms of security since a malicious connection to a single CAN bus only compromises a subset of the vehicle's communication and functionality. However, a total mapping of which functionality is present on the different busses would be needed to assess the extent of this security measure.

The replay attack vulnerability in the headlights showed that it was possible to manipulate CAN connected components through the network. This means that the in-vehicle network was compromised. While an arguably simple manipulation of the hazard lights was possible, the complex sequences needed for stealing the car are expected to be substantially harder or even impossible.

With the log file used in the successful replay attack of the front hazard lights, further analysis as described in section 4.3.4 ensued. The analysis was successful, resulting in a complete reverse engineering of the front direction indicators as seen in Figure 5.1.



```
A25#000D0D0A0F2A0F90 Original frame

A25#000D000000000000 Hazard Lights (Both)
A25#000F000000000000 Left direction indicator
A25#000E000000000000 Right direction indicator
A25#0000000000000000 Off
```

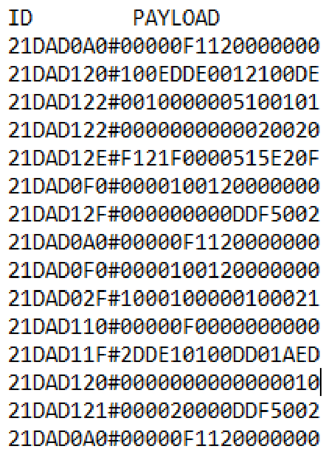
Figure 5.1: Finding nibble responsible for direction indicators. *A*25 is the ID and the hexadecimal characters following represents the payload.

The top frame in Figure 5.1 represents the original CAN frame responsible for flashing the hazard lights. By iterative bitflips on the payload of the original frame and reinjection, a single hexadecimal character responsible for controlling the direction indicators was found. The result is visualised in the remaining rows of the figure.

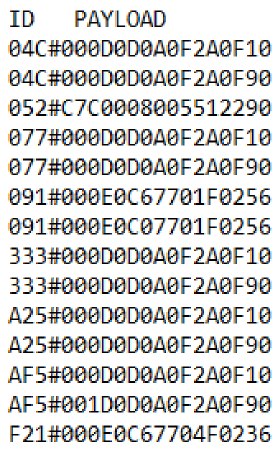
As described in section 4.3.4, this bitwise reverse engineering is made possible when the integrity of the frames is not protected with in-message checksums. The result is that it allows an uninitiated to reach a low-level understanding of the system. In our case, among the thousands CAN frames captured, we could determine that the fourth hexadecimal character in the frame with ID A25 was responsible for the direction indicators. The same complete procedure was successfully executed for flashing the high beams independently. To ensure that the success of our method was not model specific, flashing of headlights was performed on a second vehicle model with the same outcome.

With the obtained knowledge about the frame responsible for the direction indicators, more precise control of the headlights was achieved. The frame denoted "Off" in Figure 5.1 was then injected repeatedly with the result that user interaction with front direction indicators was blocked. Blocking of user input was attempted with the original large log file with the result that functions corresponding to the other frames were affected and a bus overload ensued. A high precision payload such as the "Off" frame enables an efficient DoS attack with no risk of bus overload.

The presence of a second bus led us to investigate whether a replay attack affecting both rear and front hazard lights was possible from the OBD-II port. The OBD-II port operated on the same bit rate as the internal busses. However, upon sniffing it was made evident that the OBD-II functioned differently as it was utilizing the extended CAN format made evident by the longer ID as seen in Figure 5.2.



```
ID         PAYLOAD
21DAD0A0#00000F1120000000
21DAD120#100EDDE0012100DE
21DAD122#0010000005100101
21DAD122#0000000000020020
21DAD12E#F121F0000515E20F
21DAD0F0#0000100120000000
21DAD12F#000000000DDF5002
21DAD0A0#00000F1120000000
21DAD0F0#0000100120000000
21DAD02F#1000100000100021
21DAD110#00000F0000000000
21DAD11F#2DDE10100DD01AED
21DAD120#0000000000000010
21DAD121#000020000DDF5002
21DAD0A0#00000F1120000000
```



```
ID   PAYLOAD
04C#000D0D0A0F2A0F10
04C#000D0D0A0F2A0F90
052#C7C0008005512290
077#000D0D0A0F2A0F10
077#000D0D0A0F2A0F90
091#000E0C67701F0256
091#000E0C07701F0256
333#000D0D0A0F2A0F10
333#000D0D0A0F2A0F90
A25#000D0D0A0F2A0F10
A25#000D0D0A0F2A0F90
AF5#000D0D0A0F2A0F10
AF5#001D0D0A0F2A0F90
F21#000E0C67704F0236
```

Figure 5.2: Extended CAN frames captured from OBD-II (left) compared to standard CAN frames captured from internal CAN bus (right). Notice the longer ID section of each frame.

Presuming that the two detected CAN busses both operated at above half capacity, it would be impossible for the sum of the two networks traffics to fit the CAN bus interface exposed by the OBD-II port without a doubled bit rate. In fact, the internal CAN busses would have to operate on less than half of their capacity since the extended-CAN format itself increases the overhead. Since access to all ECUs is a prerequisite for diagnostics, it leads us to believe that either the OBD-II is filtered to fit the bandwidth limit or operates differently. Attempting a replay attack as conducted on the internal CAN bus yielded no results. This implies that none of the executive frames for hazard lights were captured. This could mean that the OBD-II port operates on a request-basis, and all frames captured were response- or status messages, not executive frames. If no executive frames are sent while logging, then replay attacks are impossible to conduct. The reduced number of frames present on a request based interface could enable the use of security in the form of authentication and counters since the bus load is low when idle.

5.1.3 Use Case Door Locks

Connected to the same CAN bus as the driver side door panel, the door locks were toggled while frames were captured with CANdump. However, replay attacks showed no results on the door lock mechanism. Neither activation nor deactivation was possible. With no successful replay log, further data analysis was not possible. This implied that the network provided varying levels of resilience depending on the component, where critical functions were better protected. The strategy employed for high precision reverse engineering on headlight functions was accounted for. In order for replay attacks to have no effect, old messages have to be invalidated. A prolific protection against replay attacks on the CAN bus is a counter mechanism (see section 3.9), but other authentication methods may very well be present. Since no executive frames could be found, the existence of checksums is unknown.

5.1.4 Overall Recommendations

Security enhancements for the CAN communications bus are complex due to the limitations present such as bandwidth and timing. To reiterate, in order to provide a layer of security against replay attacks as conducted in this thesis, old messages have to be invalidated. This theoretically blocks replay attacks since recordings will not contain any valid executive frames.

Counters on the CAN bus are used in the industry today. Practically, counters are of finite size, frames once valid and then invalidated are bound to be valid again in the future. The time until a frame is valid again depends on the size of the counter as well as the update frequency.

Due to the previously mentioned limitations of the CAN bus, we propose a low-frequency global counter with no checksum attached. This will render replay attacks non-deterministic which complicate reverse engineering. The frequency and size of the counter may be adapted to suit computational limitations of the ECUs. Ideally, only a single ECU transmits the counter onto the bus at set intervals. Each executive frame would reserve space in the payload for the counter value at the time of transmission. The receiver compares the payload counter with the global counter, only accepting an acceptable delta between the global counter and the executive frame.

In summary, the proposed solution is a first barrier of defence and would not provide theoretical security.

5.2 Ethernet

In this section, we document the results from our analysis (4.4.4). First, we discuss the nature of the data traffic we were able to record with the capture module. Secondly, we list key results from our analysis that were used to identify specific components and sub-systems in the network. Thirdly, we present the extents we were able to determine the network topology. Our results below relate to the use case "Control of the infotainment system". Finally, we propose recommendations for solutions that could have prevented the methods used for our findings.

5.2.1 Nature of Captured Traffic

By performing a Man-in-the-middle attack, we were able to log the traffic sent and received from the infotainment ECU, as well as log its connections to other components.

Analysing the recorded network traffic in Wireshark, we could determine the use of multiple protocols, including TCP, UDP, DNS requests, and SOME/IP. We could also infer that the ECU is connected to some sort of network switch.
We were also able to determine that the majority of the recorded network communications followed patterns that were repeated at fixed intervals.

5.2.2 Identifying Components

It was possible to read the IP headers from all captured packages. Additionally, it was possible to read the headers in TCP and UDP packages. The source, destination IP addresses, and ports in these headers were used to analyse the flow of information between devices and the topology of the network. From the captured Ethernet frames we were also able to read the source and destination MAC addresses. These were cross-referenced as specified in (4.4.4). While not all addresses were registered with IEEE, the devices whose addresses were registered could be connected to known manufacturers.

By scanning the test bed's OBD-II port with the network scanner *Nmap*, we were able to extract information about which version of Android is running on an Android device on the network.

5.2.3 Charting the Network Topology

After understanding the nature of the captured data, we were able to draw a map of the identified devices in the network and reconstruct how these different parties communicate with each other. By listing the IP addresses from the recorded packets, a graph of the network topology could be constructed. The packet flow from the data recordings indicated two different sub-systems, both providing different functionality to the network since the intercepted packets showed no direct communication between these two sub-systems. The two sub-systems will be referred to as network 1 and network 2 [1].

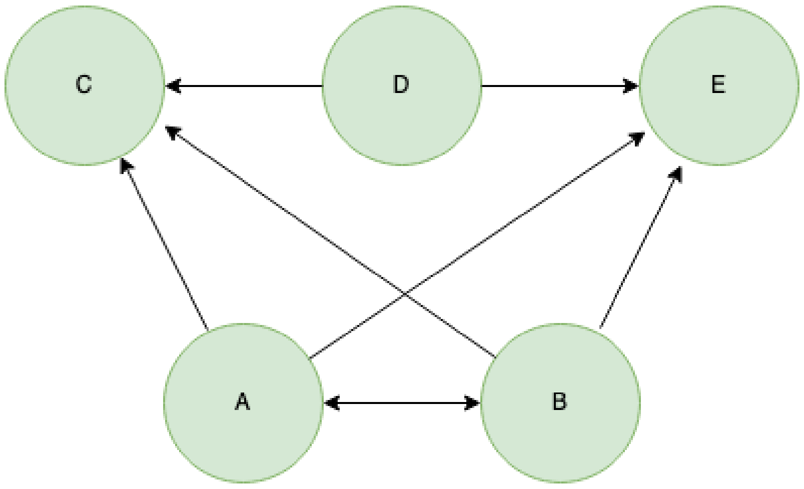
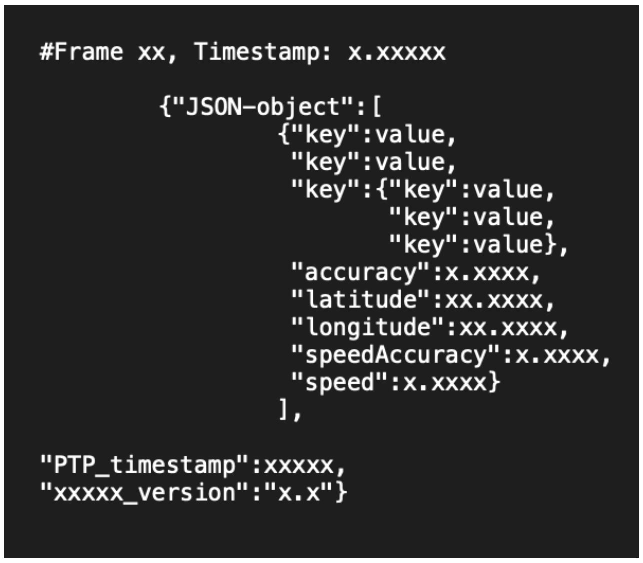


Figure 5.3: A graph showing the mapped components from network 1 with arrows showing flow of data.

Network 1 shown in Figure 5.3 is closely linked to the vehicle telemetry. This was concluded as some intercepted packets contained clear-text telemetry information. This includes information like GPS coordinates and elevation (see Figure 5.4 and 5.5). We were able to validate this information based on the coordinates of the test bed.

[1]The description of network 1, and network 2 are generalised to abstractions of the real networks. This is due to security reasons and to prevent presenting confidential information

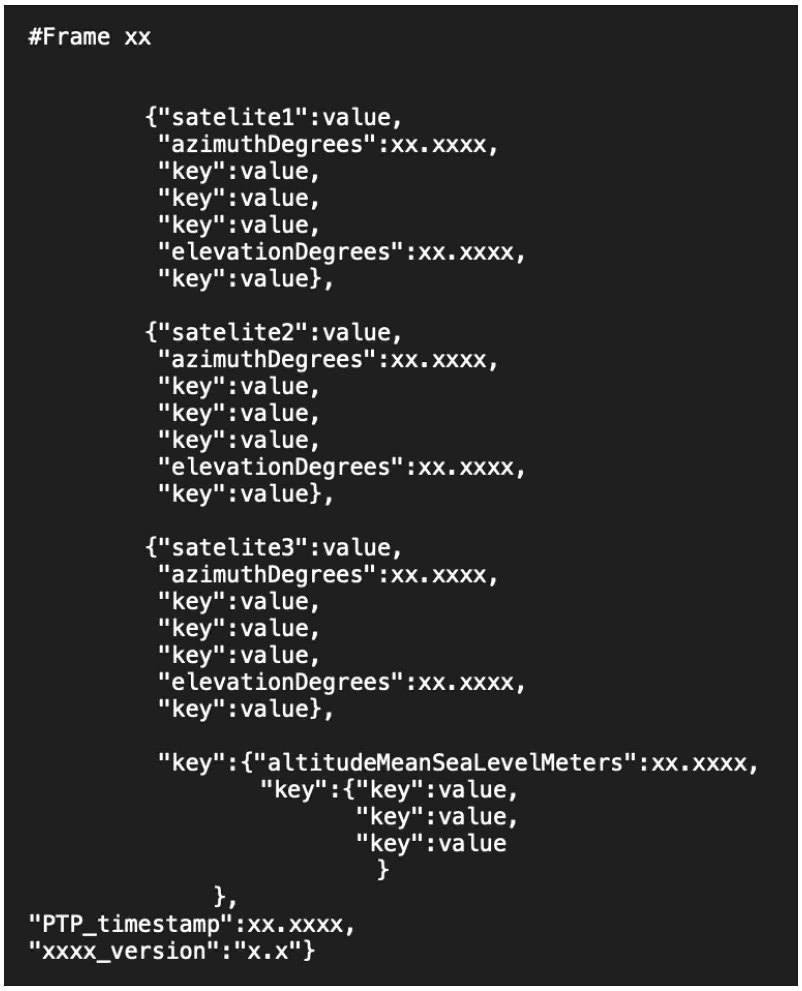


```
#Frame xx, Timestamp: x.xxxxx

        {"JSON-object":[
                {"key":value,
                 "key":value,
                 "key":{"key":value,
                        "key":value,
                        "key":value},
                 "accuracy":x.xxxx,
                 "latitude":xx.xxxx,
                 "longitude":xx.xxxx,
                 "speedAccuracy":x.xxxx,
                 "speed":x.xxxx}
                ],

"PTP_timestamp":xxxxx,
"xxxxx_version":"x.x"}
```

Figure 5.4: Abstraction of a JSON-object in clear-text, sent from component A to component B. Real key names are modified or replaced with "key", all values are replaced by *value* or a sequence of x's.



```
#Frame xx


        {"satelite1":value,
         "azimuthDegrees":xx.xxxx,
         "key":value,
         "key":value,
         "key":value,
         "elevationDegrees":xx.xxxx,
         "key":value},

        {"satelite2":value,
         "azimuthDegrees":xx.xxxx,
         "key":value,
         "key":value,
         "key":value,
         "elevationDegrees":xx.xxxx,
         "key":value},

        {"satelite3":value,
         "azimuthDegrees":xx.xxxx,
         "key":value,
         "key":value,
         "key":value,
         "elevationDegrees":xx.xxxx,
         "key":value},

         "key":{"altitudeMeanSeaLevelMeters":xx.xxxx,
                "key":{"key":value,
                       "key":value,
                       "key":value
                        }
             },
"PTP_timestamp":xx.xxxx,
"xxxx_version":"x.x"}
```

Figure 5.5: Abstraction of a JSON-object sent from component B to component A. Real key names are modified or replaced with "key", all values are replaced by *value* or a sequence of x's.

We concluded that network 2 shown in Figure 5.6 is closely linked to the user interface for the infotainment system. This is based on the internet services that component Y and Z request access from, such as requests to a satellite imagery and mapping service, and requests to internet services hosted by our industrial partner[2]. Examples of internet services used by some components are:

- Location API from a location API provider
- Multiple internet services provided by our industrial partner
- Other APIs from a subcontractor

Since the system appears to be operating a navigation service due to the location API, as well as other services provided by a subcontractor providing user interface services, we are confident that this is linked to the user interface of the infotainment

[2]The suppliers of component B in Figure 5.3, and the components X, Y, Z in Figure 5.6 was identified by name, but is obfuscated in this report due to security reasons.

system.

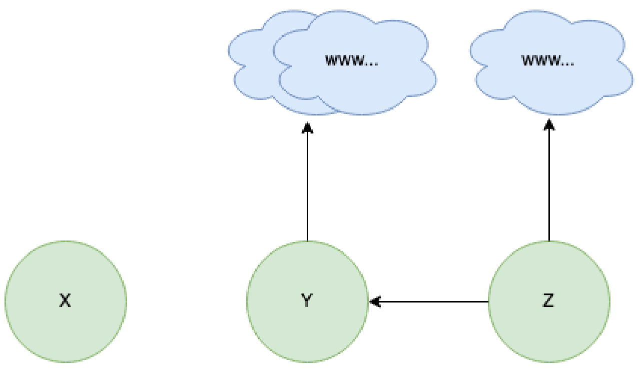


Figure 5.6: Showing the mapped components from network 2 with package flow, including requests to the world wide web.

Finally, we were able to discover which component was produced by which in-vehicle systems manufacturer. We analysed which manufacturers were hosting the internet services requested by the components on the Ethernet network, and also searched online after which standardized MAC addresses are registered for which manufacturers. We could therefore conclude which manufacturers and which vehicle systems our industry partner uses. In Figure 5.7 we believe component B, X and Y to be delivered by the same manufacturer and work together to handle a specific functionality in the system.

Component Z is believed to be a main component for the user interface for the infotainment system in the vehicle. The scan of the OBD-II port showed there is an android device connected to the network. Component Z makes DNS requests to services from an infotainment service manufacturer, therefore there is reason to believe that this component is the Android device, and handles the user inputs for the infotainment system.

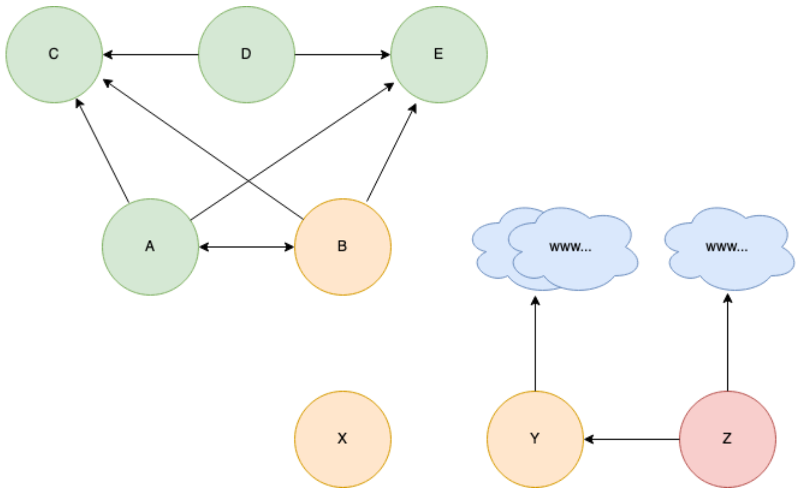


Figure 5.7: Showing components believed to be provided by one sub-contractor, colored in orange, Android device colored in red. External resources such as web API's are shown as blue clouds

All of these findings present issues for the manufacturer since there is sensitive information in the different components of the car's communication layer and their network design. There are also privacy concerns for the owner and user of the vehicle due to the telemetry information found in cleartext.

Being able to construct a topology of the network gives an attacker a better understanding of the network. This can make vulnerabilities in the network more obvious to an attacker and give suggestions for how and where malicious packets can be sent.

5.2.4 Injection

The result of the injection was ambiguous. The program constructed to send a SOME/IP request to a targeted IP-address did not give any concrete results. We were initially not sure if the SOME/IP request had been sent successfully, but based on the monitored traffic from the capture module, we were able to intercept TLS protocols we had not captured in earlier recordings where no attempt of any injections was made. While we cannot be sure of a direct correlation between our results that included the TLS protocol and the injection, the occasions that this protocol was discovered might suggest that there is. According to Zelle et al. [42], the TLS protocol is an efficient tool for encrypting the communication and ensuring authenticity as well as confidentiality and integrity. The authors' description of its implementation would explain why our injection did not work, since the injected packet would have had to use the same cryptographic algorithm as in the network. However, the authors also mention how TLS ensures confidentiality in the network since the information is encrypted, which we have stated above as not being the case since we were able to read GPS data from certain packets. This raises doubts about how this protocol is implemented and what data it protects.

5.2.5 Recommendations

Here we present our recommendations to mitigate the vulnerabilities we have discovered.

MACsec
Our findings from the Ethernet network were possible due to the link layer being vulnerable to passive listening attacks. To mitigate this vulnerability a standard such as IEEE 802.1AE or the link layer network security standard MACsec can be implemented. The reason for this choice is that MACsec protects the confidentiality of low level network protocols such as ARP, which was instrumental in discovering the network topology. Major integrated circuit manufacturers such as Broadcom and Marvell have released 1000BASE-T1 Routers with support for this technology (8957X-PB1 and 88Q222XM respectively) which could make the cost of implementing this recommendation more reasonable.

Obfuscated MAC addresses
Coupling a manufacturer to the MAC address on an otherwise unknown ECU could immensely accelerate the reverse engineering process. To hinder this, we recommend randomizing the MAC address so that this coupling cannot be made. This method is already in use in consumer devices such as smartphones, where for example IOS devices automatically generate a random MAC address every 24 hours to prevent fingerprinting, where attackers gather information about the network through scanning it or analysing its response after injecting packets [43]. Continuously regenerating MAC addresses would be unnecessary in this case and would only increase the complexity of the network. Instead generating new MAC addresses during the assembly process would be sufficient.

TLS
Finally, we propose Ethernet networks to utilise application level encryption for non critical services such as telemetry and infotainment systems. While MACsec protects the links between devices, if a malicious agent gains control of the switch or any of the devices inbetween, then the guarantee of confidentiality is lost. Implementing end-to-end encryption between services with lax latency requirements guarantees a stronger protection against reverse engineering and the loss of confidentiality of sensitive information. TLS could be a suitable candidate for such end-to-end encryption. TLS has been well tested as the standard behind HTTPS, and the 1.3 release included many latency improvements [44]. However, further research is needed to find if TLS is actually suitable for automotive networks.

5.3 Validation

The results from our topology discovery (5.2.3) as well as our results from our CAN analysis (4.3.4) was sent to our industry partner for validation to confirm the accuracy of our methods. Our deduction that component Z was a central component in the infotainment system was correct, as well as the existence of a telemetry device handling positioning data. Our industry partner also validated that checksums and counters are partly implemented within CAN, and that the headlights are not protected with these mechanisms. Our hypothesis that the OBD-II port operated in a different way compared to the internal CAN was also confirmed, it worked mostly on a request-basis in contrast to the internal CAN.

6 Discussion

We have analysed the security of the Controller Area network and the Ethernet network in a modern automotive vehicle. The experiments have been conducted with commodity hardware, and without expert knowledge of the underlying electrical system, and the outcome has been evaluated to answer our research questions.

6.1 Research Question 1

How resilient are in-vehicle networks against unauthorized access and manipulation, conducted with cost-efficient hardware and without expert knowledge?

Regarding unauthorized access and manipulation, our experiments showed that the tested in-vehicle networks were not resilient to these vulnerabilities. We demonstrated that both the CAN and Ethernet networks are accessible with cost-efficient tools and no expert knowledge. Using a reduction algorithm (4.3.4), we were able to identify the desired CAN frames and initiate a replay attack within hours of first gaining access to the test bed. The Ethernet network was also compromised by inserting a capture module between two connections, allowing us to read sensitive information in plain text.

To validate the danger of unauthorized access to an automotive network, we analyzed the Ethernet network's vulnerability to *Topology and Vulnerability Discovery* as documented by Kiravuo et al. [8]. Using the methods suggested in the paper (such as analysing ARP requests) we were able to extend the aforementioned study into the automotive domain, and document the network's topology and identify devices. With this, we can confirm that *Topology and Vulnerability Discovery* is not limited to computer networks but automotive networks as well.

By taking control of the headlights, we have shown an alternative approach to compromising components connected to the CAN bus compared to Buttigieg et al. [6]. They exploited the lack of authentication in the CAN Protocol by performing a Man-in-the-middle attack. We performed replay attacks as an alternative method and achieved similar results. This shows that the security issues presented by Buttigieg et al. are not limited to Man-in-the-middle attacks, and further strengthen their claim that with enough knowledge, time and resources, an attacker can perform such attacks. Furthermore, physically connecting to the CAN bus as a proxy leads to more control of the network since the proxy can alter sent signals, which is impossible for a node with a regular bus connection. But on the other hand, it is more complicated to establish such a connection, especially in a vehicle environment.

6.2 Research Question 2

What software solutions could have prevented methods exercised in question 1?

We have presented suggestions for software solutions that would have prevented our findings from research question 1, for both CAN (5.1.4) and Ethernet (5.2.5). For CAN we suggest the use of a reference counter to protect the integrity of CAN frames, while for Ethernet we suggest using multiple standards to protect the multiple layers of the OSI network stack.

Concerning the vulnerability mitigation in automotive Ethernet, we focused more on protecting confidentiality rather than integrity. This was in contrast to the work of Corbett et al. [13] who argued that the integrity of the network is more important than the confidentiality due to the direct life threatening dangers of losing control of a vehicle at high speeds. However, this does not make protecting confidentiality unnecessary. Protecting the confidentiality of packet content increases the cost of reverse engineering services running in the network, thus adding an additional hurdle for reliably affecting the integrity of the network.

In [4] the authors evaluate the most promising CAN authentication methods but based on the industrial criteria they reject all of them. In their conclusion, they say that the CAN bus is unsuited for secure communication and other technologies with higher bandwidth should be considered. The lack of adoption in industry of security mechanism in CAN, and the result from their evaluation indicate that a solution that guarantees complete security is not possible. But we argue that low security is better than no security. Lin's [7] security mechanism included counters to prevent replay attacks and these types of counters would have prevented our attack strategy.

6.3 Unexpected Findings

In the beginning of the project, we expected to spend a significant amount of time on reverse engineering and analysing the data to achieve the knowledge needed to inject data. Our goal with our first visit to the industrial partner was to collect data, but within four hours, we had already achieved control of the headlights. The ease of performing a replay attack was surprising. Koscher et al. [3] drew the same conclusion when experimenting on a car from 2009, and we can confirm that this is still the case for a car produced in 2020. We expected the tested CAN communication bus to implement some type of security mechanism against replay attacks. We found nothing preventing us from controlling the headlights after establishing a connection to the bus. This might be a consequence of how the vehicle's system prioritizes components in terms of security. Securing every component leads to a larger overhead than only securing a subset. Excluding the headlights may in that way make it possible to have stronger security on more safety critical components.

Another unexpected finding was data appearing in plain text when analysing the Ethernet communication. A reason for this could be that the computing device that handled the data did not have enough computational power to decrypt it. Another reason could be that the network designers found the encryption of this data to be an inefficient use of resources, in spite of it being possible to map the network based on this plain text information.

6.4 Limitations

The total number of vehicles tested in this paper was only two. As such we cannot state for a fact that their vulnerabilities are shared by other vehicles outside of those we have tested. Additionally, the transition from analysis to the implementation of new research findings within automotive networking take time. As such our test bed, which represents the electrical system in a production model, does not represent the latest developments in the field. Therefore, the vulnerabilities and recommendations may have already been accounted for in future models.

Moreover, the experiments conducted in this thesis were performed on a test bed designed for easy access. Consequently, physical obstacles such as the vehicle body and the placement of components in a real car provide an additional layer of security that we have not accounted for.

We were also not able to verify the automotive Ethernet's vulnerability to unauthorized manipulation. This was because we required specialized software to interact with the identified services that was not available and too complex to be developed in time for testing.

Finally, we have not tested whether our solutions' resource cost and operational overhead are suitable for automotive networks. Automotive networks require more strict limits in latency than most computer networks. Therefore, additional analysis and possibly modification of existing solutions are needed before being considered for implementation in new vehicles.

6.5 Future Work

We hope the results from our findings can be used for further research into automotive network security. For instance, further research is needed in analysing higher level automotive protocols such as SOME/IP for vulnerabilities. Additionally, in order to protect automotive networks from the attacks demonstrated in this paper, research into adapting existing standards to automotive networks (such as the work of Martin Lang [5]) is needed.

Another suggestion for future work is research into the computational overhead for our suggested solutions. In this project, we intentionally explored the network's security from a black box perspective, making it impossible to calculate the cost in terms of operational resources for real-time systems.

The CAN protocol analysed in this thesis was an in-vehicle network implementation. But the CAN protocol is also utilised in other domains such as elevators, robotics and railways. Future research should be conducted into whether the methods examined and vulnerabilities found in this thesis apply to other domains as well.

Finally, the examined attacking methods relied on a wired connection to a stationary computer next to the test bed. Since the found vulnerabilities could pose a threat to the driver and passengers in a moving vehicle, remote access to the in-vehicle networks should be explored and assessed.

7
Ethical Aspects

Performing a security analysis on a vehicle platform that is currently in production poses a substantial ethical dilemma. Any vulnerabilities found in the design of the in-vehicle network and our methods for data extraction, analysis and possible tampering had to be exhaustively discussed in terms of presentation such that the project's outcome does not end up being a guide for tampering with the in-vehicle network. The goal is to propose software related solutions to the identified problems in order to increase the security of a vehicle, but publishing a thesis that contains instructions on how to hack a car could lead to the opposite by making the information publicly available to people with malicious intent. Furthermore, tampering is not limited to malicious acts only, since layman tampering could result in severe or even fatal consequences.

This responsibility is not only towards the public but also our industry partner who has made this project possible. Any findings during the project were reported to them and possibly sensitive information was also redacted upon their request. Respecting this has been important since the employees of the car manufacturer are the domain experts, and they have a better understanding of the consequences if sensitive information is published.

The vehicles on which we performed our experiments on were supplied by our industry partner and inevitably led to the project being rather manufacturer specific. Nonetheless, this report sheds light on the broader more structural security issues, some of which are flaws in the network standards and could thus potentially affect the industry as a whole.

8 Conclusion

The functionality in a modern vehicle relies heavily on the in-vehicle network and electronic devices. A network failure or vulnerability could result in a substantial economic loss or even lethal consequences. With an increasing amount of sensitive and critical data sent over the in-vehicle network, network security in the domain is an ever increasing topic.

In this thesis we have shown that expert vehicular network domain knowledge is not needed to perform an attack on a modern vehicle. Limited to commodity hardware, it was not only possible to sniff the in-vehicle networks CAN and Ethernet, but also inject data. CAN was susceptible to replay attacks which were exercised for reverse engineering instructions and blocking user input. In the Ethernet network, the Man-in-the-middle attack resulted in unencrypted information as well as enough information to draw the topology of the network. Thus, our assessment is that the in-vehicle networks experimented on could not be regarded as resilient or secure. However, as mentioned before, the test bed used in our experiments was a model in production and therefore might not have all the security measures that newer models do.

We have also proposed software security enhancements to the identified vulnerabilities in the form of low security counters for the CAN bus, TLS encryption for high level encryption of Ethernet traffic and MACsec for low level obfuscation of vendor IDs which would complicate reverse engineering.

This thesis concludes that the in-vehicle networks are insecure, in terms of the vulnerabilities we discovered. More research has to be concluded to assess whether the same insecurities are exploitable from remote access. Additionally, future work has to assess how the overhead of our proposed solutions compares to existing security technologies used in the industry.

Bibliography

[1] D. Morris, G. Madzudzo, and A. Garcia-Perez. “Cybersecurity threats in the auto industry: Tensions in the knowledge environment”. In: *Technological Forecasting and Social Change* 157 (2020). ISSN: 0040-1625. DOI: `https://doi.org/10.1016/j.techfore.2020.120102`. URL: `https://www.sciencedirect.com/science/article/pii/S0040162520309288`.

[2] Z. Levi. *Automotive Ethernet: The Future of In-car Networking?* URL: `https://www.electronicdesign.com/markets/automotive/article/21806349/automotive-ethernet-the-future-of-incar-networking` (visited on 03/21/2021).

[3] K. Koscher et al. “Experimental Security Analysis of a Modern Automobile”. In: *2010 IEEE Symposium on Security and Privacy* (July 2010). DOI: `10.1109/sp.2010.34`.

[4] N. Nowdehi, A. Lautenbach, and T. Olovsson. “In-Vehicle CAN Message Authentication: An Evaluation Based on Industrial Criteria”. In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. 2017, pp. 1–7. DOI: `10.1109/VTCFall.2017.8288327`.

[5] M. Lang. “Secure Automotive Ethernet: Balancing Security and Safety in Time-Sensitive Systems”. MA thesis. Faculty of Computing: Blekinge Institute of Technology, Aug. 2019.

[6] R. Buttigieg, M. Farrugia, and C. Meli. “Security issues in controller area networks in automobiles”. In: *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2017, pp. 93–98. DOI: `10.1109/STA.2017.8314877`.

[7] C. Lin and A. Sangiovanni-Vincentelli. “Cyber-Security for the Controller Area Network (CAN) Communication Protocol”. In: *2012 International Conference on Cyber Security*. 2012, pp. 1–7. DOI: `10.1109/CyberSecurity.2012.7`.

[8] T. Kiravuo, M. Sarela, and J. Manner. “A Survey of Ethernet LAN Security”. In: *IEEE Communications Surveys Tutorials* 15.3 (2013), pp. 1477–1491. DOI: `10.1109/SURV.2012.121112.00190`.

[9] B. Aboba et al. *Extensible Authentication Protocol (EAP)*. RFC 3748. ipUnplugged, June 2004, pp. 1–67. URL: `https://tools.ietf.org/html/rfc2104`.

[10] P. Meyer et al. “DoS Protection through Credit Based Metering - Simulation Based Evaluation for Time-Sensitive Networking in Cars”. In: *Proceedings of the 6th International OMNeT++ Community Summit*. 2019.

[11] J. Choi, S. Min, and Y. Han. “MACsec Extension over Software-Defined Networks for in-Vehicle Secure Communication”. In: *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. 2018, pp. 180–185. DOI: `10.1109/ICUFN.2018.8436963`.

[12] Keen security lab. *Experimental Security Assessment of BMW Cars: A Summary Report*. Tech. rep. 2018, pp. 1–24. URL: `https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf`.

[13] C. Corbett et al. “Automotive Ethernet: Security opportunity or challenge?” In: *Sicherheit 2016 - Sicherheit, Schutz und Zuverlässigkeit* (2016).

[14] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach*. Pearson, 2017.

[15] D. Durisic. “AUTOSAR standard”. In: *Automotive Software Architectures*. Ed. by M. Staron. Springer International, 2017, pp. 81–116.

[16] H. Yu and C. Lin. “Security concerns for automotive communication and software architecture”. In: *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2016), pp. 600–603. DOI: `10.1109/INFCOMW.2016.7562147`. URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7562147&isnumber=7561992`.

[17] H. Zhang et al. “CANsec: A Practical in-Vehicle Controller Area Network Security Evaluation Tool”. In: 20 (2020). DOI: `10.3390/s20174900`.

[18] Steve Corrigan. “Introduction to the Controller Area Network (CAN)”. In: (Aug. 2002).

[19] Miroslaw Staron. “Evaluation of Automotive Software Architectures”. In: *Automotive Software Architectures*. Springer International, 2017, pp. 157–158.

[20] J. Xia et al. “Real-time and reliability analysis of time-triggered CAN-bus”. In: *Chinese Journal of Aeronautics* 26.1 (2013), pp. 171–178. ISSN: 1000-9361. DOI: `https://doi.org/10.1016/j.cja.2012.12.017`. URL: `https://www.sciencedirect.com/science/article/pii/S1000936112000258`.

[21] H. Daigmorte and M. Boyer. “Evaluation of Admissible CAN Bus Load with Weak Synchronization Mechanism”. In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS ’17. Grenoble, France: Association for Computing Machinery, 2017, pp. 277–286. ISBN: 9781450352864. DOI: `10.1145/3139258.3139261`. URL: `https://doi.org/10.1145/3139258.3139261`.

[22] K. Tindell, A. Burns, and A.J. Wellings. “Calculating controller area network (can) message response times”. In: *Control Engineering Practice* 3.8 (1995), pp. 1163–1169. ISSN: 0967-0661. DOI: `https://doi.org/10.1016/0967-0661(95)00112-8`. URL: `https://www.sciencedirect.com/science/article/pii/0967066195001128`.

[23] T. Steinbach, F. Korf, and T. Schmidt. “Real-time Ethernet for automotive applications: A solution for future in-car networks”. In: *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin).* 2011, pp. 216–220. DOI: `10.1109/ICCE-Berlin.2011.6031843`.

[24] C. Kozierok et al. *Automotive Ethernet: The Definitive Guide.* Intrepid Control Systems, 2014.

[25] P. Laygude. *Overview on Automotive Ethernet.* Vector India Conference 2019. July 9, 2019. URL: `https://assets.vector.com/cms/content/events/2019/VH/VIC2019/Track_1_5_Glancing_Ethernet_communication_for_Automotive_1.pdf` (visited on 05/12/2021).

[26] K. Matheus and T. Königseder. *Automotive Ethernet.* 2nd ed. Cambridge University Press, 2017. DOI: `10.1017/9781316869543`.

[27] C. Burgardt. “In-Vehicle Network Security: Attack and Countermeasures”. MA thesis. Faculty of Information Technology: Federal University of Pernambuco, Dec. 2018.

[28] T. Steinbach. *Ethernet-basierte Fahrzeugnetzwerkarchitekturen für zukünftige Echtzeitsysteme im Automobil.* Springer, 2018.

[29] D. Kim et al. *Study for Recovering Crypto Key Mismatch of In-Vehicle Network DTLS.* 2018.

[30] CSS Electronics. *OBD2 Explained - A Simple Intro (2021).* URL: `https://www.csselectronics.com/screen/page/simple-intro-obd2-explained#obd2-pid-frame` (visited on 05/14/2021).

[31] Z. El-Rewini et al. “Cybersecurity challenges in vehicular communications”. In: *Vehicular Communications* 23 (2020), p. 100214. ISSN: 2214-2096. DOI: `https://doi.org/10.1016/j.vehcom.2019.100214`. URL: `https://www.sciencedirect.com/science/article/pii/S221420961930261X`.

[32] W. Stallings and L. Brown. *Computer Security: Principles and Practice.* Fourth Global edition. Pearson Education Limited, 2018. ISBN: 9781292220611, 1292220619.

[33] M. Bozdal et al. “Hardware Trojan Enabled Denial of Service Attack on CAN bus”. In: *Procedia Manufacturing* 16 (2018), pp. 47–52. ISSN: 2351.9789. DOI: `10.1016/j.promfg.2018.10.158`. URL: `https://www.sciencedirect.com/science/article/pii/S2351978918312794`.

[34] K. Strandberg, T. Olovsson, and E. Jonsson. “Securing the Connected Car”. In: *IEEE Vehicular Technology Magazine* 13 (Mar. 2018), pp. 56–65.

[35] J. Kreissl. “Absicherung der SOME/IP Kommunikation bei Adaptive AUTOSAR”. MA thesis. Faculty of Computing: University of Stuttgart, Nov. 2017.

[36] D. Möller and R. Haas. *Guide to Automotive Connectivity and Cybersecurity: Trends, Technologies, Innovations and Applications.* Springer, 2019.

[37] C. Abad and R. Bonilla. “An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks”. In: *Proceedings of 27th International Conference on Distributed Computing Systems Workshops*. IEEE Computer Society, 2007.

[38] Y. Mirsky, N. Kalbo, and Y. Elovici. “Vesper: Using Echo-Analysis to Detect Man-in-the-Middle Attacks in LANs”. In: *IEEE Transactions on Information Forensics and Security* (2019). DOI: 10.1109/TIFS.2018.2883177.

[39] J. Liu et al. “In-Vehicle Network Attacks and Countermeasures: Challenges and Future Directions”. In: *IEEE Network* 31.5 (2017), pp. 50–58. DOI: 10.1109/MNET.2017.1600257.

[40] A. Hevner et al. “Design Science in Information Systems Research”. In: *Management Information Systems Quarterly* 28.1 (2004), pp. 75–105.

[41] *INTERNET PROTOCOL*. RFC 791. University of Southern California, Sept. 1981, pp. 1–45. URL: https://tools.ietf.org/html/rfc791.html (visited on 04/25/2021).

[42] D. Zelle et al. “On Using TLS to In-Vehicle Networks”. In: *ARES ’17: Proceedings of 12th International Conference on Availability, Reliability and Security*. Association for Computing Machinery (ACM), 2017, pp. 1–10. ISBN: 9781450352574. DOI: 10.1145/3098954.3105824.

[43] *Use private Wi-Fi addresses in iOS 14, iPadOS 14 and watchOS 7*. URL: https://web.archive.org/web/20210424120018/https://support.apple.com/en-gb/HT211227 (visited on 04/29/2021).

[44] *Differences between TLS 1.2 and TLS 1.3*. URL: https://web.archive.org/web/20190919000200/https://www.wolfssl.com/differences-between-tls-12-and-tls-13-9/ (visited on 05/01/2021).