CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

CIVID - Collaborative In-vehicle Intrusion Detection

Design and evaluation of a AUTOSAR-compliant framework for collaborative intrusion detection in vehicular fleets

Master's thesis in Software engineering and technology

Daniel Aryan
Kristoffer Söderberg

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

Master's thesis 2021

CIVID - Collaborative In-vehicle Intrusion Detection

Design and evaluation of a AUTOSAR-compliant framework for collaborative intrusion detection in vehicular fleets

Daniel Aryan & Kristoffer Söderberg

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

CIVID - Collaborative In-vehicle Intrusion Detection
Design and evaluation of a AUTOSAR-compliant framework for collaborative intrusion detection in vehicular fleets
Daniel Aryan & Kristoffer Söderberg

Supervisor: Rodi Jolak, Department of Computer Science and Engineering
Examiner: Christian Berger, Department of Computer Science and Engineering

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

Typeset in LaTeX
Gothenburg, Sweden 2021

CIVID - Collaborative In-vehicle Intrusion Detection
Design and evaluation of a AUTOSAR-compliant framework for collaborative intrusion detection in vehicular fleets
Daniel Aryan & Kristoffer Söderberg
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Context: Developments within the automotive domain have caused an increased vulnerability to attacks against connected vehicles. To mitigate this threat, vehicles utilize intrusion detection systems.
Problem: Intrusion detection systems can be very effective in detecting and stopping ongoing attacks against vehicles. However, these systems are not infallible and would benefit from increased accuracy in their attack detection.
Objective: By leveraging the access that these connected vehicles have to other vehicles and the outside world, this thesis has designed and evaluated a framework for collaborative intrusion detection (CIVID) with the stated goal of increasing detection accuracy.
Approach: A design science methodology has been applied to conceptualize the problem, design a solution and validate this solution through the simulation of a virtual vehicle fleet.
Result: The aforementioned validation of the collaborative framework shows a marginal increase in accuracy measures through the utilization of a collaborative intrusion detection approach. However, the results also show that the implementation of CIVID yields increased time-to-detection of security events that require consultation.
Conclusion: Despite showing increased accuracy measures, it is unclear whether or not the costs and risks associated with the CIVID framework outweigh the marginal improvements in accuracy measures that it provides. Also, there are many additional challenges that need to be dealt with when implementing the CIVID framework, such as trust- and resource management. How these are to be implemented as well as alternative implementations of the CIVID framework, is left to be explored in future research.

Keywords: Automotive, AUTOSAR, Anomaly-based detection, Collaborative IDS In-vehicle networks, Intrusion detection systems

Acknowledgements

Firstly, we would like to give our thanks to our supervisor Rodi Jolak for his support and constructive feedback during the course of writing this thesis.

Secondly, we would like to extend our gratitude towards Christian Sandberg and Afshin Soltani Esterabadi from AB Volvo, for their inputs and ideas, and for taking their time to discuss this topic with us.

Furthermore, we would like to express our gratitude to Thomas Rosenstatter, for his input and constructive feedback.

Lastly, we would like to give our thanks to our examiner Christian Berger.

Dnaiel Aryan & Kristoffer Söderberg, Gothenburg, June 2021

Contents

List of Figures

List of Tables

Nomenclature

AUTOSAR
AUTOSAR AUTomotive Open System ARchitecture
IDSM Intrusion detection manager
IDSR Intrusion detection reporter
QSEv Qualified security event
SEM Security Event Memory
SEv Security event
SOC Security operations center

Intrusion detection
IDN Intrusion detection network
IDS Intrusion Detection System
IPS Intrusion Prevention System

Machine learning
DNN Deep neural network
ML Machine Learning
RNN Recurrent neural network

Automotive terms
CAN Controller area network
ECU Electrical control unit
IVN In-vehicle network
V2I Vehicle-to-infrastructure
V2V Vehicle-to-vehicle
VANET Vehicular ad-hoc network

1

Introduction

During the last few decades, the number of electronic devices and software systems embedded into manufactured vehicles has been rapidly increasing. Control systems within vehicles that had traditionally been implemented mechanically, such as the brakes, throttle and steering, have gradually been replaced with electronic systems [35]. These systems are run and controlled by small microcontroller-based computers, known as electronic control units (ECUs) [10] [18] [20].

These electrical components aim to provide higher levels of convenience, efficiency and safety for the vehicle's driver and its passengers, for example by means of providing advanced driver assistance systems (ADAS) and by exchanging massive amounts of data between the vehicle's various components [18]. Modern vehicles usually contain more than 100 ECUs [39].

However, while this development does extend the capabilities of the vehicle, it also increases the number of potential cybersecurity threats that the vehicle is exposed to and may make the vehicle more prone to attacks[10] [20]. This, combined with the safety-critical nature of a connected vehicle operating in traffic, increases the need for high-security measures in vehicular IT systems [10].

Concurrently, so-called connected vehicle fleets are becoming increasingly common, where a network-connected group of motor vehicles is owned or leased by private businesses, governmental agencies, or other organizations. The addition of network connectivity to these vehicles creates a wide range of possibilities within fleet management. As such a question arises: Is it possible to leverage the access that these organizations have to a fleet of connected vehicles to detect ongoing attacks against the vehicles in the fleet more accurately?

This study aims to answer the above question by outlining a potential framework for collaborative intrusion detection within a vehicle fleet. This framework will adhere to the intrusion detection standards laid out by the standardization body AUTOSAR, and will be assessed using threat modeling. Following this, the framework will be exemplified with a system implementation. This implementation will in turn be used to further assess the usefulness of the framework, through an empirical evaluation of the accuracy metrics outlined in chapter 4.

1.1 Statement of the problem

As vehicles are becoming more and more complex and the amount of software running on each vehicle increases, the number of potential attack surfaces and vulnerabilities in the vehicle increases accordingly. While vehicle manufacturers put extensive effort into threat modeling, defining security requirements and securely designing systems, these efforts cannot mitigate all the threats facing the vehicle. Additionally, the vehicle's different software components are often developed by a variety of different vendors, with each vendor having its own security auditing routines [10]. This further increases the difficulty of designing secure systems. As such, preemptive threat identification and mitigation efforts are not enough and need to be augmented with active intrusion detection and response activities.

Nowadays, many new vehicles are equipped with intrusion detection systems (IDS), which actively try to detect any incoming attacks targeting the vehicle, or other anomalies [19]. However, these systems often have high false-positive rates [34] [19], which results in many false alarms being raised and logged by the vehicles. This may in turn make it harder for security experts to analyze data, identify the root cause of the security incident and decide on a suitable response.

By using a collaborative intrusion detection approach within a fleet, as opposed to only analyzing a suspected attack from the viewpoint of one single vehicle, the accuracy of intrusion detection systems may increase, which would ideally give security experts more manageable data to work with.

1.2 Purpose of the study

Cyber-attacks impose a constant threat on vehicles. Methods to defend a vehicle against such attacks is an ongoing process where failing to detect an attack, can result in life-threatening consequences. The purpose of this study is to design and evaluate a collaborative in-vehicle intrusion detection framework, henceforth referred to as *CIVID*, that aims to improve intrusion detection accuracy by enabling the vehicles within a fleet to collaborate with each other. If such a framework can be utilized to improve the detection accuracy of vehicles, an additional step towards making vehicles more secure can be made.

The evaluation of the framework is to be performed through the analysis of a prototype implementation of the CIVID framework, where outlined accuracy metrics will be used to determine the usefulness of the framework. This framework intends to provide an additional building block to the area of collaborative intrusion detection networks within vehicular fleets and is intended to pave the way for future research on collaborative intrusion detection systems in the automotive domain.

1.3 Research questions

The research questions for this study cover both the framework design as well as the framework evaluation.

- RQ 1- How can vehicles within a vehicle fleet cooperate with each other in order to better detect ongoing attacks?

- RQ 2- What is the effect of a collaborative intrusion detection approach in vehicle fleets on the accuracy & efficiency of identifying ongoing attacks?

RQ1 will be answered through the design of the CIVID framework for collaborative intrusion detection systems tailored specifically for the automotive domain. The framework will be assessed through threat modeling efforts

RQ2 will be answered through empirical evaluation of an implementation of the CIVID framework. More specifically, this evaluation will consist mainly of evaluating an implemented anomaly-based in-vehicle IDS which conforms to the proposed framework, to see if increased accuracy measures are enabled by it. More details regarding this evaluation and the precise metrics used are provided in chapter 4.

1.4 Scope and limitations

The goal of this thesis is to design a theoretical framework that is generally applicable within vehicular IDSs. In other words, the CIVID framework itself should not be limited to only work with certain kinds of detection methods or attack types. As such, the framework will describe the general architecture and information flow within a distributed and collaborative automotive IDS, but will not enforce implementation of any specific intrusion detection methodology.

However, for the prototype implementation of the proposed framework that will be used for empirical evaluation, certain limitations will be applied. For instance, it is not possible to validate the implemented system against every imaginable kind of attack. Instead, the framework will only be tested against a select few types of network injection attacks during the empirical evaluation phase of this thesis. These attacks include flooding-, spoofing, fuzzing and replay attacks. Further details regarding the empirical evaluation and the selected attack examples are provided in chapter 4 and section 6.1.

In addition, the prototype system will not implement the proposed CIVID framework in full, but will instead only focus on implementing the core functionality required to evaluate the proposed framework. More specifically, all of the modules included in the framework will be developed and the flow of information will conform to the flow proposed in the framework. However, certain aspects of the CIVID framework that are not needed to evaluate the framework and provide a

proof-of-concept, but that would be essential for a real-world implementation of the framework, will be left out of this prototype implementation. The parts of the framework that have been left out of the prototype are the trust management and suitability check aspects discussed in chapter 5. Additionally, the aspect of efficient resource allocation as is discussed in section 3.2.1, has also been left out of the framework.

2
Background

This chapter provides some background into topics relevant to this thesis. The first section briefly explores the inner workings of the modern vehicle, with its electrical components and network connectivity.
Thereafter, some background is provided regarding intrusion detection systems and the various techniques used to detect ongoing attacks. The chapter then goes on to detail the standards for in-vehicle intrusion detection outlined by the standardization body AUTOSAR.
Finally, some commonly used methodologies for threat modeling within the automotive domain are provided, including a comprehensive attack model which outlines some of the various assets that are vulnerable to attack within vehicles.

2.1 The connected vehicle and in-vehicle networks

The modern vehicle contains many electrical control units (ECUs), which control many of the vehicle's key features [10] [18] [20]. The various ECUs within the vehicle are connected through a network called an in-vehicle network (IVN), consisting of several different kinds of communication buses [18]. In addition, modern connected vehicles can provide additional features by communicating with a wide range of external devices and services through interfaces such as Wi-Fi or Bluetooth [18]. These external communication interfaces are also a part of the in-vehicle network.

2.1.1 In-vehicle networks

As previously mentioned, there exist several different types of communication buses that serve to connect the various components within a vehicle. Figure 2.1 shows an illustration of a modern IVN. The kind of bus that is used for a specific subsystem depends on its functionality and requirements. A commonly used communication protocol in the in-vehicle network is the Controller Area Network (CAN), which is used to allow ECUs to broadcast important information over the CAN bus and even to other buses. Other commonly used protocols include Media Oriented Systems Transport (MOST), which is a high-speed multi-media protocol used to support entertainment systems in the vehicle, and Local Interconnect Network (LIN), which is used for communication within component groups that are working closely together as a unit [18] [19].

As the number of ECUs in the vehicle grows, so does the complexity and size of the

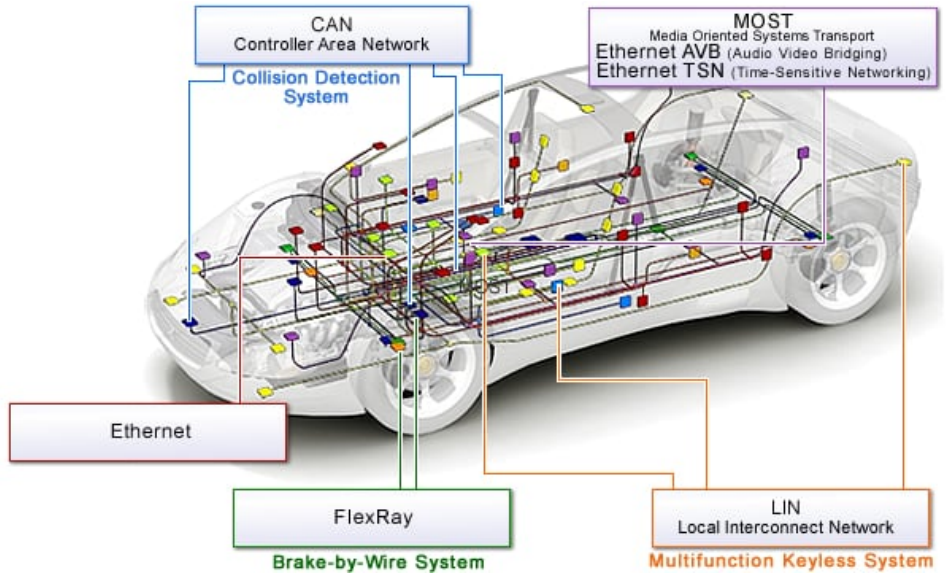


Figure 2.1: Illustration of an in-vehicle network [6]

information flow within these buses, perhaps most notably the CAN bus. The CAN protocol is the most widely used communication protocol within the IVN [41].

2.1.2 External communication

As mentioned, modern connected vehicles can enable a wide range of additional functionality by communicating with the external world, through interfaces such as Wi-Fi, Bluetooth or external gateways in the CAN-bus used for vehicle-to-infrastructure (V2I), vehicle-to-vehicle (V2V) [19] or even vehicle-to-everything [4] communication. For instance, these features include receiving traffic reports and service information and allowing the passengers to access the internet from the vehicle. While the usage of external communication can extend the capabilities of the vehicle and introduce conveniences for its users, it also introduces additional attack surfaces that could potentially be targeted by attackers.

One mode of communication is Vehicle-to-Vehicle (V2V). V2V allows a vehicle to communicate with other vehicles directly, often by utilizing a protocol called VANET (Vehicular ad-hoc network). This protocol allows vehicles to create and maintain temporary ad-hoc networks, based on a peer-to-peer architecture. This direct communication allows vehicles to share information with each other. Such information could be the exchange of traffic information to prevent accidents by informing vehicles to maintain safe distances.

2.1.3 Controller Area Networks

The controller area network (CAN) is a broadcast-based communication bus [41] that was originally developed in the early 1980s by the German automotive supplier

Robert Bosch in an effort to reduce the complexity and weight of wiring within vehicles [25]. The latest version of the CAN protocol, known as CAN 2.0, was published in 1991 [2] and is still used as the primary means of communication between ECUs in modern vehicles [41].

Data is broadcast over the CAN-bus in frames, and each frame that is sent over the CAN-bus is received by all of the devices that are connected to the bus [2]. The general shape of a CAN-frame is shown in figure 2.2.
Each CAN-frame contains an arbitration ID, which is used for both identification and prioritization of CAN-messages. When several ECUs try to broadcast messages over the CAN-bus simultaneously, the message with the lowest arbitration ID will be prioritized [41].
In addition, each CAN-frame also includes a data payload, which carries control instructions and status information regarding the vehicle state. Additional low-level bus control fields are also included in the CAN-frame, such as the DLC field (which specifies the number of bytes in the data payload) [41].

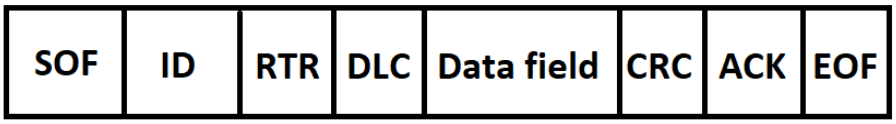


Figure 2.2: CAN frame format

While the use of CAN-buses in automotive- and other applications does serve to reduce complexity and wiring costs, the simplicity of the protocol also introduces certain weaknesses into the network [41]. For instance, since messages are broadcast to all devices connected to the CAN-bus, the confidentiality of messages cannot be guaranteed on the CAN-bus. The CAN-protocol does not require any message encryption to be implemented on the CAN-bus [2], and while it is still possible to adopt end-to-end encryption for messages sent over the CAN-bus, this remains a difficult task due to the limited processing power of the ECUs. In addition, devices connected to the CAN-bus do not have any means of validating the origin of a CAN message, which enables many different network injection attacks such as *Spoofing* or *Flooding* attacks [41].

2.2 Intrusion detection systems

The National Institute of Standards and Technology (NIST) specifies intrusion detection as the process of monitoring events in networks and computer systems in order to identify whether these events are malicious or not. If an event is considered to be malicious, this event will be classified as a security incident [33]. An incident can have several causes, ranging from malicious code injections to deliberate intrusion attempts. A security incident can also be caused by authorized users trying to alleviate or in other ways misuse their privileges on a system [33].

2.2.1 IDPS Technology

An Intrusion Detection System is a software that automates the detection process and triggers alerts if security incidents are detected. The difference between an IDS and the closely connected Intrusion Prevention System (IPS), is that an IPS has the same detection possibilities as an IDS but can also take action in preventing security incidents from taking effect. As an example, one way to enforce this prevention could be for the IPS to close the connection between the malicious source and the intended target. The IDS and IPS technology also operate under the shared name of IDPS technology. Since this thesis aims more toward attack analysis over attack prevention, the main focus in this thesis will be intrusion detection systems [33].

There exist several different types of intrusion detection systems but they all have some key functionality in common. This key functionality is the ability to perform logging of information related to an incident, alerting security administrators of observed events and producing reports that summarize the observed events that were identified as security incidents [33].

2.2.2 Network- and Host-based IDS

Deepha et al [7] discuss several different types of IDSs where the difference between them is the area of the system that the intrusion detection monitors. Two types of IDSs are network-based- and host-based IDSs.
A network-based IDS monitors network traffic and established connections by implementing techniques like packet sniffing to identify malicious activity and attacks. On the other hand, a Host-based IDS monitors system calls and application log files when implementing its detection method.

2.2.3 Detection methods

There are several different types of detection methods that can be used to detect an intrusion. Some of these are signature-based detection, anomaly-based detection, specification-based detection and stateful protocol analysis [33].

Signature-based detection operates by checking traffic or events on a system for signatures that are known to be malicious. This detection method is very efficient at detecting threats that are already known, but can not detect novel attacks [33].

Anomaly-based detection is another detection method which, as opposed to signature-based detection, can detect previously unknown attacks [34]. Anomaly-based detection often operates by implementing machine learning algorithms to learn what a system's normal state is. The IDS does this by performing a training period where it monitors a specific system, creates a profile to act as a baseline of what is to be considered as normal behavior and if events on the system deviate from what is considered to be normal, the IDS will classify the event as an intrusion.

A common issue with anomaly-based detection is that many of the detected attacks are false alarms, also known as false positives. The machine learning model is trained on the system to know what is normal user behavior, however a lot of events on any system are often exhibiting currently unseen, but legitimate behavior. This means that even though the anomaly-based detection method can be proficient in detecting novel attacks, many of the triggered alerts will be false alarms [34]. Another issue with anomaly-based detection is that it could by accident include malicious behavior when learning what the normal state is. This would of course be detrimental to the accuracy of the IDS. However, as previously mentioned, the main advantage of using an anomaly-based detection method is its effectiveness in identifying novel attacks [33].

Stateful protocol analysis is another detection method that operates by keeping a state of protocols in order to identify malicious behavior. This behavior is based on sequences of commands that by themselves may not be malicious, but as a sequence can constitute malicious behavior [33].

Specification-based detection is similar to anomaly-based detection in that it also detects attacks as behavior that differs from a normal state. However, unlike anomaly-based detection which usually relies on machine learning techniques, the specification-based detection method instead manually specifies what normal behavior is, and triggers an alert when system events deviate from this specified behavior. The benefit of using specification-based detection is that it has been shown to produce a low amount of false positives [34]. However, a limitation with specification-based detection is that it can be very time-consuming to specify what the normal behavior of a system should be. Additionally, it is not as proficient in detecting novel attacks as the anomaly-based detection method [34].

Even though several different detection methods exist, there is nothing that prevents utilizing a hybrid-based IDS that combines two or more of the described detection methods to increase the accuracy of the IDS, as has been performed in [13].

2.3 Standardisation of intrusion detection systems for vehicles

The main purpose of embedding intrusion detection systems within vehicles is to detect security incidents at an early stage. To achieve this, vehicular IDSs log information regarding detected security events that it deems to be alarming so that this can be evaluated at a later stage [22]. In addition, these events are usually reported to a central backend, which can perform further aggregation and analysis of the reported security incidents, to try to figure out the root cause of the problem. If an incident is determined to have been caused by some security issue within the vehicle, some countermeasure can be developed and deployed [22].

2.3.1 AUTOSAR

AUTomotive Open System ARchitecture (AUTOSAR) is a global partnership between a wide range of parties active within the automotive domain and the development of automotive software. AUTOSAR was founded in 2003 and its primary directive includes establishing an open and standardized software architecture for ECUs. By establishing an open software architecture standard that can be implemented by any developer of automotive software, AUTOSAR hopes to improve the transferability of software between different vehicle models and manufacturers, improve the scalability of automotive systems and ensure that availability and safety concerns are properly considered within these systems. Today the AUTOSAR standards are widely used within the automotive industry and are considered the de-facto standard within the automotive domain [39]. There are currently more than 280 companies participating in the partnership [1].

2.3.2 Standard for in-vehicle network intrusion detection

AUTOSAR is currently in the process of introducing a new standard for how intrusion detection systems are to be implemented within automotive systems. This standard is a part of the *Foundation* platform and is primarily described in [28] [36] [37] [38]. Some of the key terms used in the standard are highlighted in table 2.1.

Table 2.1: Key terms used in the AUTOSAR standard for intrusion detection

Term	Abbreviation
Security sensors	-
Security event	SEv
Qualified security event	QSEv
Intrusion detection manager	IDSM
Intrusion detection reporter	IDSR
Security operations center	SOC
Security event memory	SEM

The AUTOSAR standard described in [28] [36] [37] [38] takes into consideration that a vehicle in effect operates as a distributed system in and of itself. The vehicle consists of many ECUs that are connected through the IVN. The IVN is in turn divided into several different domains and subsystems and consists of multiple different kinds of communication buses such as CAN, LIN, MOST, etc. As such, there exist many different attack surfaces within the vehicle that could be targeted by attackers.
Since attacks targeting the vehicle do not only occur in one central point of the IVN, it is not enough to embed an IDS at a single point within the system architecture. Therefore, the AUTOSAR standard calls for a distributed onboard IDS, which embeds *Security sensors* and *IDSMs* throughout several locations throughout the IVN architecture [28], as shown in figure 2.3.

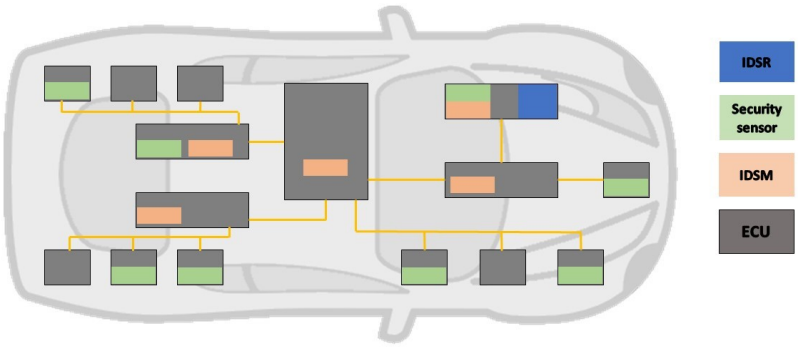


Figure 2.3: Illustration of the AUTOSAR intrusion detection standard

2.3.3 Security sensors and security events

The security sensors are spread throughout the IVN in various ECUs and gateways, and actively try to detect security events (SEvs) when they occur [28]. The precise definition of a SEv can vary between different sensors based on the tasks performed by the ECU it monitors and can be configured by the vehicle's manufacturer by sending updated SEv definitions to the sensor using an *Security Extract* [28]. The security sensors may utilize a vast variety of detection techniques to identify SEvs, such as anomaly- or specification-based detection.

2.3.4 Intrusion detection manager (IDSM)

When a security sensor detects that a SEv has occurred, it passes this information along to the local IDSM. IDSMs are placed in every key domain within the IVN. The IDSMs main tasks are to receive the SEvs from the security sensors, buffer them and then process them. In addition, an IDSM can be used as a security sensor as well, generating SEvs for later processing [36] [37].
When an IDSM processes SEvs, it applies a set of consecutive filters to the security event. These filter chains consist of slightly differing layers depending on whether the *AUTOSAR Classic Platform* or *AUTOSAR Adaptive Platform* standard is being

implemented. However, the filters included in both standards serve similar purposes [36] [37]. For instance, both include a sampling filter, where vehicle manufacturers can configure the IDSM to only consider a sample of the reported SEvs. Both standards also include an aggregation filter, which aggregates all reported SEvs of a given type for a configured time interval and then processes these as a single SEv instead. Both also implement a threshold filter, which can be configured to drop SEvs of a given type if their count is smaller than a certain threshold value [36] [37].

The main purpose of these filter chains is to allow vehicle manufacturers to better manage the resources used within the vehicle. Resources such as processing power, RAM and nonvolatile memory are scarce in the ECUs of a vehicle. These limitations force vehicle manufacturers to pursue optimizations that usually aren't required in traditional IT systems [22]. In addition, a vehicle is capable of generating a very large amount of security events. As such developers of automotive software must consider these resource limitations when configuring the IDSMs.

Security events that pass through the entire filter chain of an IDSM are deemed as qualified security events (QSEv) [28].

2.3.5 Intrusion detection reporter (IDSR) and security event memory (SEM)

Once a QSEv has been identified by an IDSM, the IDSM needs to decide what to do with the logged data. How the QSEv data should be handled by the vehicle largely depends on the configuration decided by the vehicle manufacturer. Broadly speaking, there are two potential ways of handling the QSEv data. One way of handling it is to send the logged data to a local dedicated storage location within the vehicle, an SEM [28]. The number- and placement of SEMs within a vehicle are left up to the vehicle manufacturer.

Another way of dealing with the QSEv is to have the IDSM forward the logged data through the IVN to the vehicle's intrusion detection reporter. The IDSR is a module normally placed within the vehicle's telematic control unit. The IDSR then forwards this information along to a centralized backend, the SOC, for further analysis and response. In addition, the IDSR has the ability to enrich the QSEvs before sending them to the SOC by attaching relevant context data such as a timestamp or geolocation [28].
The flow of security events described in sections 2.3.3 through 2.3.5 is summarized in figure 2.4.

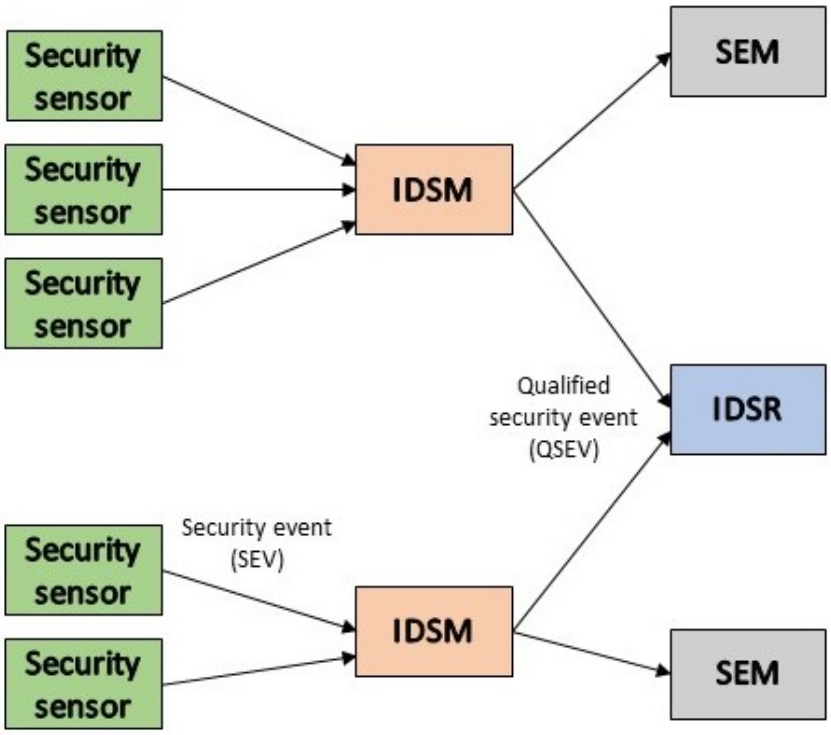


Figure 2.4: Security event flow

2.4 Threat modeling in the automotive domain

In order to design a resilient framework for collaboration between vehicles, it is important to first consider the potential threats that such a system would be exposed to. This is commonly done through the process of threat modeling. Myagmar et al. [23] describe threat modeling as the systematic process of identifying all the potential threats to a system. This process requires an in-depth understanding of the system and its complexities and should ideally be incorporated into the system design process as early as possible. Threat modeling is useful for developing meaningful security requirements and prioritizing between different security requirements based on their criticality and likelihood. Therefore, using this approach reduces the attackers' ability to misuse the system, while also helping the system designers to anticipate attack goals [23]. Furthermore, threat modeling provides system designers with the tools to answer questions regarding exactly what the system is designed to protect, and from whom [23].

According to Ma and Schmittner [20], the following steps are generally included in software threat modeling:

- Modelling of the system by means of drawing the system architecture in a Data-flow Diagram (DFD), adding system details to each node in the DFD and drawing trust boundaries
- Identification of various threats originating from data flows by using some threat identification methodology
- Addressing each threat by redesigning the system, performing threat mitigation actions, or ignoring the threat if the risk is acceptable
- Validation of the threat modeling diagram against the actual system

Within the academic community, the subject of threat modeling has been extensively researched and much research effort is still being put towards this field. In addition, threat modeling is actively used and researched upon within the industry, and many threat identification methodologies have been developed for various purposes and functions. These include the STRIDE, DREAD, SWOT, and OCTAVE threat models [15].

Within the automotive domain in particular, many research papers have been published regarding suitable threat modeling practices within the field. Many of these try to build upon the STRIDE model [20] [12], while others try to combine several established models to cover all threats within the domain in question [10].

Rosenstatter et al. [30] use the latter approach, and try to establish an attack model specific to the domain by identifying the different asset types surrounding vehicles that are vulnerable to attack and various examples of attacks towards these assets. These asset types include:

- **Hardware** - One attackable asset is the hardware infrastructure of the vehicle, i.e the hardware of ECUs, sensors and actuators in the vehicle. These are exposed to attacks that can disrupt, directly intervene with or otherwise com-

promise the availability and integrity of these devices. Typically, attacking the hardware often requires physical access to the vehicle. One such attack could be that an existing component of the vehicle is replaced with a malicious one. Another potential attack could be performed by installing ill-intended hardware to be used as a mediary in order to gain control of the vehicle.

- **Software** - Attackable assets with regards to software include the software of ECUs, libraries and operating systems. These assets are exposed to the manipulation of software, measurements or control signals. Examples of attacks towards the software assets are many but generally attacks that in any way manipulate the software are considered. An example of such an attack is a privilege escalation attack, which enables attackers to reprogram devices such as ECUs and can by doing that gain remote access to a vehicle.

- **Network/Communication** - Vehicle assets with regards to the network and communication include the controller area network (CAN), mobile-, WiFi- and Bluetooth networks to name a few. The general threat against the network and communication domain are communication failures, protocol vulnerabilities or attacks that generally compromise the confidentiality, integrity and availability of data in the respective network. As an example, a spoofing attack can be used to act as a legitimate node by suspending an authentic ECU and send fabricated messages that seem to come from the same ECU.

- **Data Storage** - The asset of data storage is also an attackable surface where information such as user data, logs, forensic data and cryptographic material can be exposed. If the storage of such data is not secured against malicious handling, this data can be subject to various attacks such as removal of forensic data or reverse engineering of secret keys by extraction and analysis of firmware.

3
Related work

This chapter begins by briefly detailing some of the previous works that have been made towards the field of collaborative intrusion detection as well as the current state of vehicular IDSs.
Thereafter, the concept of intrusion detection networks and the research behind this is explained in further detail. Finally, some of the challenges of implementing and managing IDNs are described.

3.1 Previous works

A common issue with IDSs that implement anomaly-based detection is that they are prone to produce many false alarms, as mentioned in section 2.2. [40] proposed in the year 2003 the use of a Collaborative Intrusion Detection System (CIDS) and showed that having IDSs collaborate could improve the accuracy of detection compared to using only one IDS for detection.

[14] presents how collaborative IDSs efficiently can decrease detection time due to the correlative nature of many attacks. Additionally, [14] also shows how this decrease in detection time effectively decreases the number of triggered alarms.
Furthermore, Fung and Boutaba propose in their study "Design and management of collaborative intrusion detection networks" [9], how IDSs can collaborate with each other and thereby create an intrusion detection network (IDN) to both increase detection accuracy but at the same time mitigate common challenges facing these IDNs. These challenges are further outlined in 3.2

[47] proposed an implementation of a collaborative intrusion detection system for identifying betrayal attacks against a VANET network. This collaborative IDS would utilize dynamic behavior analysis in order to detect malicious behavior from any of the vehicles in the network. Thus this paper proposes a collaborative intrusion detection approach to better meet the challenges of trust management.

State-of-the-art accuracy of vehicular IDSs varies depending on the detection method used and how this detection method is implemented. For example, in a car hacking challenge in 2020 [13], one of the best performing IDSs used was one implementing a rule-based detection method that produced an F1-score of 0.869. However, the runner-ups in this competition did hold an F1-score of 0.864 when using a hybrid-based IDS implementing several different detection methods. Additionally, [41] man-

aged to gain an impressive F1-score of 99.9 percent when using a hybrid-based IDS implementing both rule-based and machine learning based (RNN) detection methods. These measures are interesting as comparison metrics to the F1-scores obtained and presented in section 7.

This thesis will continue the research on how collaborative intrusion detection networks can be used to achieve gains in attack detection, but also extending this research by proposing a generic attack-detection framework for the automotive domain that not only implements a collaborative intrusion detection network to increase the accuracy of the attack detection but at the same time adhering to the newly defined AUTOSAR standards.

3.2 Collaborative IDS - IDN

According to [9], a downside with having a single IDS responsible for detection is that an IDS only has knowledge of information that has traversed the area in a system or subsystem which that specific IDS is set to monitor. Therefore it is not presented with a global view and thus is not efficient at detecting fast-spreading new attacks. [9] concludes that an IDS could become more effective by collaborating with other IDSs and sharing information between them. A network of collaborating IDSs forms an Intrusion Detection Network (IDN) where participating IDSs benefits from knowledge and experience sharing to enhance the accuracy of the detection [9].

There are two main types of IDNs, *information-based* and *consultation-based* [9]. An information-based IDNs continuously share information such as detection data and possible new attacks that the separate IDSs have found suspicious. This type of IDN is proficient at detecting fast-spreading new attacks. An example of such a fast-spreading attack could be a worm, which has proven to be a very efficient way to infect many different systems in the past, where the Morris worm can act as a famous example. Even though the information-based IDN is efficient at detecting fast-spreading new attacks, they often pay for this increased effectiveness with a large message overhead. Messages, of which some may not even be useful to the other IDSs [9].

A consultation-based IDN operates in the way that when an IDS detects suspicious activity but needs more confidence to declare it as an incident, the IDS can send consultation messages to other IDSs in the IDN to get feedback from other IDSs in the network. The requester can then use the provided feedback to make a more informed, final decision in whether to classify the activity as a security incident or not. The consultation-based IDN has less communication overhead than information-based IDNs [9].

3.2.1 Challenges with IDNs

One challenging aspect of IDNs is that of trust management. If an adversary could manipulate some of the IDSs on the network and send false or untrustworthy in-

formation to uncompromised IDSs on the network, that could effectively lower the detection accuracy when using the falsified feedback for the final decision [9]. A possible countermeasure for this scenario is to implement some protocol for trust management to decrease the risk of making such a scenario successful.

Common protocols for trust management in distributed systems, which an IDN can be defined as, is to implement some consensus protocol such as a byzantine fault agreement protocol. Such a protocol essentially works by having a majority vote among the nodes to rule out malicious nodes [9]. Of course, using this approach does mean that if an adversary could control a large amount of the nodes on the network they could then thwart the consensus protocol in their favor or in other ways harm the decision process. [9] proposed the use of a "Dirichlet-based trust model" which uses Bayesian statistics to calculate the trustworthiness of other peers and uses this as a weighted factor in choosing how much to trust the information of certain IDSs before calculating the final decision based on the given feedback.

Additional challenges with IDN networks include how to use the collective information from the other IDSs and calculate a final decision. [9] proposed a Bayesian optimization problem and specified whether to raise an alarm or not based on a threshold value. This approach proved to have the lowest cost compared to a simple average cost function [9].

Another issue with IDNs is how to manage resources. IDSs usually have limited resources and may therefore not be able to respond to every request from other IDSs. Also, how should the IDSs choose which collaborators to contact for feedback and how to manage this list of collaborators as this will also require additional resources. [9] provides a solution to these questions as well, but this is considered to be outside the scope of this thesis and will therefore be omitted.

4
Methodology

This chapter describes the methodology used when producing this thesis. A brief introduction to design science methodology is provided and followed by a road map as to how this methodology will be applied. This road map spans from the initial step of conceptualizing the problem, to the validation of a designed solution that addresses this problem.

4.1 Design science methodology

In accordance with Runesson et al [31] a design science methodology was implemented for conducting this research. [31] defines design sciences as the "aim to understand and improve human-made designs in an area of practice". [31] continues with specifying that design science is a suitable research methodology for software engineering since the software itself is made by humans.

Design science research generally studies specific problem instances in order to address general problems. The design science paradigm is comprised of three distinct steps. These are *problem conceptualization*, *solution design*, and *validation*. Furthermore, the activity of problem conceptualization is to be performed in connection to an envisioned solution. Thus, the problem conceptualization step is often tightly connected to the activity of solution design, where alternative solutions, as well as related research, are to be considered. Additionally, per Runesson et al. [31], the steps of problem conceptualization, solution design and empirical validation are to be executed in a cyclic iterative manner until a finalized version is reached.

In this study, we aimed to design a framework for collaborative intrusion detection through the study of automotive intrusion detection systems and intrusion detection networks (IDNs). The designed CIVID framework has been refined through the use of established principles such as threat modeling and has been exemplified through a system implementation. Finally, this exemplification of the framework has been validated through an empirical evaluation [31] as shown in chapter 7.

Furthermore, Knauss [17] proposes 7 actionable guidelines for utilizing a design science methodology in research thesis work. The aim of these guidelines is to assist students in balancing the trade-off between academic values and practical relevance.

The guidelines proposed by Knauss include:

- Define the artifact early
- Work in iterations
- Define research questions with respect to the regulative cycle, i.e. one related to the problem, one related to potential solutions and their construction, and one related to evaluation
- Schedule regular meetings
- Shift emphasis between cycles
- Have a dedicated section to describe the artifact
- Write the thesis document as you go

The guidelines laid out by Knauss [17] have been adhered to during the process of producing this thesis. The produced artifact, i.e the proposed framework, has been defined early on as a solution to the aforementioned problem conceptualization. Throughout the thesis work, one-week iterations have been applied and meetings with the academic supervisor have been scheduled once every iteration. Furthermore, meetings with academic and industry advisors have been scheduled roughly once per cycle, i.e roughly once per month. The defined research questions have been defined with respect to the regulative cycle. The first research question addresses both the problem and potential solution. The second research question addresses the evaluation of the potential solution.
Focus has oftentimes shifted between different aspects of the solution design and evaluation between iterations. Furthermore, chapter 5 is dedicated to describing the produced artifact in full detail. Finally, the thesis document has been continuously written during the course of this thesis.

4.1.1 Problem conceptualization

Initially, in order to acquire domain knowledge, an extensive literature review was conducted. During the aforementioned review, the writers of this thesis searched the databases of Elsevier and Google Scholar for research papers by keywords: "connected vehicles", "cyberattacks on connected vehicles", "in-vehicle intrusion detection", "collaborative ids", "ids in connected vehicles". In addition, meetings were held with scholars and domain experts in the field of connected vehicles, which gave further insight into the true nature of the problem that required solving. These activities provided clarity into the actual problem to be solved and a provided guidance in the design of the proposed collaborative intrusion detection framework.

During subsequent iterations of the framework design process, additional queries for available literature were performed in order to provide further context information and clarity.

4.1.2 Solution design

During the initial literature review, a more clear view of the problem that had to be solved was gained. Following this, work began on designing the proposed CIVID framework. The design of the framework was informed by previous works on collab-

orative intrusion detection systems and automotive intrusion detection.

Concurrently with the design of the theoretical framework, the authors worked on the implementation of the framework exemplification. This implementation was realized by simulating a CAN-network by using the Python programming language and implementing a collaborative anomaly-based intrusion detection system on top of this. The setup and design of this implementation are described in more detail in chapter 6.

4.1.3 Validation

Once a viable implementation of the proposed CIVID framework had been finalized, the validation phase of the research was started. During the validation phase, empirical validation of the implemented system was performed. Details regarding how this empirical validation was performed, such as how data was collected as well as how a baseline for comparison was established, are described in further detail in section 6.5.

The primary metric used to evaluate the accuracy of the implementation is the *F1-score* measured during the evaluation. The F1-score is used in statistical analysis and is a measure of a test's accuracy. It is calculated by the test's precision (P) and recall (R) with their harmonic mean as result. The precision is calculated by a given test's True Positives (TP) divided by the sum of the test's TP and its False Positives (FP) [29] [8].

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

The recall is calculated by the test's TP divided by the sum of the test's TP and its False Negatives (FN) [29] [8].

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

The formula of the harmonic mean will produce the F1-score which will give a value ranging from 0 to 1, with 1 being the most accurate [29] [8].

$$F1\text{-}score = \frac{2}{precision^{-1} + recall^{-1}} \tag{4.3}$$

The F1-score is also used in multiclass classification where the same formulas apply, but where a mean is calculated from the results of the classes F1-score. This mean can be weighted in different ways, depending on the desired outcome, but as an example, the calculated mean can be weighted by each class's occurrence in the test data [24].
The F1-score is a common metric used when evaluating the accuracy of intrusion detection systems [41] [13], hence the choice of applying this metric for evaluation in this study as well.

An F1-score will be calculated for each sample data point collected. The data collection phase is thoroughly described in section 6.5. In order to test if statistical

significance holds between the baseline and the collaborative settings, a statistical test is to be used. Naturally, the statistical test to be used will need to be performed on data that adheres to the assumptions of the intended statistical test. Given that all of the assumptions hold to be true, a *One-way repeated measures ANOVA test* is deemed suitable for this thesis, as the simulations described in section 6.5 are performed in a triplewise manner. The repeated measures ANOVA test is used in order to see if there are statistically significant changes in the mean of a measure when different treatments are applied to the same subjects. The core difference between the repeated measures ANOVA test and the standard ANOVA test is that measurements are collected from the same subject several times in a repeated measures ANOVA test.

The assumptions of the repeated measures ANOVA test are the following [45]:

- The independent variable needs to be on a categorical (nominal or ordinal) scale and the dependent variable needs to be on a metric scale (interval or ratio)
- The subjects are randomly selected from the full population
- There should be no outliers present in the data
- The dependent variable, i.e. the measured F1-scores, should be normally distributed in the population for each level of the within-subjects factor, i.e. the different simulation modes presented in section 6.5
- The population variance of difference calculated between any two levels of a within-subjects factor is the same regardless of which two levels are chosen. This assumption is also referred to as the sphericity assumption or as the homogeneity-of-variance-of-differences assumption, and is typically tested using *Mauchley's test for sphericity*

In addition, the average elapsed real-time (i.e wall-clock time) from when a CAN-message arrives until the analysis of it is complete, was used as a secondary metric to evaluate the time efficiency of using a collaborative intrusion detection approach in comparison to only performing intrusion detection activities locally in one vehicle. The metric of elapsed real-time will be used in order to showcase the real-world value and applicability of the proposed collaborative intrusion detection approach, and as such will be used to assess the usefulness of the framework. Details regarding the measuring of this metric are also provided in section 6.5

5

The proposed framework

This chapter outlines the collaborative in-vehicle intrusion detection framework (CIVID) proposed by this thesis. This includes explanations regarding each component included in the framework, their respective roles and responsibilities, the flow of information etc. Furthermore, the various aspects of the framework that are left ambiguous and open for different implementations, such as the utilized detection methods, are explained.

5.1 CIVID

As described in section 3.2, there mainly exist two different approaches to designing a collaborative intrusion detection system, an information-based- or a consultation-based approach [9]. There are pros and cons to each approach. For instance, the information-based approach is highly useful for detecting new and fast-spreading attacks, but introduces a lot of message overhead. The consultation-based approach on the other hand introduces far less overhead, but at the cost of limiting the sharing of information between different nodes in the network.

For the automotive domain in particular, the consultation-based approach is deemed the most suitable out of these, primarily because of resource limitations. As previously discussed, the various ECUs within a vehicle are usually highly limited in both processing capabilities and memory [22]. In addition, these devices are often in charge of handling safety-critical functions in real-time, which should be considered before introducing additional overhead to these devices. For these reasons, the overhead introduced by using an information-based approach is considered to be too great.

A key assumption made in designing the CIVID framework is that other vehicles or the security backend have access to some resource or information that is not available to the requesting vehicle. In other words, it is assumed that there are potential gains to be had from using a collaborative approach. The resources that may be only be accessed by collaboration partners could be anything from network traffic logs, computational power or authorization keys. For instance, in the case of auto-adaptive intrusion detection systems, which are continually trained over time [46], other vehicles may be better equipped to identify whether or not some network traffic constitutes an intrusion.

The CIVID framework has been designed to be compatible with the AUTOSAR in-vehicle intrusion detection standard described in section 2.3 and uses the same notations as used within the standard. This standard has provided the CIVID framework with its structure, components and flow of information. However, it is important to note that the collaborative approach utilized in the CIVID framework is not limited to only functioning within the structure provided by the AUTOSAR standard, since the collaborative intrusion detection approach has been proven to be valid in contexts outside of the automotive domain as well. Neither is the framework meant to serve as an extension of the AUTOSAR standard itself.

The different components included in the framework and their respective roles are summarized in figure 5.1. The flowchart shown in figure 5.2 shows the general flow of information for requesting consultation within the framework. The key properties of the framework are described below:

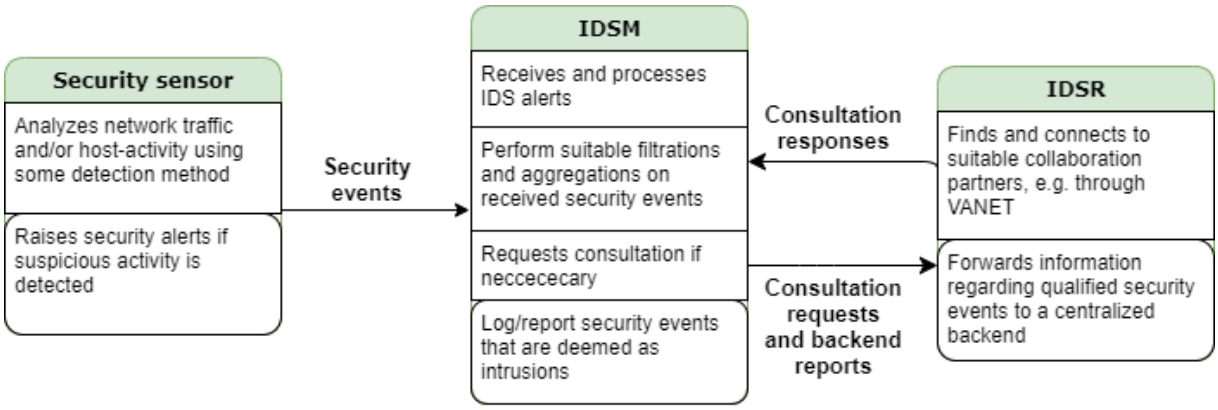


Figure 5.1: Components involved in the collaborative IDS framework

- **Security sensor** - Just as in the AUTOSAR standard, the security sensor is responsible for raising alerts whenever malicious or suspicious activity is detected. This can be done through various methods, as described in section 2.2.3. The method used to decide when to raise an IDS alarm is denoted as *Decision algorithm 1* in figure 5.2.

- **IDSM** - The IDSM also serves the same purpose that it does within the AUTOSAR standard, e.g. to filter and aggregate the various intrusion alerts raised by the security sensors within its domain. However, the CIVID framework introduces a couple of additional tasks to be performed by the IDSM.

  Firstly, the IDSM is tasked with deciding when to request consultation in regards to a security event or incident. This is denoted as *Decision algorithm 2* in figure 5.2. Depending on how the framework is implemented, this decision can be influenced by a wide range of factors such as availability of resources, the certainty of the prediction, etc.

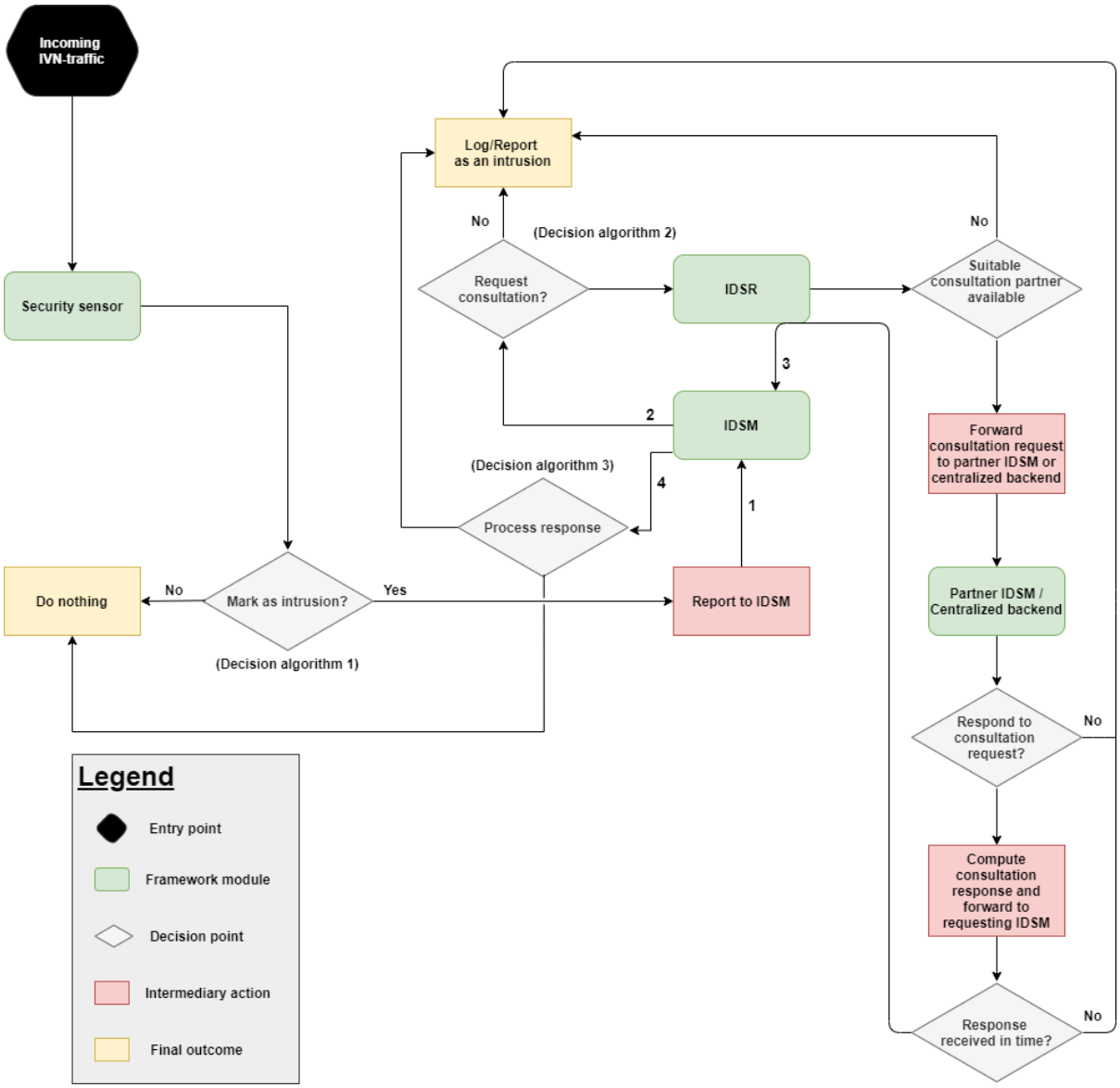


Figure 5.2: Collaborative IDS framework data-flow

Secondly, the IDSM needs to decide how it should process received responses to its consultation requests. In other words, the IDSM needs to implement an algorithm for deciding the final outcome of a security event based on the received consultation responses, i.e whether or not to mark the event as a qualified security event (QSEV) (And if so, whether or not to attach additional context information to the event). There exist several ways of implementing such algorithms, as discussed in section 3.2.1. For instance, such a decision might be based on the certainty of the response, trust relations with the consultation partners and aggregate statistics in cases where several consultation responses are received. This decision is denoted as *Decision algorithm 3* in figure 5.2.

Finally, the IDSM should be able to respond to consultation requests received

from other vehicles. As mentioned in section 2.3.4, the IDSM itself may serve as a security sensor. This is leveraged in order to enable the IDSM to analyze incoming consultation requests and formulate a response. Of course, the IDSM may opt to ignore incoming consultation requests for various reasons, such as resource constraints or security considerations.

- **IDSR** - The IDSR is responsible for finding and maintaining relations with potential consultation partners. Depending on the configuration of the IDN, the IDSR may communicate with a security backend, with other vehicles directly or both. If the IDN is configured to only allow direct communication between a single vehicle and the security backend, the act of requesting consultation becomes relatively simple. However, some complications may remain, such as how to keep the security backend well-informed and potential latency due to geographical distance.

  On the other hand, if the IDN is configured to enable direct communication between different vehicles in a peer-to-peer manner, such as by utilizing the VANET protocol mentioned in section 2.1.2, several additional complications need to be considered. One such complication is the issue of trust management, which was discussed in section 3.2.1. Additionally, the differing configurations between different vehicles may render them unable to assist each other in their intrusion detection efforts. For instance, network traffic that may be considered unusual in one vehicle may not be unusual in another. As such, the IDSR needs to implement a check to make sure that its collaboration partners are suitable for specific collaboration requests.

6
Setup and experimental design

In this chapter, the experimental setup and each step taken for the implementation of the CIVID exemplification discussed in chapter 4 is described in detail.
Firstly, the dataset of network injection attacks against a CAN-network that was used for the implementation is explained.
Secondly, details are provided regarding the utilized intrusion detection model, which was created using the machine-learning resources *TensorFlow* and *Keras*.
Thereafter, additional details are provided on how the virtual CAN-network used to simulate actual CAN-traffic was implemented and which components are included in the virtual network.
Then, a description is given of the hardware components used for the simulations and how these were connected to each other.
Finally, a detailed description is given of how data collection was performed during the simulations. This includes how the various simulation runs were chosen, what data was collected and how it was collected.

6.1 Acquiring a dataset of network injection attacks against a CAN-network

For the evaluation of the proposed framework, the authors have chosen to focus primarily on detecting attacks against the *Network* asset, as described in section 2.4. As such, a dataset of network injection attacks against a CAN-network has been utilized.

The dataset used in this study has been acquired from the HCRL (Hacking and countermeasures research lab), a research laboratory affiliated with the School of Cybersecurity, Korea University [16] [13]. HCRL mainly performs research in data-driven security, using machine learning and data mining technology on large datasets. Their publicly available datasets are collected from authentic services that range from mobile payment data, e-commerce transaction data, as well as car-driving and attack data. HCRL contributes to the field of data-driven security by sharing its datasets with the public.

The dataset used in this thesis was made publicly available by HCRL following a cybersecurity competition hosted by HCRL together with the organizations Culture Makers as well as Korea Internet & Security Agency in 2020 [16]. The competition aimed to develop attack and detection techniques towards a Controller Area Net-

work, and the target vehicle used during the competition was the Hyundai Avante CN7 [16] [13].

The dataset acquired from this competition consists of CAN traffic collected from the Hyundai Avante CN7 and portrays normal traffic as well as malicious attack messages. The dataset is divided into 8 CSV files labeled as "preliminary" and one CSV file labeled as "final". For the "preliminary" datasets, the data was acquired when the vehicle was in a stationary state as well as a driving state. The dataset labeled as "final" however, only consists of data from the vehicle while in a stationary state, due to safety reasons [16] [13]. The size and structure of the dataset are shown in table 6.1. Out of the 9 CSV-files present, three were excluded from the model training and network simulation. Two were excluded for only containing "Normal" traffic and one was excluded for only containing 5 attacks in total.

The rows of the dataset contain raw CAN-frames as described in section 2.1.3, as well as a class- and subclass column. The 'class' column indicates if the particular message is normal traffic by labeling it "Normal", or if it is an attack by labeling it with "Attack". The 'subclass' column indicates which type of attack has been injected, if any, by labeling it with either "Normal" or with the name of the corresponding attack type. [16] [13]. These attack types can be Flooding-,Spoofing-, Replay and Fuzzing-attacks. Example rows extracted from the dataset can be seen in table 6.2.

Table 6.1: Size and structure of the utilized dataset [16]

Round	Type	#Normal	#Attack	#Rows (Total)
Preliminary	Training	3,372,743	299,408	3,672,151
	Submission	3.358,210	393,836	3,752,046
Final	Submission	1,090,312	179,998	1,270,310

Table 6.2: Example rows extracted from the dataset

Timestamp	Arbitration ID	DLC	Data	Class	Subclass
1599046395.285203	164	4	00 08 12 5C	Normal	Normal
1599046433.36685	000	8	00 00 00 00 00 00 00 00	Attack	Flooding
1599046458.3852608	553	8	00 00 00 02 01 00 80 00	Attack	Spoofing
1599046512.425457	164	8	54 88 A7 2C FB FE 51 08	Attack	Fuzzing
1599046630.5071099	329	8	4A C7 7E 8C 32 2E 00 10	Attack	Replay

Furthermore, HCRL has detailed that the dataset was constructed by logging the OBD-II port from the Hyundai Avante CN7 while performing message injection attacks. Each of the injection attacks was performed for 3-5 seconds with some interleaving time between each message [16] [13]. The implementation of the different

attacks was performed as follows:

The Flooding attack, also known as a denial-of-service attack, was implemented by injecting messages with an arbitration (CAN) ID of '0000' every 0.3 milliseconds. This attack aims to consume the bandwidth of the Controller Area Network. [16] [13].

The Fuzzing attack was performed by injecting random arbitration (CAN) id as well as completely random data for the Data[] attributes every 0.5 milliseconds. A fuzzing attack aims to create unexpected behavior in the system by sending unexpected input data [16] [13].

For the Spoofing attack, messages were injected with a particular CAN ID that targets the drive gear and RPM gauge respectively, every 1 millisecond. The spoofing attack aims to inject a CAN message in order to control some desired functionality [16] [13].

More comprehensive details regarding this dataset can be acquired from [16] and [13].

6.2 Building an anomaly based intrusion detection model

To implement the CIVID framework detailed in chapter 5, an intrusion detection algorithm is required for the Security sensors. As described in section 2.2.3, there exist several different kinds of intrusion detection methods, such as specification-based and anomaly-based. For this implementation, anomaly-based intrusion detection was used. More specifically, a simple deep neural network (DNN) model was built to detect anomalous CAN traffic. This machine learning model was built using the Python deep-learning resources TensorFlow [21] and Keras [5]. These resources allow data scientists to easily implement their own deep learning neural networks.

Since the aim of this thesis isn't to develop a flawless intrusion detection system for CAN-networks, the implemented DNN model was kept simple while still producing relatively accurate predictions. Figure 6.1 shows the layers included in the model. The eleven features used as input for the DNN model are as follows: The interarrival time of each CAN-frame, the frame's arbitration ID, its DLC as well as each of its 8 bytes of data, separated into 8 separate features. For CAN-frames containing less than 8 bytes of data, the model input was padded with zeroes in order to fit the expected model input.

The utilized dataset described in section 6.1, contains far more "Normal" CAN-traffic than network injection attacks. This means that the different output variables to be classified are highly imbalanced. Training a DNN-model on such a dataset will result in the model being more inclined to predict the CAN-traffic to be "Normal",

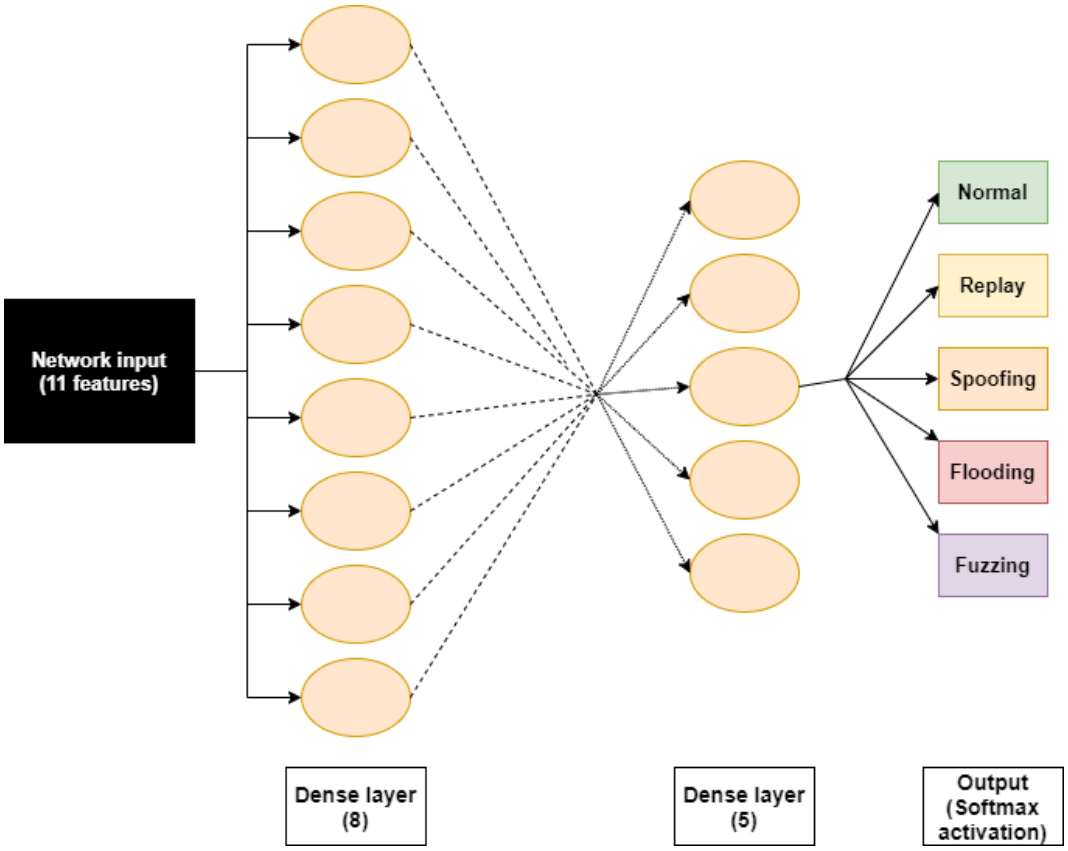


Figure 6.1: DNN-based detection model

even if this is not the case. In order to enable the DNN-model to better handle this kind of imbalanced datasets, threshold-moving has been applied [3]. As such, the different classes are weighted by their prior probabilities in order to penalize the model for misclassifying minority classes [3]. An exception to this is the 'Replay' class, which in its structure is indistinguishable from 'Normal' CAN-frames. Because of its simplicity, the designed model is incapable of distinguishing between 'Normal' and 'Replay' messages. As such, and since the detection methodology itself is not the focus of this thesis, the authors opted to give the 'Replay' class a weight of 0. In other words, the designed model will not attempt to detect replay attacks targeting the CAN-network.

6.3 Simulating a CAN-network

In order to showcase how the proposed CIVID framework could work in a practical application, several virtual CAN-networks were created using the Python programming language and the *python-can* package [11]. In addition the *python-can-isotp* package [27], which is a Python implementation of the ISO-15765 protocol, was used to facilitate higher-level communication between nodes in the CAN-network. This was for instance useful for sending alerts and consultation requests between the different nodes in the network, as these required a more complex message structure than the one provided by the low-level CAN-protocol.

Ideally, the framework would be tested directly on actual CAN-networks placed within vehicles. However, because of resource limitations and the difficulty of acquiring such hardware, a virtual environment was utilized instead. Once the virtual CAN-networks were initialized, the dataset described section 6.1 was used to simulate real-time CAN-network traffic.

To closely resemble a real-world setting, the virtual CAN-network was designed per the in-vehicle intrusion detection standard introduced in section 2.3. As such, separate modules were built for Security sensors, IDSMs and IDSRs. Each module is in charge of some aspect of the intrusion detection process, as described below:

- **Security sensor** - The security sensor is in charge of analyzing the CAN-network traffic that passes through its domain in order to try to detect possible intrusions. It does this by utilizing the intrusion detection model described in section 6.2. Whenever a possible intrusion is detected, an alarm is raised which contains relevant context information such as the type of attack that was detected, the "certainty" of the prediction as well as the row number (which is used for validation purposes). Once raised, the alarm is forwarded to the IDSM by utilizing the isotp (ISO-15765) protocol.

- **IDSM** - The IDSM is the primary decision-making module within the virtual CAN-network, and is in charge of logging intrusion alerts, deciding when to request consultation from another 'vehicle' and to receive consultation requests from other 'vehicles'. In addition, the IDSM serves as a Security sensor as well, as it processes consultation requests from other vehicles through its own intrusion detection model and formulates a response.
  In order to decide whether or not to request consultation for a particular security event received from the security sensor, the IDSM looks to the certainty of the prediction made by the sensor. In cases where the certainty of the prediction falls below a certain threshold value, the IDSM will request consultation for that particular security event. For the simulation runs described in section 6.5, a threshold value of 85% was used.
  When the IDSM decides to request consultation from other 'vehicles', it sends a consultation request to the IDSR using the isotp protocol. If a response is not received within a specified amount of time, the IDSM registers the intrusion alert as it was received from the Security sensor. However, if a response is received in time, the prediction with the highest 'certainty' is registered.
  In order to enable empirical validation of the implementation's performance by means of calculating the F1-score, no filtering or aggregation mechanism has been implemented within the IDSM despite this being supported by the CIVID framework described in chapter 5.

- **IDSR** - The IDSRs primary task is to connect the vehicle to other nearby vehicles and to forward consultation requests from the IDSM to these. In our simulated scenarios, up to three devices were connected through the IDSM

by utilizing the *python-can-remote* package [32], which creates a "CAN over TCP/IP bridge" which is compatible with python-can. However, in a real-world setting, some other means of facilitating communication between different vehicles would be more suitable, such as the VANET protocol. In addition, our implementation of the framework does not incorporate any trust management nor suitability checks as mentioned in chapter 5, as these components lie outside the scope of the empirical evaluation.

6.4 Hardware setup

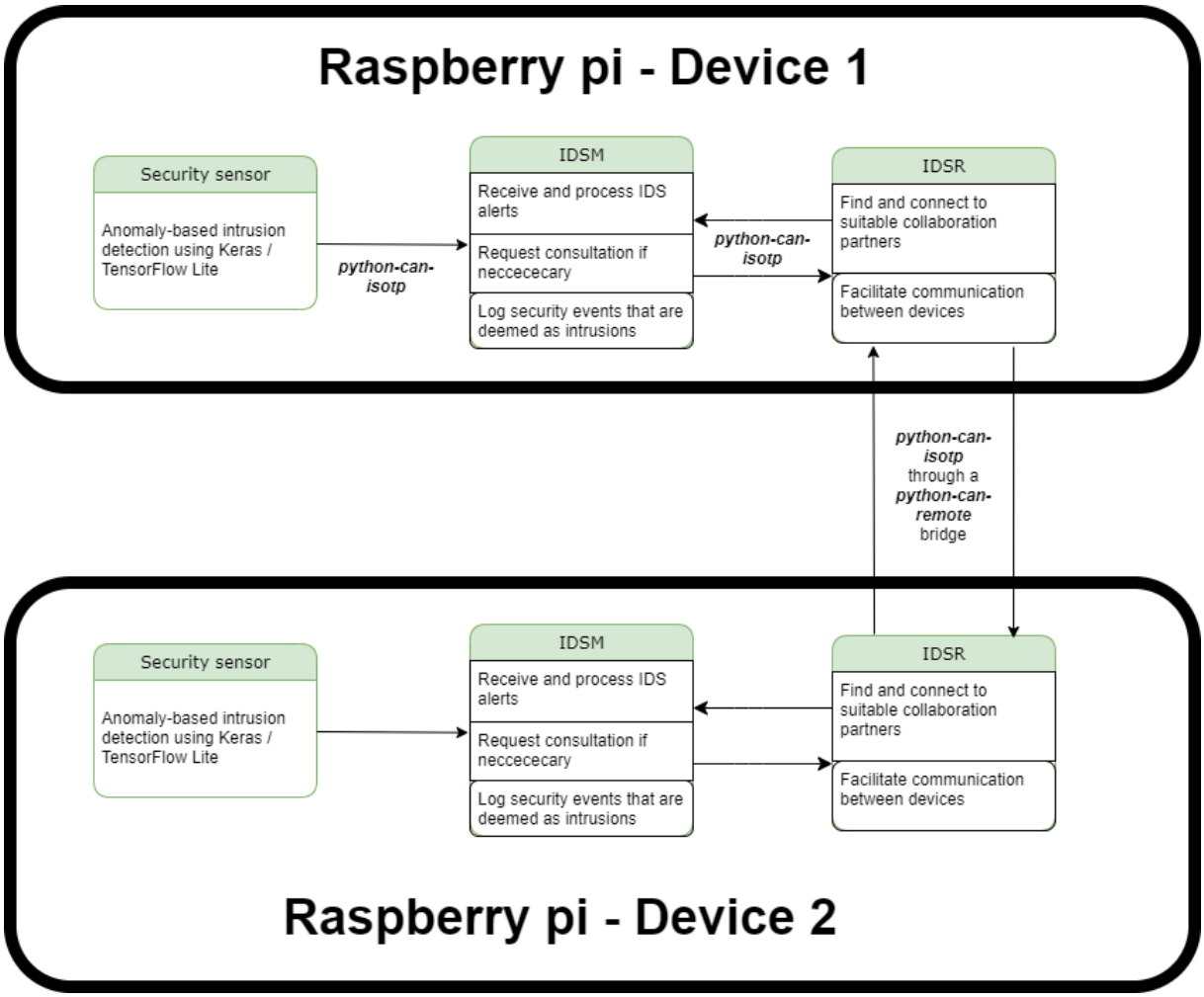


Figure 6.2: Modules included in the experimental setup

When evaluating the implemented system, up to three *Raspberry Pi 4, model b, 4GB* devices were used to simulate three vehicles collaborating with each other. The exact specifications of these devices is detailed in Appendix C. Three different configuration modes have been used when evaluating the system. These modes are denoted as "Baseline", "Collaborative 1" and "Collaborative 2". The "Baseline" mode involves using only one device when running simulations. The "Collaborative

1" mode involves using two devices when running simulations and the "Collaborative 2" mode involves using all three devices when running simulations. Figure 6.2 showcases the "Collaborative 1" mode and how these two devices are connected. For the "Collaborative 2" mode, a third device was connected to device 1 in the same manner as in figure 6.2. Further details regarding the various simulation runs and the three different configuration modes utilized are provided in section 6.5.

The primary reasons for using the Raspberry Pi devices were as follows:

- **Similarity to in-vehicle hardware** - Due to the limited processing capabilities of the Raspberry Pi 4b devices, they were considered to be closer to actual in-vehicle hardware, such as the ECUs, than any other hardware available to the authors. Of course, the differences between these and the ECUs of a vehicle may still be great, and ideally, actual in-vehicle hardware would be used for this simulation. However, given the options available, this was deemed to be the most suitable approach nonetheless.

- **Homogeneous hardware** - While there technically isn't any need for the implemented system to be run on devices with similar computing capabilities, doing so allows for a simpler experimental setup and allows the authors to more easily control for- and understand the properties and limitations of the utilized devices.

6.5 Data collection

In order to collect the data required to perform the empirical evaluation, a number of simulation runs were performed. As mentioned in section 6.1, the dataset is divided into 9 separate CSV-files, of which 6 will be used for data collection.
During each of these simulation runs, one of these files was used to simulate network traffic on a single Raspberry pi device, while other files were used to train the required DNN-models. The datasets used and their corresponding DNN-models are shown in table 6.3

Table 6.3: Utilized csv-files and their corresponding DNN-models

ID	File name	Model name
1	Pre_train_D_1	model1
2	Pre_train_D_2	model2
3	Pre_train_S_1	model3
4	Pre_train_S_2	model4
5	Pre_submit_S	model5
6	Pre_submit_D	model6

The simulation runs were conducted in a triplewise manner, with each simulation run using the "Baseline" approach having two corresponding simulation runs where the "Collaborative 1" and "Collaborative 2" approaches were utilized.

During the "Baseline" simulation runs, the collaborative features of the system had been disabled, and as such all final decisions made by the IDSM were only based on the local analysis on that particular device. These "Baseline" simulation runs will be used to establish the baseline performance of the system without using collaboration and will be performed using the same data configurations as the corresponding collaborative simulation runs.

During the "Collaborative 1" simulation runs, the collaborative features of the system were activated and a single "consultation partner" device was connected to the device with the simulated network traffic, henceforth referred to as the *primary device*. This enabled the primary device to request consultation whenever this was deemed necessary. As mentioned in section 6.3, consultation was requested whenever the prediction of the security sensor had a certainty of less than 85%. Once the primary device received a response from the consultation partner, the prediction with the highest certainty was picked and logged.

"Collaborative 2" simulation runs worked in the same manner as the "Collaborative 1" simulation runs, with the only difference being that yet another consultation partner device was connected to the primary device. In other words, for each consultation request sent out by the primary device, two consultation responses were received back. The decision algorithm for dealing with received consultation responses also worked in the same manner as in "Collaborative 1" simulation runs, i.e. the prediction with the highest certainty was logged.

In total, 90 simulation runs were performed: 30 utilizing the "Baseline" approach, 30 using the "Collaborative 1" approach and 30 using the "Collaborative 2" approach. Using 30 simulation runs for each simulation mode was a suitable number of data points for use in the planned statistical tests. Furthermore, after 90 simulation runs the measured results seemed to have stabilized and hence this was considered to be a suitable stopping point.

The data- and model configurations for each of the simulation runs were selected using a probabilistic approach, by Python script to randomly generate the configurations of each simulation run. In order to protect the integrity of the results, the authors enforced that each dataset could only be used in a single capacity during each simulation run. In other words, cases where the network traffic data had also been used to train any of the models were disallowed, as were cases where several devices utilized the exact same model. The resulting data configuration of each simulation run is detailed in Appendix A.

As mentioned, during the simulation runs CAN-network traffic was only simulated on one of the Raspberry Pis. Meanwhile, the other devices remained focused solely

on receiving and responding to consultation messages. This was done in order to keep the setup as simple as possible.

As previously discussed, the IDSM is the primary decision-making and logging module. However, the security sensor only sends alerts for suspected intrusions to the IDSM, and not all alerts end up becoming qualified security events (QSEVs). As such the IDSM does not log information regarding every row from the original dataset, but only rows that qualify as QSEVs. As such, a separate validation script had to be developed in order to compare these logs to the original datasets and calculate the F1-score.

During any given simulation run, the following information was collected and logged by the IDSM:

- **Qualified security events** - I.e. the alerts raised by the security sensor. Each QSEv consists of two components: The corresponding row number in the simulated network traffic dataset, which is required for validation purposes, as well as the suspected type of intrusion. During simulation runs using the baseline approach, the IDSM simply logs the intrusion type reported by the security sensor. However, during collaborative simulation runs, and for security events where consultation is deemed necessary, the prediction with the highest certainty is logged as described in section 6.3.
- **Average local time** - For security event, the time from when that message arrived at the Security sensor until it had been received, parsed and processed by the IDSM was measured using the *time* module included in Python. A distinction is made here between the messages that only require local processing and the ones that require consultation. At the end of each simulation run, the average time required to process the locally handled messages is calculated and logged.
- **Average consultation time** - For simulation runs where the collaborative approach is utilized, the IDSM also logs the average time it takes for messages that require consultation to be fully processed. For security events, a timer is started when the message is received by the Security sensor. In the cases where consultation requests are required, this timer is not stopped until the security event's consultation responses have been received and processed. The timer isn't stopped until the final consultation response to the security event has arrived and been processed by the primary devices' IDSM. At the end of the simulation run, the average time required for the consultation messages is calculated and logged.

The collected data, as well as the scripts used for simulation and validation, can be found on the following GitHub page *https://github.com/DanielAryan/CIVID*.

7
Results

This chapter outlines the results obtained after conducting the data collection phase described in section 6.5. Firstly, the F1-scores obtained from each of the three different simulation modes are presented in detail.
Thereafter, the statistical tests performed on these results are described, including the various assumption checks that needed to be performed.
Finally, the results pertaining to the time required for processing of local- and consultation messages are presented.

7.1 F1-scores

Table 7.1 presents the measured F1-scores obtained from the 90 simulation runs, presented in a triplewise manner. The leftmost column shows the F1-scores obtained from the "Baseline" simulation runs, where only one device was used for detection. The middle column shows the F1-scores of the corresponding "Collaborative 1" simulation runs, which utilized the collaborative approach with a single consultation partner. Finally, the last column shows the results of the "Collaborative 2" simulation run, where two consultation partner devices were used for detection.
Furthermore, table 7.2 shows the means and 95% confidence intervals of each simulation mode's F1-scores.

A comparison between the "Baseline" simulation runs and the "Collaborative 1" simulation runs shows that the F1-scores of the "Collaborative 1" simulation runs are consequently greater than the "Baseline" runs. However, they only seem to be greater by a relatively small margin. Additionally, the F1-scores obtained from the "Collaborative 2" simulation runs are also consequently greater than the "Baseline" runs, and are greater than the "Collaborative 1" runs in most cases. However, in a few cases, namely simulation runs 39, 57, 60, 63, 84 and 87, the F1-scores of the "Collaborative 2" runs are slightly lower than their corresponding "Collaborative 1" runs. A more detailed breakdown of the results of each simulation run is presented in appendix B where the precision and recall used to derive each F1-score is also presented.

As previously mentioned, the results only show a marginal increase in F1-scores from using the collaborative approach. From Appendix B, when looking at the different F1-scores for each attack, it is clear that the attack benefiting most from a collaborative approach, and thus is mainly responsible for the increase in F1-score,

Table 7.1: Rounded F1-scores measured during the simulation runs[1]

Baseline		Collaborative 1		Collaborative 2	
Sim-run	***F1-Score***	***Sim-run***	***F1-Score***	***Sim-run***	***F1-Score***
1	0.463264	2	0.495187	3	0.497803
4	0.540365	5	0.555521	6	0.556409
7	0.611609	8	0.637676	9	0.641449
10	0.629500	11	0.633065	12	0.633625
13	0.591291	14	0.603199	15	0.631814
16	0.540590	17	0.543474	18	0.545305
19	0.549778	20	0.564924	21	0.569497
22	0.643081	23	0.669723	24	0.672263
25	0.568531	26	0.588151	27	0.600198
28	0.537090	29	0.539972	30	0.540503
31	0.465695	32	0.473161	33	0.474219
34	0.464736	35	0.472763	36	0.482419
37	0.560695	38	0.572603	**39**	**0.570021**
40	0.608112	41	0.628673	42	0.632940
43	0.440935	44	0.455877	45	0.460078
46	0.507057	47	0.521624	48	0.531502
49	0.472845	50	0.481844	51	0.483209
52	0.541948	53	0.561075	54	0.561586
55	0.548979	56	0.572546	**57**	**0.571328**
58	0.658585	59	0.686092	**60**	**0.679878**
61	0.572473	62	0.632326	**63**	**0.590040**
64	0.459100	65	0.464881	66	0.465872
67	0.457182	68	0.480110	69	0.485338
70	0.443996	71	0.459744	72	0.467119
73	0.499257	74	0.512783	75	0.512853
76	0.541157	77	0.568336	78	0.569993
79	0.492692	80	0.505431	81	0.505606
82	0.485754	83	0.496986	**84**	**0.496582**
85	0.666681	86	0.686779	**87**	**0.686747**
88	0.725958	89	0.757439	90	0.758155

is the Fuzzing attack. For this individual attack type, the collaborative approach is providing a quite substantial increase. For example, comparing the results of simulation runs 1, 2 and 3 in appendix B, the "Baseline" F1-score of the fuzzing attack in particular is 0.42 but compared to the collaborative approaches the F1-scores are 0.55 and 0.56 respectively.

For the other attack types, their respective F1-scores does not seem to be greatly affected by the application of a collaborative approach. In regards to the Flooding

[1]The bold cells showcases simulation runs where the Collaborative 2 mode performs worse than Collaborative 1

Table 7.2: 95% Confidence intervals of the measured F1-scores

Mode	n	95% CI Lower bounds	Mean	95% CI Upper bounds
Baseline	30	0.515238	0.542965	0.570691
Collaborative 1	30	0.531273	0.560732	0.590191
Collaborative 2	30	0.533521	0.562478	0.591436

attack, all of the utilized models used for detection appear to be predicting this kind of attack very well, hence no consultation is ever needed when detecting these attacks.
For the Replay and Spoofing attacks however, no benefit is gained from using a collaborative approach since all models, no matter how many devices are used, are equally bad at predicting these attacks. The utilized machine learning models are simply not very adept at detecting those attacks because those kinds of attacks are structurally no different from normal CAN-traffic. The reasoning behind this is further outlined in section 6.2.

An interesting aspect of the results is the F1-scores of the spoofing attacks. In most cases, the spoofing attacks achieve an F1-score of 0, for the reasons outlined above. However, in some cases, such as in simulation runs 88, 89 and 90, the F1-scores of the spoofing attacks are much greater (often close to 1.0). This discrepancy is a result of the simplicity of the utilized detection methodology as well as a quirk of the utilized dataset. As mentioned in section 6.5, 6 dataset files were used for both model training and to simulate network traffic. Between these 6 files, the exact structure of the utilized spoofing attack varied somewhat. Therefore, since our models are incapable of detecting the true markers of a spoofing attack, in most cases no spoofing attacks were detected at all. However, in the cases where the F1-scores of the spoofing attack increased, the model utilized by the primary device and the network traffic data happened to use the same spoofing attack frames, resulting in a much higher detection rate due to overfitting. As such, these results are in no way indicative of the utilized models' actual ability to detect spoofing attacks. Despite this quirk of the utilized dataset and detection methodology, this variability of the spoofing attack F1-scores does not affect the overall integrity of the results. The reason for this is that this discrepancy is not affected whatsoever by whether or not a collaborative approach was utilized or not, hence the F1-scores of the spoofing attacks are roughly the same across all simulation run triplets.

Compared to the state-of-the-art IDSs discussed in section 3, the IDS utilized in this thesis performs much worse in terms of F1-scores. This is partially because the individual F1-scores for the spoofing and replay attacks are very low which decreases the average F1-score for the simulation run as a whole.

7.2 Statistical tests

As mentioned in section 4.1.3, given that the assumptions of the one-way repeated measure ANOVA test holds, this statistical test will be utilized to infer whether or not there are statistically significant differences between the "Baseline" and the collaborative approaches.

The dependant variable used, i.e the F1-scores of these simulation runs, does operate on a continuous scale, thus this assumption holds. Furthermore, since each simulation run was run independently, i.e under the exact same operating conditions and without affecting other simulation runs, the assumption of independent responses between subjects should also hold.

The assumption of subjects being randomly selected from the population is also considered to hold since a random generator has been used for selecting which data to use for each simulation run.

Figure 7.1 presents a boxplot of the measured F1-scores, which can be used to visually assess the variability of the data and infer the presence of outliers. Based on this table, one might infer that there are three potential outliers, one in each simulation mode. These are the topmost data points measured in each simulation mode, i.e the F1-scores from simulation runs 88,89 and 90. From a visual standpoint, it is unclear whether these points differ to such an extent that they should be deemed as outliers. To assess this, the *Turkey's fences* method [42] has been utilized.

Based on the results of Tukey's fences, as can be seen in table 7.3, the largest F1-scores seen in each simulation mode are still lower than the upper bound set by Tukey's fences. Thus, even though they appear close to the upper bounds, these data points are not deemed as outliers. Therefore, the assumption of no outliers being present is also deemed to hold.

Table 7.3: Tukey's fences method

Tukey's fences	**Q3**	**IQR**	**Q3+1.5*IQR**	**Largest F1-score**
Baseline	0.581882	0.102582	0.735755	0.725958
Collaborative 1	0.615936	0.119849	0.795710	0.757439
Collaborative 2	0.616006	0.118813	0.794226	0.758155

As for the assumption that the dependent variable should be normally distributed in each simulation mode, both Shapiro-Wilk tests (table 7.4) and Q-Q plots (figure 7.2) to assess the normality of the data. Based on the p-values shown in table 7.4, which are all higher than the standard α value of 0.05, we are not able to reject the null-hypothesis that the simulation modes' F1-scores are normally distributed. Additionally, the Q-Q plots shown in figure 7.2 roughly show the ordered quantiles of the measured F1-scores following the theoretically expected quantiles of the normal distribution in all three simulation modes. Thus, it is not unreasonable to assume

that the F1-scores are normally distributed in each simulation mode. Therefore, the normality assumption of the repeated measures ANOVA test is also deemed to hold.

Table 7.4: Shapiro-Wilk test of measured F1-scores

Mode	Statistic	p-value
Baseline	0.947006	0.140490
Collaborative 1	0.944406	0.119567
Collaborative 2	0.941054	0.097102

The final assumption of the repeated measures ANOVA is the sphericity assumption, i.e that the differences in sample variance should be equal between any two levels of the independent variable. This sphericity assumption is usually tested using *Mauchly's test of sphericity*. Using this test on the measured F1-scores of each simulation mode, resulted in a p-value of 1.0 which indicates sphericity. Furthermore, a visual inspection of the boxplot in figure 7.1 also seems to indicate the homogeneity of variances in differences. Based on this, the sphericity assumption is considered to hold.

As all of the assumptions of the repeated measures ANOVA test have been checked and deemed to be met by the collected data, this test has been applied to the F1-scores of the three different simulation modes. The results of this ANOVA test is

Figure 7.1: Boxplot of measured F1-scores

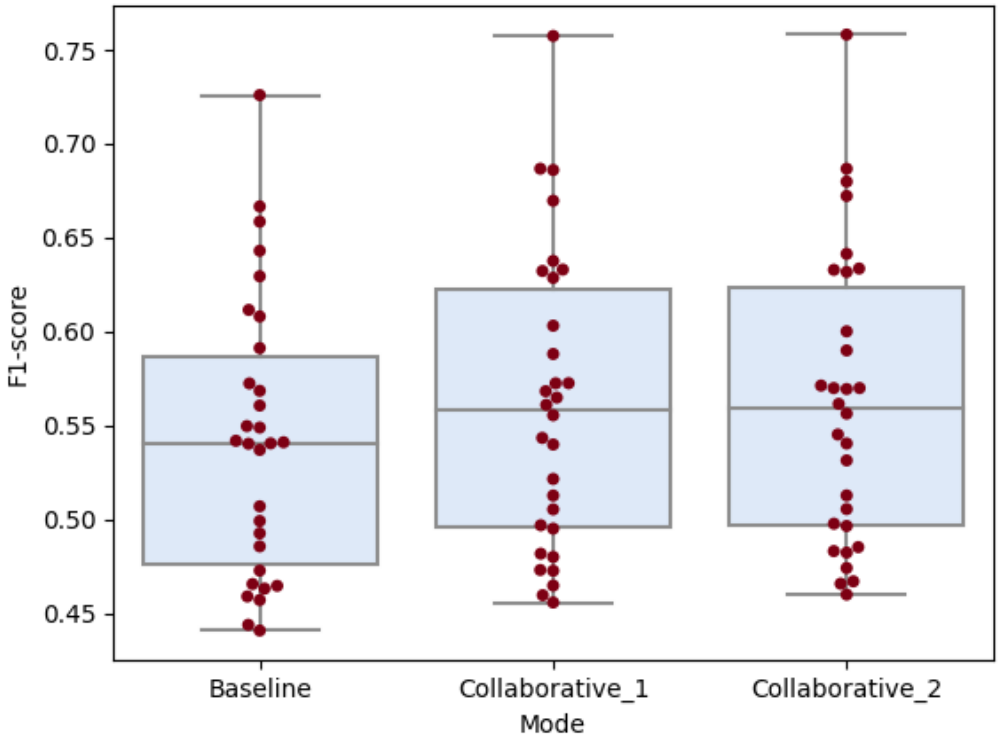
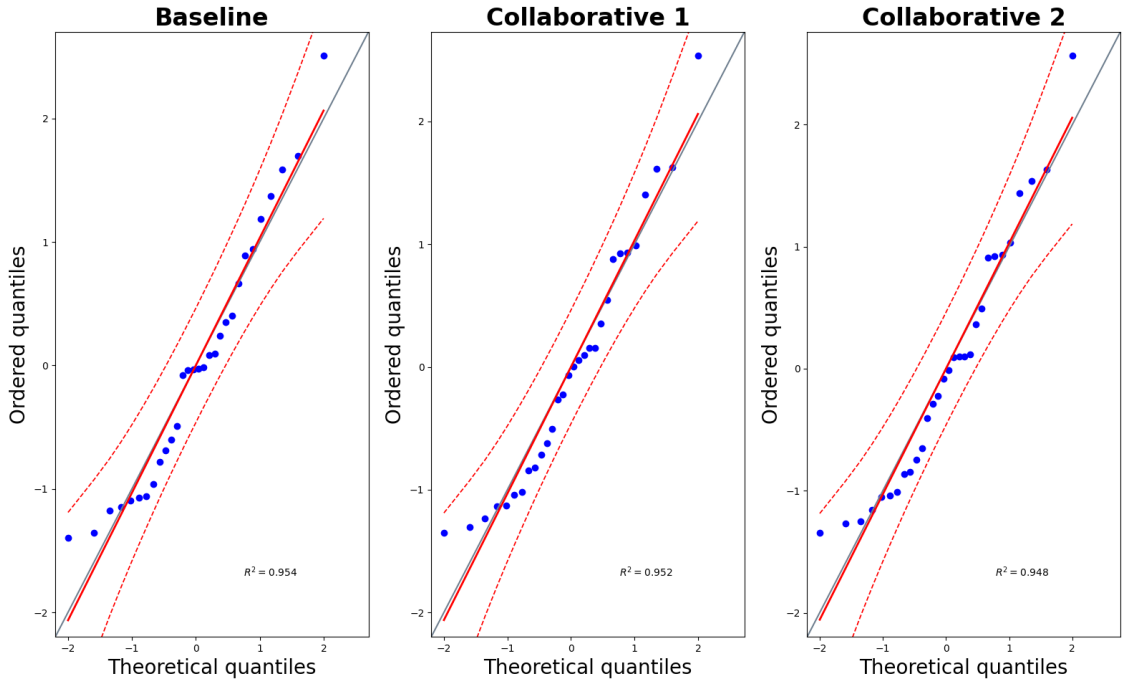


Figure 7.2: Q-Q plots for Normal distribution for each simulation mode



shown in table 7.5. The meaning of each statistic displayed in this table and how these should be interpreted is outlined in [44], however for the purpose of inferring whether or not there are significant differences between the groups it is sufficient to look at the p-value of the test (*p-unc*).

As this table shows a p-value that is significantly lower than the applied α value of 0.05, we can reject the null-hypothesis that there are no differences between the means of the three groups.

Table 7.5: One-way repeated ANOVA measures

Source	SS	DF	MS	F	p-unc	np2	eps
Mode	0.00699	2	0.00349	62.89297	2.98095e-15	0.68442	0.96372
Error	0.00323	58	5.56125e-05	NaN	NaN	NaN	NaN

In order to find out exactly which groups that differ significantly from each other, a post-hoc test, utilizing 2-way pairwise t-tests, has been applied. The results of this test are shown in table 7.6. The meaning of each statistic displayed in this table and how these should be interpreted is described in [43], however the corrected p-values (*p-corr*) of the tests are the most relevant when interpreting the results.

.

From these results, we can infer that there are statistically significant differences between the means of the "Baseline" and both collaborative simulation modes. However, since the p-value obtained by comparing the "Collaborative 1" and "Collabo-

rative 2" approaches (0.36) is higher than the α value of 0.05, we cannot infer any statistically significant differences between these two groups.

Table 7.6: Post-hoc test (2-way pairwise t-tests) using Benjamini/Hochberg FDR correction

Mode 1	Mode 2	T-statistic	DF	Tail	p-unc	p-corr
Baseline	Collaborative 1	-8.472075	29.0	two-sided	2.461940e-09	3.692910e-09
Baseline	Collaborative 2	-10.929051	29.0	two-sided	8.471511e-12	2.541453e-11
Collaborative 1	Collaborative 2	-0.928644	29.0	two-sided	0.360739	0.360739

Finally, in order to confirm the direction of the differences between the "Baseline" and collaborative approaches, a one-way post-hoc test has also been applied. The results of this test are shown in table 7.7. From these results we can infer that the means of both "Collaborative 1" and "Collaborative 2" F1-scores are statistically greater than those of the "Baseline" simulation runs.

Table 7.7: Post-hoc test (one-way pairwise t-tests) using Benjamini/Hochberg FDR correction

Mode 1	Mode 2	T-statistic	DF	Tail	p-unc	p-corr
Baseline	Collaborative 1	-8.472075	29.0	one-sided	1.230970e-09	1.846455e-09
Baseline	Collaborative 2	-10.929051	29.0	one-sided	4.235756e-12	1.270727e-11

7.3 Processing times

Table 7.8 presents the average times in seconds required for processing different kinds of messages, as measured during the simulation runs. The first row shows the times required to process a message locally, i.e when no consultation is needed.
The table also presents the average time for processing a message when collaboration with external devices is needed. The timings are calculated by measuring the time from when a message arrives at the Security sensor until all consultation responses have been processed and a final decision on the message has been reached. The "Collaborative 1" row presents the average time for processing messages that require consultation when using one collaborative partner device. The "Collaborative 2" row presents the average time for processing messages that require consultation when using two collaborative partner devices. The results presented in table 7.8 have been obtained by taking the average of the averaged processing times calculated from each simulation run. These results are further detailed in Appendix B.

Table 7.8: 95% Confidence intervals of the measured times

Mode	n	95% CI Lower bounds	Mean	95% CI Upper bounds
Average local time	90	0.01839	0.02005	0.02170
Average consultation time (Collaborative 1)	30	0.13405	0.14277	0.15149
Average consultation time (Collaborative 2)	30	0.15434	0.16736	0.18038

Table 7.8 shows that processing a consultation message using "Collaborative 1" takes approximately 7 times as long compared to processing a local message. However, the times for processing a consultation message when using two collaborative devices, i.e "Collaborative 2", takes longer time compared to "Collaborative 1", but the additional increase is only 17 percent, compared to the aforementioned comparison which had an increase of 700 percent.

8

Discussion

In this chapter, a discussion relating to the results obtained in chapter 7, is performed. Initially, a discussion regarding the trade-off between the added value of implementing the proposed CIVID framework, and the added risks of doing so. This is followed by a discussion on the pros and cons of utilizing a machine learning based detection method. Thereafter, an analysis of how the utilization of a collaborative intrusion detection approach affects the time-to-detection of potential intrusions is presented. Finally, the various threats to validity identified by the authors are presented and discussed.

8.1 The added value & risks of using a collaborative approach

Based on the results showcased in chapter 7, the collaborative approach does yield greater F1-scores than the baseline approach, having an average increase of 3.3% between the "Baseline" and the "Collaborative 1" simulation runs. Furthermore, despite the fact that the "Collaborative 2" simulation runs most often performed better than the "Collaborative 1" simulation runs, one could not infer a statistically significant improvement in F1-scores when an additional consultation partner was added. However, the fact that adding an additional consultation partner did not yield even greater gains than the ones observed is likely more reflective of the specific implementation than it is of the CIVID framework as a whole. If another decision algorithm had been implemented for processing and aggregating consultation responses, or if another detection methodology had been utilized, it is possible that even greater gains could have been observed by adding additional consultation partners. For instance, the implemented algorithm for processing consultation responses only takes the prediction with the highest certainty into consideration. A more sophisticated algorithm for processing responses, which aggregates the responses from several consultation partners might have resulted in additional gains as more consultation partners are added. Further research on this topic would be particularly interesting considering that the time-to-detection does not appear to scale proportionally as more consultation partners are added. Furthermore, for future research, it would be interesting to study how different implementation designs and decision algorithms for the CIVID framework could be used to leverage the information and resources of other vehicles in the fleet to an even greater extent.

As mentioned in chapter 7, the increase in F1-scores observed from utilizing the

collaborative approach is primarily derived from the improved detection accuracy of *Fuzzing* attacks in particular. The *Flooding* attacks however, hardly seemed to require consultation at all and were barely affected by the introduction of a collaborative approach. Therefore, it would seem that the collaborative intrusion detection approach may be more suitable for detecting certain kinds of attacks than others. One important thing to note however, is that the gains yielded by the collaborative intrusion detection framework are highly dependent on how it is implemented, what kind of detection methods are utilized and what type of information is shared during consultation.

As mentioned in chapter 5, some kind of trust management mechanism is required in order to implement the CIVID framework for a real-world setting. However, even with a sophisticated trust management system in place, usage of the collaborative approach does introduce an additional attack vector that could be targeted by malicious actors. For instance, if one of the vehicles in a collaborative intrusion detection network were to be compromised, it might try to utilize the intrusion detection network in order to influence other vehicles in the network. As such, there are inherent risks to the usage of intrusion detection networks that must be considered. Therefore, the potential gains of any specific implementation of the CIVID framework need to be weighed against the introduced risks. In addition, if the CIVID framework is to be implemented, e.g. in some vehicular fleet, it might be beneficial to limit the consultation to the attack types that benefit the most from collaboration.

The fact that the collaborative approach does not yield greater improvements in detection accuracy, as well as its perceived tendency to perform better with certain kinds of attacks, raises the questions of whether implementing a collaborative approach is worth the added cost and potential risks, and for what types of intrusions the collaborative approach is most useful. Therefore, it would also be interesting for future research to investigate under which circumstances it is suitable to implement the CIVID framework.

8.2 Pros and cons of the utilized detection method

The accuracy measures presented in chapter 7 show that the implemented anomaly-based IDS used in this thesis produced F1-scores averaging at approximately 0.6. As mentioned in chapter 3, state-of-the-art IDSs in the vehicular domain produces higher F1-scores than the IDS implemented in this thesis.
However, as mentioned in section 6.2, it was never the intention of the authors of this report to produce an IDS with F1-scores comparative to the aforementioned state of the art IDSs, but rather to investigate whether or not a collaborative approach could lead to increased F1-scores in an IDS. Despite this, the discrepancy between the F1-score performance of the IDS produced in this thesis and the state-of-the-art IDSs mentioned in chapter 3 is deemed interesting enough to warrant further analysis. It is also worth mentioning that the datasets- and evaluation methods used in the related works regarding other IDSs differ from the ones used in this thesis,

meaning that a direct comparison between the F1-scores obtained in the various studies might give a slightly misleading impression.

The main reason why the implemented IDS performs worse than state-of-the-art IDSs is mainly that the only detection method used is anomaly-based detection implemented using machine learning, whereas many state-of-the-art IDSs use at least a specification-based detection method, or hybrid-based IDS, which combine several different detection methods. A machine learning based detection method, such as the one used in this thesis, can be very suitable when detecting certain types of attacks, such as Flooding or Fuzzing attacks, as can be seen in Appendix B. However, it is much harder for a machine learning model to detect attacks such as Spoofing- or Replay attacks, as they are structurally indistinguishable from normal traffic. Flooding and Fuzzing attacks have characteristics, such as the time between messages or uncommon data patterns, which makes them more easily identifiable when compared to normal network traffic. The replay- and spoofing-attacks however, are seemingly just like any other normal message which makes them much harder to single out from normal traffic. Thus, a machine learning based detection method might not be the most suitable choice for detecting these kinds of attacks.

As mentioned in section 3.2, there are pros and cons with every type of detection method, therefore implementing a hybrid-based IDS, as was seen being done in [41] and [13], may be a suitable choice when combating the diverse arena of cyber-attacks in the automotive domain. It would be interesting to see how the CIVID framework proposed in this thesis, could be leveraged in combination with other types of detection methods as well. As an example, if anomaly-based detection, which seemingly works well on flooding attacks, would collaborate with an IDS that implements some other detection method that works well on spoofing and replay attacks, then their collaborative gains could likely produce an even greater increase in accuracy than what was observed in this thesis. This is a topic suitable for future research.

8.3 The effect of using a collaborative approach on time-to-detection

The average times from when messages are received by the IDSM until they have been fully processed are presented in table 7.8. As mentioned in 7.3, this table shows that messages that require consultation require significantly more time before a final decision can be made than messages that are only processed locally. This difference can primarily be attributed to the transmission time required to send messages back and forth to the consultation partners. Interestingly, as was also highlighted in section 7.3, this increase in processing times do not seem to scale proportionally as the number of consultation partners increases. As such, it should be possible to consult with a large number of consultation partners without inducing an adverse effect on the time-to-detection, as opposed to only consulting with a few consultation partners.

Worth noting here is that, as mentioned in chapter 6.3, the implemented IDSM does not perform all of the tasks that an IDSM should perform according to the AUTOSAR standard, e.g. the filtering and aggregation of security events. As such, one might expect to see higher local processing times in a real-world setting.

While the time until a final decision can be made is significantly greater while using the collaborative approach, the IDSM does not simply sit idle while waiting for consultation responses to arrive. Instead, it continues to process incoming network traffic as it arrives, and processes the consultation responses as they arrive. Therefore, the time it takes to process collaboration messages should not greatly affect the system as a whole. In addition, only a very small percent of security events require consultation. However, for the messages that do require consultation, the increased duration of time might constitute a security risk, if intrusions are not dealt with in a timely manner. To mitigate this risk, it might be necessary to perform some intermediary action on security events that are awaiting consultation responses. For instance, the messages in question could be suspended until the consultation responses arrive.

8.4 Threats to validity

This section outlines the various threats to validity identified by the authors while writing this thesis. These threats are divided into four categories. These are internal-, external-, construct- and conclusion validity.

8.4.1 Internal validity

The obtained F1-scores, as presented in 7, are the product of the chosen anomaly-based detection method with its implemented threshold set to 85%. This value of 85% could just as well have been set to another value, such as 75% or 95%. The reason for choosing a threshold value of 85% was only because it was a value that was deemed by the authors of this thesis to generate a reasonable amount of consultation messages. Choosing a different threshold value, would very likely yield different F1-scores. Additionally, since the detection method used was based on anomaly detection, it is not clear what the results would have been if a different detection method, such as specification-based detection, had been used instead.

The data used for the training of the models consists of 6 CSV-files of CAN-frames. However, these files do differ in the amount of CAN-frames, as well as how many attacks are included in each file. Since the models are trained on different files, if one model is trained on a file with 800 000 rows and another on a file with 2000 000 rows, therefore if the data would have been assigned differently, then maybe the outcome would also differ. Additionally, from the attacks included in the files, only two of four attacks are suitable for anomaly-based detection. If other, from a machine learning perspective, more easily identifiable attacks had been chosen, then a higher F1-score would probably also have been produced. On the other hand, if

none of the included attacks would be manageable by the anomaly-based detection method, the simulation runs wouldn't even be sensible to perform and would probably also produce a different result.

The anomaly-based detection method used was a simplistic DNN model. As previously stated, since the goal was not to implement a flawless IDS, the generated results of the simulation runs might have been different if a differently designed model had been used. To address this issue, a replication of this approach would be advised with different intrusion detection models.

8.4.2 External validity

As previously mentioned, the packages python-can, python-isotp and python-can-remote were used to simulate a network that operates on CAN-frames. This setup has then been utilized to run the simulations used to validate the proposed framework. One threat to validity is that the authors of this report have not seen these packages used to any extent in other research, thus it is unclear how well these packages have managed to implement the underlying protocols. This is unclear as no evaluation of the underlying source code has been conducted and no comparisons to real-world vehicle implementations have been performed. However, given that these packages have implemented the underlying protocols correctly and that the authors of this report haven't introduced bugs when using these packages to simulate the network implementation, the difference between the implementation used in this thesis, and a real-world setting, is deemed to only differ regarding the performance metric of time. Based on this, the results derived from the validation regarding processing times probably differ from a real-world setting, but perhaps it doesn't differ much in quota. This is something that needs to be further investigated. Furthermore, the results regarding the F1-scores should not differ based on the network implementation since the F1-scores are not dependent on transmission times, but only on the implementation of the IDS and the data processed.

Three Raspberry PIs have been used as computing resources. How well these Raspberry PIs compare to ECUs in computational performance and storage capacity, are not known. Even though our proposed collaborative attack-detection seems to work well based on the performed simulation runs, it could be the case that actual ECU hardware is not able to handle the additional computing efforts or storage requirements necessary for implementing the CIVID framework.

Additionally, when performing simulation runs, the three Raspberry PIs used have been in very close proximity to one another and the network router. They have also been connected to the same local network. Therefore the actual implication on processing times in a real-world setting and how these relate to one another is not known and should be considered as a threat to validity.

This thesis has concluded that implementing a collaborative approach to attack de-

tection can increase the accuracy of an IDS. As already known, this conclusion is based on the F1-scores provided by experimental simulation runs. However, the question remains whether these F1-scores actually measures the accuracy of a real in-vehicle IDS. From vehicle research articles presented in this thesis, the F1-score has been identified as a commonly used metric for evaluating the accuracy of IDSs, however, it is not clear whether this specific metric is actually used by vehicle manufacturers or if some other metric is used. If some other metric is used, there is the possibility that the conclusions drawn from the F1-scores differ from the results obtained when using some other accuracy metric.

When validating the framework, the focus has been to validate the performance benefits of implementing a collaborative approach compared to a baseline approach. However, the applicability of the CIVID framework will be affected by other factors as well. As mentioned in section 2.2, important aspects to consider when implementing an IDN is how to implement trust management as well as resource management. This thesis has shown that a collaborative approach can be beneficial in increasing the accuracy of an in-vehicle IDS, but the question is whether this benefit is beneficial enough when adding all the message-overhead needed when implementing trust management? Will the hardware of vehicles be able to handle the added overhead of storing messages and tables for possible collaborators? The framework has not been validated under these constraints and will therefore need further investigation. This will be left for future research.

8.4.3 Construct validity

Before running the 90 simulation runs, each model was randomly paired with a dataset to be trained with and the data used as network data for each simulation run were chosen. The random pairing was done in order to ensure comparable groups and eliminate a source of bias when pairing the models with the data. This process was scripted in python using randomization and some basic rules, such as not allowing the network data to be the same as any of the current models had been trained with. From what the authors have been able to tell from the generated set-up, this script seems to work as expected, but there is always the possibility of a bug being implemented either in this phase or anywhere else in the implementation which could potentially be a threat to the validity of the results.

8.4.4 Conclusion validity

The conclusions drawn from the statistical tests are not completely clear-cut. The decision to use a confidence level of 95 percent in all statistical tests is a potential threat to the validity of this thesis. The foremost reason that this confidence level has been applied is that it is a commonly used α value when dealing with frequentist statistics. Using this α value does constitute a 5% risk of making a type 1 error, which should be taken into consideration. However, for values close to a confidence

level boundary, the consequences of using this α value are not always clear cut.

Additionally, in this thesis we used the outlier determination rule-of-thumb known as Tukeys' fences to find out whether some data points were outliers or not. Tukeys' fences concluded that the data points were not to be considered as outliers. However, they were very close to the boundary and could just as well have been considered to be outliers. Therefore, concluding that there are no outliers in the data used for a repeated measures ANOVA test, is a potential threat to the validity of the results.

9
Conclusion

In this thesis, a study has been performed to explore whether or not connected vehicles can benefit from a collaborative approach when detecting attacks. To this end, the CIVID framework for collaborative intrusion detection in vehicle fleets has been developed, which utilizes a consultation-based collaboration mechanism. The CIVID framework is designed to adhere to the latest AUTOSAR standard for in-vehicle intrusion detection, however the utilized collaborative approach is not dependent on the AUTOSAR standard, but is applicable for other architectures as well.

In order to evaluate whether or not the CIVID framework can yield improvements in detecting ongoing attacks against a vehicle, a prototype implementation of the framework has been created by utilizing a machine-learning-based anomaly-detection method.

The results obtained from the aforementioned evaluation have shown that implementing the CIVID framework yields improved F1-score accuracy compared to not implementing the framework. The observed improvements in accuracy from implementing the CIVID framework with a single collaborator averages around 3.3%. In addition, no statistically significant improvement in accuracy could be observed when adding an additional collaboration partner. However, it is believed by the authors of this thesis that alternative implementation decisions, such as utilizing different detection methods or utilizing a different algorithm for processing consultation responses, could likely result in even greater improvements in accuracy as well as further improvements when additional collaborators are added.

It has also been shown that implementing the CIVID framework significantly increases the time-to-detection when consultation is required. However, this increase does not appear to scale proportionally as more collaborators are introduced, thus it should be possible to utilize the CIVID framework with many collaborators.

In conclusion, the CIVID framework has been shown to enable improved accuracy measures in in-vehicle intrusion detection. However, the improved accuracy measures need to be weighed against the increased time-to-detection and other added costs of implementing the framework. Thus, it would be beneficial to conduct further research on how to best implement the CIVID framework, as well as finding ways to combine the CIVID framework with some of the state of the art detection algorithms to see if even greater accuracy measures can be achieved there as well.

9.1 Future work

Our study shows that the accuracy of vehicular IDSs can be improved by utilizing the collaborative CIVID framework proposed in this thesis. However, the improvements in accuracy measured during the evaluation phase of this thesis were marginal, and thus it is unclear if the added value from this accuracy improvement outweighs the potential risks and costs of implementing the CIVID framework. Additionally, it is unknown whether or not greater improvements in accuracy could be achieved through the CIVID framework if other implementation decisions had been made. Thus, it is left for future research to identify whether or not other implementations of the CIVID framework could yield further gains.

Furthermore, this study shows that the utilization of a collaborative approach increases the time-to-detection of certain security events. This increased time does not appear to scale proportionally with the number of consultation partners. However, when compared to the time required to process security events locally only, this increase proved quite significant. Thus, future implementations of the CIVID framework will need to find ways to address this issue, for instance through intermediary action on the security events in question.

Bibliography

[1] AUTOSAR. *GENERAL INFORMATION ABOUT AUTOSAR*. 2021. URL: https://www.autosar.org/about/ (visited on 03/10/2021).

[2] Robert Bosch. "CAN specification version 2.0". In: *Robert Bosch GmbH, Postfach* 300240 (1991), p. 72.

[3] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (2018), pp. 249–259. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2018.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0893608018302107.

[4] Shanzhi Chen, Jinling Hu, Yan Shi, Ying Peng, Jiayi Fang, Rui Zhao, and Li Zhao. "Vehicle-to-Everything (v2x) Services Supported by LTE-Based Systems and 5G". In: *IEEE Communications Standards Magazine* 1.2 (2017), pp. 70–76. DOI: 10.1109/MCOMSTD.2017.1700015.

[5] Francois Chollet. *Keras*. https://keras.io. 2015. (Visited on 03/06/2021).

[6] Renesas Electronics Corporation. *In-Vehicle Network block diagram*. URL: https://www.renesas.com/us/en/application/automotive/in-vehicle-networking-application (visited on 04/14/2021).

[7] AJ Deepa and V Kavitha. "A comprehensive survey on approaches to intrusion detection system". In: *Procedia Engineering* 38 (2012), pp. 2063–2069.

[8] Leon Derczynski. "Complementarity, F-score, and NLP Evaluation". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 261–266. URL: https://www.aclweb.org/anthology/L16-1040.

[9] Carol J. Fung and Raouf Boutaba. "Design and management of collaborative intrusion detection networks". In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. 2013, pp. 955–961.

[10] Mohammad Hamad, Marcus Nolte, and Vassilis Prevelakis. "Towards comprehensive threat modeling for vehicles". In: *Proc. Workshop Secur. Dependab. Crit. Embedded Real-Time Syst.(CERTS)*. 2016, pp. 31–36.

[11] hardbyte. *python-can*. Version 3.3.4. URL: https://github.com/hardbyte/python-can (visited on 03/11/2021).

[12] Mohammad A. Hoque and Ragib Hasan. "Towards a Threat Model for Vehicular Fog Computing". In: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 2019, pp. 1051–1057. DOI: 10.1109/UEMCON47517.2019.8993064.

[13] Hyunjae Kang, Byung I. Kwak, Young H. Lee, Haneol Lee, Hwejae Lee, and Huy K. Kim. "Car Hacking and Defense Competition on In-Vehicle Network". In: *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*. Vol. 2021. 2021, p. 25.

[14] Sachin Katti, Balachander Krishnamurthy, and Dina Katabi. "Collaborating against Common Enemies". In: *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*. IMC '05. Berkeley, CA: USENIX Association, 2005, p. 34.

[15] Salman Khan. "Fuzzy STRIDE Model based on Werners Aggregation Operator for Computer Network Threat Modelling". In: *International Journal of Computing and Digital Systemss* 6 (Mar. 2017), pp. 83–88. DOI: 10.12785/IJCDS/060204.

[16] Hyunjae Kang; Byung I. Kwak; Young H. Lee; Haneol Lee; Hwejae Lee; Huy K. Kim. *Car Hacking: Attack & Defense Challenge 2020 Dataset*. 2021. DOI: 10.21227/qvr7-n418. URL: https://dx.doi.org/10.21227/qvr7-n418 (visited on 05/01/2021).

[17] Eric Knauss. "Constructive Master's Thesis Work in Industry: Guidelines for Applying Design Science Research". In: *CoRR* abs/2012.04966 (2020). arXiv: 2012.04966. URL: https://arxiv.org/abs/2012.04966.

[18] Seungho Lee, Wonsuk Choi, Hyo J. Jo, and Dong H. Lee. "T-Box: A Forensics-Enabled Trusted Automotive Data Recording Method". In: *IEEE Access* 7 (2019), pp. 49738–49755. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2910865.

[19] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. "Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (July 2019), p. 184. ISSN: 1687-1499. DOI: 10.1186/s13638-019-1484-3. URL: https://doi.org/10.1186/s13638-019-1484-3.

[20] Zhendong Ma and Christoph Schmittner. "Threat modeling for automotive security analysis". In: *Advanced Science and Technology Letters* 139 (2016), pp. 333–339.

[21] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/` (visited on 04/28/2021).

[22] Eduard Metzker. *Reliably Detecting and Defending Against Attacks - Requirements for Automotive Intrusion Detection Systems*. Published on Vector's website. Mar. 2020. URL: `https://assets.vector.com/cms/content/know-how/_technical-articles/Security_Intrusion_Detection_AutomobilElektronik_202003_PressArticle_EN.pdf` (visited on 03/20/2021).

[23] Suvda Myagmar, Adam J. Lee, and William Yurcik. "Threat modeling as a basis for security requirements". In: *Symposium on requirements engineering for information security (SREIS)*. Vol. 2005. Citeseer. 2005, pp. 1–8.

[24] Juri Opitz and Sebastian Burst. "Macro f1 and macro f1". In: *arXiv preprint arXiv:1911.03347* (2019).

[25] Keith Pazul. "Controller area network (can) basics". In: *Microchip Technology Inc* 1 (1999).

[26] RaspBerry Pi. *Raspberry Pi 4 model B: Specifications*. URL: `https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/` (visited on 04/14/2021).

[27] pylessard. *Hadoop*. Version 1.6. URL: `https://github.com/pylessard/python-can-isotp` (visited on 03/11/2021).

[28] *Requirements on Intrusion Detection System*. R20-11. Foundation. AUTOSAR. Nov. 2020. URL: `https://www.autosar.org/fileadmin/user_upload/standards/foundation/20-11/AUTOSAR_RS_IntrusionDetectionSystem.pdf` (visited on 04/15/2021).

[29] Cornelis J. Van Rijsbergen. "Foundation of evaluation". In: *Journal of documentation* (1974), pp. 365–373. DOI: `https://doi.org/10.1108/eb026584`.

[30] Thomas Rosenstatter, Kim Strandberg, Rodi Jolak, Riccardo Scandariato, and Tomas Olovsson. "REMIND: A Framework for the Resilient Design of Automotive Systems". In: *2020 IEEE Secure Development (SecDev)*. IEEE. 2020, pp. 81–95. DOI: `10.1109/SecDev45635.2020.00028`.

[31] Per Runeson, Emelie Engström, and Margaret-Anne Storey. "The Design Science Paradigm as a Frame for Empirical Software Engineering". In: *Contemporary Empirical Methods in Software Engineering*. Springer, 2020, pp. 127–147. ISBN: 978-3-030-32488-9. DOI: 10.1007/978-3-030-32489-6_5.

[32] Christian Sandberg. *python-can-remote*. Version 0.2.1. URL: https://github.com/christiansandberg/python-can-remote (visited on 03/11/2021).

[33] Karen Scarfone and Peter Mell. "Guide to intrusion detection and prevention systems (idps)". In: *NIST special publication* 800.2007 (2007), p. 94.

[34] Ramasubramanian Sekar, Ajay Gupta, James Frullo, Tushar Shanbhag, Abhishek Tiwari, Henglin Yang, and Sheng Zhou. "Specification-based anomaly detection: a new approach for detecting network intrusions". In: *Proceedings of the 9th ACM conference on Computer and communications security*. 2002, pp. 265–274.

[35] Hyun M. Song, Jiyoung Woo, and Huy K. Kim. "In-vehicle network intrusion detection using deep convolutional neural network". In: *Vehicular Communications* 21 (2020), p. 100198. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2019.100198. URL: http://www.sciencedirect.com/science/article/pii/S2214209619302451.

[36] *Specification of Intrusion Detection System Manager*. R20-11. Classic Platform. AUTOSAR. Nov. 2020. URL: https://www.autosar.org/fileadmin/user_upload/standards/classic/20-11/AUTOSAR_SWS_IntrusionDetectionSystemManager.pdf (visited on 03/08/2021).

[37] *Specification of Intrusion Detection System Manager for Adaptive Platform*. R20-11. Adaptive Platform. AUTOSAR. Nov. 2020. URL: https://www.autosar.org/fileadmin/user_upload/standards/adaptive/20-11/AUTOSAR_SWS_AdaptiveIntrusionDetectionSystemManager.pdf (visited on 03/08/2021).

[38] *Specification of Intrusion Detection System Protocol*. R20-11. Foundation. AUTOSAR. Nov. 2020. URL: https://www.autosar.org/fileadmin/user_upload/standards/foundation/20-11/AUTOSAR_PRS_IntrusionDetectionSystem.pdf (visited on 03/08/2021).

[39] Miroslaw Staron. *Automotive Software Architectures. An Introduction.* Springer International Publishing, 2017. ISBN: 9783319586106. DOI: 10.1007/978-3-319-58610-6. URL: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.SPRINGERLINK9783319586106&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[40] Yu-Sung Wu, Bingrui. Foo, Yongguo. Mei, and Saurabh. Bagchi. "Collaborative intrusion detection system (CIDS): a framework for accurate and efficient

IDS". In: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* 2003, pp. 234–244. DOI: 10.1109/CSAC.2003.1254328.

[41] Shahroz Tariq, Sangyup Lee, Huy Kang Kim, and Simon S. Woo. "CAN-ADF: The controller area network attack detection framework". In: *Computers & Security* 94 (2020), p. 101857. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.101857. URL: https://www.sciencedirect.com/science/article/pii/S0167404820301292.

[42] John W. Tukey. *Exploratory data analysis.* Vol. 2. Reading, Mass., 1977.

[43] Raphael Vallat. *pingouin.pairwise$_t$tests*. 2021. URL: https://pingouin-stats.org/generated/pingouin.pairwise_ttests.html (visited on 04/29/2021).

[44] Raphael Vallat. *pingouin.rm$_a$nova*. 2021. URL: https://pingouin-stats.org/generated/pingouin.rm_anova.html (visited on 04/29/2021).

[45] J.P. Verma. *Repeated measures design for empirical researchers.* John Wiley & Sons, 2015.

[46] Zhenwei Yu, Jeffrey Tsai, and Thomas Weigert. "An Adaptive Automatically Tuning Intrusion Detection System". In: *TAAS* 3 (Aug. 2008). DOI: 10.1145/1380422.1380425.

[47] Man Zhou, Lansheng Han, Hongwei Lu, and Cai Fu. "Distributed collaborative intrusion detection system for vehicular Ad Hoc networks based on invariant". In: *Computer Networks* 172 (2020), p. 107174. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107174. URL: https://www.sciencedirect.com/science/article/pii/S1389128619311879.

A

Appendix 1

Table A.1: Simulation runs

Simulation run	Mode	Data	Device	Partner1	Partner2
1	Baseline	1	3	N/A	N/A
2	Collaborative1	1	3	5	N/A
3	Collaborative2	1	3	5	6
4	Baseline	5	2	N/A	N/A
5	Collaborative1	5	2	3	N/A
6	Collaborative2	5	2	3	1
7	Baseline	1	4	N/A	N/A
8	Collaborative1	1	4	5	N/A
9	Collaborative2	1	4	5	6
10	Baseline	6	5	N/A	N/A
11	Collaborative1	6	5	3	N/A
12	Collaborative2	6	5	3	4
13	Baseline	3	2	N/A	N/A
14	Collaborative1	3	2	4	N/A
15	Collaborative2	3	2	4	6
16	Baseline	1	6	N/A	N/A
17	Collaborative1	1	6	4	N/A
18	Collaborative2	1	6	4	3
19	Baseline	5	1	N/A	N/A
20	Collaborative1	5	1	4	N/A
21	Collaborative2	5	1	4	3
22	Baseline	3	1	N/A	N/A
23	Collaborative1	3	1	2	N/A
24	Collaborative2	3	1	2	5
25	Baseline	2	1	N/A	N/A
26	Collaborative1	2	1	4	N/A
27	Collaborative2	2	1	4	5
28	Baseline	2	6	N/A	N/A
29	Collaborative1	2	6	1	N/A
30	Collaborative2	2	6	1	5
31	Baseline	1	5	N/A	N/A
32	Collaborative1	1	5	6	N/A

33	Collaborative2	1	5	6	3
34	Baseline	6	2	N/A	N/A
35	Collaborative1	6	2	4	N/A
36	Collaborative2	6	2	4	3
37	Baseline	4	1	N/A	N/A
38	Collaborative1	4	1	6	N/A
39	Collaborative2	4	1	6	2
40	Baseline	4	2	N/A	N/A
41	Collaborative1	4	2	5	N/A
42	Collaborative2	4	2	5	1
43	Baseline	6	4	N/A	N/A
44	Collaborative1	6	4	3	N/A
45	Collaborative2	6	4	3	1
46	Baseline	6	1	N/A	N/A
47	Collaborative1	6	1	4	N/A
48	Collaborative2	6	1	4	3
49	Baseline	3	5	N/A	N/A
50	Collaborative1	3	5	1	N/A
51	Collaborative2	3	5	1	6
52	Baseline	4	5	N/A	N/A
53	Collaborative1	4	5	3	N/A
54	Collaborative2	4	5	3	6
55	Baseline	3	6	N/A	N/A
56	Collaborative1	3	6	5	N/A
57	Collaborative2	3	6	5	2
58	Baseline	2	4	N/A	N/A
59	Collaborative1	2	4	6	N/A
60	Collaborative2	2	4	6	3
61	Baseline	1	2	N/A	N/A
62	Collaborative1	1	2	6	N/A
63	Collaborative2	1	2	6	5
64	Baseline	4	3	N/A	N/A
65	Collaborative1	4	3	1	N/A
66	Collaborative2	4	3	1	5
67	Baseline	6	3	N/A	N/A
68	Collaborative1	6	3	4	N/A
69	Collaborative2	6	3	4	2
70	Baseline	2	3	N/A	N/A
71	Collaborative1	2	3	4	N/A
72	Collaborative2	2	3	4	6
73	Baseline	2	5	N/A	N/A
74	Collaborative1	2	5	3	N/A
75	Collaborative2	2	5	3	4
76	Baseline	5	4	N/A	N/A

77	Collaborative1	5	4	3	N/A
78	Collaborative2	5	4	3	6
79	Baseline	4	6	N/A	N/A
80	Collaborative1	4	6	5	N/A
81	Collaborative2	4	6	5	3
82	Baseline	5	3	N/A	N/A
83	Collaborative1	5	3	4	N/A
84	Collaborative2	5	3	4	2
85	Baseline	3	4	N/A	N/A
86	Collaborative1	3	4	6	N/A
87	Collaborative2	3	4	6	1
88	Baseline	5	6	N/A	N/A
89	Collaborative1	5	6	3	N/A
90	Collaborative2	5	6	3	1

B

Appendix 2

Table B.1: Results of each simulation run

Simulation run 1					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9745	0.2724	0.9849	0.0000	0.0110
Recall	1.0000	0.9430	0.8033	0.0000	1.0000
F1-score	0.9871	0.4227	0.8849	0.0000	0.0217
F1-score average	0.4632642240805171				
Average local time(s)	0.00924413881012882				
Average consultation time(s)	N/A				
Total number of CAN-frames	806390				
Total number of security events	207962				
Consultation messages sent	0				

Simulation run 2					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9805	0.3947	0.9851	0.0000	0.0114
Recall	1.0000	0.9400	0.8410	0.0000	1.0000
F1-score	0.9902	0.5559	0.9074	0.0000	0.0225
F1-score average	0.49518730618379686				
Average local time(s)	0.02422968879251890				
Average consultation time(s)	0.12509782695955882				
Total number of CAN-frames	806390				
Total number of security events	207962				
Consultation messages sent	31341				

Simulation run 3					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9805	0.4068	0.9851	0.0000	0.0114
Recall	1.0000	0.9395	0.8432	0.0000	1.0000
F1-score	0.9902	0.5677	0.9086	0.0000	0.0225
F1-score average	0.4978029830030538				
Average local time(s)	0.02715964448692883				
Average consultation time(s)	0.17785932701193363				
Total number of CAN-frames	806390				

Total number of security events	207962
Consultation messages sent	31341

Simulation run 4					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6058	0.9697	0.0000	0.0000
Recall	1.0000	0.9133	0.9771	0.0000	0.0000
F1-score	1.0000	0.7284	0.9734	0.0000	0.0000
F1-score average	0.5403649056584626				
Average local time(s)	0.01269283785650726				
Average consultation time(s)	N/A				
Total number of CAN-frames	1751313				
Total number of security events	180334				
Consultation messages sent	0				

Simulation run 5					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7177	0.9695	0.0000	0.0000
Recall	1.0000	0.9049	0.9848	0.0000	0.0000
F1-score	1.0000	0.8005	0.9771	0.0000	0.0000
F1-score average	0.5555214562925321				
Average local time(s)	0.02245297089636788				
Average consultation time(s)	0.14894742371576017				
Total number of CAN-frames	1751313				
Total number of security events	180334				
Consultation messages sent	24471				

Simulation run 6					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7238	0.9695	0.0000	0.0000
Recall	1.0000	0.9061	0.9852	0.0000	0.0000
F1-score	1.0000	0.8048	0.9773	0.0000	0.0000
F1-score average	0.5564089183097984				
Average local time(s)	0.02322356693386549				
Average consultation time(s)	0.1941606768316235				
Total number of CAN-frames	1751313				
Total number of security events	180334				
Consultation messages sent	24471				

Simulation run 7					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.1931	0.9838	0.0000	0.6800
Recall	1.0000	0.9664	0.8758	0.0000	1.0000
F1-score	1.0000	0.3219	0.9267	0.0000	0.8095

F1-score average	0.611609252384845
Average local time(s)	0.01028034987697497
Average consultation time(s)	N/A
Total number of CAN-frames	806390
Total number of security events	153199
Consultation messages sent	0

Simulation run 8					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2326	0.9838	0.0000	0.7737
Recall	1.0000	0.9629	0.9023	0.0000	1.0000
F1-score	1.0000	0.3747	0.9413	0.0000	0.8724
F1-score average	0.6376758107221018				
Average local time(s)	0.02891414778971760				
Average consultation time(s)	0.1348762946569698				
Total number of CAN-frames	806390				
Total number of security events	153199				
Consultation messages sent	30491				

Simulation run 9					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2447	0.9839	0.0000	0.7737
Recall	1.0000	0.9614	0.9087	0.0000	1.0000
F1-score	1.0000	0.3900	0.9448	0.0000	0.8724
F1-score average	0.6414492867367638				
Average local time(s)	0.02907465045904520				
Average consultation time(s)	0.15621831090158392				
Total number of CAN-frames	806390				
Total number of security events	153199				
Consultation messages sent	30491				

Simulation run 10					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.1918	0.9776	0.0000	0.8049
Recall	1.0000	0.9673	0.8968	0.0000	1.0000
F1-score	1.0000	0.3201	0.9355	0.0000	0.8919
F1-score average	0.6295004383722211				
Average local time(s)	0.01051485650591716				
Average consultation time(s)	N/A				
Total number of CAN-frames	2000733				
Total number of security events	350262				
Consultation messages sent	0				

Simulation run 11

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2122	0.9775	0.0000	0.7790
Recall	1.0000	0.9639	0.9085	0.0000	1.0000
F1-score	1.0000	0.3479	0.9417	0.0000	0.8758
F1-score average	0.6330647439953994				
Average local time(s)	0.01792383643000037				
Average consultation time(s)	0.1168857001649193				
Total number of CAN-frames	2000733				
Total number of security events	350262				
Consultation messages sent	28339				

Simulation run 12					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2130	0.9775	0.0000	0.7813
Recall	1.0000	0.9639	0.9089	0.0000	1.0000
F1-score	1.0000	0.3489	0.9420	0.0000	0.8772
F1-score average	0.6336254737921314				
Average local time(s)	0.01800044240286858				
Average consultation time(s)	0.13191625237267196				
Total number of CAN-frames	2000733				
Total number of security events	350262				
Consultation messages sent	28339				

Simulation run 13					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2965	0.9833	0.0000	0.3800
Recall	1.0000	0.9154	0.9336	0.0000	1.0000
F1-score	1.0000	0.4479	0.9578	0.0000	0.5507
F1-score average	0.5912906152742234				
Average local time(s)	0.01138513168497635				
Average consultation time(s)	N/A				
Total number of CAN-frames	799292				
Total number of security events	108292				
Consultation messages sent	0				

Simulation run 14					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3419	0.9831	0.0000	0.3837
Recall	1.0000	0.9112	0.9460	0.0000	1.0000
F1-score	1.0000	0.4972	0.9642	0.0000	0.5546
F1-score average	0.6031986590261011				
Average local time(s)	0.03525482519863726				
Average consultation time(s)	0.1874087551690297				
Total number of CAN-frames	799292				

Total number of security events	108292
Consultation messages sent	43659

Simulation run 15					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.4791	0.9835	0.0000	0.3840
Recall	1.0000	0.9109	0.9690	0.0000	1.0000
F1-score	1.0000	0.6280	0.9762	0.0000	0.5549
F1-score average	0.6318140159755838				
Average local time(s)	0.03519608717974126				
Average consultation time(s)	0.24761949069020664				
Total number of CAN-frames	799292				
Total number of security events	108292				
Consultation messages sent	43659				

Simulation run 16					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.5815	0.9839	0.0000	0.0000
Recall	1.0000	0.9515	0.9784	0.0000	0.0000
F1-score	1.0000	0.7218	0.9811	0.0000	0.0000
F1-score average	0.5405901858681553				
Average local time(s)	0.01296681092699768				
Average consultation time(s)	N/A				
Total number of CAN-frames	806390				
Total number of security events	76756				
Consultation messages sent	0				

Simulation run 17					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6008	0.9837	0.0000	0.0000
Recall	1.0000	0.9482	0.9799	0.0000	0.0000
F1-score	1.0000	0.7356	0.9818	0.0000	0.0000
F1-score average	0.5434744621168063				
Average local time(s)	0.01949592119582156				
Average consultation time(s)	0.13201598918346494				
Total number of CAN-frames	806390				
Total number of security events	76756				
Consultation messages sent	6256				

Simulation run 18					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6131	0.9837	0.0000	0.0000
Recall	1.0000	0.9469	0.9808	0.0000	0.0000
F1-score	1.0000	0.7443	0.9822	0.0000	0.0000

F1-score average	0.5453048605054158
Average local time(s)	0.01933875602356931
Average consultation time(s)	0.14563795249632863
Total number of CAN-frames	806390
Total number of security events	76756
Consultation messages sent	6256

Simulation run 19					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6649	0.9706	0.0000	0.0000
Recall	1.0000	0.9374	0.9712	0.0000	0.0000
F1-score	1.0000	0.7780	0.9709	0.0000	0.0000
F1-score average	0.5497778046589369				
Average local time(s)	0.01239404026101591				
Average consultation time(s)	N/A				
Total number of CAN-frames	1751313				
Total number of security events	191234				
Consultation messages sent	0				

Simulation run 20					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7821	0.9702	0.0000	0.0000
Recall	1.0000	0.9319	0.9782	0.0000	0.0000
F1-score	1.0000	0.8505	0.9742	0.0000	0.0000
F1-score average	0.5649244731880099				
Average local time(s)	0.02267848227725682				
Average consultation time(s)	0.15100586334997237				
Total number of CAN-frames	1751313				
Total number of security events	191234				
Consultation messages sent	28097				

Simulation run 21					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.8213	0.9703	0.0000	0.0000
Recall	1.0000	0.9301	0.9801	0.0000	0.0000
F1-score	1.0000	0.8723	0.9752	0.0000	0.0000
F1-score average	0.5694974719317341				
Average local time(s)	0.02246912009309587				
Average consultation time(s)	0.17421387636683167				
Total number of CAN-frames	1751313				
Total number of security events	191234				
Consultation messages sent	28097				

Simulation run 22

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.5765	0.9851	0.0000	0.3509
Recall	1.0000	0.9385	0.9783	0.0000	1.0000
F1-score	1.0000	0.7143	0.9817	0.0000	0.5195
F1-score average	0.6430809696594382				
Average local time(s)	0.01275120077003419				
Average consultation time(s)	N/A				
Total number of CAN-frames	799292				
Total number of security events	76582				
Consultation messages sent	0				

Simulation run 23					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7678	0.9848	0.0000	0.3522
Recall	1.0000	0.9274	0.9904	0.0000	1.0000
F1-score	1.0000	0.8401	0.9876	0.0000	0.5210
F1-score average	0.669722599955785				
Average local time(s)	0.02340435794868040				
Average consultation time(s)	0.17431849472723393				
Total number of CAN-frames	799292				
Total number of security events	76582				
Consultation messages sent	15090				

Simulation run 24					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7842	0.9849	0.0000	0.3526
Recall	1.0000	0.9325	0.9911	0.0000	1.0000
F1-score	1.0000	0.8520	0.9880	0.0000	0.5214
F1-score average	0.672263213415282				
Average local time(s)	0.02304780522405865				
Average consultation time(s)	0.20697769298705146				
Total number of CAN-frames	799292				
Total number of security events	76582				
Consultation messages sent	15090				

Simulation run 25					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9994	0.4273	0.9829	0.0000	0.1701
Recall	1.0000	0.9351	0.9490	0.0000	1.0000
F1-score	0.9997	0.5865	0.9656	0.0000	0.2908
F1-score average	0.5685306979723106				
Average local time(s)	0.01188307035534849				
Average consultation time(s)	N/A				
Total number of CAN-frames	889395				

Total number of security events	105883
Consultation messages sent	0

Simulation run 26					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9996	0.4979	0.9823	0.0000	0.1917
Recall	1.0000	0.9289	0.9597	0.0000	1.0000
F1-score	0.9998	0.6483	0.9709	0.0000	0.3218
F1-score average	0.5881509643801524				
Average local time(s)	0.02869033111675804				
Average consultation time(s)	0.15312072131383783				
Total number of CAN-frames	889395				
Total number of security events	105883				
Consultation messages sent	22175				

Simulation run 27					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9996	0.5670	0.9825	0.0000	0.1920
Recall	1.0000	0.9314	0.9660	0.0000	1.0000
F1-score	0.9998	0.7049	0.9742	0.0000	0.3221
F1-score average	0.6001979079730219				
Average local time(s)	0.02841550306588697				
Average consultation time(s)	0.1787600277402893				
Total number of CAN-frames	889395				
Total number of security events	105883				
Consultation messages sent	22175				

Simulation run 28					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.5632	0.9799	0.0000	0.0000
Recall	1.0000	0.9503	0.9767	0.0000	0.0000
F1-score	1.0000	0.7072	0.9783	0.0000	0.0000
F1-score average	0.5370897234921677				
Average local time(s)	0.01328485558534588				
Average consultation time(s)	N/A				
Total number of CAN-frames	889395				
Total number of security events	80533				
Consultation messages sent	0				

Simulation run 29					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.5841	0.9796	0.0000	0.0000
Recall	1.0000	0.9411	0.9785	0.0000	0.0000
F1-score	1.0000	0.7208	0.9791	0.0000	0.0000

F1-score average	0.5399721247890498
Average local time(s)	0.02031302319711752
Average consultation time(s)	0.13343317589570314
Total number of CAN-frames	889395
Total number of security events	80533
Consultation messages sent	6869

Simulation run 30					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.5858	0.9797	0.0000	0.0000
Recall	1.0000	0.9455	0.9786	0.0000	0.0000
F1-score	1.0000	0.7234	0.9792	0.0000	0.0000
F1-score average	0.5405029986484851				
Average local time(s)	0.02015493282939847				
Average consultation time(s)	0.14404288742597454				
Total number of CAN-frames	889395				
Total number of security events	80533				
Consultation messages sent	6869				

Simulation run 31					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2413	0.9827	0.0000	0.0000
Recall	1.0000	0.9673	0.9050	0.0000	0.0000
F1-score	1.0000	0.3862	0.9423	0.0000	0.0000
F1-score average	0.4656948381698919				
Average local time(s)	0.01075891183010454				
Average consultation time(s)	N/A				
Total number of CAN-frames	806390				
Total number of security events	130652				
Consultation messages sent	0				

Simulation run 32					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2663	0.9827	0.0000	0.0000
Recall	1.0000	0.9629	0.9169	0.0000	0.0000
F1-score	1.0000	0.4172	0.9486	0.0000	0.0000
F1-score average	0.4731613501119729				
Average local time(s)	0.02012780384547831				
Average consultation time(s)	0.11802650368760892				
Total number of CAN-frames	806390				
Total number of security events	130652				
Consultation messages sent	12896				

Simulation run 33

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2700	0.9827	0.0000	0.0000
Recall	1.0000	0.9634	0.9180	0.0000	0.0000
F1-score	1.0000	0.4218	0.9493	0.0000	0.0000
F1-score average	0.4742188375753115				
Average local time(s)	0.01998012149668836				
Average consultation time(s)	0.12945322137760465				
Total number of CAN-frames	806390				
Total number of security events	130652				
Consultation messages sent	12896				

Simulation run 34					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2385	0.9647	0.0000	0.0000
Recall	1.0000	0.9156	0.9265	0.0000	0.0000
F1-score	1.0000	0.3785	0.9452	0.0000	0.0000
F1-score average	0.46473646798932117				
Average local time(s)	0.01149169180056112				
Average consultation time(s)	N/A				
Total number of CAN-frames	2000733				
Total number of security events	272991				
Consultation messages sent	0				

Simulation run 35					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2672	0.9648	0.0000	0.0000
Recall	1.0000	0.9104	0.9370	0.0000	0.0000
F1-score	1.0000	0.4132	0.9507	0.0000	0.0000
F1-score average	0.47276312564914297				
Average local time(s)	0.03768504468279315				
Average consultation time(s)	0.1829225367339803				
Total number of CAN-frames	2000733				
Total number of security events	272990				
Consultation messages sent	104596				

Simulation run 36					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.3043	0.9651	0.0000	0.0000
Recall	1.0000	0.9090	0.9474	0.0000	0.0000
F1-score	1.0000	0.4560	0.9561	0.0000	0.0000
F1-score average	0.48241933025973305				
Average local time(s)	0.03738557330610345				
Average consultation time(s)	0.2377954632563837				
Total number of CAN-frames	2000733				

Total number of security events	272989
Consultation messages sent	104595

Simulation run 37					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3351	0.9811	0.0000	0.2152
Recall	1.0000	0.9397	0.9308	0.0000	1.0000
F1-score	1.0000	0.4941	0.9553	0.0000	0.3542
F1-score average	0.5606952135328033				
Average local time(s)	0.01122287994840310				
Average consultation time(s)	N/A				
Total number of CAN-frames	817042				
Total number of security events	115242				
Consultation messages sent	0				

Simulation run 38					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3856	0.9808	0.0000	0.2167
Recall	1.0000	0.9330	0.9422	0.0000	1.0000
F1-score	1.0000	0.5457	0.9611	0.0000	0.3562
F1-score average	0.5726028396273561				
Average local time(s)	0.03157324821573743				
Average consultation time(s)	0.1522965395519949				
Total number of CAN-frames	817042				
Total number of security events	115242				
Consultation messages sent	28188				

Simulation run 39					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3744	0.9807	0.0000	0.2166
Recall	1.0000	0.9303	0.9402	0.0000	1.0000
F1-score	1.0000	0.5340	0.9600	0.0000	0.3561
F1-score average	0.5700205868365913				
Average local time(s)	0.03134830780146734				
Average consultation time(s)	0.18166102649337673				
Total number of CAN-frames	817042				
Total number of security events	115242				
Consultation messages sent	28188				

Simulation run 40					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2997	0.9799	0.0000	0.4636
Recall	1.0000	0.9180	0.9316	0.0000	1.0000
F1-score	1.0000	0.4519	0.9551	0.0000	0.6336

F1-score average	0.6081124947063008
Average local time(s)	0.01127589320044494
Average consultation time(s)	N/A
Total number of CAN-frames	817042
Total number of security events	113884
Consultation messages sent	0

Simulation run 41					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3273	0.9798	0.0000	0.5405
Recall	1.0000	0.9118	0.9409	0.0000	1.0000
F1-score	1.0000	0.4817	0.9599	0.0000	0.7017
F1-score average	0.628672971724123				
Average local time(s)	0.02439368822560710				
Average consultation time(s)	0.12486498998532236				
Total number of CAN-frames	817042				
Total number of security events	113884				
Consultation messages sent	15470				

Simulation run 42					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3452	0.9798	0.0000	0.5405
Recall	1.0000	0.9117	0.9452	0.0000	1.0000
F1-score	1.0000	0.5008	0.9622	0.0000	0.7017
F1-score average	0.6329403472802897				
Average local time(s)	0.02427980572855567				
Average consultation time(s)	0.1414569181400341				
Total number of CAN-frames	817042				
Total number of security events	113884				
Consultation messages sent	15470				

Simulation run 43					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.1663	0.9644	0.0000	0.0000
Recall	1.0000	0.9661	0.8812	0.0000	0.0000
F1-score	1.0000	0.2837	0.9209	0.0000	0.0000
F1-score average	0.44093511994260454				
Average local time(s)	0.01052775516929245				
Average consultation time(s)	N/A				
Total number of CAN-frames	2000733				
Total number of security events	356915				
Consultation messages sent	0				

Simulation run 44

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9987	0.2089	0.9649	0.0000	0.0000
Recall	1.0000	0.9592	0.9106	0.0000	0.0000
F1-score	0.9994	0.3430	0.9370	0.0000	0.0000
F1-score average	0.4558769171958651				
Average local time(s)	0.02998754099933197				
Average consultation time(s)	0.14023613173707833				
Total number of CAN-frames	2000733				
Total number of security events	356913				
Consultation messages sent	74013				

Simulation run 45					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9991	0.2218	0.9650	0.0000	0.0000
Recall	1.0000	0.9591	0.9172	0.0000	0.0000
F1-score	0.9995	0.3603	0.9405	0.0000	0.0000
F1-score average	0.46007841529413085				
Average local time(s)	0.02992565699742371				
Average consultation time(s)	0.15940093272548406				
Total number of CAN-frames	2000733				
Total number of security events	356914				
Consultation messages sent	74012				

Simulation run 46					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9995	0.4099	0.9669	0.0000	0.0000
Recall	1.0000	0.9364	0.9637	0.0000	0.0000
F1-score	0.9998	0.5702	0.9653	0.0000	0.0000
F1-score average	0.5070570849234042				
Average local time(s)	0.01279946520011168				
Average consultation time(s)	N/A				
Total number of CAN-frames	2000733				
Total number of security events	207546				
Consultation messages sent	0				

Simulation run 47					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9996	0.4855	0.9666	0.0000	0.0000
Recall	1.0000	0.9310	0.9737	0.0000	0.0000
F1-score	0.9998	0.6382	0.9701	0.0000	0.0000
F1-score average	0.5216236624847614				
Average local time(s)	0.02674672122280557				
Average consultation time(s)	0.1676054527258302				
Total number of CAN-frames	2000733				

Total number of security events	207544
Consultation messages sent	43127

Simulation run 48					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9996	0.5429	0.9668	0.0000	0.0000
Recall	1.0000	0.9291	0.9781	0.0000	0.0000
F1-score	0.9998	0.6853	0.9724	0.0000	0.0000
F1-score average	0.5315017741514232				
Average local time(s)	0.02624313487719361				
Average consultation time(s)	0.19111344604059655				
Total number of CAN-frames	2000733				
Total number of security events	207545				
Consultation messages sent	43127				

Simulation run 49					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2640	0.9829	0.0000	0.0000
Recall	1.0000	0.9683	0.9180	0.0000	0.0000
F1-score	1.0000	0.4149	0.9493	0.0000	0.0000
F1-score average	0.47284487467968805				
Average local time(s)	0.01103758130640841				
Average consultation time(s)	N/A				
Total number of CAN-frames	799292				
Total number of security events	119610				
Consultation messages sent	0				

Simulation run 50					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2965	0.9828	0.0000	0.0000
Recall	1.0000	0.9635	0.9302	0.0000	0.0000
F1-score	1.0000	0.4534	0.9558	0.0000	0.0000
F1-score average	0.4818438923656969				
Average local time(s)	0.02224680805400842				
Average consultation time(s)	0.12417991897588011				
Total number of CAN-frames	799292				
Total number of security events	119610				
Consultation messages sent	13724				

Simulation run 51					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3016	0.9828	0.0000	0.0000
Recall	1.0000	0.9628	0.9319	0.0000	0.0000
F1-score	1.0000	0.4594	0.9567	0.0000	0.0000

F1-score average	0.48320960150380865
Average local time(s)	0.02213600945572240
Average consultation time(s)	0.1410630242673762
Total number of CAN-frames	799292
Total number of security events	119610
Consultation messages sent	13724

Simulation run 52					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.5891	0.9781	0.0000	0.0000
Recall	1.0000	0.9693	0.9757	0.0000	0.0000
F1-score	1.0000	0.7329	0.9769	0.0000	0.0000
F1-score average	0.5419479997470582				
Average local time(s)	0.01270618425773808				
Average consultation time(s)	N/A				
Total number of CAN-frames	817042				
Total number of security events	79185				
Consultation messages sent	0				

Simulation run 53					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7184	0.9780	0.0000	0.0000
Recall	1.0000	0.9665	0.9844	0.0000	0.0000
F1-score	1.0000	0.8242	0.9812	0.0000	0.0000
F1-score average	0.5610752362761042				
Average local time(s)	0.02061397127532327				
Average consultation time(s)	0.15747332546820397				
Total number of CAN-frames	817042				
Total number of security events	79185				
Consultation messages sent	9129				

Simulation run 54					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7228	0.9779	0.0000	0.0000
Recall	1.0000	0.9652	0.9847	0.0000	0.0000
F1-score	1.0000	0.8266	0.9813	0.0000	0.0000
F1-score average	0.5615859693428462				
Average local time(s)	0.02029755559089563				
Average consultation time(s)	0.18277092311139148				
Total number of CAN-frames	817042				
Total number of security events	79185				
Consultation messages sent	9129				

Simulation run 55

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6344	0.9839	0.0000	0.0000
Recall	1.0000	0.9521	0.9830	0.0000	0.0000
F1-score	1.0000	0.7615	0.9834	0.0000	0.0000
F1-score average	0.5489794444528007				
Average local time(s)	0.01320500561442549				
Average consultation time(s)	N/A				
Total number of CAN-frames	799292				
Total number of security events	72211				
Consultation messages sent	0				

Simulation run 56					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.8115	0.9839	0.0000	0.0000
Recall	1.0000	0.9479	0.9928	0.0000	0.0000
F1-score	1.0000	0.8744	0.9883	0.0000	0.0000
F1-score average	0.5725456548698753				
Average local time(s)	0.02088532523344809				
Average consultation time(s)	0.16259432618608946				
Total number of CAN-frames	799292				
Total number of security events	72211				
Consultation messages sent	9205				

Simulation run 57					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.8018	0.9839	0.0000	0.0000
Recall	1.0000	0.9473	0.9924	0.0000	0.0000
F1-score	1.0000	0.8685	0.9881	0.0000	0.0000
F1-score average	0.5713283301234593				
Average local time(s)	0.02061403540888654				
Average consultation time(s)	0.19136733757549496				
Total number of CAN-frames	799292				
Total number of security events	72211				
Consultation messages sent	9205				

Simulation run 58					
	Flooding	*Fuzzing*	Normal	*Replay*	*Spoofing*
Precision	1.0000	0.2542	0.9827	0.0000	0.8863
Recall	1.0000	0.9646	0.9210	0.0000	1.0000
F1-score	1.0000	0.4024	0.9508	0.0000	0.9397
F1-score average	0.6585852937026102				
Average local time(s)	0.01121088262143662				
Average consultation time(s)	N/A				
Total number of CAN-frames	889395				

Total number of security events	128821
Consultation messages sent	0

Simulation run 59					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3407	0.9826	0.0000	0.9278
Recall	1.0000	0.9593	0.9482	0.0000	1.0000
F1-score	1.0000	0.5029	0.9651	0.0000	0.9625
F1-score average	0.6860918087155801				
Average local time(s)	0.02874992386869768				
Average consultation time(s)	0.14243647631099784				
Total number of CAN-frames	889395				
Total number of security events	128821				
Consultation messages sent	26032				

Simulation run 60					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3456	0.9826	0.0000	0.8617
Recall	1.0000	0.9595	0.9489	0.0000	1.0000
F1-score	1.0000	0.5082	0.9655	0.0000	0.9257
F1-score average	0.6798782952473199				
Average local time(s)	0.02865986921500828				
Average consultation time(s)	0.16373201245791208				
Total number of CAN-frames	889395				
Total number of security events	128821				
Consultation messages sent	26032				

Simulation run 61					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2635	0.9830	0.0000	0.3361
Recall	1.0000	0.9141	0.9197	0.0000	1.0000
F1-score	1.0000	0.4090	0.9503	0.0000	0.5031
F1-score average	0.5724729581024459				
Average local time(s)	0.01108266144745935				
Average consultation time(s)	N/A				
Total number of CAN-frames	806390				
Total number of security events	119946				
Consultation messages sent	0				

Simulation run 62					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.5244	0.9836	0.0000	0.3504
Recall	1.0000	0.9065	0.9731	0.0000	1.0000
F1-score	1.0000	0.6644	0.9783	0.0000	0.5189

F1-score average	0.6323256286040959
Average local time(s)	0.03865799916539484
Average consultation time(s)	0.17935920611537934
Total number of CAN-frames	806390
Total number of security events	119946
Consultation messages sent	46933

Simulation run 63					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3180	0.9830	0.0000	0.3505
Recall	1.0000	0.9070	0.9385	0.0000	1.0000
F1-score	1.0000	0.4709	0.9602	0.0000	0.5190
F1-score average	0.5900404918491323				
Average local time(s)	0.03838480249856085				
Average consultation time(s)	0.22142436434107926				
Total number of CAN-frames	806390				
Total number of security events	119946				
Consultation messages sent	46933				

Simulation run 64					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9957	0.2265	0.9815	0.0000	0.0311
Recall	1.0000	0.9457	0.7844	0.0000	1.0000
F1-score	0.9978	0.3654	0.8719	0.0000	0.0603
F1-score average	0.4590997718962645				
Average local time(s)	0.00907380395375699				
Average consultation time(s)	N/A				
Total number of CAN-frames	817042				
Total number of security events	225855				
Consultation messages sent	0				

Simulation run 65					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9964	0.2445	0.9810	0.0000	0.0311
Recall	1.0000	0.9410	0.7940	0.0000	1.0000
F1-score	0.9982	0.3882	0.8777	0.0000	0.0603
F1-score average	0.46488099573725855				
Average local time(s)	0.01296815704643392				
Average consultation time(s)	0.10321753614773922				
Total number of CAN-frames	817042				
Total number of security events	225855				
Consultation messages sent	9420				

Simulation run 66

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9964	0.2477	0.9811	0.0000	0.0311
Recall	1.0000	0.9422	0.7954	0.0000	1.0000
F1-score	0.9982	0.3922	0.8785	0.0000	0.0604
F1-score average	0.4658721544037692				
Average local time(s)	0.01300273314581389				
Average consultation time(s)	0.11433701381055666				
Total number of CAN-frames	817042				
Total number of security events	225855				
Consultation messages sent	9420				

Simulation run 67					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9737	0.2244	0.9696	0.0000	0.0322
Recall	1.0000	0.9437	0.8024	0.0000	0.3217
F1-score	0.9867	0.3625	0.8781	0.0000	0.0586
F1-score average	0.4571818731623981				
Average local time(s)	0.00926444469179732				
Average consultation time(s)	N/A				
Total number of CAN-frames	2000733				
Total number of security events	511897				
Consultation messages sent	0				

Simulation run 68					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9810	0.2993	0.9701	0.0000	0.0332
Recall	1.0000	0.9399	0.8324	0.0000	0.3217
F1-score	0.9904	0.4540	0.8960	0.0000	0.0602
F1-score average	0.4801101613293661				
Average local time(s)	0.02457990077196037				
Average consultation time(s)	0.12626114170728656				
Total number of CAN-frames	2000733				
Total number of security events	511896				
Consultation messages sent	80120				

Simulation run 69					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9810	0.3199	0.9701	0.0000	0.0332
Recall	1.0000	0.9385	0.8375	0.0000	0.3217
F1-score	0.9904	0.4771	0.8989	0.0000	0.0602
F1-score average	0.4853384832832968				
Average local time(s)	0.02445319888792081				
Average consultation time(s)	0.1407474543045195				
Total number of CAN-frames	2000733				

Total number of security events	511896
Consultation messages sent	80119

Simulation run 70					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9480	0.1997	0.9827	0.0000	0.0276
Recall	1.0000	0.9434	0.7698	0.0000	1.0000
F1-score	0.9733	0.3297	0.8633	0.0000	0.0537
F1-score average	0.44399609167496423				
Average local time(s)	0.00900703845882573				
Average consultation time(s)	N/A				
Total number of CAN-frames	889395				
Total number of security events	253698				
Consultation messages sent	0				

Simulation run 71					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9631	0.2417	0.9828	0.0000	0.0278
Recall	1.0000	0.9396	0.7949	0.0000	1.0000
F1-score	0.9812	0.3844	0.8789	0.0000	0.0542
F1-score average	0.4597435013243105				
Average local time(s)	0.02127223472596935				
Average consultation time(s)	0.1206767295378248				
Total number of CAN-frames	889395				
Total number of security events	253698				
Consultation messages sent	31928				

Simulation run 72					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9631	0.2664	0.9829	0.0000	0.0278
Recall	1.0000	0.9394	0.8050	0.0000	1.0000
F1-score	0.9812	0.4151	0.8851	0.0000	0.0542
F1-score average	0.4671189126956188				
Average local time(s)	0.02120892985000297				
Average consultation time(s)	0.13202957308283428				
Total number of CAN-frames	889395				
Total number of security events	253698				
Consultation messages sent	31928				

Simulation run 73					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3669	0.9794	0.0000	0.0000
Recall	1.0000	0.9669	0.9498	0.0000	0.0000
F1-score	1.0000	0.5319	0.9644	0.0000	0.0000

F1-score average	0.49925697862518426
Average local time(s)	0.01199094708738224
Average consultation time(s)	N/A
Total number of CAN-frames	889395
Total number of security events	102384
Consultation messages sent	0

Simulation run 74					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.4295	0.9793	0.0000	0.0000
Recall	1.0000	0.9640	0.9602	0.0000	0.0000
F1-score	1.0000	0.5943	0.9697	0.0000	0.0000
F1-score average	0.5127828595173121				
Average local time(s)	0.02270185109935061				
Average consultation time(s)	0.13490751541744234				
Total number of CAN-frames	889395				
Total number of security events	102384				
Consultation messages sent	12566				

Simulation run 75					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.4298	0.9794	0.0000	0.0000
Recall	1.0000	0.9641	0.9603	0.0000	0.0000
F1-score	1.0000	0.5946	0.9697	0.0000	0.0000
F1-score average	0.512852615560583				
Average local time(s)	0.02249040781274375				
Average consultation time(s)	0.14737918005804718				
Total number of CAN-frames	889395				
Total number of security events	102384				
Consultation messages sent	12566				

Simulation run 76					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.5883	0.9714	0.0000	0.0000
Recall	1.0000	0.9656	0.9779	0.0000	0.0000
F1-score	1.0000	0.7311	0.9747	0.0000	0.0000
F1-score average	0.5411571133525003				
Average local time(s)	0.01281932621653754				
Average consultation time(s)	N/A				
Total number of CAN-frames	1751313				
Total number of security events	181614				
Consultation messages sent	0				

Simulation run 77

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7805	0.9712	0.0000	0.0000
Recall	1.0000	0.9591	0.9911	0.0000	0.0000
F1-score	1.0000	0.8606	0.9810	0.0000	0.0000
F1-score average	0.5683364000548614				
Average local time(s)	0.02091047717288571				
Average consultation time(s)	0.17159639621317987				
Total number of CAN-frames	1751313				
Total number of security events	181614				
Consultation messages sent	26992				

Simulation run 78					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.7934	0.9712	0.0000	0.0000
Recall	1.0000	0.9596	0.9917	0.0000	0.0000
F1-score	1.0000	0.8686	0.9813	0.0000	0.0000
F1-score average	0.569993276727365				
Average local time(s)	0.02057189805622596				
Average consultation time(s)	0.20402609196087984				
Total number of CAN-frames	1751313				
Total number of security events	181614				
Consultation messages sent	26992				

Simulation run 79					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.3434	0.9772	0.0000	0.0000
Recall	1.0000	0.9510	0.9412	0.0000	0.0000
F1-score	1.0000	0.5046	0.9589	0.0000	0.0000
F1-score average	0.4926920217606742				
Average local time(s)	0.01167097021042733				
Average consultation time(s)	N/A				
Total number of CAN-frames	817042				
Total number of security events	104573				
Consultation messages sent	0				

Simulation run 80					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.3996	0.9773	0.0000	0.0000
Recall	1.0000	0.9472	0.9532	0.0000	0.0000
F1-score	1.0000	0.5621	0.9651	0.0000	0.0000
F1-score average	0.5054307193365066				
Average local time(s)	0.02193113432495054				
Average consultation time(s)	0.11076986213530864				
Total number of CAN-frames	817042				

Total number of security events	104573
Consultation messages sent	10407

Simulation run 81					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9999	0.4004	0.9773	0.0000	0.0000
Recall	1.0000	0.9478	0.9533	0.0000	0.0000
F1-score	1.0000	0.5630	0.9651	0.0000	0.0000
F1-score average	0.5056057216924295				
Average local time(s)	0.02191854739071162				
Average consultation time(s)	0.12338044683035437				
Total number of CAN-frames	817042				
Total number of security events	104573				
Consultation messages sent	10407				

Simulation run 82					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9850	0.3317	0.9732	0.0000	0.0312
Recall	1.0000	0.9439	0.8177	0.0000	0.3062
F1-score	0.9924	0.4909	0.8887	0.0000	0.0567
F1-score average	0.4857537666996848				
Average local time(s)	0.00931420431380791				
Average consultation time(s)	N/A				
Total number of CAN-frames	1751313				
Total number of security events	441293				
Consultation messages sent	0				

Simulation run 83					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9863	0.3776	0.9731	0.0000	0.0315
Recall	1.0000	0.9403	0.8301	0.0000	0.3062
F1-score	0.9931	0.5388	0.8959	0.0000	0.0571
F1-score average	0.4969858011277147				
Average local time(s)	0.01480238286663409				
Average consultation time(s)	0.10824584629193988				
Total number of CAN-frames	1751313				
Total number of security events	441293				
Consultation messages sent	25657				

Simulation run 84					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	0.9863	0.3761	0.9730	0.0000	0.0315
Recall	1.0000	0.9392	0.8298	0.0000	0.3062
F1-score	0.9931	0.5371	0.8957	0.0000	0.0571

F1-score average	0.496582074073548
Average local time(s)	0.01475588529714444
Average consultation time(s)	0.11903079099581183
Total number of CAN-frames	1751313
Total number of security events	441294
Consultation messages sent	25657

Simulation run 85					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.2531	0.9845	0.0000	0.9677
Recall	1.0000	0.9686	0.9150	0.0000	1.0000
F1-score	1.0000	0.4014	0.9485	0.0000	0.9836
F1-score average	0.6666805094051171				
Average local time(s)	0.01104313598860028				
Average consultation time(s)	N/A				
Total number of CAN-frames	799292				
Total number of security events	122895				
Consultation messages sent	0				

Simulation run 86					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3226	0.9843	0.0000	0.9782
Recall	1.0000	0.9628	0.9398	0.0000	1.0000
F1-score	1.0000	0.4833	0.9616	0.0000	0.9890
F1-score average	0.6867787984075168				
Average local time(s)	0.02719627027846706				
Average consultation time(s)	0.13694631473029373				
Total number of CAN-frames	799292				
Total number of security events	122895				
Consultation messages sent	21753				

Simulation run 87					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.3226	0.9843	0.0000	0.9782
Recall	1.0000	0.9623	0.9398	0.0000	1.0000
F1-score	1.0000	0.4832	0.9616	0.0000	0.9890
F1-score average	0.686746536090552				
Average local time(s)	0.02701090538457659				
Average consultation time(s)	0.15197091230177548				
Total number of CAN-frames	799292				
Total number of security events	122895				
Consultation messages sent	21753				

Simulation run 88

	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.6549	0.9834	0.0000	0.7734
Recall	1.0000	0.9506	0.9807	0.0000	1.0000
F1-score	1.0000	0.7755	0.9821	0.0000	0.8722
F1-score average	0.725958373763487				
Average local time(s)	0.01239648328296182				
Average consultation time(s)	N/A				
Total number of CAN-frames	1751313				
Total number of security events	196471				
Consultation messages sent	0				

Simulation run 89					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.9102	0.9833	0.0000	0.7725
Recall	1.0000	0.9440	0.9942	0.0000	1.0000
F1-score	1.0000	0.9268	0.9887	0.0000	0.8717
F1-score average	0.7574392245255972				
Average local time(s)	0.02038490765084561				
Average consultation time(s)	0.16132129927770034				
Total number of CAN-frames	1751313				
Total number of security events	196471				
Consultation messages sent	26713				

Simulation run 90					
	Flooding	*Fuzzing*	*Normal*	*Replay*	*Spoofing*
Precision	1.0000	0.9166	0.9834	0.0000	0.7726
Recall	1.0000	0.9442	0.9944	0.0000	1.0000
F1-score	1.0000	0.9302	0.9889	0.0000	0.8717
F1-score average	0.7581546040039193				
Average local time(s)	0.02032375087102205				
Average consultation time(s)	0.18919854690174187				
Total number of CAN-frames	1751313				
Total number of security events	196470				
Consultation messages sent	26713				

C

Appendix 3

C.1 Raspberry PI 4 Model B

Specifications of the Raspberry PI 4 Model B, derived from the official Raspberry PI webpage [26]:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient