A Concept for an Intrusion Detection System over Automotive Ethernet

HANNA LINDWALL
PONTUS OVHAGEN
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

A Concept for an Intrusion Detection System over Automotive Ethernet

Hanna Lindwall dic14hli@student.lu.se
Pontus Ovhagen dic14pov@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Paul Stankovski Wagner

Examiner: Thomas Johansson

March 2, 2020

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

A modern automotive vehicle is a complex technical system, containing many electronic, mechanical, and software parts. Typically, a high-end vehicle contains 70 or more electronic control units (ECUs) on average. These are controlling a large number of distributed functions, of which many are safety-critical, and adding complexity, which is surpassing 100 million lines of code.

Furthermore, the communication link in the automotive architecture is also being upgraded from the traditional controller area network (CAN) bus to Automotive Ethernet, in order to enable higher communication bandwidth and handle the increasing complexity. However, introducing Ethernet opens up for new attacks and loopholes to be exploited by hackers. Attacks on ECUs are even more dangerous than web attacks, as these involve the safety of the persons inside the vehicle. To secure the in-vehicle communication the automotive industry needs to look into traditional cybersecurity protection techniques from an automotive perspective. One security solution gaining more and more attention regarding in-vehicle security is the concept of an intrusion detection system (IDS).

In this thesis, we propose a concept for a host-based IDS relying on two different detection methods. We suggest a combination of specification-based, focusing on message sequencing and allowed elapsed time in between a request and its respective response, and anomaly-based detection, evaluating the frequency, payload length and timeout for request-response pairs. To evaluate our IDS we execute five different attack scenarios, where we calculate binary classification metrics and measure its classification speed. Our evaluation shows that the proposed IDS successfully detects malicious events such as delay, packet injection, exhaustion and two different flooding attacks.

Based on our experience designing an in-vehicle IDS, we describe potential difficulties, limitations and future improvements that engineers can use to implement or improve their adaptation of an in-vehicle IDS system. We believe the results of this master's thesis can be applied in more advanced research, especially in the field of IDS for in-vehicle networks, and can hopefully contribute to a safer driving experience.

Keywords: Intrusion Detection System, Deep Packet Inspection, Specification-based Detection, Anomaly-based Detection, V2G, Automotive Ethernet.

Acknowledgements

Throughout this master's thesis, we have acquired help from ESCRYPT and other research teams at Bosch. The IDS team at ESCRYPT conducted a demonstration of their solution and always answered our questions. Due to tough competition in the industry and for security reasons, it is not easy to find public research on security solutions for vehicles, so we are grateful for their support.

Popular Science Summary

A Concept for an Intrusion Detection System over Automotive Ethernet by Hanna Lindwall and Pontus Ovhagen

With the transition to automotive Ethernet as a standard network bus, the faster network speeds and additional bandwidth benefits will be the main advantages for internal vehicle networks. However, there are also inevitable security risks by introducing Ethernet in vehicles and this is the main problem we are trying to address in our master's thesis.

The automotive industry has to investigate traditional cybersecurity solutions used to secure traditional networks, in order to reach the goal of securing in-vehicle automotive Ethernet networks. Today, firewalls are commonly used as a first layer of protection, but as attacks have become increasingly sophisticated, a firewall's detection mechanisms can easily be bypassed. Therefore, the automotive industry has started to look into proactive security solutions that can detect new threats, which have not been made public or discovered yet. One solution is implementing an intrusion detection system (IDS), which has been practiced in various industries since the early days of network security and is now being applied for automotive use cases. An IDS monitors and detects attacks or other threats present inside the network. Compared to a firewall, the IDS takes a more proactive position by reporting, alerting and logging detected threats against a system. An IDS often relies on extensive deep packet inspection to inspect deeper into a packet and can, therefore, screen packets on an application-level basis. Common functionalities of IDS include, for example, updating the system for future attacks, analysis of potential attack patterns on the network, alerting other security mechanisms in place or generating warnings for network administrators.

In this master's thesis we propose a concept for a host-based in-vehicle IDS. We then implement and test its capabilities by launching a series of attacks against the IDS and measure its detection performance. To detect threats we incorporated two different detection methods: specification-based and anomaly-based detection.

The specification-based detection focuses on deviations from specified behavior in a protocol. We chose to focus on the ISO 15118 specification, which describes a protocol for perform-

ing Vehicle-to-Grid charging sessions. In our implementation, we mainly focus on deviations in message sequences and timeouts outlined in the protocol.

After further research, it became apparent that the specification-based detection did not have full detection coverage. For this reason, we decided on integrating an anomaly-based detection method into our IDS. Anomaly-based detection is also a method for detecting anomalous behavior and relies on a statistical approach. In our IDS, we collect data from different examples of charging sessions and based on this data the IDS classifies the incoming packet either as a threat or a normal packet. The anomaly-based detection method evaluates the expected frequency, payload length and the time elapsed between a message request and its respective response.

The IDS overall performed very well for the five attack scenarios we deployed and shows that an IDS with this hybrid approach is a promising security solution for in-vehicle automotive Ethernet networks.

Table of Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current
CA	Certificate Authority
CAN	Controller Area Network
DC	Direct Current
DCU	Domain Control Unit
DoS	Denial of Service
DPI	Deep Packet Inspection
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
E/E	Electrical/Electronic
EIM	External Identification Means
EV	Electric Vehicle
EVCC	Electric Vehicle Communication Controller
EXI	Efficient XML Interchange
FNR	False Negative Rate
FPR	False Positive Rate
HIDS	Host-based Intrusion Detection System
HSM	Hardware Security Model
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IoT	Internet of Things

IV	Initialization Vector
LIN	Local Interconnect Network
LnUB	Lower- and Upper Bound
MAC	Media Access Control
MCU	Microcontroller Unit
MOST	Media Oriented Systems Transport
NIDS	Network-based Intrusion Detection System
OEM	Original Equipment Manufacturer
PKI	Public Key Infrastructure
PnC	Plug and Charge
ROC	Receiver Operating Characteristic
SDP	SECC Discovery Protocol
SECC	Server Communication Controller
SOME/IP	Scalable Service-Oriented Middleware over IP
SSL	Secure Socket Layer
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPR	True Positive Rate
UDP	User Datagram Protocol
V2G	Vehicle-to-Grid
V2GTP	V2G Transfer Protocol
V2I	Vehicle-to-infrastructure
V2V	Vehicle-to-vehicle
XML	Extensible Markup Language

Chapter 1
Introduction

This chapter gives the reader a brief introduction to the master's thesis. We also address the problem at hand and what goals have been set for the project. Furthermore, this section recognizes previous work and further explains the contributions of this thesis.

1.1 Background

The modern automobile industry has to provide drivers with a wide range of safety, comfort and assistance functionalities to keep up with consumer demands and the harsh competition. This development has resulted in an evolution of additional features in new vehicle generations. Most of the functional domains are electronic, which has led to the emergence of advanced electrical/electronic (E/E) architectures. More and more of the vehicle system is dependent on connectivity and communications within and outside the vehicle, hence manufacturers are nowadays required to develop systems that can handle high bandwidth in addition to being well-timed [1].

To cope with the increase of data streams, automotive Ethernet has been introduced as a solution. Automotive Ethernet has established itself as the future protocol for data communication in vehicles and can be used as the central backbone to connect multiple domain control units (DCUs) with a gateway [2], see Figure 1.1. A DCU is a coordinating control unit which processes data and provides operations within a specific domain in a vehicle. With its fast data speeds up to several Gbit/s, automotive Ethernet outperforms previous standards. Automotive Ethernet is based on the Ethernet protocol, but with additional requirements for the automotive industry. Ethernet is well-known and tested, due to the fact that it has been around for over 30 years, but so has its security flaws.

Imagine driving down a highway and suddenly the engine is shut off, or you start the engine and are not able to control the steering. This may be a possibility due to modern auto industry being a nodal point in the internet of things (IoT). While IoT increases convenience and versatility, it also presents a bigger target for cyber attackers [4].

Ever since the introduction of ECUs in vehicles, these units have been the target for malicious attacks [5]. Introducing communication over Ethernet to vehicles comes with inevitable security risks, which can be combated with traditional

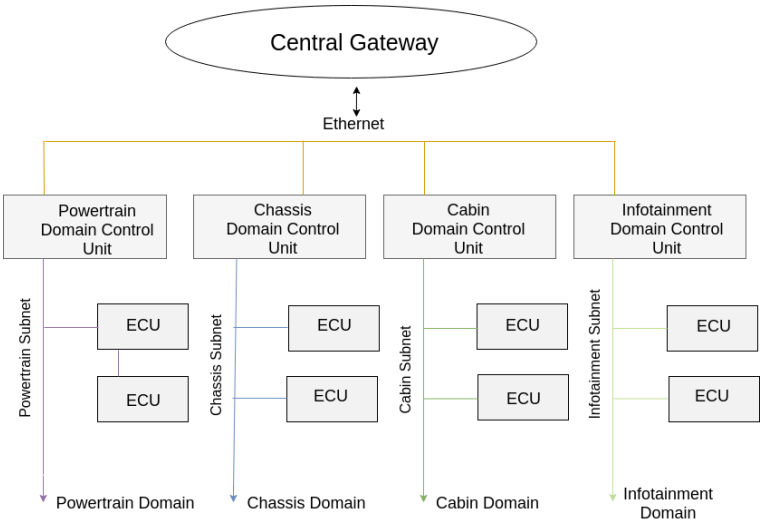

Figure 1.1: Future vehicle architectural setup with a distributed gateway and DCUs. Figure inspired by [3].

security solutions, such as a firewall or an intrusion detection system. However, as mentioned, the automotive industry has requirements for in-vehicle communication, such as low latency and well-timed communication, which has to be taken into account when developing security solutions for vehicles.

Robert Bosch GmbH is a renowned international company in the technology industry and delivers vast amounts of products and software within multiple fields. One of these fields is security in the automobile industry, where Bosch is developing embedded software for devices that will be utilized for the automotive Ethernet standard.

The ambition is to introduce new features using deep packet inspection (DPI) in combination with an intrusion detection system to discover potential threats. This master's thesis is a cooperation with ESCRYPT, a wholly-owned subsidiary of a Bosch company called ETAS GmbH, and will focus on DPI and IDS for automotive Ethernet.

1.2 Problem Statement and Goals

The problem addressed in this thesis is: *How can one integrate IDS using DPI in an automotive setting to increase security coverage?* Moreover, the answer to this problem is based on specified key questions that can be seen below. These questions are essential to the scope of this report and serve as a foundation for the model implementation, discussions and conclusions presented.

Academic questions

1. Which requirements are associated with DPI for automotive Ethernet?

2. Which requirements are associated with IDS in the context of automotive Ethernet?

    - What functions of IDS are valuable from the perspective of automotive Ethernet?

To measure progress and set tangible milestones for the project, a set of goals have been established. These goals, in addition to the key questions mentioned above, are further evaluated in the closing chapters of this thesis.

Goals

- Analyse which IDS features would fit the automotive domain from security and resource point of view.
- Propose a proof of concept of how some of those features would be designed and implemented.
- Implement the proof of concept for at least one attack vector.

1.3 Delimitations

This report mainly focuses on the use of DPI and IDS in the context of automotive Ethernet.

With the use of Ethernet in automobiles, the car's internal systems now interact as any device connected to the Internet. Hence, there is extensive amount of protocol implementations which could be studied in detail for security threats. The scope of this thesis is therefore delimited to the ISO 15118 specification, both theoretically and for the implementation of IDS features. The specification covers Vehicle-to-Grid (V2G) interaction and the thesis will focus on this type of communication. The main focus is on delivering a working proof of concept that we can test and evaluate.

In this project, we will not evaluate the actual integration aspects of deploying an IDS into a vehicle, we leave this part of the project for others to pursue and evaluate. The reason being the time constraints on completing the project within the set deadline. Furthermore, this implementation cannot be seen as production-ready, and we do not address further challenges that come with porting, running and testing successfully in a different environment than the one mentioned in this thesis.

V2G charging is a fairly new and unexplored area for ESCRYPT. Consequently, there is no prior work for us to build on. With no luck finding captures with ISO 15118 traffic, we have to rely on the fairly static session generation from open-source projects. The two projects OpenV2G and RISEV2G were used as basis for training and testing our implementation. This will make the IDS partially biased towards our training examples, with no complete assurance of similar results being produced for authentic in-vehicle session data.

1.4 Contributions

The major contributions of this report can be stated as follows.

- We select an evaluation framework based on binary classification and state how this framework is used to evaluate the IDS in Chapter 3.
- We outline requirements for an IDS implemented for automotive Ethernet in Chapter 4.
- We outline requirements for DPI incorporated into a security solution for automotive Ethernet in Chapter 4.
- We propose a proof of concept for a host-based IDS (Chapter 5) and show that the implementation works by launching a series of attacks against it. The best configurations of our IDS are presented in Chapter 6. The configurations can be used for other studies or as a stepping stone for developers to use in their implementation.

1.5 Related Work

1.5.1 Design and Implementation of an IDS for In-vehicle Networks

The controller area network data link protocol has been the standard for the automobile industry under a considerable amount of time. The science community has over the years researched CAN thoroughly and many contributions have pointed out its security flaws [6, 7, 8].

Salman and Bresch [9] implemented an IDS system for a CAN network topology, but rather than being present in a network switch, the IDS is deployed on a single ECU in the car network. Their implementation highlights how to design a hybrid IDS using specification-based and anomaly-based detection methods. Some valuable features include: detection of unauthorized messages in specific domains on the network, setting a threshold for fluctuations of speedometer data and accounting cycle times for messages sent on the CAN bus. Even though the CAN protocol differs from the ISO 15118 specification in several aspects, these features accentuate what type of functions are of interest from an automotive perspective when implementing an IDS. Finally, they conclude that having one detection approach is not enough. Instead, using a combination of two methods is a more promising alternative.

1.5.2 Anomaly Detection for SOME/IP using Complex Event Processing

Herold et al. [10] also implemented an IDS but for the SOME/IP protocol, which is a service-oriented protocol that manages both CAN and automotive Ethernet related messages. Their solution incorporated specification-based and anomaly-based detection where they used static rule sets and checked both frequency and timing. Their prototype was implemented in Java, and Esper was chosen as the Complex Event Processing engine. It accepts rules written in the Event Processing Language, a subset of SQL, extended with features for stream processing.

1.6 Literature Study

1.6.1 Security Analysis of Ethernet in Cars

Talic [11] conducted a thorough security investigation of automotive Ethernet with its associated protocols. The evaluation in this thesis was carried out with the help of a framework called OSSTMM. This framework allowed him to perform an extensive comparison between different configurations of his testbed and effectively measure the impact on security from adding or removing features he discovered in his analysis. Notable attack surfaces that were successfully discovered include ARP cache poisoning, SOME/IP spoofing of real-time data and denial of service (DoS).

1.6.2 Deep Packet Inspection for Intrusion Detection Systems: A Survey

Abuhmed et al. conducted a survey over techniques, challenges, and algorithms for DPI in 2007 [12]. Their work highlights several key challenges such as which search algorithm should be used, how to handle an increasing number of intruder signatures and also the challenge of inspecting locations of unknown signature locations. The latter is a central problem when performing DPI and dictates the degree of packet inspection performed at certain locations in security architecture. Furthermore, they compared hardware implementations of DPI from a collection of different sources where a matching algorithm using a Quad Bloom filter was found to be the highest performer with an output of 20.4 Gbit/s, compared to ternary content-addressable memories (TCAMs) that had a throughput of 12.35 Gbit/s.

1.6.3 Firewall and IDPS Concept for Automotive Ethernet

An important aspect of security architectures in vehicles is to divide the responsibilities between different subsystems or devices. In *Firewall and IDPS concept for automotive Ethernet* [13] written by Yilmaz in cooperation with Bosch, he derives a concept for an automotive firewall and IDPS (Intrusion Detection and Prevention System) to secure in-vehicle communication over automotive Ethernet. His model consists of a firewall that acts as an initial filter of incoming packets, where mostly header data is checked for malicious signatures. These signatures are identified as sequences of bytes or as a set of malicious instructions. The firewall is equipped with a TCAM and a rate meter for fast initial processing of packets at wire-speed. However, TCAM has its limitations due to e.g., high resource consumption. Moreover, unclassified packets from the firewall are routed to an IDPS for an extended search in the packet payload, where appropriate measures are taken to filter or block malicious packets. Both subsystems interchange information about detected threats and create dynamic rules as they adapt to newly discovered threats.

1.6.4 An approach to Specification-based Attack Detection for In-vehicle Networks

Larson et al. [14] presented a specification-based approach based on the CAN v2.0 and CANopen v3.01 specifications. Some features include the specification rules for message correctness, expected values and message cycle times. They further analyze the implications of placement and find that placing one detector on each ECU was optimal since they can omit transmitter and receiver identities. Conclusively, they state that their approach readily shows that most attack vectors specified can be detected from the attack scenarios they considered.

1.6.5 An Assessment Method for Automotive Intrusion Detection System Performance

On the behalf of the United States Department of Transportation, Stachowski et al. [15] present an assessment method on how to test and assess IDS solutions in a vehicle. Through simulating attacks over a CAN bus (by injecting packets), they evaluate the implementation by viewing the IDS as a binary classifier. With the use of receiver operating characteristic (ROC) plots and calculated ratio metrics, they evaluate several suppliers' IDSs by running the tests on three different vehicles. ROC plots are used to graphically illustrate the diagnostic ability of the binary classifier system (i.e., the IDS).

Even though this work is tailored towards an IDS designed for CAN use cases, the test methodology can be applied on IDS implementations supporting other protocols.

1.6.6 Road Vehicles - Vehicle-to-Grid Communication Interface

The ISO 15118 specification [16] describes a communication protocol for V2G charging that is expected to be one of the leading charging standards in the future. With the increased adoption of this standard, Hauck et al. [17] made a high-level overview of some of the security challenges and mitigation techniques that can be used to limit risks associated with modern vehicles. Among other threats, they highlight the risk of tampering in a charging station – enabling the attacker to affect the power quality of the local power system or obtain free charging sessions. This attack vector can be prevented with the use of strong encryption, removal of all external input jacks and integrating alerting mechanics. Furthermore, they suggest mitigation approaches such as always utilizing transport layer security (TLS) when possible, having highly protective storage for digital certificates and private/public keys as well as using digital signatures and encryption for all vulnerable messages.

Chapter 2
Background

This chapter covers both basic and more advanced material in network security and automotive Ethernet. The chapter is meant to inform the reader of all the facts needed to better understand the discussions presented in the subsequent chapters in the report.

The experienced reader with knowledge in network basics, network security or automobile technologies may skip parts of the background, though it is advised to at least cover the material for security in vehicles (Section 2.4) and V2G charging (Section 2.3.1).

2.1 Network Basics

2.1.1 Transmission Control Protocol/Internet Protocol Model: TCP/IP Model

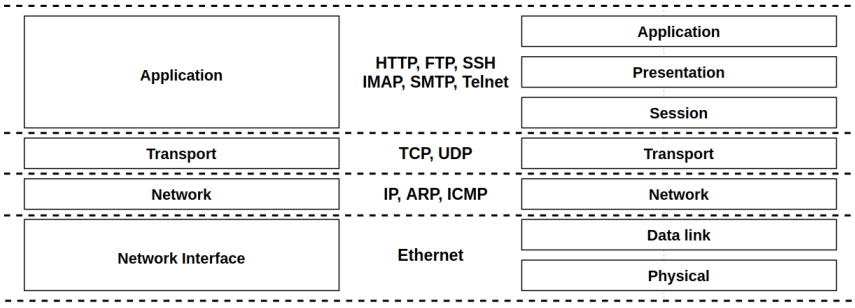

Figure 2.1: The TCP/IP model compared with the OSI model.

In 1979, the International Organization for Standardization published the OSI reference model. This model can be seen as a framework to guide the development of existing and new network standards. In the OSI model, there are seven different layers: application, presentation, session, transport, network, data link and physical. To pass data from one application to another, the data has to pass from the first application layer via the physical layer in order to get to the receiving application layer [11].

In contrast, the TCP/IP model provides a framework for networking protocols and uses a four-layer architecture, which combines the three upper layers and combines the first and second lower layers from the OSI model. The model was developed to provide a mechanism for implementing the internet by the defense advanced research projects agency DARPA [18]. TCP and IP were developed almost hand in hand, creating this widely used model. One can see the comparison between the two models in Figure 2.1. In this report, we commonly refer to the application- or transport layer defined in this model.

Ethernet

Ethernet is a collection of several computer network technologies used to transfer data between devices – defined in the ISO 802.3 specification.

There exist multiple standards for Ethernet, the first commercially available standard was 10BASE5 and it allowed for speeds of 10 Mbit/s. However, the succeeding development of 10BASE-T introduced a full-duplex which makes it possible to send and receive simultaneously [19]. Advances in Ethernet have progressed significantly during the last two decades, the introduction of Fast Ethernet (i.e., 100BASE–X) and Gigabit Ethernet (i.e., 1000BASE–X) have greatly increased performance with speeds of up to 1 Gbit/s.

Ethernet is in its design a connectionless protocol with support for broadcast, multicast, and unicast communication. Typically, all participants are connected to a switch where frames are processed and exchanged between endpoints. A device connected to an Ethernet network uses a media access control (MAC) address to identify itself and other hosts on the network, though, there is no routing based on MAC addresses that enables cross-network communications. Features such as network routing are left to protocols applied on top of Ethernet.

An example of an Ethernet II frame can be seen in Figure 2.2.

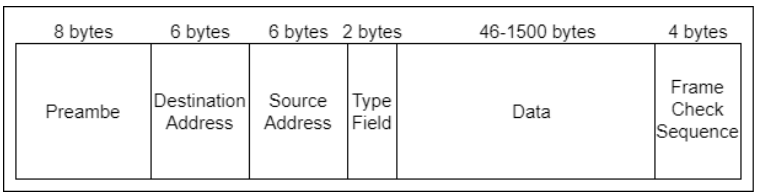

Figure 2.2: An illustration of a frame for Ethernet II.

Transmission Control Protocol: TCP

TCP is a connection-oriented protocol that establishes a connection between two hosts before any data is transmitted. This makes it possible to verify whether packets have been received or not and arrange retransmission in case of loss of packets. TCP provides many functions, e.g., fragmentation of large data that cannot be sent as one packet, data stream reconstruction, socket services for multiple port connections on a remote host, flow control and packet sequencing and reordering. With all these built-in functions, transmission over TCP implies bigger header sizes and longer processing times.

The IP layer handles routing between machines with IP addresses but does not know which process on the machine that should receive the message. Finding the correct process is up to the transport layer and is achieved by using port numbers. Each process on a machine has a port number, for example, HTTP uses port number 80. In Figure 2.3, one can observe that a port is composed of 16 bits [19].

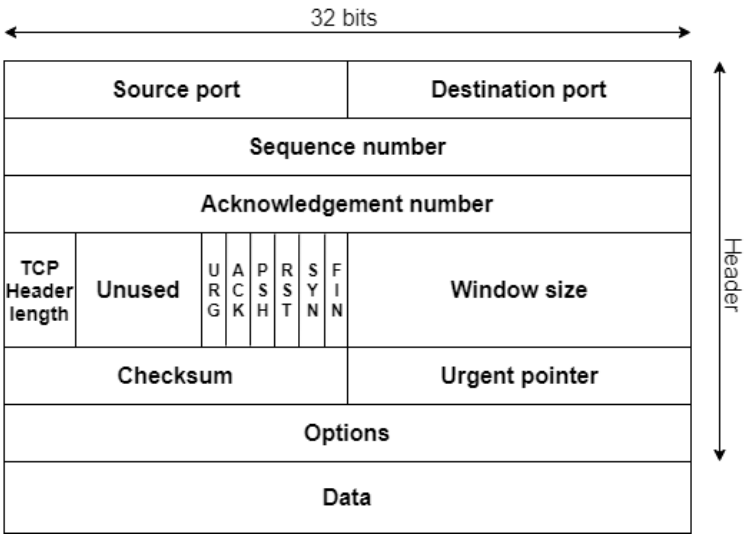

Figure 2.3: TCP header

A TCP full-duplex connection is setup through a three-way handshake and this handshake process is executed at the start of each TCP session. Figure 2.4 visualizes the three-way handshake process. In combination with Figure 2.3, one can see the header fields involved in the handshake.

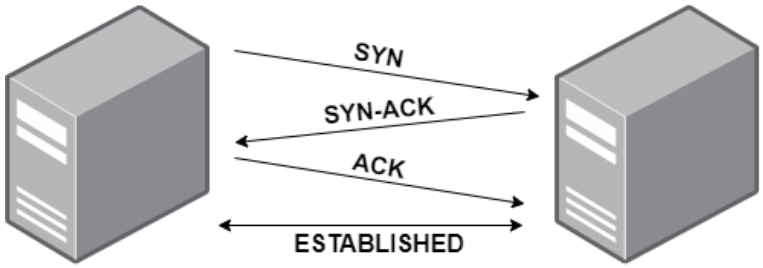

Figure 2.4: TCP connection establishes through a three-way handshake.

User Datagram Protocol: UDP

The second protocol that occupies the transport layer is the user datagram protocol (UDP). UDP is a connectionless transport layer protocol, meaning there are no

handshakes, no acknowledgments, no way to account for lost datagrams, and no flow control. There is no need for a connection to be established before transmitting data. Therefore, it is often described as unreliable compared to TCP. Packets are still sent to sockets or ports, but there is no support for retransmission as for TCP.

To send a UDP datagram there is not a lot of header data required, there are no synchronization or priority parameters, no sequence numbers nor timers and no retransmission of packets. The header is small and the protocol is faster than TCP. Below in Figure 2.5, a UDP header is shown.

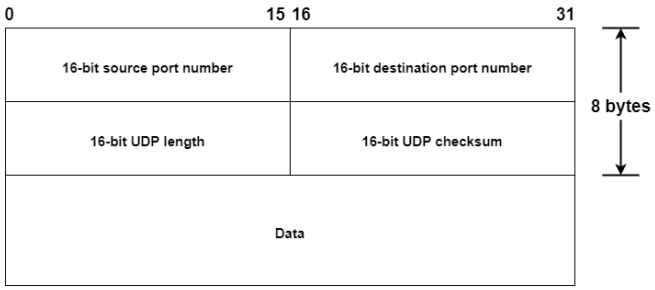

Figure 2.5: UDP header.

Transport Layer Security: TLS

The transport layer security and its predecessor, secure socket layer (SSL), are application protocols, which are designed to provide communication security. The connection is secure because symmetric cryptography is used to encrypt the data transmitted between the two parties. The keys for this symmetric encryption are generated uniquely for each connection and are based on a shared secret that was negotiated at the start of the session. The identity of the communicating parties can be authenticated using public-key cryptography. The connection is also reliable because each message, which is being transmitted, includes a message integrity check using a message authentication code. With this code, one can prevent the alteration of message data during transmission [20].

2.2 The Modern Automotive Industry

Automobiles were seen as mechanical machines until the introduction of electronics. Electronics have been taking over mechanic and hydraulic systems since the '70s, with ECUs taking over the central importance for many new automotive applications and services [13]. ECUs are embedded systems that control electrical subsystems, everything from brake control to engine control to transmission control. The cost of electronics is approximately 35% of the manufacturing costs and as one can see from Figure 2.6 below, the cost-share has been increasing since the beginning of the automobile industry. Furthermore, it is approximated that by 2030 automotive ECUs will account for 50% of the total cost of the car [21].

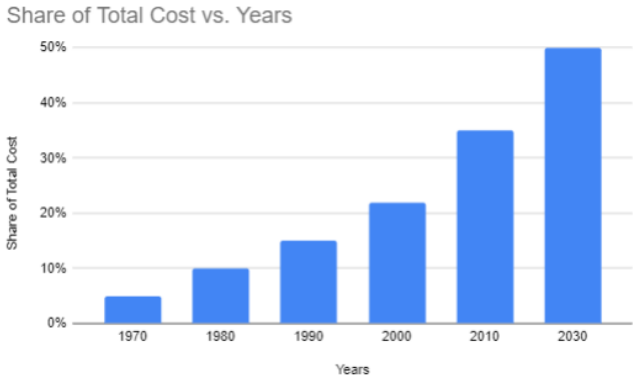

Figure 2.6: Automotive electronics cost as a percentage of total car cost worldwide from 1950 to 2030. Credit to Deloitte for statistics [21].

Back in 2006, the average compact-class vehicle had around 40 ECUs, while the more high-end class vehicle had almost 70 ECUs [22]. Today a modern car has around a hundred built-in or installed ECUs, with luxury cars having as many as 150 ECUs [23]. The interconnection of all these electronic devices creates what is commonly referred to as the internal network of a car.

With the many ECUs and the growing complexity of automobile systems, one can understand why point-to-point communication with cables is not feasible. During the '80s the cabling was becoming increasingly complex, the cables were long, and weighed more than 100 kg [24]. Fieldbuses were introduced where several ECUs could communicate over the same bus, in order to decrease weight and complexity. There are four embedded networks commonly used in the automotive industry: CAN, local interconnect network (LIN), media oriented systems transport (MOST) and FlexRay. These protocols have been around for a long time in the automotive industry, nevertheless, the usage of automotive Ethernet is increasing. However, before diving into why in Section 2.3, here is a brief description of the protocols mentioned above.

2.2.1 The Local Inteconnect Network: LIN

In 1998, the LIN Consortium was founded by five automakers: BMW, Volkswagen Group, Audi, Volvo Cars and Mercedes-Benz [25]. LIN is a serial bus system and serves as a communication infrastructure for low-speed control applications in vehicles, with bitrate support from 1 kbit/s to 20 kbit/s.

The LIN was created to function as a standard and low-cost alternative to CAN for less demanding and noncritical applications, such as door modules and air conditioning systems [26].

2.2.2 The Control Area Network: CAN

The CAN is a widely-used communication fieldbus system, which efficiently supports distributed real-time control, developed by Robert Bosch GmbH in 1983 and standardized in 1994 [11].

Nowadays the CAN is used as a society of automotive engineers network for real-time control in, e.g., the powertrain, body and chassi domains in vehicles. It is a faster protocol, compared to the LIN, supporting bitrates up to 1 Mbit/s. Carrier sense multiple access with collision avoidance is used in the broadcast bus CAN. CAN also implements fixed priority to guarantee that real-time processes receive messages in time. Nevertheless, with the growing number of ECUs connected to the CAN and computing power, the maximum speed of the CAN is not enough for distributed real-time systems [27].

2.2.3 FlexRay

In 2000, the two automotive original equipment manufacturers (OEMs) Daimler-Chrysler and BMW joined together with the two chip producers Motorola and Philips to create the foundation FlexRay Consortium [28]. FlexRay is a fault-tolerant protocol designed for high data rates, advanced control applications [27]. Safety- and time-critical automotive applications were the focus when creating this protocol. FlexRay evolved as an alternative to CAN for tasks which needed better performance and higher data rate [11, 28]. Today FlexRay is used for anti-lock braking, electronic power steering, and vehicle stability functions.

2.2.4 Media Oriented Systems Transport: MOST

The development of the MOST protocol was initiated in 1998 and it is designed for infotainment and media-oriented communication, such as audio, video, navigation and telecommunication systems. MOST is a high-speed protocol with a maximum data rate of 24.8 Mbit/s and it provides support for up to 64 nodes. The high data rates make the MOST bus a good fit for real-time audio and video transmission applications [29]. However, in the advent of Ethernet, the usage of MOST is decreasing.

2.2.5 E/E Architecture

As mentioned previously, the continuous development of the automotive industry is inevitable, and a big reason for this is that the industry is moving more and more towards autonomous systems, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. Both the amount of ECUs and micro-controller units (MCUs) have been and probably will continue increasing, with emphasis on the amount of MCUs [30]. The growing number of control units has led to an evolution of embedded memory, power, and processing systems. In a near-future vehicles will generate and consume approximately 40 TB of data every eight hours of driving, and the average vehicle will produce 4 TB of data a day for merely one hour of driving. Today, cameras alone generate 20-40 Mbit/s and the radar system generates 10-100 kbit/s [31].

Looking at current wired networking technologies in the automotive domain, LIN and CAN are the strongest technologies. The MOST protocol is the most widely used networking technology for multimedia and infotainment systems. In a modern automotive system, there is a need for several communication technologies to co-exist [32].

For a long time, the distributed E/E architecture has been the top choice for automobile manufacturers. In this architecture, each ECU is created for specific functionality. They are grouped and placed in domains based on the relation with each other. The intercommunication of ECUs at different vehicle domains is established via a central gateway [13]. However, with increased functionality, the amount of data sent within the system increases and more bus systems with higher bitrates are added, so that the central gateway architecture simply reaches its limits.

A new architecture is currently underway to the wider market, where the central gateway function is split into domains. These domains are then interconnected by an Ethernet backbone [33]. Nowadays the automotive industry is developing hardware and software systems for domain-centralized E/E architecture, see Figure 1.1. Each set of related functions are grouped under a domain control unit, each of which is interconnected via the central gateway where Ethernet is used as the central communication backbone. Commonly used definitions of vehicle domains are chassis, powertrain, body, and infotainment. With Ethernet as the backbone, the different domains can use different network techniques. For example, LIN is good enough for certain sub-domains, there is no need to replace it with Ethernet and acquire a higher bandwidth than necessary. Because the domain-centralized approach enables layered architecture this system offers greater depth in security [13].

One essential problem with security in cars is the storage limitations, a car has a finite amount of space and to keep the costs down manufacturers cannot always install the largest and best hardware for storage. However, if the security solutions are divided into domains or down to individual ECUs the solutions become both more customized and they separate the memory load as well. OEMs can use smaller storage solutions and possibly decrease the cost of the security solution.

2.3 Automotive Ethernet

The initial problem with implementing Ethernet as the backbone was at the physical layer. The Ethernet transceivers did not fulfill the automotive standards and the physical medium used in vehicles did not fulfill Ethernet requirements. Although, when the single-twisted pair cable was invented, the obstacles of the physical layer seemed possible to overcome [33].

The automotive industry is entering a new era where connectivity and high-speed data transfer is the name of the game. In the Ixia whitepaper, the current state of the industry is stated as follows:

> *"The industry is highly motivated by the significant benefits of expanded bandwidth, reduced labor cost and vehicle weight. Ethernet is*

poised to replace the disparate communication means, or to tie them together in a single backbone." ([34], p.18)

The Ethernet backbone is formed by an Ethernet switch, which in fact maintains point-to-point connections to the domain gateway. The advantage of this architecture is that the gateway processing is split, and at the same time, the domain gateway can perform additional tasks for its local domain [33]. This architecture also prevents message congestion and enables a more customized security solution for each domain [13].

Automotive Ethernet has four main areas of usage, which include advanced driver assistance systems, diagnostics over IP, infotainment and communication backbone [11]. The reason why automotive Ethernet is preferred as the communication network for these services is that it can handle and adapt to the growing bandwidth requirements. In addition to the high data rates Ethernet technology can support, it is a protocol that is stable, long-established and well-understood as a result of the widespread deployment.

Below in Figure 2.7,[1]one can observe an ISO model of ESCRYPT's CycurGATE, which serves as an example of protocols used in the automotive domain. There are the seven different layers from the OSI model and their respective protocol(s). The color grey represents a protocol that communicates over automotive Ethernet.

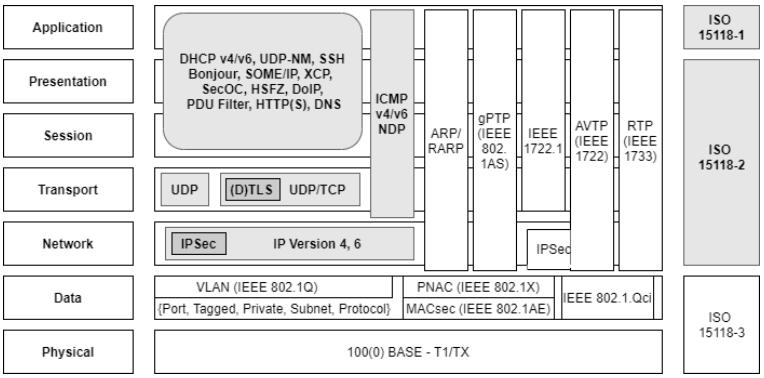

Figure 2.7: An ISO model of ESCRYPT's CycurGATE. Protocols used for vehicle communication, with protocols in grey representing protocols used over automotive Ethernet [35].

In the future, automotive Ethernet will expectantly form the foundation upon which all in-vehicle communications will occur, resulting in a network where all protocols and data formats will be consistent. Consequently, the network will be scalable [36].

[1]©ESCRYPT, used with permission.

2.3.1 Vehicle to Charging Grid: V2G

There are many factors which have driven the automotive industry towards electric vehicles (EVs), not only the fact that ECUs weigh less and enable more services, it is also due to political and environmental factors the industry has taken steps to move away from fossil fuel-driven vehicles [37]. Furthermore, the improvements in EV technology have changed the outlook for the automotive industry.

EVs are considered promising candidates to replace fossil fuel-powered vehicles. Not only do they have the potential to lead to cleaner transportation, but can also provide electric storage capabilities for other applications, such as V2G, Vehicle-to-Home, Vehicle-to-Load, V2I and V2V [38]. This is why EVs are such an important part of the IoT industry.

V2G technology enables bi-directional sharing of electricity between EVs and the electric power grid. The technology turns each vehicle into a power storage system, increasing power reliability and the amount of renewable energy available to the grid during peak power usage [39]. There are 50 V2G projects around the world trying to figure out how to make it work for all participants — EV owners, EV utilities, auto OEMs and EV charging developers [40].

ISO 15118

In Sweden, there are 3670 public charging stations with a total of 17097 charging plugs [41]. Whereas most of the charging is approximated to be done at home, where the projected influx of EVs could increase stress on the grid during peak demand hours of 6-9 a.m. and 5-9 p.m [40].

In 2010, the creation of the ISO 15118 standard for V2G communication was initiated. The international standard outlines the digital communication protocol that an EV and charging station should use to recharge the EV's battery. This includes wired (AC and DC) and wireless charging applications.

The smart charging mechanism built into the standard enables the electrical grid to match the capacity of the grid with the energy demand for the growing number of EVs that connect to the electrical grid. ISO 15118 enables bi-directional energy transfer in order to realize V2G applications by feeding energy from the EV back to the grid when needed. Additionally, the standard makes it possible to integrate EVs into the smart grid. A smart grid is an electrical grid that interconnects energy producers, consumers, and grid components like transformers by means of information and communication technology, see example in Figure 2.8.[2]

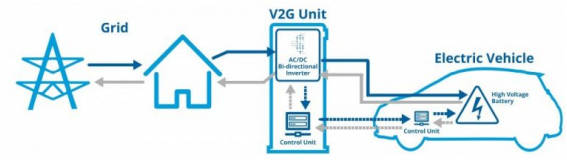

Figure 2.8: Simple overview over communication between EV, V2G unit, home and power grid [40].

[2] ©Cenex, non-commercial use in Copyright, Designs and Patents Act 1988.

Furthermore, this allows the EV and charging station to exchange information dynamically and a proper charging schedule can be negotiated. It is of great importance that the EVs operate in a grid-friendly manner, which means that the charging system supports the charging of multiple vehicles at the same time – while preventing the grid from overloading. To ensure that these smart charging applications calculate an individual charging schedule for each EV, they use the information available about the state of the electrical grid, the energy demand of each EV, and the departure time and driving range of each driver [42]. In this report, the charging system is defined as the supply equipment communication controller (SECC) and the charging system deployed by the EV is called the EV communication controller (EVCC), based on definitions from the ISO 15118 specification.

As one might have noticed previously in Figure 2.7, there are several parts of the ISO 15118 standard, in total there are eight specifications of the standard that all focus on different parts of the communication. See Figure 2.9 below.

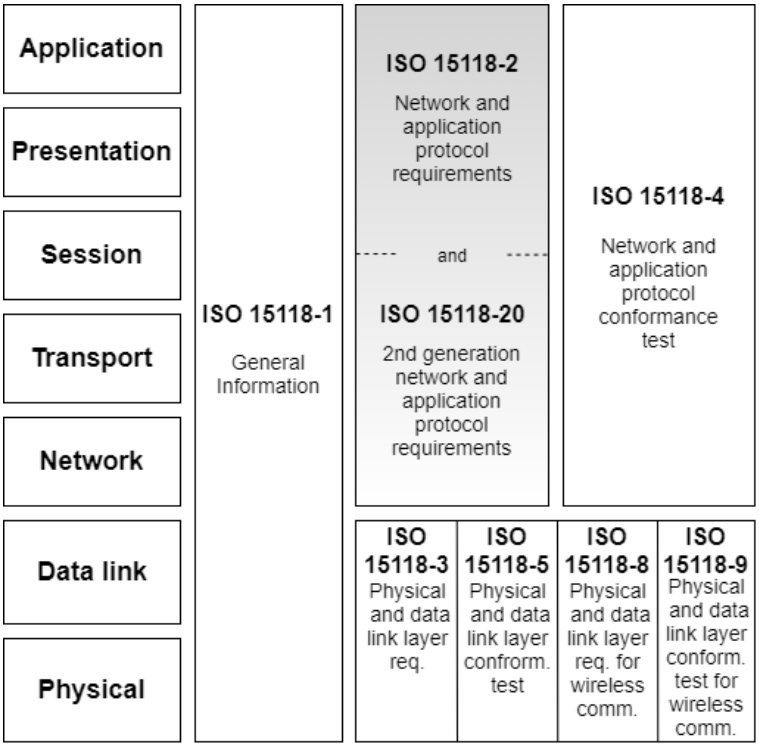

Figure 2.9: The eight different specifications within the ISO 15118 family. Source Wikipedia [43].

In this paper, we focus on ISO 15118-2: Network and application protocol requirements. This specification specifies the V2G communication, including the different message types with their respective contents and flow charts over the order messages should arrive to the client or server. All the different message

types can be found in Appendix A.

Before going into specifics about how a session works, it is important to mention how data is encapsulated. In the ISO 15118 specification [16], the V2G transfer protocol (V2GTP) is the transport protocol for V2G data transfers (used at the session layer) and can be considered the HTTP of V2G communications. The V2GTP frame contains a header and a payload field. The header has a length of eight bytes with four different parameters: protocol version, inverse protocol version, message type, and payload length, while the V2GTP payload field represents the actual data to be transferred. Regardless of which message type is used, a V2G message will always be encapsulated according to this format.

A session starts with the client multicasting its demand for charging, a so-called SECC discovery protocol (SDP) request is sent via UDP. Through the SDP request-response pair a security parameter, which indicates whether TCP or TLS should be used for the session, is set. Following this, a server initiates the TCP or TLS session setup with a handshake [16].

The messages sent after the SDP request-response pair use the XML/EXI data representation format. Efficient XML interchange (EXI) is a compact representation of the extensible markup language (XML). Using a relatively simple algorithm, which is amenable to fast and compact implementation, and a small set of datatype representations, EXI reliably produces efficient encodings of XML event streams [44].

V2G communication can be AC or DC specific and there are message types that are only sent for AC respective DC communication. It is stated in Appendix A which messages are AC or DC-specific [16].

There is also the matter of authentication of user and payment method. One can use external identification means (EIM), the traditional method of payment with card or cash. In addition to EIM, the ISO 15118 specification introduces a feature called plug & charge (PnC) [42]. PnC both enables the EV to automatically identify itself to the charging station and to gain authorized access to the energy it needs to recharge. Moreover, PnC deploys several cryptographic mechanisms to ensure that the communication maintains confidentiality, integrity, and authenticity of all exchanged data.

Security in ISO 15118

The security of the ISO 15118 specification is based on the digital certificates and public key infrastructure made available through the PnC feature. The ISO 15118 specification follows a common hybrid approach, using asymmetric-key algorithms to create and verify digital signatures, in addition to agreeing upon a symmetric key. The symmetric key can then be used to encrypt and decrypt all messages during a charging session with a symmetric key algorithm. In order to achieve a successful PnC session, both the EV and the charging system must support the three main security characteristics: confidentiality, integrity, and authenticity [45].

First off, the messages must be encrypted to ensure that no third party or malicious actor will be able to eavesdrop on the communication. Secondly, it is important that if a message has been modified, it is detected in order to attain data integrity. Lastly, to ensure that the other party is who they claim to be,

the EV and charging system must verify its communication counterpart. To make sure that these three security characteristics are fulfilled, a TLS session is used to secure the V2G communication.

For the security of the transport layer, the communicated messages between EVCC and SECC are encrypted using a symmetric key negotiated during the TLS key negotiation phase. The key agreement protocol used to agree upon a shared symmetric TLS session key is called elliptic curve Diffie-Hellman (ECDH) [45]. ECDH is a key agreement protocol that allows two parties, each having a private/public key pair (asymmetric), to establish a shared secret over an insecure channel [46]. The symmetric key is used to encrypt and decrypt the messages using a block cipher suite called AES_128_CBC_SHA256. To decrypt the sender's message the receiver needs the symmetric key. Therefore, a message cannot be decrypted if one does not have access to the key, ensuring confidentiality [16].

Similar to the TLS cipher suite, the ISO 15118 specification provides another key agreement scheme that establishes an encryption key for certain messages with sensitive data. This encryption key is used to provide message security – mainly utilized to encrypt private keys sent between the EVCC and another third party. In this scheme, the ephemeral-static ECDH protocol is used, which allows for the creation of a shared session key from a static public key already present inside the vehicle. This is further explained in Annex G in the ISO 15118 specification [16].

Verifying authenticity and data integrity are features that are realized through asymmetric cryptography, using a key pair composed of a private/public key. The private key must be kept secret and is only used by the entity to which it belongs in order to create digital signatures. The public key is distributed to peers in the same ecosystem and used to verify the signature that was created with the associated private key. The public-key cryptography is used to create and verify signatures in order to control the authenticity of the sender and the integrity of the received message. The cryptographic algorithm used for this is the elliptic curve digital signature algorithm (ECDSA) [45]. In addition to the security measurements described by the ISO 15118 specification, there are also security methods built into TLS. For example, the message authentication code algorithm, which efficiently verifies the authenticity of the message given a symmetric key and a tag. The message is accepted when the message and tag are not tampered with or forged, otherwise, the message is rejected.

The ISO 15118 specification outlines an ecosystem of digital certificates that need to be in place for PnC to work. This is where public key infrastructures (PKIs) come into play. A PKI defines an infrastructure including different entities, policies, and devices that can manage, distribute and revoke digital certificates based on asymmetric cryptography. In a PKI system, public keys are associated with an identity through a process of registration and issuance of certificates. The connection is established both at and by a certificate authority (CA) [47]. These CAs manage the creation, storage, distribution, and revocation of digital certificates. A digital certificate is an electronic document used to verify that a public key belongs to an authorized party. Therefore, it is also known as a public key certificate.

In the case of the ISO 15118 specification, there are SECC certificates that are signed and placed in the charging station by a certified charging operator. The

EVCC uses these to authenticate the SECC over TLS. There are also contract certificates in the EVCC to authenticate against an SECC and/or secondary actor, these are used for signing in the V2G session. Lastly, there are V2G root certificates and potential Sub-CA certificates that certify the aforementioned SECC certificates and contract certificates. These are all essential to maintaining trust and security in the PnC process.

One of the biggest improvements with the PnC feature is that the driver does not need to do anything in order to authenticate the vehicle, beyond plugging the charging cable into the vehicle and charging station. There is neither a need for entering a credit card, nor scanning a QR code, nor opening an app. This makes the ISO 15118 specification very user-friendly [42].

2.4 Security in Vehicles

Confidentiality means that one should ensure that communication between authorized parties is secret and unauthorized access to data transmission in vehicles should be impossible. It is also essential to make sure that unauthorized modification of data is infeasible or at minimum detected. The prospect of unauthorized modification to system assets or actual transmitted data is obviously bad and includes writing, changing and deleting messages. It is also important that unauthorized modifications of any hardware or software in the vehicle are either infeasible or at least detectable. The purpose of authentication and identification is to establish and assure that the origin of the message is correctly identified. In a vehicle, there is a great need for this, because only authenticated and identifiable assets should be able to communicate with a certain ECU [48]. Security in vehicles is becoming a serious matter in the car manufacturing industry and there is engagement from a legislative perspective as well (e.g., the SPY Car Act in the US [49]). With autonomous driving and integration of V2V and V2I features gaining momentum, new security vulnerabilities are inevitable. This places the responsibility on the manufacturers to use proper and proven security architecture in their vehicles.

When considering safety in vehicles, one of the most prominent aspects is the concern of passenger safety. When Miller and Valasek managed to gain remote control over a 2014 Jeep Cherokee, they show that a breach of the internal network can have fatal consequences [50, 51]. With their remote exploitation, they sent CAN messages to activate blinkers, lock the car, prompt remote steering and kill the engine. Whereas safety is the number one priority, privacy and integrity concerns are expected to be upheld as well, just as it is for modern websites today. As a practical example, no one should be able to track a driver by accessing GPS data or eavesdrop on a hands-free phone call.

Table 2.1 below shows the three general different groups of possible system intruders of a vehicle: first-hand user, such as the car owner; car manufacturers, car mechanics and garage personnel; different unauthorized parties, such as an institution [6].

The first two attacker groups have full physical access to all transmission media and respective affected devices of the automotive network. As the car owner

Table 2.1: Possible attackers or persons performing unauthorized modifications [52].

Attackers	Knowledge	Physical Access
Owner	Varied (mostly low)	Full
OEMs, mechanics etc.	High	Full
Unauthorized third party	Varied (can be high)	Limited or none

normally has only low theoretical and technical capabilities, the bigger risk factor is the second group. The people in this group might have both adequate background knowledge and the technical tools for completing a successful intrusion attack, which might result in permanent damages to both software and hardware in the vehicle. Possible motivations for an unauthorized attack from a third party might be to acquire private passenger information, through phone tapping or data theft [6].

2.4.1 Challenges

The majority of the foreseeable security issue arise as by-products of interconnecting the various vehicle bus systems. LIN, CAN or MOST are bus systems interconnected, which basically may access and send messages to any other ECU. Furthermore, a single compromised bus system endangers the entire in-vehicle communication network.

In combination with the increasing integration with outer networks, such as V2V, V2G, and V2I, future attacks on automotive communication systems can be accomplished without any physical contact, just by passing a car or via cellular phone from almost anywhere in the world [6].

Lightweight Devices

Modern communication over the Internet implements security solutions for authentication, integrity checks and provide confidentiality with protocols and algorithms based on cryptography. In many cases, these security properties can be introduced quite effortless and they introduce a limited amount of latency with today's modern computing power. However, consider a modern car network; consisting of multiple smaller devices (switches, MCUs, ECUs, sensors, etc.) that operate on lower wattage with lower rates of processing power and memory resources available. If one is to apply a cryptographic protocol scheme or another security measurement in the automotive setting, the selection of algorithms used needs to be carefully considered to not strain the performance of these lightweight devices.

Access Mediums

In-vehicle security is not only limited to remote threats but also needs to address physical access aspects. Hence, even with physical access to the car, it should not be possible to compromise its internal systems.

Connectivity

One could argue connectivity is a double-edged sword in the context of in-vehicle security. As car network architecture is growing, with the introduction of V2V, V2G, and V2I concepts there will be new useful features available. For instance, vehicles will be able to find available parking spaces at a parking lot, share information about immediate dangers on the road or support autonomous driving systems with traffic information. With all these benefits, there will be trade-offs between conveniences and the potential attack surfaces, introduced for exploitation or violation of privacy. It is therefore crucial that independent components and sub-domains, as well as the entire internal network of the car, are secured for this new paradigm of connectivity.

2.4.2 Firewall

A firewall is a system used to prevent unauthorized access between networks, primarily screening traffic entering or exiting a private network. It is either implemented in hardware or software depending on the network, or in some cases, a combination of both. Firewalls have been used as the first barrier of entry for attacks over a long period of time, and they are now being integrated into modern vehicles. A firewall typically has one or several of the following functionalities (as mentioned by Cisco in [53]):

- Packet filtering: The firewall inspects each and every packet entering or leaving the network. Based on a certain set of rules, it accepts or rejects packets.
- Stateful filtering: A stateful firewall remembers the state of previous communications between parties. Thus, it can filter based on expected state or connection properties as communication is taking place.
- Operating as a proxy: The firewall acts as a proxy for applications by handling e.g., connection establishment in their place.
- Application layer filtering: Newer generations of firewalls can inspect certain application protocols such as HTTP, FTP or DNS.

Stateful Inspection

Stateful packet inspection is a type of inspection method done in software and is carried out with shallow packet inspection into the transport layer. It recalls the state of previous communications with an entry into a table, this entry will serve as a means to compare incoming packets, belonging to the same session, allowing for faster processing of arriving packets. A typical characteristic of a stateful filter is high latency at the start of reception – since no previous matching has been done in a clean state – but it will quickly become effective as entries are added to the table. A downside to using stateful packet filters are the memory requirements, maintaining the state of active connections on the network [54].

2.4.3 Intrusion Detection System: IDS

Intrusion detection has been a practiced method in various industries since the early days of network security and it is now being applied for automotive use cases. An IDS is commonly applied as a dedicated physical device on the network or as a software program. Furthermore, it is utilized as a second-line of defense in a network security architecture, often strategically placed behind a firewall to detect ongoing attacks. An example of IDS placement in a network topology can be seen in Figure 2.10.

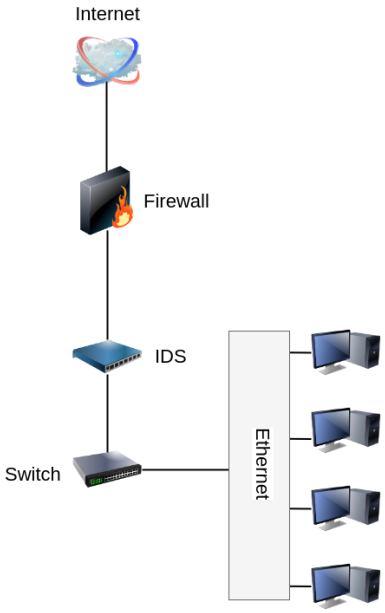

Figure 2.10: IDS placement inside a network.

An IDS differs from a firewall, whose sole purpose is to screen traffic between networks and prevent intrusions, instead, an IDS monitors and detects attacks or threats present inside the network. Its goal is not to prevent malicious threats observed on the network (unless it possesses prevention functionalities), but rather it takes a more proactive position by alerting and logging information [55]. An IDS often relies on extensive DPI, which can screen packets on an application-level basis. Common functionalities of an IDS include, for example, updating the system for future attacks, analysis of potential attack patterns on the network, alerting other security mechanisms in place or generating warnings for network administrators.

There are generally two methods of IDSs: host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS).

An HIDS is an IDS system that is installed on a host machine and monitors dynamic behavior on the host system. It typically looks at system audit logs and tries to detect malicious activity. For instance, an HIDS can check the integrity of

files or check against known signature patterns in a database.

A NIDS on the other hand, is referring to an IDS residing on the network. It acts as an inline sniffer, sampling packets as they arrive at the network. Typically, you would see a NIDS incorporated into a network switch or central gateway, to be able to process all network traffic. Consequently, a NIDS device must allow for high-speed processing of packets to avoid introducing bottlenecks in the network.

There are different detection methods for how to detect anomalous behavior in an IDS. Namely, signature-, anomaly- and specification-based methods. One approach is to use one of these features to secure a system, or they could be used in unison as a hybrid IDS implementation.

Signature-based Detection

A signature-based detection engine relies on pattern matching to detect known malicious behavior in sequences of bytes or subsets of malicious instructions. To realize a signature-based approach one has to define unique rule sets for a specific network. For instance, detecting and not responding to an `ICMP ECHO` request allows a host to not be discovered by a network mapper (`nmap`) scan. The command `nmap` is used to discover hosts and services on a computer network by sending packets and analyzing the responses. It can be used to perform port scan- or port sweep attacks.

In signature-based methods pattern matching algorithms are the central component determining the performance. Since a software-based IDS operates within the boundaries of a typical application, there are subjects of concern regarding CPU usage, memory, and power consumption. There have been multiple studies on enhanced algorithms, some examples are Myers algorithm [56] and Wu-Manber [57]. Nevertheless, which algorithm should be applied depends on the network requirements and accessible hardware on a given system.

A clear advantage of signature-based implementations is the ability to use a multitude of signature patterns that can protect against already known attack patterns, this safely secures the system against discovered threats and gives the incentive to share rules in security communities. Although, the signature-based method has its downsides as well, mainly since it cannot protect against zero-day attacks (attacks on vulnerabilities that have not been patched or made public) and signatures need to be updated continuously to keep up with new upcoming threats.

Anomaly-based Detection

Anomaly-based methods take another approach to detect threats and they find malicious traffic through discrepancies from normal behavior instead. Conversely, from a signature-based method, an anomaly-based IDS tries to detect previously unknown attacks and enables protection against future attack vectors. Although, there are negative aspects of anomaly-based detection methods. One is the tendency for higher false positive rates [58].

For statistical approaches, the goal is to find a baseline for the characteristics of communications over the network such as type of packets sent, frequency of packets, changes in payload size or data content and sequences of packets. After

reaching a baseline, the IDS then classifies traffic as normal or abnormal, sending out an alert if the system receives any suspicious packets. For example, an anomaly-based system can detect if a flooding or replay attack is being performed if it keeps track of how often a message type is usually sent through the network. It may also detect whether packets are being dropped based on the same frequency check.

As mentioned, anomaly-based approaches try to categorize a baseline and find outliers that are new unknown threats, such challenges can also be addressed with the use of machine learning. Yihunie et al. [59] presented an analysis using the NSL-KDD dataset, evaluating the performance of several machine learning models including stochastic gradient descent, random forests and support vector machine. Their work highlights valuable comparisons between these models where random forests outperformed the other classifiers. A random forests classifier works by training many decision trees on random subsets of the features, then averaging out their predictions.

Specification-based Detection

Another method for detection in an IDS is a specification-based approach, where deviations from set specification requirements are used to detect malicious packets. Usually, this is achieved by setting up explicit rules for each protocol's specification and thereby finding deviations from normal behavior. Properties such as source and destination correspondence, data volume, message times or header values are all examples of metrics that could be taken into consideration. Another more general example could be not allowing processes certain sequences of system calls [60]. Differing from a pure anomaly-based method, it does not tend to suffer from high false positive rates. In part since its rule-based principles yield deterministic performance similar to that of signature-based methods.

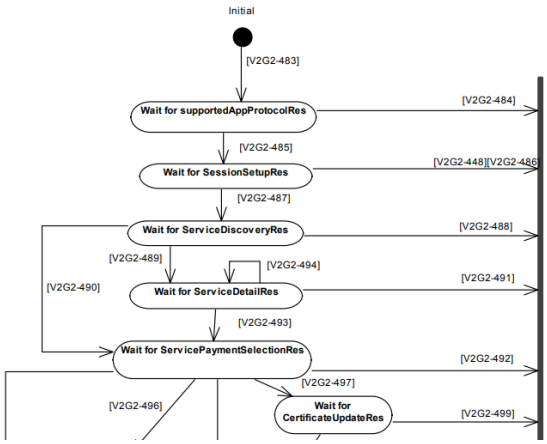

Figure 2.11: A snippet of V2G communication from the ISO 15118 specification.

For example, the ISO 15118 standard specifies a flow chart where one can follow the exact types of messages that should be following each other. Therefore, a specification-based IDS might include some sort of a decision tree ensuring that

the V2G communication is following its specification. See Figure 2.11 for a small snippet of a V2G session flowchart.

SNORT

SNORT is an open source NIDS solution, which can perform real-time traffic analysis and logging of packets on IP-based networks [61]. SNORT uses a signature-database to match rules in the detection engine, where a signature is defined as distinctive marks or characteristics being present in an exploit [62]. These rules are continuously updated by its community to stay on top of current vulnerabilities.

2.4.4 Deep Packet Inspection: DPI

With the increase of data being sent over the internet, depending on the application there is a need to screen packets for security threats. DPI enables inspection of packets in the higher layers of the OSI stack, effectively inspecting beyond protocol headers – allowing for greater control over which packets can flow through the network [63]. Deep inspection is carried out with a DPI engine in real-time, which classifies packets either as normal or attack packets. Internet service providers commonly deploy DPI for filtering content, targeted advertising and blocking of undesired web traffic. Furthermore, DPI enables prioritizing of packets that are of interest to the automotive domain. This further complements the real-time requirements present in a car network.

Another related use of DPI is shallow packet inspection, which inspects headers up to the transport layer. As mentioned in Section 2.4.2 about firewalls, this type of inspection is used to perform stateful packet inspection. A comparison between each inspection method is illustrated in Figure 2.12.

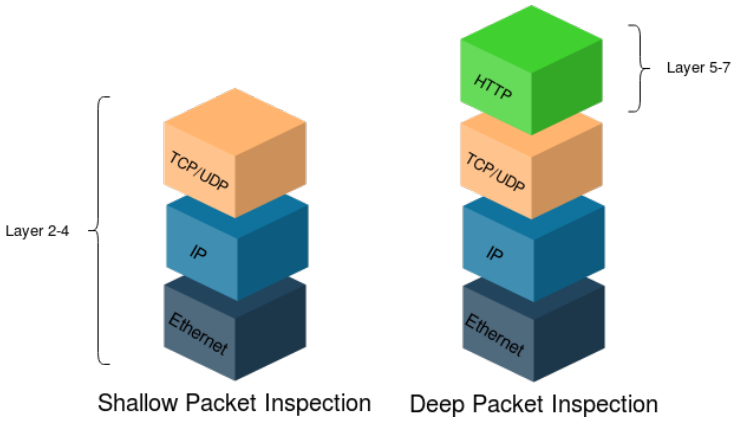

Figure 2.12: Visual comparison of inspection range between shallow and deep packet inspection.

Firewalls and IDSs are heavily reliant on inspecting packets to various degrees, to match packet information against entries or signatures.

There exist two types of DPI: hardware and software solutions. Under normal circumstances, regular switches or processors have limited capabilities to host a DPI system due to a lacking performance under heavy loads of traffic. Hardware DPI uses specialized hardware to accelerate the inspection engine's processing time, which enables wire-speed inspection of packets. Since hardware-based solutions have a higher price tag, larger corporations often use these. The nDPI library [64] and Solarwinds NPM [65] are, on the other hand, practical examples of software-based DPI. Instead of relying on hardware, these products apply a software-based DPI engine for inspecting packets on an application level.

Chapter 3
Method

This chapter describes the methodology, which has been used in order to receive the results in Chapter 6 and helped us draw our conclusions. The chapter includes a description of hardware and software used in order to implement our IDS. Furthermore, test and performance evaluation methods are also described.

3.1 General Research

In the beginning, we did not quite know what to anticipate or expect of this project with no previous experience in embedded security. Upon meeting Bosch and ESCRYPT a couple of times the master's thesis took form. We then started to outline the scope and conduct general research. There had been a student completing his master's thesis at ESCRYPT the previous semester [13] and our project was meant to be a continuation of that. We used the resulting report as our base in the beginning. Thereafter, trying to collect as much information about implementations of IDSs, network systems for in-vehicle communication and automotive security. The essential research papers used as basis for e.g., establishing requirements, choosing the different components and selecting features for the IDS are presented in Section 1.6.

After some general research, we realized the sizable challenge of continuing the previous work and implementing a complete firewall and IDPS solution. Our supervisors recommended that we narrow our scope down to only focus on one protocol, namely ISO 15118. With access to the specification for ISO 15118, we strove to understand how the different actors in a V2G communication session operate, what the different message types were and which functionalities they provide. Thereby, we were able to gain a more in-depth view of how an in-vehicle IDS would operate and which features were of importance.

3.2 Establish Requirements and Select a Concept

After we had gained insight into how an IDS works and what requirements are associated with automotive networks, we established the requirements for an in-vehicle IDS from our research. We settled on implementing two detection methods for our IDS: specification-based and anomaly-based detection. The requirements, motivations, and decisions related to the IDS are discussed in Chapter 4. Through

this process, we answer the question of *Which requirements are associated with IDS in the context of automotive Ethernet?* Also, we further address the matter of DPI in in-vehicle networks, and try to answer the question *Which requirements are associated with DPI for automotive Ethernet?*

Furthermore, we explain and motivate a proposed design concept for a V2G IDS in Section 4.4. By motivating our design choice, we reach the goal to *Analyse which IDS features would fit the automotive domain from security and resource constraint point of view*, as well as the goal to *Propose a concept on how some of those features would be designed and implemented.*

To select the specific feature set for the IDS, a security analysis of the V2G protocol in the ISO 15118 specification was conducted in Section 4.3. Additionally, outlining which security practices were already in place in the protocol. Following this, we defined potential security threats and weaknesses in the protocol which became inspiration for some of the features implemented.

3.3 Implementation

To achieve the defined goal *Implement a Proof of Concept for at least one attack vector*, we developed a V2G IDS presented in Chapter 5.

The previous research on how to design an IDS served as the basis for the architecture and features to be implemented. The IDS was exclusively written in C since it is generally the preferred programming language for MCUs in vehicles.

3.3.1 Hardware Setup

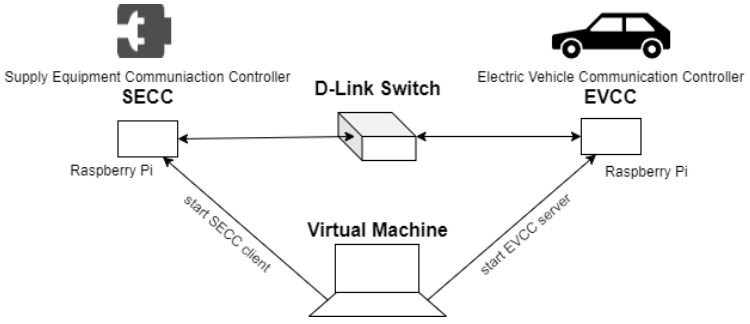

Figure 3.1: Hardware setup for the project development environment.

Two Raspberry Pi 3 Model B with Raspbian OS (version 4.19) was used to simulate the traffic, one acting as the SECC server, the other as the EVCC client. The model B has a Quad Core 1.2GHz Broadcom BCM2837 64-bit CPU with 1GB RAM and support for 100 BASE Ethernet [66]. The reason for choosing this setup was partly due to the lack of dedicated V2G hardware, but mainly it allowed us to develop on a Linux-based system. Consequently, giving us great flexibility to port the project

to any Linux-based environment of our choice and test the software continuously on the virtual machines used for development.

The switch used was D-Link's 8-port fast Ethernet easy desktop switch (model GO-SW-8E). This was used to set up the local network between the Raspberry Pis, allowing the two of them to act as a server and a client for V2G sessions. See Figure 3.1 for an illustration of the setup.

3.3.2 Software

In the developing stages, there was a need for data examples that could be used for training and testing. Without either the time or knowledge to implement the protocol ourselves, the two open source projects RISEV2G and OpenV2G were utilized to quickly progress in the development stage of the project. Wireshark was used to collect examples of V2G sessions in `pcap` format.

Wireshark

Wireshark is a free software for taking captures of network traffic [67]. Wireshark can be used to sniff, inspect, filter and analyze network protocols in real-time along with support for offline screening of packet captures. When developing network applications (such as an IDS) Wireshark is a powerful tool for inspecting inputs and outputs of the application, not to mention the ability to capture generated traffic scenarios. Wireshark was important to us namely because it allowed us to inspect the V2G packets being sent between the Raspberry Pis. Which made it possible to decide the order, size, and frequency of the messages. We also used Wireshark in order to capture and store entire V2G sessions as `pcap` files, which could be used as examples of both normal and attack data traffic.

RISEV2G

To inspect and analyze V2G sessions, the open source V2G software RISEV2G library is a promising option with comprehensive examples written in Java [68]. The setup was fairly easy and it was even possible to enable authentic TLS communication with generated dummy certificates on the EVCC and the SECC. RISEV2G lets one start a client and server application that exchanges messages over TCP or TLS according to the ISO 15118 specification, also their website offers tutorials and other valuable information for anyone interested in learning more about ISO 15118.

We used RISEV2G in the beginning to get a better understanding of the protocol. The library also allowed us to collect normal data examples in `pcap` format through Wireshark.

OpenV2G

OpenV2G is another example of an open source adaptation of ISO 15118 [69]. Different from RISEV2G the OpenV2G software includes dedicated codec functionality for efficient conversion between XML documents and EXI compressed format and it is written in C. With the codec API one can integrate parsing of V2G data in virtually any setting which makes it a valuable tool for the purposes of this project.

libpcap

Another important C library used in the project is `libpcap` (full name of the package is `libpcap0.8-dev`), specifically we used version 1.8.1 to achieve compatibility with the corresponding Debian 10 distribution deployed on the Raspberry Pi. Its API provides function calls for reading of `pcap` files as well as live capturing of packets with filtering capabilities.

Virtualbox

For development, we used Virtualbox to run VMs on our PCs. To test our software we used a virtual machine with Ubuntu 18.04.3 LTS (Bionic).

3.4 Evaluation Framework

To test and evaluate our implementation, a series of metrics were collected during a real-time session, where different attack scenarios were deployed on top of the session. The attack client generates data for which packets contain threats and will be compared against detections made by the IDS. After conducting the tests we were able to answer the question *What functions of IDS are valuable from the perspective of automotive Ethernet?*

3.4.1 Performance Evaluation

An IDS can be seen as a binary classification method and therefore it is suitable to evaluate it accordingly. To evaluate the performance of the V2G IDS we based our evaluation metrics on the recent work of Stachowski et al. [15]. In their work, they present an assessment approach where quantitative and qualitative metrics are recorded by rendering the IDS as a binary classifier. The metrics chosen for evaluating and testing the IDS can be seen in Table 3.1 and are based on their report. The detection speed defined in [15] was redefined as classification speed and further include normal packet classifications. Our motivation for this is that only measuring how long time it takes to detect attacks does not capture the performance for the majority of packets – which are normal packets (negatives). We further expanded the classification speed to also include best, worst and average speeds, which allows for further insight into the speed dynamics observed in the IDS. Finally, we switched out accuracy for balanced accuracy to evaluate the overall accuracy of our model, weighing both correct true positive and true negative classifications rates equally. The balanced accuracy metric is better than the normal accuracy for imbalanced data sets. We had imbalanced data sets in our implementation, as there was a significantly larger amount of normal packets compared to malicious packets [70].

As previously mentioned, the IDS can be seen as a binary classification system where packets are either classified as a positive (a detection) or a negative. A positive classification implies an attack is ongoing in the system while a negative represents all other possible events (including normal traffic and erroneous messages).

Table 3.1: Table of metrics taken for evaluation.

Metric	Description
False Negative Rate	Rate of incorrectly identified attacks when attacks were ongoing.
False Positive Rate	Rate of falsely identified attacks when no attacks were present (false alarms).
True Positive Rate (Recall)	Fraction of actual attacks alerted by the IDS.
Precision	What portion of identified attacks were true attacks.
Balanced Accuracy	Balanced accuracy metric determining fraction of correctly classified packets.
F-score	Metric accounting for precision/recall. Varies depending on β selected.
Classification speed	How long time it takes to classify a packet as normal or issue an alert after an attack has started.

The confusion matrix shown in Figure 3.2 depicts the four outcomes of a binary classification system, which can be used to calculate the metrics in Table 3.1 above. These four outcomes are true negatives, false negatives, true positives, and false positives.

		Predicted Class	
		Normal	Attack
Actual Class	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

Figure 3.2: The confusion matrix for binary classification in an IDS [71].

Negative and Positive Rates

True positive rate (TPR) or recall is defined as the correctly identified attacks against the system. This metric is used to calculate the balanced accuracy and F-score. Therefore, a classifier that produces no false negatives will attain a recall of 1.0.

$$TPR\,(Recall) = \frac{TP}{TP + FN}$$

Moreover, the false positive rate (FPR) defines the incorrectly identified attacks against the system; normal packets being classified as attacks. The optimal FPR would be a rate of 0 with no false positives.

$$FPR = \frac{FP}{FP + TN}$$

The false negative rate (FNR) shows the rate of incorrectly rejected attacks against the system; attacks being identified as normal packets. Similar to the FPR, the FNR best case performance has a rate of 0 with no false negatives.

$$FNR = \frac{FN}{FN + TP}$$

Precision

Precision is defined as the proportion of positive identifications (attacks) that were correctly detected. Hence, deciding how many positives were actually correct. A perfect precision score is 1.0 with no false positives. An important observation is that the precision does only account for positive classifications and does not consider negatives.

$$Precision = \frac{TP}{TP + FP}$$

Precision together with recall (TPR) are both important metrics to measure the effectiveness of a classifier. The performance of a classifier considering precision and recall typically shows a trade-off between these two metrics. Where the increase of one will dampen the other.

F-score

The F-score is a measurement of a test's accuracy, moreover, its the harmonic mean of precision and recall. As mentioned before, models normally experience a trade-off between precision and recall, and F-score tries to combine both these metrics into one score. The F-score is also a weighted metric that can be tuned by changing the β variable. A β value of 1 (called f_1-score) balances the score evenly between precision and recall, while a value of 0.5 (known as $f_{0.5}$-score) would weigh precision higher than recall. As a result, depending on if a supplier or OEM prefers higher recall or precision, the F-score can be weighted towards

their preferred configuration of an IDS. The perfect F-score that can be attained is 1.0.

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

Balanced Accuracy

Balanced accuracy can be defined as the fraction of predictions the IDS got right. It can further be seen as a more conclusive metric displaying how accurate the overall performance of a classifier is – considering both true positive and true negative classification rates. This is important since the balanced accuracy compensates for imbalanced data sets (which normal accuracy does not) and yields more reflective results as to how the model is actually performing. A score of 1.0 is a perfect result for this accuracy metric.

$$Balanced\,Accuracy = \frac{TPR + (1 - FPR)}{2}$$

3.4.2 Testing Different Configurations

When implementing the anomaly-based detection we had a lot of issues figuring out which statistical approach we should use. After some trials and errors, we settled on using the minimum and maximum value from expected data. The frequency, payload length and timeout for request-response pairs were checked by comparing them to expected normal V2G session data. All values from the V2G session examples were considered to be normal, and we could use the lowest possible values for minimum and the highest possible values for maximum. For example, in the session examples, a `ChargeParameterDiscovery` request occurred once or twice per session. Therefore, the anomaly-based part would detect if the request occurred five times during a session. It is not expected for the `ChargeParameterDiscovery` request to be sent that many times, and it would be classified as abnormal behavior. The same logic was applied for payload length and timeout detection.

There were three reasons why we chose the minimum and maximum approach. Firstly, we saw that Salman and Bresch [9] successfully used this approach when implementing their IDS for the CAN bus. Secondly, when we asked the research team for IDS at Bosch they also utilized the static bound approach. Lastly, to use normal data as acceptable data seemed logical. There is no data from the V2G session examples that can be seen as attack data. Because we had captures of normal data sessions, it seemed logical to use this data in order to set lower- and upper bound (LnUB) thresholds. This LnUB threshold is synonymous with a classification threshold, which determines the boundary for when a message is considered an attack or threat. We collected data for each message type, and it was not too complex to give each message type its own LnUB.

ESCRYPT also suggested the idea of adding a tolerance threshold. They argued that a few errors could go undetected without affecting the detection performance too negatively. The tolerance threshold turned out to be effective, and

in Chapter 6 there are several attack scenarios when the IDS performs better when applying a higher tolerance threshold.

An illustration of the effect from the threshold and tolerance can be seen in Figure 3.3, this could represent the detection output from any of the message types for either frequency, payload or timeout predictions. As can be seen, the classification threshold sets the boundary for when the IDS starts detecting threats. To the right, there are classified detections and to the left is the classified normal traffic. If the IDS makes detections right of the threshold, they are either true or false positives. All packets left of the threshold are either true or false negatives.

Here the effects of the tolerance are apparent, as it will only affect the right side of the threshold. Figure 3.3 displays that two false positives will be correctly classified as non-malicious, but one actual malicious packet will be neglected and incorrectly classified.

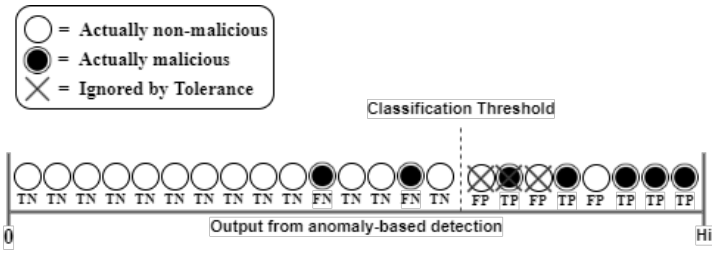

Figure 3.3: An illustration of anomaly-based detection outputs with an upper bound threshold and a tolerance of three. The x-axis represent an arbitrary class value ranging from zero to higher values.

The combination of the tolerance and the LnUB thresholds decides which values are seen as abnormal and will trigger the detection logging. For testing the performance those were the two thresholds we modified in order to see which combination was most effective for each attack scenario. In Table 3.2 below, one can see the different configurations that were tested.

Table 3.2: The different configurations used when testing the IDS.

LnUB	Tolerance	Configuration ID
0	0	Conf1
0	3	Conf2
0.1	0	Conf3
0.1	3	Conf4
0.25	3	Conf5
0.4	3	Conf6

3.4.3 Attack Scenarios

The goal was to implement at least one attack vector to test the IDS. The final result was a total of five different attacks; one attack over UDP and four attacks over TCP. Since there was one message type sent over UDP, we settled on only implementing one attack over UDP. We have outlined all of the attacks in Table 3.3. All of the attacks were launched in separate sessions to evaluate the performance of a single attack vector. The tests were separated for the two selected detection methods to evaluate each method in isolation from the other.

In our test environment, we assumed simple behavior from the server, where requests are always answered with responses. However, in reality, OEMs would more likely implement proper response behavior from the server-side, for example, by not answering spontaneous messages injected into the stream.

Table 3.3: Implemented attack scenarios for testing of the IDS.

Attack	Description
SDP Flooding	Floods the server with more than 50 SECCDiscovery requests.
Message delay attack	Delays messages from the server-side, signifying either a congested network or dropping of packets.
Injection attack	Injects random V2G packets, trying to fuzz or prompt certain responses from the server.
Exhaust server attack	Floods the server with ChargeParameterDiscovery requests,
Charging session flood	Floods the server with additional ChargingStatus and CurrentDemand requests while in a AC or DC charging session.

Our tests were randomized for the delay attack to an extent. The attacks were performed on a random 5% of all the messages sent during a session, and of those 5% the message types with the timeout of two seconds allowed were the ones which triggered a detection (see Table A.2 in Appendix A).

For packet injection, it was also a 5% chance of an attack being launched. This percentage was randomly divided into three message types: `SessionSetup`, `ServiceDiscovery`, and `CableCheck`. Resulting in a varied amount of attacks per the three message types. The reason why we chose these three different groups was to expand our coverage for the test. Performing attacks with several different message types, the more secure we knew our solution was if it could detect well.

The exhaustion attack only performed the attack using one message type, `ChargeParameterDiscovery`, and the amount of those attacks being sent was randomized ranging between 80-200.

Both the SDP and charging flooding attack scenarios had a static number of attacks launched.

Chapter 4
Design Concept

In this chapter, we outline requirements for an IDS and DPI, followed by a security analysis. Based on these requirements, the security analysis, and general research we motivate the design choices for the implemented IDS concept.

4.1 IDS Requirements

When establishing requirements for a security-related product the underlying technology will impose limitations on any new component. In modern vehicle networks, we have mentioned that the real-time requirements are essential. For instance, the control domain requires speeds around 15 microseconds for prioritized traffic [72]. It is therefore essential that the IDS does not introduce further bottlenecks or degrade performance through added latency.

The responsibility of an IDS is to detect all threats against the system within its scope. Many of these vulnerabilities or attack signatures are known and can successfully be mitigated by extending the IDS static rule set. However, the IDS should also have the capability to detect previously unseen attack vectors in the form of zero-day attacks. Therefore, detection features monitoring deviations from normal session behavior are desired as well. Salman and Bresch (see Section 1.5.1) further confirm that the combination of these two principals allows for better detection capabilities in their hybrid implementation [9].

In any IDS, you strive for perfect performance, but in reality, it is almost impossible to achieve due to the complexity of vehicular systems. Nonetheless, the absence of a detection could have severe consequences for the driver if the attack would be successful [50]. Additionally, false positives also need to be kept low in quantity, since if combined with prevention mechanisms, the IDS could remove legitimate traffic. Therefore, it can be concluded that it is desirable to keep false positive and negative rates at a minimal level.

There is a wide variety of policies, standards, and practices employed at different automakers and OEMs. Not only does this mean every implementation in each car model is unique, but it also demands that the internal devices integrated in the vehicle must be uniquely configured. For this reason, an IDS component deployed in these vehicles should be configurable to adapt to any type of car. Hence, developers working for OEMs should be able to extend the logic or customize detection thresholds as they please to fit their specific requirements.

Finally, the resources available for a hosting ECU (e.g., number of cores, available RAM, etc.) are limited. CPU requirements are one of these factors where CPU usage, CPU time and power usage need to be optimized in order for the IDS software to run on an ECU properly. If the hardware is stressed too severely, it could reduce the lifetime of these micro-controllers and introduce unexpected errors or shutdowns.

In conclusion, the requirements for an in-vehicle IDS in modern vehicles can be summarized as seen in the bullet points below. This list further answers the academic question: *Which requirements are associated with IDS in the context of automotive Ethernet?*

1. Low latency – respects high real-time requirements in vehicle communications.
2. The IDS shall be able to detect known and unknown attack patterns.
3. Low false positive and negative ratio.
4. Configurable – can easily be extended or configured to adjust to a new host network.
5. The CPU requirements should be kept as low as possible.

4.2 DPI Requirements

The DPI is a core component for many systems plugged in the network including proxies, packet filters, sniffers, IDS, and intrusion prevention systems. Network components use DPI as an essential inspector where it is applied in different layers of the OSI model [12]. As mentioned in Section 2.4.4, DPI makes it possible to inspect and analyze all layers in the OSI model.

The ability to examine the packets at the respective depth, at wire speed is not an easy task, and such a process decreases the throughput of the system and introduces latency. When designing an effective DPI system one has to consider two main implementation features: the design of an efficient data structure with an optimized memory access rate, and the design of a high throughput algorithm to process intruder signature [12]. Inspection of packets on application level is a demanding task, especially considering the number of packets that need to be processed. Not to mention the importance of addressing the hard real-time requirements in-vehicle communications. Careful consideration needs to be taken not to introduce bottlenecks. Hence, the performance of the DPI engine is of utmost importance.

There are two approaches of performing DPI for any case of application. One can either choose to perform the inspection simultaneously but separately as the data traffic enters the system. There is no bottleneck for the real traffic this way, the traffic is handled completely separately by the DPI. However, this also means that the system might work retroactively and the attack can be long foregone before the DPI raises a red flag. This type of DPI solution cannot perform actions at wire speed. The other approach is to capture and analyze the data directly at wire speed as it enters the system. This means attacks will be detected right away,

but it is harder to avoid the bottleneck when all packets have to be screened. This is the type of approach used for most in-vehicle security solutions [13].

One of the most significant aspects is to choose a search algorithm with an appropriate level of complexity, which can properly compare signatures against intruder signatures without decreasing the throughput of the system. Furthermore, the data structure has to operate and process data traffic regardless of the state of the network. The goal is for the DPI to perform at the same level at all times and under all circumstances.

Memory access time is one of the biggest causes of latency and bottlenecks within a system. Naturally, a high memory-efficient design is preferable, however, this is one of the hardest criteria to fulfill. What we have gathered from ESCRYPT, is that the process of developing a concept includes both hardware and software solutions. A team sets out to develop the concept and completes the software, while the hardware is most commonly outsourced to other manufacturers. When presenting the complete solution, the cost is often the most important thing for the client. Money and time have already been spent developing the code, and the only cutback that can be done is on the part that has not been developed yet: the hardware. Therefore, this requirement is the hardest to achieve in the real world. With less money spent on hardware, there will be more pressure on developing memory-efficient software. The optimal solution would be the signature analysis divided in between software and hardware; with hardware handling shallow inspection and software handling deep inspection [13].

There is also the matter of the signatures the DPI system should handle. Preferably, the system should be able to handle an infinite amount of signatures. This means there has to be some sort of server storing many signatures, because the ECU has very limited data storage. This problem of large amounts of signatures is why anomaly-based detection is such a hot topic. Instead of looking at static signatures – which could reside in overlapping packets or anywhere in the packet frame being sent – the system can look for normal frames in terms of size and content. The abnormal behavior could be reported and evaluated if it should be stored as a static signature on a backlog server. This leads to an important security requirement overall; the ability to dynamically update signatures.

DPI enables the system to find a signature wherever it is within a frame, by looking at the entire frame and not only the data provided by the first couple of OSI layers. This is one of the biggest advantages of DPI and a cornerstone of its functionality. Stateful security is one possible solution to avoid overlapping signatures. If the system saves sessions and packet states it is harder to perform these types of attacks.

To summarize the requirements for DPI in vehicular systems, we have made a short list for them in order to answer the question *Which requirements are associated with DPI for automotive Ethernet?*

1. Implement an appropriate search and analytical algorithm to acquire high memory efficiency and low latency.

2. The system should be operational and computational regardless of traffic circumstances.

3. Design system for high memory efficiency. Preferably through the combination of software and hardware.
4. The system should support a large set of different signatures.
5. Overlapping signatures: the system should be able to detect a signature, even if it is overlapping several frames.
6. The system should be able to handle a signature being anywhere in a frame.
7. Implement a method to dynamically update currently stored signatures with new signatures.

Limitations

A liability of DPI engines is the inability to process encrypted traffic on a network. Hence, communications relying on the use of common security protocols such as SSL/TLS cannot be inspected by an inline DPI engine. Though there exist solutions for this today called DPI-SSL (e.g., SonicWall products support DPI-SSL [73] as well as nDPI [64]), where the hosting DPI device acts as a man-in-the-middle, decrypting and encrypting communications on the fly as packets are inspected. However, DPI-SSL needs special support by a firewall (or other hosting devices) and generally requires better hardware facilities.

Another aspect to consider is that DPI is a resource-intensive task to perform. Unless deploying specially designed hardware there is a trade-off in time spent on each packet and how deep you can inspect a packet's payload, especially if hosted on a device with limited resources, such as an ECU. There are 8-bit processors used for small switching tasks, there are also numerous 16-bit processors being used for ECUs. These are computationally too weak to run cryptographic protocols, so to secure an ECU, considerable processing power is required [13].

4.3 Security Analysis

In this section, we analyze the security aspects of the ISO 15118 specification and later identify important conditions to be reflected in a V2G implementation. An important note is that we do not focus on third party interaction, which is a significant part of realizing an ISO 15118 deployment. Instead, we mainly address concerns for security in the context of a PnC V2G session and subjects related to how a V2G session is set up and performed.

4.3.1 Threat picture

If one considers the intuitive and practical use of PnC in V2G charging, there is no denying we will see further deployment of PnC in the future. Growth in charging infrastructure, charging networks, and the establishment of new secondary actors will open up for new attack surfaces to be exploited by attackers. Hence, securing this growing ecosystem will become all the more prevalent as the adoption of V2G charging progresses.

There are multiple scenarios to consider when looking at attack surfaces for a V2G session, but we have identified four main scenarios that are of concern from a security point of view. The scenarios are depicted in Table 4.1.

Table 4.1: Possible attacker scenarios in V2G communications.

Scenario	Description	Attack goal
Malicious EVCC	Attacker has control over the EVCC ECU that runs the V2G client software and is able to direct a series of attacks.	Exhaust server, deny client charging, commence open TCP communications, DoS on SECC.
Malicious SECC	Attacker has control over the SECC server and is able to direct a series of attacks.	Deny EVCC service, DoS of EVCC, zero-cost charging, affect power quality in system.
Man-in-the-middle	Attacker has control over an intermediary node and can modify, inject, replay or send messages at will.	DoS on EVCC by modifying packets, sniff information, send incorrect metering information.
Remote attacker	Attacker is on the charging network and can sniff, replay or send packets.	DoS on server or client, sniff information, hijacking a session.

As connectivity in modern vehicles increases, there is the persistent threat of hackers being able to breach the internal network of the car remotely. Not only that, but physical tampering threats are also a reality, where an adversary could potentially modify the firmware in the ECUs over the OBD port (on-board diagnostics) [74]. Physical access threats are also present on the side of the charging station, where exposed external ports could give an adversary access to configuration files or controller firmware etc. With such access possibilities, security solutions should expect and look for malicious behavior from fake or remotely controlled devices in a charging session.

During a V2G session the vehicle controller (EVCC) and the charging station controller (SECC) mutually exchange information over HomePlugGreenPhy (physical medium for smart charging). Essentially, they interact as two computers on a public network and a malicious party could be present in that network. It is therefore of importance that these networks are properly secured and maintain a high level of security.

4.3.2 Certificate handling

The first major point of concern is the handling of certificates, which were briefly introduced in Section 2.3.1. If certificates in the PKI system were to be compromised or if an adversary could forge signatures, this would render the whole V2G communication insecure on several levels. To mention two examples, an adversary could charge on another person's behalf (if in possession of the contract certificate) or the attacker could set up a malicious charging station (with the SECC certificate). The specification implicitly tries to mitigate this issue by limiting the lifetime of a contract certificate (maximum of two years expiration), forcing the client to engage in sending a `CertificateUpdate` request to be able to continue using PnC. Furthermore, this is why proper action would have to be taken by CAs and Sub-CAs to control the issuance of new certificates and responsibly revoke any known compromised certificates in the system. In addition to this, OEMs and other involved parties would also have to secure the private keys stored in the EVCC as well as the SECC. An option would preferably include the use of a hardware security model (HSM) for efficient cryptographic computation and most importantly restrict access to the private keys.

4.3.3 UDP in V2G

TLS is the basis for all security properties upheld on the transport layer. Without TLS, the communication would be open and readable, which opens up for many attack vectors to be executed. For example, this increases the possibility for remote TCP hijacking attacks with IP spoofing, making it possible for an adversary to send messages that appear to originate from the EVCC client. Therefore, concerns for remote exploits and other similar attacks are effectively mitigated by the security properties of TLS. With that in mind, the protocol setup phase is of concern since it negotiates the terms for TLS. If the security parameter in the SDP request is changed, communication could commence without TLS by downgrading the client to use a non-TLS setup. It is therefore essential that OEMs require TLS to be used when deploying their custom build of the ISO 15118 specification.

There could be a possibility to perform DoS with the UDP multicast feature utilized during the startup phase of the protocol. This can be done through a malicious EVCC client or a remote attacker, who floods the SECC with large SDP requests over UDP. As a result, the attacker could drain the bandwidth of the SECC – making the charging station unable to properly continue its services. To prevent this, the SECC server should apply the basic protection of only expect requests with a length of two bytes and not respond to more than 50 SDP requests from a single IP (as it is defined in the protocol).

Dudek et al. [75] studied the HomePlugGreenPhy standard used for V2G communications. They found that it was possible for an attacker to fake a SDP response by sending crafted arbitrary IPv6 address and port pairs. It was only possible to establish the physical presence (to send UDP packets) by quickly obtaining the network membership key used to create or join a HomePlug network. As the attacker has been accepted as the SECC, the attacker could act as a man-in-middle or as a malicious charging station; capable of editing, injecting or dropping packets as desired. This would however not adventure the security of the vital

private keys sent in `CertificateInstallation` or `CertificateUpdate` message pairs. Since these message types demand the use of TLS (according to the ISO). Moreover, this attack presumes that the session can be downgraded from TLS to use open communication (otherwise, the attacker will fail to be authenticated by the EVCC since he does not have a signed SECC certificate).

4.3.4 TCP in V2G

The V2G protocol defined in the ISO 15118 specification is a deterministic protocol in which messages are sent only at certain stages in the protocol. From a security standpoint, this deterministic behavior can be considered an advantage, since an attacker needs to abide by the next expected message structure if an attack is to go undetected. A practical example would be e.g., after sending a `SessionSetup` request, the server should not answer such requests again. This narrows the attack surface and prevents malicious behavior such as fuzzing, bogus messaging and flooding attacks. With this in mind, the OEMs deploying the protocol should strictly enforce these states and message sequences in their implementations of ISO 15118.

A DCU that hosts the V2G application can have multiple purposes and would realistically operate several applications simultaneously. It is therefore important that the number of associated addresses and open ports are restricted, and does not reveal any information on port scans or accept random connection attempts.

4.3.5 XML and EXI security

The V2G protocol uses XML-based messages in compressed EXI format with XML security properties for signatures and encryption. Signatures are applied to actions where authentication, authorization and integrity requirements are needed (i.e., when authorizing a vehicle for charging or signing of metering information). Encryption on the other hand is only used to obscure the private keys delivered through `CertificateInstallation` and `CertificateUpdate` messages. Note that this encryption sits on the application layer and still implicitly benefits from the encryption properties of TLS. Together, this hybrid scheme serves to secure critical points of information exchanges and safeguards sensitive data sent during a session.

Regarding how encryption keys are established; the key used for signing is already present in the vehicle from the contract certificate. A shared key is derived for encryption through ephemeral-static ECDH using the private key associated with the contract certificate. If the private key were to be compromised by an adversary the shared key could be recreated from the EVCC private key. This shared key can then be used to reveal all data of past and future sessions sent with this key. Again, we see the further need to safely secure the keys stored in the vehicle.

When signing documents with the contract certificate using the ECDSA algorithm, it is important to maintain high entropy in the scheme. The IV, nonces and challenges should all be created using a source of proper randomness. Otherwise, an adversary could reveal parts or obtain the full private key if the same IV is

used for all signatures. This is further mentioned in the specification [16].

XML standard has its advantages and is currently used through widespread adoption as a way to structure, decode and encode data. However, security researchers have criticised XML for its unnecessary complexity and performance flaws [76], and OWASP has an extensive record of its vulnerabilities [77]. W3C describes in their definition of EXI format 1.0 [44] that EXI shares the same security issues as XML, defined in RFC3023 [78]. As an example, EXI is compatible with user-defined datatype representations, which can be processed incorrectly if not careful. This may introduce new vulnerabilities that can be exploited. Thus, the ISO 15118 implementations utilizing XML/EXI should be deliberately analyzed not to introduce security weaknesses.

4.3.6 Vehicle Charging

Another aspect of V2G sessions is how charging progress in exchanged between the server and the client. When a vehicle is charging, using either AC or DC transfer, there exists a field that represents the current charging progress of the vehicle. The parameter is called `chargeProgress` and is present in the `PowerDelivery` request. An attacker who is able to modify this value could effectively deny the client service by modifying the parameter and setting it to "Stop". Another variant of this malicious act would be to trigger constant renegotiations between the EVCC and the SECC by continuously modifying the value to "Renegotiation". In a best case scenario, the operator receiving EV reports from the SECC could perhaps pick-up on this anomalous activity, and report the vehicle's status to responsible agencies. To perform the described attack, the adversary would have to be in control over the ECU or be established as a man-in-the-middle.

In V2G the charging station needs to employ proper power management not to overload the grid. Hauck et al. [17] (mentioned in Section 1.6) also made a security analysis of the ISO 15118 specification. In their quite extensive analysis, they highlight the danger of gaining physical access to a charging station and what implications it could have for the grid infrastructure. For example, a malicious actor could change the `PowerDelivery` or `CurrentDemand` messages (which deliver information about charging configurations, power and voltage demands, etc.) to affect the power quality of the local grid, or delay or even exit charging sessions on command. Even more beneficial for the adversary controlling the EVCC, they could change the metering information sent from the charging station and subsequently charge for zero cost. To lower the exposure to physical access threats, Hauck et al. [17] suggest mitigation techniques using tampering alarms and encrypting the flash memory used in the charging station, among others.

4.3.7 Error Handling

Another topic to consider is how errors are handled during a session. In the ISO 15118 specification, there are explicitly specified ways to handle errors on an application level, e.g., specifying when certain error codes should be sent from the server or client. An aspect not covered by the specification is how the EVCC and SECC should behave in such instances, and it is mostly up to the OEMs and

suppliers to define this on their own. For instance, the client could be configured to exit a session after a certain error has occurred a set number of times, this could be leveraged to effectively stop a session by prompting a series of such errors. Another example could be a configuration where sending a certain error from the server-side will prompt the client to send a new request. This could be exploited in order to try to force the client to be stuck in an endless loop, without any means to exit or progress in the session. In conclusion, it can be stated that defined behaviors for error handling must be carefully analyzed to not enable exploits as mentioned above.

4.3.8 Recommendations

Based on the findings presented in this section, we here summarize the key takeaways into a list of recommendations for realizing a secure deployment of ISO 15118 PnC solutions.

- TLS should always be the default for ensuring secure communications. OEMs should, to whatever extent possible, avoid open data exchange since it opens up several attack vectors.
- The SECC server should be stateful and not respond to messages outside its state progression path during communications.
- The storage of private keys in the EVCC as well as SECC should be cryptographically secured.
- Proper random number generators should be used for ECDSA nonce, challenge and IV to prevent an adversary from guessing the secret key.
- The charging station should have tamper-proof mechanisms (tampering alarms, encrypted flash memory, removal of external input ports, etc.)
- Error handling should be carefully configured not to introduce exploitable behaviors/actions in the EVCC/SECC.

In conclusion, the V2G communication protocol in the ISO 15118 specification can be considered mostly secure and employs proper security practices to safeguard user data and mitigate several potential exploits and vulnerabilities. The main security flaws are dependent on the party implementing the protocol, and whether this implementation is done properly.

4.4 V2G IDS Proof of Concept

This section presents the reader with our motivations for the design choices of the implemented IDS.

4.4.1 HIDS

The progress of E/E-architectures moving towards domain-centralized topologies and the fact that the ISO 15118 protocol most likely will be concerning only one domain (the domain controlling charging of the vehicle), it made sense for us to design and implement a HIDS for this type of protocol. In practice, this would mean that the detection system would be hosted on a single dedicated ECU or a multipurpose DCU. By developing a security solution for one host it is easier to focus on one specific protocol, as we did for this master's thesis. Moreover, the design principles proposed for this IDS could be applied on more general concepts, such as an in-vehicle NIDS or central gateway.

4.4.2 Detection Methods

There were several reasons why we chose to implement two methods of detection, namely specification-based and anomaly-based detection. First of all, the previous master's thesis completed at ESCRYPT by Yilmaz (see Section 1.6.3) resulted in a security concept for a central gateway, where Yilmaz stated that the combination of an anomaly-based and signature-based detection could perform the DPI. We considered using SNORT, but they did not ship rules for the ISO 15118 specification, which only left the option of writing our own rules and signatures for detection. This would require a deeper knowledge regarding possible threats and attacks enabled by the ISO 15118 specification, which we did not have time to investigate further in this project. After further research, we found a few promising articles using specification-based methods as an alternative to signature-based detection. Both Larson et al. (see Section 1.6.4) and Salman and Bresch (see Section 1.5.1) used a specification-based method with restrictions on e.g., message type and frequencies. Overall, the specification-based approach seemed as a promising option, which could potentially allow for similar detection coverage as signatures. We found the approach to be quite intuitive to implement, by basically reapplying a selected number of conditions defined in the specification.

In *Anomaly Detection for SOME/IP using Complex Event Processing* (see Section 1.5.2) the authors list the anomalies they have considered for their implementation of an anomaly-based IDS for the protocol SOME/IP, which is a protocol used for service-oriented communication in vehicles. Herold et al. considered malformed packets, protocol violations, system-specific violations and timing issues [10]. We would argue malformed packets and timing issues would go under both categories of detection methods, while protocol violations goes under specification-based detection. Malformed packets are packets whose structure does not comply with the protocol, they are either not expected by the specification or they behave abnormally in some other way. Malformed packets may be used to trigger bugs in the protocol implementation or covert the channel to exchange undetected information. Protocol violations occur when a packet deviates from the standard. As the name might indicate, timing issues has to do with violations of time-related conditions.

Another master's thesis, which influenced our choice of implementing the combination of specification-based and anomaly-based detection was *Design and implementation of an intrusion detection system (IDS) for in-vehicle networks* by

Salman and Bresch (see Section 1.5.1). They implemented an IDS for the CAN bus, incorporating both of the detection methods [9].

Overall, the recent reports we found in our research about IDSs in vehicles seem to lean towards this kind of two method approach, and therefore we decided to implement an IDS with specification-based and anomaly-based detection.

4.4.3 DPI

We wanted to incorporate DPI into our solution as well, the IDS covers detection from the IP layer and up all the way to the application layer of the OSI model. The packets sent through a V2G session are analyzed through all the layers, since OpenV2G uses codec functionality that parses EXI/XML data. If the data were incorrect, the parser would generate an error, discard the packet and continue. Therefore, some of the specification-based detection is already covered by the OpenV2G library, packets are required to be well-formed or the parser will not decode or encode the EXI/XML data. By using open source libraries, we chose to focus on implementing the IDS, rather than implementing the ISO 15118 standard from scratch.

4.4.4 Specification-based Detection

When implementing the specification-based detection we were required to read the ISO 15118 protocol to some extent. As one could see Figure 2.9 in Section 2.3.1 there are eight different parts of the ISO 15118 family, which adds up to over 700 pages, and frankly, we found that amount of pages a little overwhelming. We discussed our concerns with our supervisors and came to the conclusion that we could not cover all parts of the specification, and they recommended that we should only focus on the message types and sequence of messages. This was covered in ISO 15118-2, which we started to read rather thoroughly. The standard was straightforward and easy to understand, there were flow charts of both AC and DC sessions using both external payment and PnC, which helped us to understand the allowed message sequences. After getting the background knowledge, we tried to come up with possible attack scenarios and flaws within the protocol, see Table 4.1 in Section 4.3.1 for our threat picture of the standard.

As mentioned, the ISO 15118 specification included some flowcharts describing the exact message sequence allowed, and we decided to start out with implementing stateful logic for our IDS. We defined states for all message types, both requests and responses. Then when a message entered the system its type was compared with the waiting state of the IDS, to check whether the message was indeed expected based on the current state of the IDS. The specification-based detection moved the states along, confirming that the incoming message followed the allowed sequence order. This prevents the attacker from performing replay attacks to some extent, dropping packets, sending bogus packets and exhausting the server.

The protocol also included a table stating all the allowed timeouts between a request and its respective response, in other words the time allowed to elapse between a request being sent and the response to arrive, see Table A.2. We decided to implement a check for these timeouts as well, this could detect a possible drop

of packet or that an attacker tried to sniff and modify a packet.

We also saw it as important to make sure that AC and DC specific messages were not allowed in the other respective session. Even if a packet is well-formed and well-structured it should not be allowed if it is not the right energy transfer mode. Therefore, the energy transfer mode was also stored in the IDS, to ensure that only messages with the valid energy transfer mode was allowed.

It was the same with method of payment; PnC requires that a TLS session is set up between the EVCC and SECC. In order to set up a TLS session one is required to exchange certificates and certificate messages requires that TLS is used. The method of payment and a TLS flag are stored in the IDS for us to check that these types of messages are allowed to be sent. There were no frequency limitations except for SDP messages, and we used the frequency limitation on these SDP messages to implement protection against SDP flooding. The protocol offered no real protection against replay, flooding or DoS attacks for other message types. The payload length limitation for all messages except SDP messages is set to 4294967295 bytes in the protocol, which is a quite substantial size. For SDP messages, there were a static size for the request and response, and we implemented a check for the payload length for these types of messages. Again, for general messages there is no protection against exhaustion or DoS attacks.

There is a need for storing the V2G session ID, it is not allowed to change through the entire session when it has been set and it is sent in every header. Therefore, the IDS stores the session ID to ensure that it has not been modified. This prevents some impostor attacks, where the attacker spoofs the session and tries to send his or her own messages.

4.4.5 Anomaly-based Detection

Anomaly-based detection is one of the more effective approaches to detect future attacks. It analyzes packets to decide whether they are deviating from expected behavior. With the anomaly-based part, we wanted to fill some of the security loopholes that the specification-based detection did not cover. One of the major ones being DoS attacks, for the frequency, payload length and timeout of a message type.

The protocol had a clear definition of which order the messages were supposed to arrive, but had poor protection against flooding attacks. There were some messages that were allowed to be sent repeatedly, which opens up for flooding and replay attacks for those types of messages. There was no real payload length limit for the messages either; the fact that they had to be well-formed and well-structured helped a bit. However, an attacker could still exhaust the server with large messages.

The protocol actually had strict timeouts for all message types, but some of them had up to five seconds. A potential attack could be to resend these types of messages in order to deny charging for others.

Timeout and payload length checks prevent an attacker from sending sniffed and to some extent modified packets, the modification may result in an abnormal delay or payload length, which will be detected by the IDS.

We decided to separate the anomaly-based detection based on the energy trans-

fer mode, AC or DC. The ISO 15118 specification differentiates the two energy transfer modes and we came to the conclusion that it would be beneficial to implement this as well. Not only for the specification-based part, but for the anomaly-based part as well.

4.4.6 Prevention through IP Address and Port Number Storage

We decided to store the IP address and port number of the SECC and EVCC for the TCP or TLS session. This way, if another party tried to spoof the charging session the IDS would detect if they used another IP address or port number. This also prevents port scanning, which means that someone is trying to send client requests to a range of server port addresses on a host, with the goal of finding an active port and exploiting a known vulnerability of that service. For example, the IDS will detect if a rouge charging station is trying to set up another session during an ongoing one.

One major exploit of the ISO 15118 standard is when the sender is not yet authenticated, which is before the handshake. There is a limit of 50 on the amount of SDP requests, a client can send before the server terminates the session. This is opens up the system for potential DoS attacks. We chose to store the IP addresses which were unsuccessful with setting up a V2G session and the respective amount of times an SDP request had been sent from that IP address. If an IP address sent more than 50 requests the IDS detected this. We chose to store five IP addresses, so no more than five different IP addresses were allowed to try to connect to the same SECC. They were stored for two hours before they were flushed. These security measures helps the IDS to detect SDP flooding and SDP flooding from multiple addresses.

Chapter 5
Implementation

This chapter will describe the implementation of the IDS concept. It will outline the different detection methods within the IDS and how we managed to set up attack sessions between an SECC server and an EVCC client.

5.1 IDS

Since the ISO 15118 specification is expected to be deployed on a single ECU or DCU, we decided that the IDS would be designed as a HIDS, which in turn adopted a hybrid approach with two main parts: specification- and anomaly-based detection. We further decided that the IDS was to be implemented as a passive packet sniffer that generates reports, and not have any prevention mechanisms in place. Furthermore, the IDS was designed to have awareness of the higher protocol layers in the OSI model, starting from the IP layer. Figure 5.1 below illustrates the implemented IDS and its different components.

The specification-based detection is implemented based on the ISO 15118 specification and its main detection feature is to inform the IDS if an incoming package is out of order based on the specification standard. There are also some messages that are AC or DC specific, in addition to messages that depend on the selected method of payment. This is described in the specification and therefore, this type of detection is handled by the specification-based part.

A data collector was implemented in order to collect data from the example `pcap` files for the anomaly-based detection methods. The data is stored in binary data files and categorized based on if the transfer mode of the example file was AC or DC. The anomaly-based part detects whether a message has been sent too few or too many times during a session (frequency). It also detects if the payload length of the package is seen as larger or smaller than usual. Finally, both the specification- and anomaly-based detection detects whether the time elapsed between a request-response pair is out of bounds. From these data categories, the collector separates the data into distinct data files for frequency, payload length and time elapsed for each respective request-response pair.

The anomaly-based detection was implemented to use the data files to extract the message-specific data, compute individual thresholds for each category and use these thresholds as a reference for classifying packets. These thresholds are then compared to the actual values for incoming packets in real-time. The anomaly-

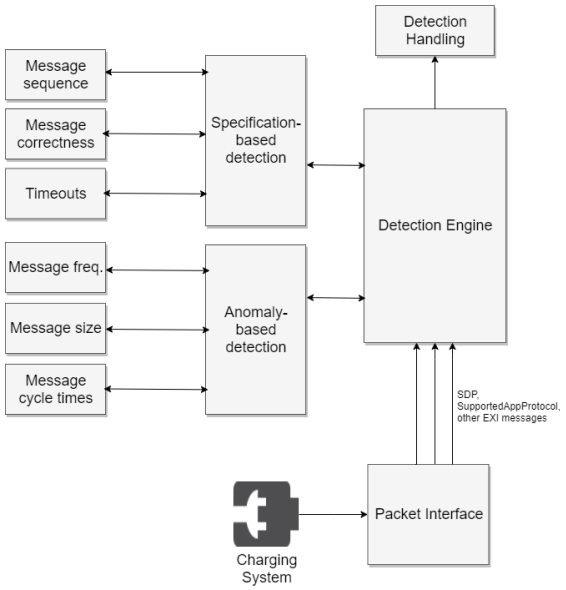

Figure 5.1: Illustration of the implemented IDS.

based detection uses the energy transfer mode, which the IDS stores, in order to get the normal minimum and maximum values for that specific type of message. The anomaly-based part also persistently stores the frequencies of all the message types for the current session. With this information, it is possible for the anomaly-based part to check whether the incoming message type has exceeded its expected frequency, and thus should be ruled as a positive detection. This is realized by continuously comparing against the upper bound frequency, whereas the lower bound frequency needs to be reported at the end of a session.

A V2G session starts with UDP for the SDP request-response pair, after that the protocol has stated that either TCP or TLS may be used for the remaining communication of the session. During the implementation of this IDS only TCP has been used.

The IDS handles incoming packets either from a `pcap` file (offline) or a live capture from an ongoing V2G session. Following this, the packet handler forwards the incoming packet to the detection engine based on three different message categories: SDP, Supported Application Protocol or other EXI messages. OpenV2G was used to encode and decode the incoming packets, and this open source library was designed to differentiate the three different message categories and that is why our implemented IDS solution also differentiates them. The IDS then identifies every message according to its message type defined in the ISO 15118 specification, they can be found in Appendix A. The IDS is stateful and keeps track of the session state, which is changed based on the message type. In Figure 5.2, one can observe the different characteristics of the V2G session and the incoming message.

```
typedef struct v2g_session_t {
    uint8_t session_status;     //Session status: IDS state
    uint8_t msg_type;           //Current message type used
    uint8_t resp_code;          //Current response code
    uint64_t msg_timestamp_elapsed; /*Elapsed message time
                                    from start*/
    uint32_t evcc_ip_address[4];
    uint16_t evcc_port;
    uint32_t secc_ip_address[4];
    uint16_t secc_port;
    uint8_t tls_security;       /*0x10 if TLS is not used,
                                otherwise 0*/
    uint8_t session_id[8];      /*Session ID for current
                                session*/
    uint8_t payment_method;     //Contract or ExternalPayment
    uint8_t energy_transfer_mode;   //0 for DC and 1 for AC
    uint32_t payload_length;    /*Payload length of current
                                packet*/
} v2g_session_t;
```

Figure 5.2: The V2G session object (`struct`) which the IDS updates depending on the incoming message.

5.1.1 SDP Messages

First of all, the SDP request is the only message type that has a limit on the number of times it can be sent before one should terminate the communication between the SECC and EVCC. According to the ISO 15118 specification an SDP request may only be sent 50 times, therefore, there is a method to control the amount of times an IP address has sent such a request.

Additionally, only five different IP addresses may fail to connect within the span of two hours. The IDS stores the different IP addresses and when two hours has passed, the storage is flushed. The storage is also flushed when an SDP request-response pair has been accepted as valid messages by the IDS.

The SDP response carries information whether TLS is used for the communication or not. If TLS is not used this is detected and reported.

Another check, which is specification-based, is that the current message type is checked against the session status of the IDS. This is controlled to decide whether the current incoming message is expected or if it is out of order based on the ISO 15118 specification. This is done for all incoming messages during the V2G session, not only for SDP messages.

Finally, a specification-based method tells the IDS whether the payload length of the incoming packet is seen as normal; two bytes for request and twenty bytes for response.

5.1.2 Supported App Protocol Handshake Messages

The purpose of this request-response pair is to establish which protocols are supported by both the SECC and EVCC.

As with the SDP messages the current message type is checked against the session status of the IDS. The specification-based detection also covers timeouts for every message type's request-response pair, except for SDP messages according to the ISO 15118 specification. In Appendix A, one can see all the timeouts for the respective message types. This timeout is the time elapsed between the request and response. For example, the timestamp for the `SupportedAppProtocol` request is stored upon arrival to the anomaly-based detector and when the corresponding response arrives, the elapsed time in between the pair is calculated. If the time exceeds two seconds a detection is reported, see Table A.2 in Appendix A. As previously mentioned, this control is performed for all messages except for SDP messages.

As there is a specification-based check performed, similarly there is an anomaly-based check performed. The anomaly-based part detects on frequency, payload length and time elapsed. This is done for all the remaining message types, except for SDP messages.

There is also an IP address and port number control from now on for all the remaining messages during the session. As can be seen in Figure 5.2, port and address pairs are stored for EVCC as well as SECC. This is to make sure that a V2G session has the same receiver and sender throughout the complete communication. If an incoming message has a different IP address or port number this is detected.

5.1.3 Other EXI Messages

As mentioned previously, the categorization is due to how the encoding and decoding of OpenV2G is performed. This makes it possible to handle all the remaining EXI messages in the same manner.

Firstly, there is an IP address and port number control for every other EXI message. Moreover, during the session setup the SECC decides a random and unique session ID for the established connection and informs the EVCC of the session ID through the `SessionSetup` response. Similarly to IP address and port number control, this is also checked for every remaining incoming message to make sure the session ID has not been modified. The SECC is not allowed to send another `SessionSetup` response with a new session ID during the rest of the session, and other incoming messages are not allowed to modify the attached session ID.

The method of payment is set by the EVCC through a message type called `PaymentServiceSelection` request, as can be seen in Table A.1 in Appendix A certificates are only allowed to be handled if the payment method *contract* is chosen by the EVCC. The contract payment method is used when PnC is deployed for charging the vehicle. Naturally, the specification-based detector controls whether the correct payment is chosen before handling certificates.

There is also the matter of energy transfer mode, which has been mentioned to play a part in the detection for both the specification-based and anomaly-

based parts. The transfer mode is set by the EVCC through a message type called `ChargeParameterDiscovery` request. The messages that are dependent on a specific transfer mode are controlled by the specification-based detector.

Similarly as for the handshake messages, other EXI messages have specification-based detection controlling the message sequence order and request-response pair timeout table. While the anomaly-based detector checks the frequency, payload length and request-response pair timeout.

The upper frequency bound is checked for every message type during the session, and when the `SessionStop` response has arrived, the lower frequency bound is checked. The `SessionStop` response is sent from the SECC – signaling the termination of the V2G session.

5.1.4 Detection Handling

Whenever an anomaly is detected, whether it is by the specification-based or anomaly-based detector, a detection is generated and forwarded to an incident response manager. The manager logs the detection based on a detection code, all detection codes can be found in the Table A.3 in Appendix A. The manager also has a set tolerance threshold for the anomaly-based detections. The effect of this tolerance threshold will be reviewed further in Chapter 7.

5.1.5 Statistical Data Tracker

When evaluating the IDS we used the binary classification metrics as mentioned in Section 3.4.1. The IDS uses a utility we call the statistical data tracker to gather data for which detection codes were generated for a given packet. This can seamlessly be switched on or off by adding an ending flag for tracking in the command line. Together with expected detection codes generated from the attacker client, the final performance metrics are then calculated at the close of the IDS client (by e.g., exiting with Ctrl-C).

The tracker also registers classification speed – the time it takes for a packet entering the IDS until it has been ruled as a negative or positive – and calculates average, worst and best case classification times. The classification speed measurements can be started by changing the tracker flag to 2 as the last parameter of the IDS terminal command.

5.2 V2G Session

To establish a V2G session we have based the client and server software on the test examples available in the OpenV2G source code (present in the file `main_example.c`). The examples accessible in OpenV2G use byte streams exclusively to simulate a session between client and server, there is no differentiating between the server and client. Since we wanted to simulate real communications according to the ISO 15118 specification, the server- and client-specific calls were divided into separate `c` files (`evcc_client` and `secc_server`). Depending on the command line input upon starting the executable `main` file; either an EVCC or SECC is started. To run a V2G session one starts the SECC first and then the EVCC, as they are

started as separate processes. With the additional integration of UDP and TCP using IPv6, we could successfully execute V2G sessions.

5.2.1 The EVCC Client

The client simulates an EVCC and was implemented as the initiating party during a V2G session according to the ISO 15118 specification. During the session, the client dictates what messages will be sent next, unless there is a server error. Then the server never initiates correspondence. An example of such errors could be TCP or TLS errors (e.g., socket timeouts).

The client is started through a terminal window as seen in Figure 5.3. It takes the following parameters: network interface, server IPv6 address, energy transfer mode (either AC or DC), a flag with an attack pattern (a sum indicating which attacks should be performed, explained further in Section 5.2.3) and finally a flag informing the client which statistical data tracking mode should be used.

There are two different static example sessions that can be executed, using either AC or DC charging. Which one of these sessions are launched depend on the input parameter given to the client application.

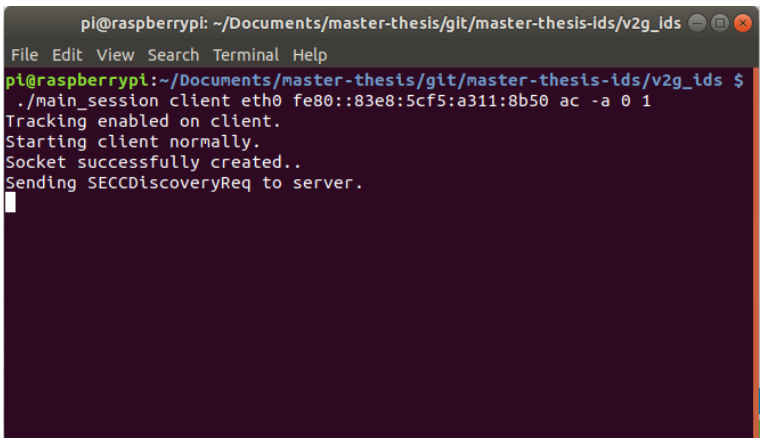

Figure 5.3: A terminal window running the EVCC application.

5.2.2 The SECC Server

The SECC server was implemented as a passive listener with limited capabilities for error handling and V2G aware decision making. It always answers a client's request with a response regardless of that message's context in the V2G communication, e.g., if a `SessionStop` request is received the server will end the session by responding with a `SessionStop` response regardless of it being a finished session or not. The server would not behave in such a way in a real-world implementation of the ISO 15118 specification. But since we had a rather short deadline for the thesis, implementing a fully-featured server application was not within our time scope. The server is started similarly to the client as seen in Figure 5.4.

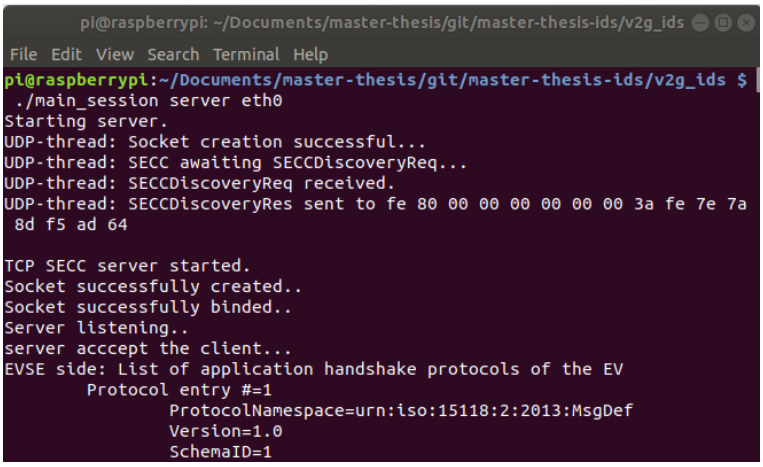

Figure 5.4: A terminal window running the SECC server application with a successful connection made to EVCC client.

5.2.3 Attack Integration

As indicated in Section 3.4.3, we needed to integrate attack scenarios into our implementation to be able to test the security of our IDS. Consider these two examples: we want to perform an attack by dropping responses from the server, effectively denying the client its service; or we want to flood the server with an abundance of messages from the client. In the light of such examples, it is clear that both parties need to deviate from normal session behavior, thus, we decided on having the EVCC client and SECC server adapt a particular behavior for certain attacks. In other words, we integrated attack functionality directly into the server and client respectively. The attacks are exclusively directed from the EVCC client (for example, flooding the server with multiple requests) and the client also prompts attack procedures on the server-side for certain attack scenarios (e.g., when delaying response messages).

A single or several attacks can be performed by using the flag `-a <attack_pattern>` informing both the client and the server which attack pattern to execute, where an attack pattern is a number parsed as two bytes with each bit representing an individual attack. See Figure 5.5, where number one triggers an SDP flooding attack and number two triggers a message delay attack, the attack pattern three would then trigger both SDP flooding and the message delay attack. This is due to that the number three is the sum of the attack patterns for the respective attacks. If several attacks are launched, a separate session instance will be executed for each attack.

If an attack pattern is entered for a session, the detection codes for every packet are stored in a binary file using the aforementioned statistical data tracker (see Section 5.1.5). These are the expected detection codes and are later compared to the detection codes generated by the IDS to decide how well the IDS performed for the specific attack pattern. The EVCC client solely performs the process of writing correct detection codes as attacks are executed.

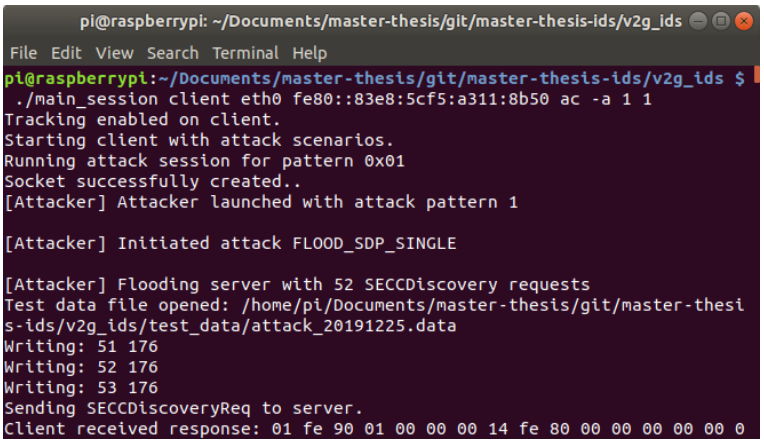

Figure 5.5: A terminal window running the EVCC client application with attacks. The initiated attack is SDP flooding from a single source address (indicated by the number one following -a flag).

Chapter 6
Result

In this chapter, we present the result from running the attack scenarios described in Section 3.4.3. Only relevant results for later discussion are showcased in this chapter, and we refer to Appendix A for all the measurements recorded during the tests.

6.1 Specification-based Results

The specification-based tests were evaluated separately from the anomaly-based part for both AC and DC sessions and showed great results overall. In all the attacks we set up we found that the specification-based anomaly detector had 100% coverage for SDP flooding, delay, injection as well as exhaust attacks. This can be seen in Table 6.1. In contrast, the attack when flooding a session with additional `ChargingStatus` and `CurrentDemand` showed results of 0% detection (if excluding detections where TLS is not used). The last result in Table 6.1 was to be expected since these are allowed sequences in the specification-based detection and should only be reported by the anomaly-based detection method. To be more specific, request-response pairs in the charging phase are allowed to be looped, therefore this is a way to perform a flooding attack without the specification-based method detecting it.

Table 6.1: The specification-based results for both AC and DC tests.

Attack	Detection rate
SDP Flooding	100%
Delay attack	100%
Injection attack	100%
Exhaust attack	100%
Charging session flood	0%

6.2 Anomaly-based Results

For the anomaly-based detection, there were two configuration metrics to evaluate: the tolerance threshold and LnUB threshold.

As mentioned in Section 3.4.2, the tolerance threshold is the threshold we have decided is an acceptable number of anomaly-based detection codes to go unnoticed. The different tolerance values we have tested are 0 and 3, meaning 0 or 3 unreported detection codes for each type of anomaly-based detection: frequency, payload length, and request-response pair timeout. No more than the chosen tolerance threshold of each anomaly-based detection type may go unreported by the IDS.

The LnUB threshold decides how much above the minimum or maximum value the IDS will allow an incoming packet value to go unreported. We have tested four numbers of LnUB thresholds: 0, 0.1, 0.25 and 0.4. For example, if a `SessionSetup` request has been sent during a DC session and the IDS is checking the payload length, the length cannot be lower than 36 or higher than 68 if the LnUB threshold zero has been chosen. In the case of the threshold being 0.25, the LnUB would be 27 respective 85.

The tables below are showing the different configurations used when testing attack scenarios, which resulted in the best case scenario for balanced accuracy, f_1-score and $f_{0.5}$-score for each type of attack scenario. The best metric value has a colored background. There are tables that contain merely one row because using the same configuration aligns the three values. There are also tables were different configurations lead to separate rows because the values do not align.

Each attack is either in AC or DC energy mode, therefore the results are presented in separate sections for each energy mode type. Furthermore, the AC sessions recorded between 850–1300 packets while the DC sessions registered between 1200–1700 packets per session.

Lastly, the classification speed is obtained in microseconds for both AC and DC charging sessions. The classification speed is the time it takes from the entry of a packet until the IDS has either let the packet through or classified it as a detection. Best, worst and average speeds are calculated continuously as the session progresses. For one of the measurements, we run the IDS under a normal session to evaluate speeds during normal circumstances. In the other case, we run the IDS with attacks that trigger multiple detections.

6.2.1 AC Results

Delay Attack

When we performed the attack that delayed a random number of packets there were two best case scenarios. First, there was the configuration that had the tolerance threshold of 0 and the LnUB threshold of 0.1, which resulted in the best balanced accuracy score. And the tolerance threshold 3 and the LnUB of 0.25 resulted in the best f_1-score and $f_{0.5}$-score. See Table 6.2 below for all metrics for the two best configurations.

Balanced accuracy prioritizes that the TPR is high and that the FPR is low. One may note that the FPR actually is lower in Conf5. But due to the FPR being relatively small for Conf3 and that the TPR is higher, the balanced accuracy is still better for Conf3.

Both of the F-scores prioritize precision and recall (TPR), and one can observe that the precision for Conf5 is 1, which is the highest value possible. As mentioned,

the TPR is actually higher for Conf3, but since its precision is lower, the F-scores are not as good for Conf3.

By just looking at the two cases, one can see that Conf3 leads to more normal packets being incorrectly detected as attacks. On the other hand, all of the actual malicious packets were detected. Conf5 had a higher precision, which in this case means there were no normal packets being incorrectly identified as attacks. However, with more actual normal packets being classified as normal packets, there were also more attack packets being classified as normal packets.

Table 6.2: The configurations resulting in the best performance for the anomaly-based detection performing an attack delaying a random number of message types with timeout 2 seconds, see Table 3.2

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Conf3	0	0.0073	1	0.7857	0.9964	0.88	0.8209
Conf5	0.0952	0	0.9048	1	0.9524	0.95	0.9794

Packet Injection Attack

When deploying the packet injection attack, one can note that the resulting table of the best case scenarios has the two F-scores spread out on two rows, instead of one, as in the result for the delay attack. Conf3 resulted in the best $f_{0.5}$-score, while Conf5 resulted in the best balanced accuracy and f_1-score, see Table 6.3 below.

The f_1-score is evenly weighted between precision and recall, while the $f_{0.5}$-score is weighted more towards precision. If the weight would be 1.5, the F-score would be weighted more against recall. In other words, the f_1-score considers incorrectly classifying normal packets (false positives) equally as much as incorrectly classifying malicious packets (false negatives). While the $f_{0.5}$-score considers false positives to be worse than false negatives.

One can see that Conf5 has a substantially lower FPR, leading to a better precision score. Because the $f_{0.5}$-score is weighted against precision this results in the better $f_{0.5}$-score than when using Conf3. The weight also explains why the $f_{0.5}$-score is closer to the precision score compared to the f_1-score. When the weight is shifted towards TPR, as for the balanced accuracy score and f_1-score, the relatively low TPR has more effect on those two metrics. That is why Conf3 produced such high balanced accuracy and f_1-score, the TPR of 1 is just too high for the slightly increased FPR to affect them.

Conf5 produced the least amount of false positives and Conf3 leads to the highest amount of true positives. Conf1 actually had the same TPR, FPR and FNR as Conf3, which lead to very similar results as for Conf3. Nevertheless, Conf1 lead to the worst $f_{0.5}$-score for this particular attack, seen in Table B.3, and was therefore dismissed as one of the better settings.

Table 6.3: The configurations resulting in the best performance for the anomaly-based detection when performing an attack injecting a random number of `SessionSetup`-, `ServiceDiscovery`- and `CableCheck` request-response pairs.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf3	0	0.007	1	0.8929	0.9965	0.9434	0.9124
Conf5	0.1304	0.0024	0.8696	0.9524	0.9336	0.9091	0.9346

Exhaustion Attack

One of the attack scenarios was an attempt to exhaust the server by replaying `ChargeParameterDiscovery` request that returned a large response message. The best configurations to handle this type of attack was Conf3, which resulted in the best balanced accuracy, and Conf6, which resulted in the best F-scores, see Table 6.4 below. Conf6 has the tolerance threshold of 3 and the LnUB threshold of 0.4. Other than the flooding attack, which was supposed to test the anomaly-based part only, the anomaly-based detection had the best performance during this attack.

The TPR was between 0.98-0.99 and the precision score was 1, which is the highest possible, for a lot of different configurations. The best and worst scores were dependent on the third decimal for balanced accuracy and f_1-score and the fourth decimal for the $f_{0.5}$-score. The highest TPR was produced by Conf3, and therefore resulted in the best balanced accuracy. Among the configurations that had the precision score of 1, Conf6 had the highest TPR. Even though the best TPR was determined based on the third decimal, this was enough to produce the highest F-scores.

If one looks at Table B.4 one can see that the tolerance threshold 3 lowers the FPR, the amount of normal packages being identified as attacks is zero when setting the tolerance threshold to 3. This is the reason why the precision is the highest possible score, leading to the excellent F-scores. Conf3 produces the most accurate classification of attack packets. This implies that a tolerance of 0 means a higher TPR score for this type of attack but also a higher amount of false positives.

Table 6.4: The configurations resulting in the best performance for the anomaly-based detection when performing an attack trying to exhaust the server by sending 80-200 `ChargeParameterDiscovery` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf3	0.0078	0.0070	0.9922	0.9883	0.9926	0.9903	0.9891
Conf6	0.0158	0	0.9842	1	0.9921	0.9921	0.9968

Flooding Attack

As indicated previously, the flooding attack was the only attack primarily designed to test the anomaly-based detection. Therefore, the best configuration was Conf1. Where no thresholds had been applied and no packets were identified incorrectly. The only detections were for actual attack packets and no normal packets that deviated from our collected data were sent. Therefore applying no thresholds resulted in the best performance, see Table 6.5 below.

Table 6.5: The configurations resulting in the best performance for the anomaly-based detection when performing an attack trying to flood the server by sending 210 `ChargeStatus` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0	1	1	1	1	1

6.2.2 Overall Classification Speed

We can clearly see that the performance in speed goes down as more detections are being made. The results for the AC case is shown in Table 6.6 below.

Table 6.6: Classification speed measurements in microseconds during AC charging sessions.

Scenario	Best	Worst	Average	Nbr packets	Detections
Normal session	5.0	96.61	23.9	1278	3
Session with attacks	4.89	209.94	144.64	1290	74

6.2.3 DC Results

Delay Attack

When we performed the attack which delayed a random number of packets, there were two best case scenarios with the same result, see Table 6.7 below. One can see that the result worsened when setting the tolerance threshold to 3 in Table B.6. This was because no normal packets were mistakenly detected for attacks.

For Conf1 and Conf3 no packets were incorrectly classified, only when thresholds were set to larger values the attack packets were incorrectly identified as normal packets. Therefore, these were the configurations resulting in the best performance for this type of attack.

Table 6.7: The configurations resulting in the best performance for the anomaly-based detection performing an attack delaying a random number of message types with timeout 2 seconds, see Table 3.2

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Conf1	0	0	1	1	1	1	1
Conf3	0	0	1	1	1	1	1

Packet Injection Attack

When deploying the packet injection attack one can observe that in the resulting Table 6.9, the best case scenarios include Conf1 for the best balanced accuracy. Also, Conf2 is included since it produced the best F-scores. Conf2 had the tolerance and LnUB threshold of 3 respective 0.

Conf1 produced the highest TPR resulting in the best balanced accuracy score, the anomaly-based detection identified all the attack packets correctly and a relatively low amount of normal packets were classified as attacks. However, one may note that the precision is very low compared to the precision produced by Conf2. This is because the number of false positives is higher for Conf1 than Conf2, and as mentioned previously this aspect affects the precision more than it affects the balanced accuracy. Conf2 was the only configuration producing such a high precision for this type of attack; decreasing the number of false positives. This resulted in a better precision score and better F-scores. The precision scores of the other configurations were too low to produce the best F-scores, see Table B.7.

Table 6.8: The configurations resulting in the best performance for the anomaly-based detection when performing an attack injecting a random number of `SessionSetup`-, `ServiceDiscovery`- and `CableCheck` request-response pairs.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Conf1	0	0.0110	1	0.7586	0.9945	0.8628	0.7971
Conf2	0.0968	0.0048	0.9032	0.9032	0.9492	0.9032	0.9032

Exhaustion Attack

One of the attack scenarios was an attempt to exhaust the server by replaying `ChargeParameterDiscovery` request which returned a large response message. The best configurations to handle this type of attack was Conf1, which resulted in both the best balanced accuracy and F-scores, see Table 6.9 below.

For this attack there were no false positives at all, the results produced by all the configurations were more or less perfect. The anomaly-based detection performs remarkably well for this attack scenario. But as with the delay attack,

the result becomes inferior when choosing the higher thresholds, even if just by a slight difference.

The reason why this type of attack scenario does not result in a perfect score is that the collected data allows for some of the attack packets to be classified as normal packets. This is due to the expected frequency being higher than the first couple of attacks, which then can go through undetected.

Table 6.9: The configurations resulting in the best performance for the anomaly-based detection when performing an attack trying to exhaust the server by sending 80-200 `ChargeParameterDiscovery` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Conf1	0.0078	0	0.9922	1	0.9961	0.9961	0.9984

Flooding Attack

The flooding attack is as stated previously a test for only the anomaly-detection part, and is supposed to perform well. For DC, a few normal packets were identified as attacks. Therefore the score is not as great as for the AC session. The two configurations resulting in the best accuracy and F-scores were Conf1 and Conf2, see Table 6.10 below.

Table 6.10: The configurations resulting in the best performance for the anomaly-based detection when performing an attack trying to flood the server by sending 310 `ChargeStatus` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Conf1	0	0.008	1	0.8	0.9960	0.8889	0.8333
Conf2	0.15	0.0032	0.85	0.8947	0.9234	0.8718	0.8854

All the attack packets were classified correctly using Conf1. Even though the TPR was the highest possible, the higher amount of false positives led to a slight decrease of the precision score compared to Conf2. Conf2 had the higher precision, but also the higher amount of false negatives. The high TPR resulted in the balanced accuracy being better when using Conf1. The higher amount of false positives was not enough to affect the balanced accuracy. Though, it did slightly affect the f_1-score. One may notice that the f_1-score is very similar for both configurations, but the TPR was still too high using Conf1 and resulted in the better f_1-score.

When moving the weight towards precision one can see why Conf2 produced the best $f_{0.5}$-score, the precision is higher and the lower TPR is not low enough to drag the resulting $f_{0.5}$-score down.

No attacks were being incorrectly classified using Conf1, though there were a few false positives. The increased tolerance threshold decreased the FPR as well as the TPR. But overall, Conf2 produced the better precision score out of the two configurations.

6.2.4 Overall Classification Speed

Precisely as for the AC case, the results for the DC sessions show a clear decline in the classification speed as more detections are generated. Results are presented in Table 6.11 below.

Table 6.11: Classification speed measurements in microseconds during DC charging sessions.

Scenario	Best	Worst	Average	Nbr packets	Detections
Normal session	4.38	151.25	30.65	1878	3
Session with attacks	7.19	212.92	120.53	1898	80

Chapter 7
Discussion

In this chapter, we discuss the result of the tests and evaluate the design of our IDS implementation. We also present potential improvements for the implementation and future work based on our master's thesis.

7.1 Evaluation of the IDS Implementation

Overall, the implemented IDS showed promising results, one can see that the performance of specification-based detection was at a 100% for all cases except for the flooding attack. The charging flood attack shows that there are attack scenarios in which the specification-based part does not cover. This attack scenario proves that there is a need for another detection method. This is something we expected when we outlined the possible attack vectors. We did not have the time to test all the possible vectors on our IDS. The anomaly-based detection engine performed well during the test of the flooding attack, which shows that anomaly-based detection could be an appropriate compensating method for specification-based detection.

The advantage of the anomaly-based detection is that it is not dependent on a specification, if a specification is modified or the OEM wants an IDS partially covering the standard protocol, the specification-based detection may have to be updated as well. The disadvantage with the anomaly-based detection is that it requires a substantial amount of training data to make accurate predictions, preferably on authentic V2G session data. From our results, we could see that with different configurations, the performance of the anomaly-based detection varied. We would have preferred more example data for more trustworthy results. The anomaly-based detection engine performed at its best for the flooding attack and exhaustion attack. It performed very well overall for attacks dependent on frequency. However, the results varied for the delay and packet injection attacks. The reason for this was mainly that it classified several normal packets as attacks, which is not optimal. We would still argue that the IDS cannot merely depend on a specification-based detection engine for covering the entire threat picture of the ISO 15118 specification. Since the anomaly-based detection covers attack scenarios the other cannot.

The Best Configurations for Anomaly-based Detection

As displayed in the result different configurations affected the performance of the anomaly-based detection, and we would like to discuss why we prioritized the best results based on balanced accuracy and the F-scores. One thing we have learned through this master's thesis is that the automotive industry is full of customized solutions and all OEMs operate differently.

Both balanced accuracy and F-scores are metrics for classifier evaluation, that to some extent handle class imbalance. Depending on which of the two classes (negative or positive) outnumbers the other, one of the metrics is preferred over the other. If the data scenario includes more normal packets than attacks, the F-score is most preferable. On the other hand, if the attacks are more common than normal packets, the balanced accuracy is most preferable [79]. As explained earlier, the difference between the two F-scores is the weight put on precision versus recall. Depending on what the OEM values as most plausible and useful, they can use either one of the metrics to choose an appropriate configuration. See Table 7.1 and 7.2 below for the best configurations and their respective detection performance for AC respective DC sessions.

Table 7.1: Best configurations for anomaly-based detection for each attack for AC.

Attack type	**Balanced Accuracy**	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Delay attack	Conf3: 0.996	Conf5: 0.95	Conf5: 0.979
Packet injection attack	Conf3: 0.997	Conf3: 0.943	Conf5: 0.935
Exhaustion attack	Conf3: 0.993	Conf6: 0.992	Conf6: 0.997
Charging flood attack	Conf1: 1	Conf1: 1	Conf1: 1

Table 7.2: Best configurations for anomaly-based detection for each attack for DC.

Attack type	**Balanced Accuracy**	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
Delay attack	Conf1 or Conf3: 1	Conf1 or Conf3: 1	Conf1 or Conf3: 1
Packet injection attack	Conf1: 0.995	Conf2: 0.903	Conf2: 0.903
Exhaustion attack	Conf1: 0.996	Conf1: 0.996	Conf1: 0.998
Charging flood attack	Conf1: 0.996	Conf1: 0.889	Conf2: 0.885

Our example data included data that is more varied for AC sessions for all attack scenarios, in DC sessions there were not that many occurrences of false positives. The exception being the final attack scenario, charge flooding, where the DC session included more false positives than the AC session. The low amount of false positives resulted in a decrease in actual attacks being detected. Since those

are more affected by the tolerance and LnUB threshold, and the amount of false negatives increases. Therefore, in data sets where there were no false positives, the lowest thresholds were the best performer. However, under real circumstances, sessions with no false positives are very unlikely to occur. Therefore, we decided to review the attack scenarios with sessions that included false positives.

In Table 7.3 below, we have listed the configurations from best to worst for each performance metric. These scores are based on the attack scenarios where sessions included false positives, see Tables in Appendix B for attack scenarios which sessions resulted in non-zero FPR. The configuration applied during an attack scenario which resulted in the best metric score received a 0. The remaining inferior configurations received gradually increasing scores, the worst configuration received a 5. Then we summed up all the configuration scores, the best configuration resulted in the lowest total sum and the worst in the highest total sum. See Tables in Appendix C for the scores of the metrics for each attack scenario. In Table 7.3 below, we have included all the configurations and their respective summed up performance score. We ranked them according to their performance separated by the three metrics: balanced accuracy, f_1-score and $f_{0.5}$-score.

Table 7.3: Scoreboard for balanced accuracy, f_1-score and $f_{0.5}$-score showcasing which configuration worked best overall.

Ranking	**Balanced Accuracy**	$\mathbf{F_1}$	$\mathbf{F_{0.5}}$
1	Conf3: 6	Conf2: 7	Conf2: 4
2	Conf1: 7	Conf1: 11	Conf5: 12
3	Conf2: 10	Conf5: 14	Conf6: 14
4	Conf4: 17	Conf3 or Conf6: 16	Conf4: 15
5	Conf5 or Conf6: 19	Conf4: 17	Conf1: 16
6			Conf3: 20

It is quite clear that the tolerance threshold of 3 improves the $f_{0.5}$-score, the threshold decreases the FPR and the precision increases. This leads to higher $f_{0.5}$-scores. One can also observe that it is the complete opposite for the balanced accuracy score, adding a tolerance threshold decreases the amount of actual attacks being detected, which leads to a lower TPR. However, it also means a lower FPR, and even though this affects the balanced accuracy as well as the F-scores, balanced accuracy is more affected by the decreasing TPR. As one might have noticed in the result Chapter 6, the f_1-score seems to sometimes align with the balanced accuracy, while sometimes with the $f_{0.5}$-score. This is the consequence of the f_1-score being weighted equally between precision and TPR, and we would like to argue that using the f_1-score to decide the best configuration can either be seen as a compromise between the two other metrics or inconclusive based on the data we have used.

Generally, it seems like a higher LnUB threshold results in a decrease of the balanced accuracy. However, Conf3 has a higher LnUB threshold than Conf1, but the scoreboard shows that Conf3 has the better score. If one looks closer at the balanced accuracy, the metric was quite similar for those two configurations

for most of the attack scenarios and therefore resulting in similar scores. The scoreboard shows that Conf3 has the score 6 and Conf1 has the score 7, which are close values. According to us, if an OEM is prioritizing a high TPR and balanced accuracy score, Conf1 and Conf3 can be seen as interchangeable. However, we would like to interpose that more studies should be conducted using data from real use cases and investigating the effect different tolerance thresholds may have on the balanced accuracy score.

For the $f_{0.5}$-score using a higher LnUB does not necessarily lead to a better score. As we can see Conf2 was the configuration with the best overall performance for this metric, followed by Conf5 which received the score 12. One may note that there is a significant jump between the best and second best score (4 vs 12), which might imply that if the LnUB threshold is to result in a better performance it is with the combination of a tolerance threshold. What we can say is that by not setting a tolerance threshold the $f_{0.5}$-score gets worse. Again, there should be more studies conducted on the detection performance effects of different LnUB values. We conducted tests with the two thresholds set at the same time, but if one might have had the time, one could conduct more tests with different LnUB and tolerance thresholds both combined and separately.

Classification Speed

The implementation shows promising results based on the classification speeds recorded in Table 6.6 and Table 6.11. As can be seen, there is a significant contrast between the results obtained under normal circumstances versus when attacks are performed. This is to be expected since additional procedures are triggered after detection (such as report generation). Best speed detections remained stable around 4-7 microseconds with minor fluctuations, which shows that a higher number of detections does not considerably affect the IDS ability to deliver peak performance in speeds. Different from the best speed, the worst speed increased by roughly 116% for AC and 40% for DC. Also, the average speed increased by almost 6x (24 to 144) in the AC case and 4x (30 to 120) for DC. From these results, we can conclude that overall detection performance of the IDS will suffer from higher detection rates in a session. On the other hand, the average speed recorded could be considered sufficient for the automotive use case. As the measured average speeds are in the order of 20-150 microseconds, while end-to-end delays in car domains could fluctuate between 20 microseconds and single-digit milliseconds [72]. This indicates that the IDS would not contribute to any major performance downgrades in latency or introduce bottlenecks in its current state from a classification speed perspective – even when detection rates are high. It is however important to note that these results might not be consistent with an actual in-vehicle implementation of this IDS. Nonetheless, we remain optimistic about its performance capabilities.

DPI

Our IDS rely on full DPI, since the EXI/XML messages need to be parsed by the codec methods. Decoding and encoding does not add any overhead and is a

fast process, we did not face any issues incorporating DPI into our solution. The charging station is not a part of the vehicle and the vehicle is probably shut off while charging, hence, the same time-sensitive requirements for most other protocols used in vehicles does not apply on the same level for this type of communication. Therefore, we are certain that DPI is something that one can incorporate in an IDS for the ISO 15118 specification.

Improvements

The previous master's thesis done at ESCRYPT by Yilmaz (Section 1.6.3) investigated a concept for a complete security solution, including both a firewall and an IDS. We would have liked to extend our solution to incorporate a firewall, which could cooperate with the IDS. With this set up the IDS could trigger dynamic updates in the firewall, based on new signatures that is within a firewall's domain. As a result, the firewall could help levitate the responsibilities of the IDS, which could help in improving system performance. On the contrary, integrating a firewall with dynamic updates would increase the complexity of the security architecture, and might be more bothersome than levitating.

Yilmaz also integrated hardware used for shallow inspection, which would have been an interesting aspect to investigate. Shallow inspection tasks in our solution included storing and checking IP addresses and port numbers, and we could have extended the IDS to check for MAC addresses as well. These are examples of shallow inspection features that could have been added to extend the scope of the IDS.

Overall, it would have been beneficial to run the IDS solution in an authentic V2G environment and see how it performs. Our solution is not inline, it is a passive sniffer and does not affect latency. Nevertheless, if the IDS is to be deployed on an ECU it would have to be inline. A normal ECU typically only has one core with limited capacity for multi-threading, and the IDS must be able to operate simultaneously as the V2G software in the vehicle. By further adapting the IDS to be inline, we could have evaluated its system performance with metrics such as latency and bandwidth. These metrics would have better displayed how the IDS would have performed under real-world circumstances.

In regard of the architecture and design choices, further investigation into the complete ISO 15118 protocol would have helped to extend the specification-based detection engine. There was a limit to how much time we had and thus we had to prioritize which features to select. Furthermore, our lack of knowledge on vehicle mechanics and manufacturing surely affected the extent to which we could operate. If we would have known more about what was viewed as normal V2G session data, in regards to e.g., expected voltage and current limits. For example, how you could integrate anomaly-based profiling of the vehicle's charging progress. This could have been integrated into the IDS and would perhaps improve its detection capabilities.

Limitations

The biggest limitation was the lack of normal data sessions, which we have mentioned being an issue throughout the report. The lack of example data did not

only affect our training data but also our knowledge of what real data should look like. The examples from RISEV2G were very different from the ones acquired from OpenV2G and the ones captured from the live V2G sessions we set up. Furthermore, there were no one from ESCRYPT or Bosch with knowledge in V2G sessions whom we could ask for advice. Therefore, we are not sure which frequencies, payload lengths or timeouts can be seen as normal. More training data from real use cases would have helped us get a deeper understanding and it would probably have affected our results. A statistical study over normal data would have helped us setting appropriate thresholds and help minimize the amount of incorrect detections.

Additionally, a security solution can only be objectively judged on the tests conducted, and we did not have the time to extend the test suite with other attack scenarios. When we asked about the kind of security evaluation methods they use at ESCRYPT, they answered that they outsource the final security tests to other organizations abroad. Where these organizations tries to continuously hack their security solutions and iterate this process for up to over one and a half years. This was not an option for us. Nonetheless, we managed to set up five different attack scenarios which our IDS managed to detect in the majority of cases. Either by using the specification-based or the anomaly-based method.

7.2 Key Takeaways from Result

To answer our last academic questions *What functions of IDS are valuable from the perspective of automotive Ethernet?*, we here summarize our findings with this master's thesis project. We do this by highlighting essential performance, design and feature aspects of our implementation – in the context of our established IDS requirements (see Section 4.1).

First, the two detection methods chosen seem to be a promising combination to achieve high correct detection rates. The specification-based detection design performed extremely well overall, but we have acknowledged scenarios where it is flawed and the anomaly-based detection could work as a compensating party.

If considering the anomaly-based detection design, we would encourage the use of a tolerance threshold, which seems to lead to a higher precision. However, we cannot draw any conclusions as to whether the LnUB threshold is a beneficial feature.

Incorporating DPI into the software was not difficult for the ISO 15118 specification, and we would argue that it is conceivable and applicable for the V2G use case.

Overall, the IDS performed very well during our attack scenarios. There was a low FPR and FNR. It should be able to detect general abnormal behavior regarding message sequencing, frequency, payload length and timeout for request-response pairs. Nevertheless, this can truly only be confirmed through extensive testing, and we cannot say whether the IDS covers all known and unknown attack scenarios.

The classification speeds measured were relatively good, but as mentioned before, a V2G session might not be as time-sensitive as other protocols used in vehicles.

The IDS is configurable, the specification-based detection engine can be extended to include more features, until it covers the entire protocol if one chooses too. Although, then there will most likely be a trade-off in added latency. We have not explored whether the classification speed is affected by adding more detection features, though, we think it is safe to say that there would be an increased latency. The anomaly-based detection is mostly dependent on the training data and the data collector's properties can be customized as long as the collector includes the frequency, payload length and the time elapsed between each message type's request-response pair. Also, the anomaly-based detection is further configurable to add additional metrics to evaluate if desired. Finally, the LnUB threshold and tolerance can be modified to tune the performance for special detection preferences, making them configurable as well.

We did not have the opportunity to test our solution on a real ECU or under real circumstances. It is therefore hard for us to say whether the solution would satisfy the CPU requirements.

In retrospect we found our method for reaching our goals for this project quite effective. The established requirements, general research and outlining of the security for ISO 15118 all contributed in the process of designing the IDS concept. Furthermore, the combination of the specification- and anomaly-based methods was successful and holds great promise for deployment. In not only V2G use cases, but also for other in-vehicle IDS applications.

7.3 Future Work

We have shown that our implementation can achieve high correct detection rates for the attack scenarios presented in this report. In reality, getting the IDS to run on available hardware and not affecting other systems negatively (by introducing e.g., latency) are more desirable before optimizing classification performance. We therefore encourage others to take the next step and apply the IDS functions we have presented in an in-vehicle setting (i.e., by either deploying it on a dedicated ECU or another internal network security device) and further evaluate its qualities.

The anomaly-based detection method applied in this thesis leveraged an intuitive approach to configure a decision threshold. With data studies of authentic V2G sessions, we believe a more statistically sound threshold could be selected (using e.g., machine learning or confidence intervals) to better fit the detection boundary for normal session data and thereby excel its performance.

In this thesis we conducted a security analysis and we found that the security practices used in the ISO 15118 specification were motivated and helped provide confidentiality, authenticity and integrity, among other properties. The analysis further inspired some of the features implemented in our proof of concept. However, there still exist multiple attack surfaces that could be further researched. Through the discovery of new vulnerabilities and exploits in the protocol, security devices such as the IDS could be further improved to detect new threats.

Chapter 8
Conclusions

In this master's thesis we attempt to find valuable functions for an IDS over automotive Ethernet. In this endeavor, we research in-vehicle security and study a V2G charging specification to establish requirements and features for implementing an IDS and incorporating deep packet inspection. We then propose and implement a proof of concept for a V2G IDS with the use of specification- and anomaly-based methods.

The specification-based detection focuses on deviations from specified behavior in a session, which are considered as abnormal in regards to the specification defining a V2G session: ISO 15118. In our implementation, we mainly focus on message sequencing and timeouts outlined by the protocol.

Anomaly-based detection is also a method for detecting anomalous behavior, but instead, it relies on a statistical approach. In our IDS, we collect data from different examples of V2G sessions and based on this data classify the packet either as a threat or a normal packet. The anomaly-based detection engine evaluates the expected frequency, payload length and timeout for each message type's request-response pair. Furthermore, the detection of a packet depends on two different thresholds: tolerance and lower/upper bound. Both thresholds affect the number of packets allowed to go undetected.

To evaluate our implementation, we construct tests for different attack scenarios and launch attacks against the IDS. The results show that the implementation can successfully detect the attacks and is a promising solution to detect threats in a V2G charging environment.

When evaluating our IDS, we find that separately the detection methods have their respective advantages and disadvantages, but together they can operate in unison as a hybrid method to detect both previously known and unseen threats. The specification-based approach is intuitive to implement and shows almost perfect results for all attacks. In contrast, the anomaly-based approach was found to be more difficult to configure for optimal performance, but it does detect attacks in situations where the other cannot. Our tests show that during normal V2G sessions with few false positives, an increased tolerance threshold will increase the precision, while slightly decreasing the true positive rate. Further investigation and evaluation of both of the two thresholds should be done to determine their importance for the performance of the anomaly-based detection.

A great next step would be to implement the IDS in a real ECU and evaluate its performance on real use cases as well as testing additional attack scenarios.

References

[1] Corbett C et al. *Automotive Ethernet: Security opportunity or challenge?* [Online; accessed September 23, 2019]. URL: https://subs.emis.de/LNI/Proceedings/Proceedings256/45.pdf.

[2] Pesé M, Schmidt K, and Zweck H. *WCM 17: SAE World Congress Experience.* In: *Hardware/Software Co-Design of an Automotive Embedded Firewall.* SAE Mobilus. USA, 2017.

[3] Yilmaz E. *Firewall and IDPS concept for Automotive Ethernet.* Figure 2.2. MA thesis. Uppsala Universitet, 2019.

[4] ESCRYPT. *Intrusion detection and prevention for vehicles.* [Online Video; accessed February 4, 2020]. July 2017. URL: https://www.youtube.com/watch?v=QS0Dx70vHz4.

[5] Business Wire. *Autonomous and Connected Vehicles Face 300,000 Attacks Per Month, According to Karamba Security.* [Online; updated January 8, 2019; accessed February 2, 2020]. 2019. URL: https://www.businesswire.com/news/home/20190108005204/en/Autonomous-Connected-Vehicles-Face-300000-Attacks-Month.

[6] Wolf M, Weimerskirch A, and Paar C. *Security in automotive bus systems.* In: *Proceedings of the workshop on embedded security in cars (ESCAR)'04.* 2004.

[7] Huybrechts T et al. *Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques.* In: *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing.* Jan. 2018, pp. 751–761. ISBN: 978-3-319-69834-2. DOI: 10.1007/978-3-319-69835-9_71.

[8] Buttigieg R, Farrugia M, and Meli C. *Security Issues in Controller Area Networks in Automobiles.* In: *CoRR* abs/1711.05824. 2017.

[9] Salman N and Bresch M. *Design and implementation of an intrusion detection system (IDS) for in-vehicle networks*. [Online; accessed October 5, 2019]. MA thesis. Gothenburg, Sweden: Chalmers University of Technology, 2017. URL: https://odr.chalmers.se/bitstream/20.500.12380/251871/1/251871.pdf.

[10] Nadine Harold et al. *Anomaly Detection for SOME/IP using Complex Event Processing*. [Online; accessed January 26, 2020]. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=7502991%5C&tag=1.

[11] Talic A. *Security Analysis of Ethernet in Cars*. [Online; accessed September 28, 2019]. MA thesis. Kungliga Tekniska Högskolan, 2017. URL: http://kth.diva-portal.org/smash/get/diva2:1147703/FULLTEXT01.pdf.

[12] Abuhmed T, Mohaisen A, and Nyang D. *Deep Packet Inspection for Intrusion Detection Systems: A Survey*. In: *Magazine of Korea Telecommunication Society* 24.11. 2007, pp. 24–36.

[13] Yilmaz E. *Firewall and IDPS concept for Automotive Ethernet*. MA thesis. Uppsala Universitet, 2019.

[14] Larson U. E., Nilsson D. K., and Jonsson E. *An approach to specification-based attack detection for in-vehicle networks*. In: *2008 IEEE Intelligent Vehicles Symposium*. June 2008, pp. 220–225.

[15] Stachowski S, Gaynier R, and LeBlanc D. J. *An Assessment Method for Automotive Intrusion Detection System Performance (Report No. DOT HS 812 708)*. Washington, DC: National Highway Traffic Safety Administration., Apr. 2019, pp. 1–50. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/151378/UMTRI-2017-11.pdf?sequence=1%5C&isAllowed=y.

[16] ISO 15118:2014(E). *Road vehicles - Vehicle-to-grid communication Interface - Part 2: Network and application protocol requirements (ISO 15118-2:2014)*. Standard 32000– 1:2008. Geneva, Switzerland: International Organization for Standardization, 2008.

[17] Cabell H et al. *Vehicle Cybersecurity Threats and Mitigation Approaches*. Golden, CO: National Renewable Energy Laboratory. NREL/TP-5400-74247, 2019, pp. 1–27. URL: https://www.nrel.gov/docs/fy19osti/74247.pdf.

[18] Kozierok C. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. San Francisco, United States: William Pollock, 2005, pp. 122–123. ISBN: 1-59327-047-X.

[19] Reynders D and Wright E. *Practical TCP/IP and Ethernet Networking*. Oxford: Newnes an imprint of Elsevier, 2003, pp. 59–73. ISBN: 978-0-7506-5806-5.

[20] Network Working Group. *The Transport Layer Security (TLS) Protocol Version 1.2*. Introduction; section 1 [Online; accessed December 1, 2019]. Aug. 2008. URL: https://tools.ietf.org/html/rfc5246.

[21] Deloitte. *Automotive electronics cost as a percentage of total car cost worldwide from 1950 to 2030*. [Online; accessed September 26, 2019]. 2019. URL: https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/technology-media-telecommunications/deloitte-cn-tmt-semiconductors-the-next-wave-en-190422.pdf.

[22] Adelsbach A, Huber U, and Sadeghi A. *Secure Software Delivery and Installation in Embedded Systems*. Berlin, Heidelberg: Springer, 2006, p. 28. ISBN: 978-3-540-28384-3. DOI: https://doi.org/10.1007/3-540-28428-1_3.

[23] embitel. *'ECU' is a Three Letter Answer for all the Innovative Features in Your Car: Know How the Story Unfolded*. [Online; accessed September 25, 2019]. 2017. URL: https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics.

[24] Lawrenz EW. *CAN System Engineering*. London: Springer London, 2013. ISBN: 978-1-4471-5612-3. DOI: 10.1007/978-1-4471-5613-0.

[25] Wikipedia. *Local Interconnect Network*. [Online; accessed September 25, 2019]. 2019. URL: https://en.wikipedia.org/wiki/Local_Interconnect_Network.

[26] ISO. *Road vehicles - Local Interconnect Network (LIN) - Part 7: Electrical Physical Layer (EPL) conformance test specification*. [Online; accessed September 25, 2019]. 2016. URL: https://www.iso.org/obp/ui/#iso:std:iso:17987:-7:ed-1:v1:en.

[27] Nouvel F et al. *Automotive Network Architecture for ECUs Communications*. [Online; accessed September 24, 2019]. 2015. URL: https://www.researchgate.net/publication/264234515_Automotive_Network_Architecture_for_ECUs_Communications.

[28] Vector. *FlexRay Consortium*. [Online; accessed September 26, 2019]. 2018. URL: https://elearning.vector.com/mod/page/view.php?id=378.

[29] Schmid M. *Automotive Bus Systems*. [Online; accessed September 26, 2019]. URL: https://user.eng.umd.edu/~austin/enes489p/project-resources/SchmidAutoBusSystems.pdf.

[30] Kenji Suzuki ESPEC CORP. *Car electronization trend in automotive industry*. Increase of Electronics (2); Slide 28 [Online PowerPoint presentation; accessed October 1, 2019]. 2014. URL: https://www.slideshare.net/kenjisuzuki397/car-electronization-trend-in-automotive-industry-44007679.

[31] Nelson P. *Just one autonomous car will use 4,000 GB of data/day*. [Online; updated December 6, 2016; accessed September 26, 2019]. 2016. URL: https://www.networkworld.com/article/3147892/one-autonomous-car-will-use-4000-gb-of-dataday.html.

[32] Nolte T, Hansson H, and Bello LL. *Automotive communications-past, current and future*. In: *2005 IEEE Conference on Emerging Technologies and Factory Automation*. Vol. 1. Sept. 2005, 8 pp.-992.

[33] Renesas Electronics Europe Roland Lieder. *The evolution of gateway processors in the auto market*. Figure 2, New car network architecture [Online; accessed September 30, 2019]. 2013. URL: https://automotive.electronicspecifier.com/driver-assistance-systems/the-evolution-of-gateway-processors-in-the-auto-market.

[34] ixia. *Automotive Ethernet: An Overview*. [Online; accessed September 25, 2019]. 2014. URL: https://support.ixiacom.com/sites/default/files/resources/whitepaper/ixia-automotive-ethernet-primer-whitepaper_1.pdf.

[35] Holle J and Shukla S. *Gatekeeper for In-vehicle Network Communication*. In: *ATZelektronik worldwide* 13.6. Dec. 2018. Figure 3, Automotive firewall (CycurGATE) as ISO layer model (© Bosch | Escrypt); p. 43, pp. 40–43.

[36] Mash C. *How Ethernet Will Change Automotive Networks*. [Online; updated February 1, 2018; accessed October 1, 2019]. 2018. URL: https://semiengineering.com/how-ethernet-will-change-automotive-networks/.

[37] Miao Tan K, Ramachandaramurthy V K, and Ying Yong J. *Integration of electric vehicles in smart grid: A review on vehicle to grid technologies and optimization techniques*. In: *Renewable and Sustainable Energy Reviews* 53.1. 2016, pp. 720–732.

[38] Robledo C. B. et al. *Integrating a hydrogen fuel cell electric vehicle with vehicle-to-grid technology, photovoltaic power and a residential building*. In: *Applied Energy* 215.4. 2018, pp. 615–629.

[39] PG&E News Department. *Pacific Gas and Electric Company Energizes Silicon Valley With Vehicle-to-Grid Technology*. In: *PG&E News Department* 415.4. 2007.

[40] Cenex. *EV Charging*. [accessed December 4, 2019]. URL: https://www.cleantech.com/ev-charging-software-and-grid-services/.

[41] Uppladning.nu. *General statistics over charging stations in the world*. [Online; accessed December 4, 2019]. 2019. URL: https://uppladdning.nu/List.aspx.

[42] V2G Clarity. *What Is ISO 15118?* [Online; accessed January 9, 2020]. 2019. URL: https://v2g-clarity.com/knowledgebase/basics-of-plug-and-charge/#pkis-basis-of-pnc.

[43] Wikipedia. *ISO 15118*. [Online; accessed February 18, 2020]. May 2018. URL: https://en.wikipedia.org/wiki/ISO_15118.

[44] W3C. *Efficient XML Interchange (EXI) Format 1.0 (Second Edition)*. [Online; accessed January 6, 2020]. 2014. URL: https://www.w3.org/TR/exi/.

[45] V2G Clarity. *The Basics of Plug & Charge*. [Online; accessed December 4, 2019]. 2019. URL: https://v2g-clarity.com/knowledgebase/what-is-iso-15118/.

[46] Wikipedia. *Elliptic-curve Diffie–Hellman*. [Online; accessed JAnuary 9, 2020]. 2019. URL: https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%5C%E2%5C%80%5C%93Hellman.

[47] Wikipedia. *Public Key Infrastructure*. [Online; accessed January 9, 2020]. URL: https://en.wikipedia.org/wiki/Public_key_infrastructure.

[48] Wolf M, Weimerskirch A, and Wollinger T. *State of the Art: Embedding Security in Vehicles*. In: *EURASIP Journal on Embedded Systems (2007) 2007:074706*. Apr. 2017.

[49] Security Magazine. *Senators Markey and Blumenthal Reintroduce Legislation to Protect Cybersecurity on Aircrafts and in Cars*. [Online; accessed October 2, 2019]. June 2019. URL: https://www.securitymagazine.com/articles/90584-senators-markey-and-blumenthal-reintroduce-legislation-to-protect-cybersecurity-on-aircrafts-and-in-cars.

[50] Miller C and Valasek C. *Remote Exploitation of an Unaltered Passenger Vehicle*. [Online; accessed October 2, 2019]. Aug. 2015. URL: http://illmatics.com/Remote%5C%20Car%5C%20Hacking.pdf.

[51] Greenberg A. *Hackers Remotely Kill a Jeep on the Highway—With Me in It.* [Online Video; accessed October 1, 2019]. July 2015. URL: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

[52] Wolf M, Weimerskirch A, and Paar C. *Security in automotive bus systems.* In: *IN: Proceedings of the workshop on embedded security in cars (ESCAR)'04.* Table 4, Attackers in the automotive area based on [Pa03]; p. 6. 2004.

[53] Cisco. *What Is a Firewall?* [Online; accessed October 3, 2019]. URL: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html.

[54] Cuppens F et al. *Handling Stateful Firewall Anomalies.* IFIP International Federation for Information Processing, 2012, pp. 174–185.

[55] Kuang J, Mei L, and Bian J. *An innovative implement in organizing complicated and massive intrusion detection rules of IDS.* In: *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems Cloud Computing and Intelligent Systems (CCIS).* Hangzhou, China: IEEE, Oct. 2012, pp. 1328–1332. DOI: 10.1109/CCIS.2012.666460.

[56] Aldwairi M, Abu-Dalo AM, and Jarrah M. *Pattern matching of signature-based IDS using Myers algorithm under MapReduce framework.* In: *EURASIP Journal on Information Security.* 2017, pp. 1–11.

[57] Mazen Kharbutli, Monther Aldwairi, and Abdullah Mughrabi. *Function and Data Parallelization of Wu-Manber Pattern Matching for Intrusion Detection Systems.* In: *Network Protocols and Algorithms* 4. Sept. 2012, pp. 46–61.

[58] Han J, Kamber M, and Pei J. *13 - Data Mining Trends and Research Frontiers: Data Mining (Third Edition).* Boston: Morgan Kaufmann, 2012, pp. 585–631. ISBN: 978-0-12-381479-1.

[59] Yihunie F, Abdelfattah E, and Regmi A. *Applying Machine Learning to Anomaly-Based Intrusion Detection Systems.* In: *2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT).* May 2019, pp. 1–5. DOI: 10.1109/LISAT.2019.8817340.

[60] Uppuluri P and Sekar R. *Experiences with Specification-Based Intrusion Detection.* In: *Recent Advances in Intrusion Detection.* Ed. by Lee W, Mé L, and Wespi A. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 172–189.

[61] Snort. *What is Snort?* [Online; accessed September 12, 2019]. URL: https://www.snort.org.

[62] Snort. *What is a signature?* [Online; accessed September 27, 2019]. URL: https://www.snort.org/faq/what-is-a-signature.

[63] Wilson M. *Deep Packet Inspection – A Look at What It Is, Tutorial & Software/Tools for DPI.* [Online; updated March 30, 2019; accessed September 26, 2019]. 2019. URL: https://www.pcwdld.com/deep-packet-inspection.

[64] ntop. *nDPI: Open and Extensible LGPLv3 Deep Packet Inspection Library.* [Online; accessed October 8, 2019]. URL: https://www.ntop.org/products/deep-packet-inspection/ndpi/.

[65] Solarwinds. *Deep Packet Inspection and Analysis with Network Performance Monitor.* [Online; accessed October 8, 2019]. URL: https://www.solarwinds.com/network-performance-monitor/use-cases/deep-packet-inspection.

[66] Raspberry PI Foundation. *Raspberry Pi 3 Model B.* [Online; accessed December 25, 2019]. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[67] Wireshark Foundation. *About Wireshark.* [Online; accessed December 25, 2019]. URL: https://www.wireshark.org/.

[68] V2G Clarity. *RISE V2G.* [Online; accessed December 25, 2019]. URL: https://v2g-clarity.com/rise-v2g/.

[69] Käbisch S et al. *OpenV2G.* [Online; accessed December 25, 2019]. URL: http://openv2g.sourceforge.net/.

[70] scikit-learn developers. *sklearn.metrics.balanced_accuracy_score.* [Online; accessed January 24, 2020]. 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html.

[71] S McElwee. *Probabilistic Clustering Ensemble Evaluation for Intrusion Detection.* Figure 3, Confusion matrix for binary classification; p. 29. PhD thesis. Aug. 2018, p.29. DOI: 10.13140/RG.2.2.13702.83525.

[72] Lim H, Völker L, and Herrscher D. *Challenges in a future IP/Ethernet-based in-car network for real-time applications.* In: *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC).* June 2011, pp. 7–12.

[73] SonicWall. *Network Security Firewalls.* [Online; accessed October 8, 2019]. URL: https://www.sonicwall.com/products/firewalls/.

[74] Klinedinst D and King C. *On Board Diagnostics: Risks and Vulnerabilities of the Connected Vehicle*. [Online; accessed January 24, 2020]. Mar. 2016. URL: https://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_453877.pdf.

[75] Dudek S, Delaunay JC, and Fargues V. *V2G Injector: Whispering to cars and charging units through the Power-Line*. In: *Synacktiv*. 2019, pp. 1–26.

[76] Peter Gutmann. *Why XML Security is Broken*. [Online; accessed January 24, 2020]. 2004. URL: https://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt.

[77] OWASP. *XML Security Cheat Sheet*. [Online; accessed January 24, 2020]. URL: https://owasp.org/www-project-cheat-sheets/cheatsheets/XML_Security_Cheat_Sheet.html.

[78] Murata M, St.Laurent S, and Kohn D. *RFC 3023: XML Media Types*. Security considerations; section 10 [Online; accessed October 3, 2019]. 2001. URL: https://www.ietf.org/rfc/rfc3023.txt.

[79] Shashwat. *Balanced accuracy vs F-1 score*. [Online; accessed January 27, 2020]. 2013. URL: https://stats.stackexchange.com/questions/49579/balanced-accuracy-vs-f-1-score.

Appendix A

V2G Related Information

Table A.1: Different message types defined in the ISO 15118 specification for a V2G session.

Message Types in V2G Session
SDP Request and Response
SupportedAppProtocol Request and Response
SessionSetup Request and Response
ServiceDiscovery Request and Response
ServiceDetail Request and Response
ServicePaymentSelection Request and Response
CertificateInstallation Request and Response (Only allowed if Contract payment method is used)
CertificateUpdate Request and Response (Only allowed if Contract payment method is used)
PaymentDetails Request and Response
Authorization Request and Response
ChargeParameterDiscovery Request and Response
CableCheck Request and Response (DC specific)
PreCharge Request and Response (DC specific)
PowerDelivery Request and Response
ChargingStatus Request and Response (AC specific)
CurrentDemand Request and Response (DC specific)
WeldingDetection Request and Response (DC specific)
MeteringReceipt Request and Response
SessionStop Request and Response

Table A.2: Different timeouts for message types defined in the ISO 15118 specification for a V2G session.

Message type	**Timeout in seconds**
SupportedAppProtocolReq	2
SessionSetupReq	2
ServiceDiscoveryReq	2
ServiceDetailReq	5
PaymentServiceSelectionReq	2
PaymentDetailsReq	5
AuthorizationReq	2
ChargeParameterDiscoveryReq	2
ChargingStatusReq	2
MeteringReceiptReq	2
PowerDeliveryReq	5
CableCheckReq	2
PreChargeReq	2
CurrentDemandReq	0,25
WeldingDetectionReq	2
SessionStopReq	2
CertificateInstallationReq	5
CertificateUpdateReq	5

Table A.3: The different definitions of the detection codes and their respective hexadecimal representation used by the IDS.

Detection Code Definition	**Det. Code**
SDP_REQ_COUNTER_VIOLATION	0xB0
SDP_REQ_TOO_MANY_DIFFERENT_IP_ADDRESSES	0xB1
SESSION_ID_ALREADY_SET	0xB2
SESSION_ID_VIOLATION	0xB3
IP_ADDRESS_VIOLATION	0xF0
PORT_VIOLATION	0xF1
TLS_NOT_USED	0xF2
SERVICE_REQUEST_VIOLATION	0xF3
CERTIFICATE_ACCESS_VIOLATION	0xF4
MESSAGE_SEQUENCE_VIOLATION	0xF5
TRANSFER_MODE_VIOLATION	0xF6
TIMESTAMP_VIOLATION	0xF7
PAYLOAD_OUT_OF_BOUNCE_DETECTED	0xA3
TIMEOUT_OUT_OF_BOUNCE_DETECTED	0xA4
COUNTER_OUT_OF_BOUNCE_DETECTED	0xA5

Appendix B

Anomaly-based detection Results

Table B.1: Different configurations used when testing the IDS.

LnUB	Tolerance	Configuration ID
0	0	Conf1
0	3	Conf2
0.1	0	Conf3
0.1	3	Conf4
0.25	3	Conf5
0.4	3	Conf6

B.1 Anomaly-based Detection Performance for AC V2G Sessions

Table B.2: Performance for the anomaly-based detection when performing an attack delaying random number of message types with timeout 2 seconds, see Table A.2.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0.0073	1	0.8333	0.9964	0.9091	0.8621
Conf2	0.0741	0.0024	0.9259	0.9259	0.9618	0.9259	0.9259
Conf3	0	0.0072	1	0.7857	0.9964	0.88	0.8209
Conf4	0.0833	0.0024	0.9167	0.9167	0.9571	0.9167	0.9167
Conf5	0.0952	0	0.9048	1	0.9524	0.95	0.9794
Conf6	0.1111	0	0.8889	1	0.9444	0.9412	0.9756

Table B.3: Performance for the anomaly-based detection when performing an attack injecting random number of `SessionSetup-`, `ServiceDiscovery-` and `CableCheck` request and response pairs.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0.0070	1	0.8846	0.9965	0.9388	0.9055
Conf2	0.1429	0.0023	0.8571	0.9474	0.9274	0.9	0.9278
Conf3	0	0.0070	1	0.8929	0.9965	0.9434	0.9124
Conf4	0.1429	0.0024	0.8571	0.9474	0.9274	0.9	0.9278
Conf5	0.1304	0.0024	0.8696	0.8672	0.9336	0.8684	0.8677
Conf6	0.1818	0.0023	0.8182	0.9475	0.9079	0.8781	0.9184

Table B.4: Performance for the anomaly-based detection when performing an attack trying to exhaust the server by sending 80-200 `ChargeParameterDiscovery` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0.0092	0.0516	0.9908	0.9038	0.9696	0.9453	0.9199
Conf2	0.0164	0	0.9836	1	0.9918	0.9917	0.9967
Conf3	0.0078	0.007	0.9922	0.9883	0.9926	0.9903	0.9891
Conf4	0.0165	0	0.9836	1	0.9918	0.9917	0.9967
Conf5	0.0185	0	0.9816	1	0.9908	0.9907	0.9963
Conf6	0.0158	0	0.9842	1	0.9921	0.9921	0.9968

Table B.5: Performance for the anomaly-based detection when performing an attack trying to flood the server by sending 210 `ChargeStatus` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0	1	1	1	1	1
Conf2	0.15	0	0.85	1	0.925	0.9189	0.9659
Conf3	1	0	0	0	0.5	0	0
Conf4	1	0	0	0	0.5	0	0
Conf5	1	0	0	0	0.5	0	0
Conf6	1	0	0	0	0.5	0	0

B.2 Anomaly-based Detection Performance for DC V2G Sessions

Table B.6: Performance for the anomaly-based detection when performing an attack delaying random number of message types with timeout 2 seconds, see Table A.2.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0	1	1	1	1	1
Conf2	0.2	0	0.8	1	0.9	0.8889	0.9524
Conf3	0	0	1	1	1	1	1
Conf4	0.25	0	0.75	1	0.875	0.8571	0.9375
Conf5	0.2222	0	0.7778	1	0.8889	0.875	0.946
Conf6	0.1714	0	0.8286	1	0.9143	0.9063	0.9603

Table B.7: Performance for the anomaly-based detection when performing an attack injecting random number of `SessionSetup-`, `ServiceDiscovery-` and `CableCheck` request and response pairs.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0.0110	1	0.7586	0.9944	0.8627	0.7971
Conf2	0.0968	0.0048	0.9032	0.9032	0.9492	0.9032	0.9032
Conf3	0	0.0126	1	0.75	0.9937	0.8571	0.7895
Conf4	0.0962	0.0164	0.9038	0.6912	0.9437	0.7833	0.7253
Conf5	0.1087	0.0179	0.8913	0.6406	0.9366	0.7455	0.6788
Conf6	0.0938	0.0209	0.9063	0.5179	0.9427	0.6591	0.5664

Table B.8: Performance for the anomaly-based detection when performing an attack trying to exhaust the server by sending 80-200 `ChargeParameterDiscovery` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0.0078	0	0.9922	1	0.9961	0.9961	0.9984
Conf2	0.01362	0	0.9863	1	0.9932	0.9931	0.9972
Conf3	0.0092	0	0.9908	1	0.9954	0.9954	0.9981
Conf4	0.0209	0	0.9791	1	0.9895	0.9894	0.9957
Conf5	0.0156	0	0.9844	1	0.9922	0.9922	0.9968
Conf6	0.0249	0	0.9751	1	0.9875	0.9874	0.9949

Table B.9: Performance for the anomaly-based detection when performing an attack trying to flood the server by sending 310 `ChargeStatus` requests in a row.

Conf.	FNR	FPR	TPR	Precision	Balanced Accuracy	F_1	$F_{0.5}$
Conf1	0	0.0079	1	0.8	0.9960	0.8889	0.8333
Conf2	0.15	0.0031	0.85	0.8947	0.9234	0.8718	0.8854
Conf3	1	0.0079	0	0	0.4960	0	0
Conf4	1	0	0	1	0.5	0	0
Conf5	1	0	0	1	0.5	0	0
Conf6	1	0	0	1	0.5	0	0

Appendix C

Scoreboards for Configurations

Table C.1: Scoreboard for balanced accuracy for each attack scenario with sessions which included false positives.

Attack	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
AC: Delay	1	2	0	3	4	5
AC: Packet injection	1	3	0	4	2	5
AC: Exhaustion	5	2	0	3	4	1
DC: Packet injection	0	2	1	3	5	4
DC: Charge flooding	0	1	5	4	4	4
Total sum	7	10	6	17	19	19

Table C.2: Scoreboard for f_1-score for each attack scenario with sessions which included false positives.

Attack	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
AC: Delay	4	2	5	3	0	1
AC: Packet injection	1	3	0	4	2	5
AC: Exhaustion	5	1	4	2	3	0
DC: Packet injection	1	0	2	3	4	5
DC: Charge flooding	0	1	5	5	5	5
Total sum	11	7	16	17	14	16

Table C.3: Scoreboard for $f_{0.5}$-score for each attack scenario with sessions which included false positives.

Attack	**Conf1**	**Conf2**	**Conf3**	**Conf4**	**Conf5**	**Conf6**
AC: Delay	4	2	5	3	0	1
AC: Packet injection	5	1	4	2	0	3
AC: Exhaustion	5	1	4	2	3	0
DC: Packet injection	1	0	2	3	4	5
DC: Charge flooding	1	0	5	5	5	5
Total sum	16	4	20	15	12	14

Printed by Tryckeriet i E-huset, Lund 2020

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2020-747
http://www.eit.lth.se