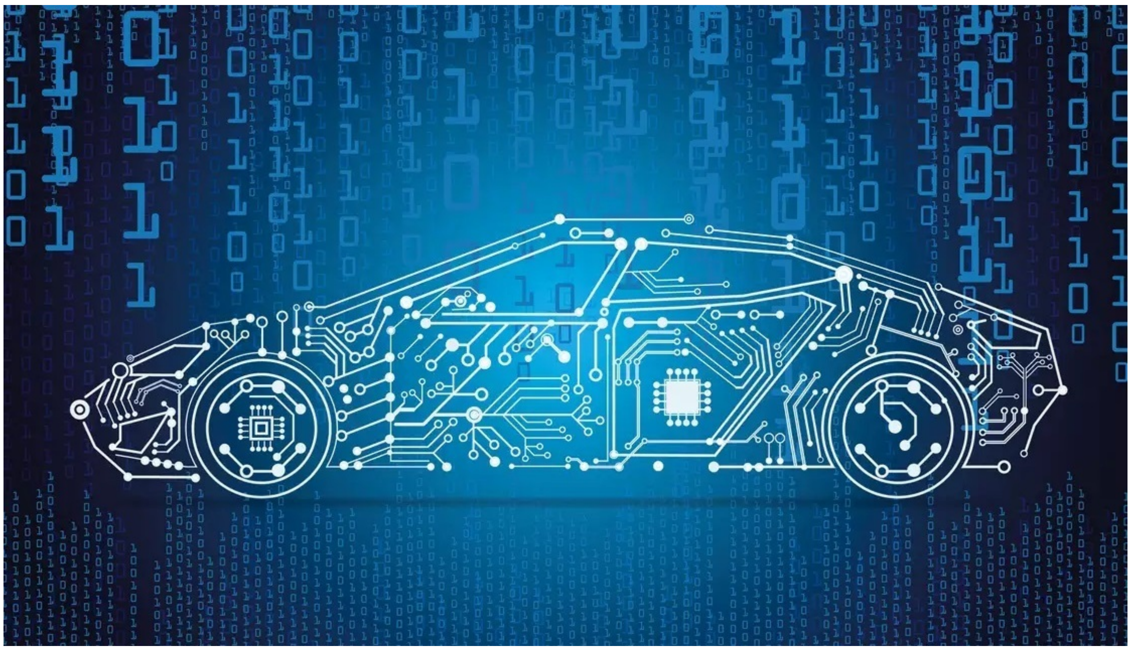


Riposte: A Collaborative Cyber Attack Response Framework for the Automotive Systems

Master's thesis in Software Engineering and Technology

SAIF ALDAGHISTANI

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

Master's thesis 2021

Riposte: A Collaborative Cyber Attack Response Framework for the Automotive Systems

SAIF ALDAGHISTANI

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Riposte: A Collaborative Cyber Attack Response Framework for the Automotive Systems

SAIF ALDAGHISTANI

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
Division of Software Engineering and Technology
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image of connected lines and dots representing a car, and digits of zeros and ones representing binary data [1].

Gothenburg, Sweden 2021

Riposte: A Collaborative Cyber Attack Response Framework for the Automotive Systems
Saif Aldaghistani
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Context: Today, sophisticated technologies and brilliant solutions, that were considered impossible before, are all around us. Many of these advancements in technologies were possible thanks to the introduction of the Internet. This wide connectivity between different devices and the ability to communicate no matter the distance have boosted the possibilities to levels unseen before. The automotive domain has got its own share of advancements from the Internet, providing more services and functionalities. However, this wide connectivity and openness to the Internet raise cyber security concerns. These services that depend on online connectivity can serve as entry points for hackers to perform attacks on different assets of the vehicle. Some of these attacks could be considered minor, but some could be considered severe and potentially threaten human lives.
Problem: That being said, choosing the right and most suitable response technique for an ongoing cyber attack could be challenging, especially when there are many response techniques to choose from.
Objective: Thus, this study aims to provide a solution in order to help identify and evaluate different cyber attack response techniques for a specific cyber attack in real-time.
Approach: To achieve this goal, the study will follow the design science research methodology by performing three iterations. The first iteration would focus on finding out and discussing the available state-of-the-art cyber attack response techniques from the literature. The second iteration would discuss and propose a solution to help evaluate suitable response techniques in real-time. Finally, the third iteration would focus on evaluating the proposed solution from the safety aspect.
Results: The results for this study were a taxonomy of state-of-the-art cyber attack response techniques regarding the software asset and their descriptions, a collaborative framework that evaluates different suitable response techniques when an attack is in progress, and an evaluation of safety by conducting a qualitative study with different experts through a questionnaire.
Conclusion: In conclusion, the proposed framework was deemed slightly unsafe based on the perception of participants. That being said, the participants provided some insight on how to improve the overall safety of the framework in order to make it safer for deployment.

Keywords: connected car, automotive security, response techniques, cyber security, vehicle collaboration.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my academic supervisor Rodi Jolak for his constant support and advice throughout the thesis work. I would also like to thank Christian Sandberg and Thomas Rosenstatter for sharing their time and invaluable knowledge whenever needed. I would like to thank my examiner Christian Berger for all the constructive feedback received throughout the thesis work. Furthermore, I would like to thank my family for their unlimited support throughout the master's study journey. Finally, I would like to thank my fiancé, Maryam, for her patience and all the love and support she provided during this study.

Saif Aldaghistani, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1 Introduction

With the introduction of 5G cellular technology, the automotive systems will have an increased level of automation, including tighter integration with other vehicles, traffic infrastructures, and different cloud services [14]. While this improves the overall safety of the passengers and opens up the system to new opportunities, it would also increase the complexity and uncertainty of such systems [14]. The wide connectivity and high reliability over Internet services raise cyber security concerns, especially in the automotive domain; being a safety and real-time critical system that is being operated by millions of individuals [15]. These online services can provide attackers with entry points that can be used to perform attacks on different assets of the vehicle. Some of these attacks can be considered minor, but some can be considered severe and can threaten human lives.

To deal with such vulnerabilities, a great effort has been spent in the past years. This in order to establish guidelines and standards for the security of the automotive domain and in identifying security principles [15]. In 2017, the European Union Agency for Network and Information Security (ENISA) published cyber security guidelines and recommendations for the automotive domain [16]. Additionally, the International Organization for Standardization (ISO) has an upcoming ISO/SAE standard (ISO 21434) for cyber security engineering for road vehicles [17].

To comply with these guidelines and standards, different techniques are proposed for cyber attack detection, analysis, and response. To explain these terms, Stojanović et al. [18] define cyber attack detection as the ability for a system to recognize unauthorized activity or access that happens in a network-based environment. Moreover, Ochieng [19] describes cyber attack analysis as "the process of assessing the cyber activities and capabilities of unknown intelligent entities or criminals"; this is to measure the impact of a cyber attack and its feasible path within the system. As for cyber attack response, it is the process of blocking, quarantining, or generally dealing with a threat that has been identified by the cyber attack detection system, preferably with a minimum amount of losses caused to the system.

This study aims to explore available cyber attack response techniques for cyber security attacks against the automotive systems. It also aims to investigate collaborative ways for evaluating these cyber attack response techniques and finding which one of them is the most suitable for a given ongoing attack.

1.1 Statement of the Problem

Vehicles nowadays are not considered as a mere means of transportation any longer. A vehicle is considered to be a network of nodes that can deliver different driving needs and services to its user. Being able to share data across surrounding vehicles on road, the ability to anticipate a sudden stop of other vehicles, and all other real-time internet services like streaming of high-definition videos and ultimately self-driving vehicles, are all services that the industry is racing towards. Some services are already implemented, and some will be with the deployment of 5G cellular networks. By providing high bandwidth, millisecond latencies, and improved reliability, the dream of self-driving vehicles is closer to reality than ever before.

However, the automotive domain is a safety and real-time critical field, and millions of individuals use it every day [15]. This wide connectivity and openness to the internet provide users with uncountable services, but it comes with the risk of increasing cyber security threats. These services can serve as entry points and allow for cyber attacks against such systems. These cyber attacks can target different assets and different quality attributes of the system. These assets are hardware, software, network, and data, while attributes could be vehicle performance, resource usage, privacy of users, the safety of the passengers as well as the pedestrians around the vehicle, and more.

The approach to solve such vulnerabilities can be challenging, with many security standards and protocols being researched and developed continuously. But eventually, attacks will happen, and as the ex-CEO of Sony 'Howard Stringer' once said:" Nobody's system is 100 percent secure" [20]. Therefore, an extra effort could be spent on cyber attack response techniques to reduce the impact of cyber attacks on a system and/or eliminate them with minimum losses.

1.2 Purpose of Study

The purpose of this study is to explore the available cyber attack response techniques on the software asset of the vehicle and then design and develop a framework in which helps evaluate these cyber attack response techniques collaboratively, using the Continuous Experimentation practice. By contributing a dataset of cyber attack response techniques for the software asset; a collaborative framework to identify an effective and most efficient cyber attack response technique against an ongoing attack; and an evaluation of safety of the proposed framework. The goal of providing a safer and better cyber attack response technique against such a cyber threat is abridged.

This study is intended to benefit both the researchers and practitioners; by providing a dataset of available cyber attack response techniques for the software asset of the vehicle, and a collaborative framework proposal for evaluating the best cyber attack response variant for that specific cyber attack for the latter.

1.3 Significance of the Study

Overall, the study aims to explore the state-of-the-art cyber attack response techniques that are available at the current time and to develop a collaborative framework that helps in evaluating these cyber attack response techniques against cyber attacks.
The concrete contributions of this study are to:

- Provide a dataset of cyber attack response techniques for the software asset of the vehicle
- Design and implement a collaborative framework to help evaluate and pick an effective and most efficient cyber attack response technique against an ongoing cyber attack
- Evaluate this framework in terms of safety

1.4 Research Questions

The design of this thesis is to first conceptualize the problem by learning the domain and identifying the available state-of-the-art cyber attack response techniques from literature, then design a solution (artefact) for evaluating these techniques in terms of effectiveness and efficiency along with its implementation, and finally evaluate the safety aspect of the proposed artefact. Therefore, this thesis is broken down into three iterative research questions that will be answered using the design science research (DSR) methodology.
RQ1: *What are the suitable response techniques for cyber attacks against the software asset of vehicles?*

This research question aims to categorize suitable cyber attack response techniques for attacks based on the software asset of the vehicle.

RQ2: *How to support online collaborative cyber attack response between different vehicles to select the right cyber attack response technique when an attack is in progress?*

This research question aims to investigate the design of collaboration between different vehicles against a cyber attack by evaluating different suitable cyber attack response techniques when an attack is in progress. This in order to find out an effective and most efficient cyber attack response technique for that specific attack to be shared with the vehicles.

RQ3: *What is the perception of experts on the safety of the proposed solution?*

This research question aims to look for safety concerns and considerations when implementing the proposed solution by conducting a qualitative study with security and/or safety experts.

1.5 Scope

The study will use the taxonomies provided by [15] to elicit the state-of-the-art response techniques for specific attacks on the different assets of the system, any additional taxonomies will not be considered. Furthermore, this study will consider the software asset of the vehicle and not consider any other asset and will focus on the collaborative response of one attack that is related to that asset. Additionally, the study will only consider three suitable response techniques based on the considered attack on the software asset.
That being said, the cyber attack response technique usually sits at the end of the security chain (Firewall -> Intrusion detection system -> Antivirus -> Response technique). And since this study's main consideration is to explore response techniques and propose a solution to evaluate them, it will not focus on implementing a complex attack scenario nor a sophisticated detection mechanism.

1.6 Thesis Outline

The remaining parts of the thesis will be structured as follows: Chapter 2 provides background knowledge and related work on fields that are related to this study, chapter 3 describes the research methodology of the study and the data collection/-analysis methods used, chapter 4 presents the results and discussion of the study, chapter 5 describes the threats to validity of this study, and finally the conclusion of this study is discussed in chapter 6.

2
Background

This chapter provides theory and background information that covers the technical and scientific concepts presented in this thesis. Section 2.1 describes the modern connected car and the technologies used to make it as such. This is followed by some background information in automotive cyber security concerns in 2.2. In section 2.3, the continuous practices are explained. Finally, related work is discussed in 2.4.

2.1 The Connected Car

Up to the 1970s, vehicles were mainly built and operated using purely mechanical parts. It is during the eighties when the first adoption of electronic components was introduced. Since then, vehicles were built using a mix of both digital hardware and software components. Nowadays, software is gaining central importance in the process of vehicular applications and services development enabling sophisticated functionalities that cannot be achieved using hardware components alone. Examples on software applications in a vehicle can range from simple control units to full autonomous driving. In fact, vehicles nowadays are considered to be the most software-intensive systems in the universe [21] with millions of lines of code. Figure 2.1 shows a comparison between the number of lines of code between different systems.

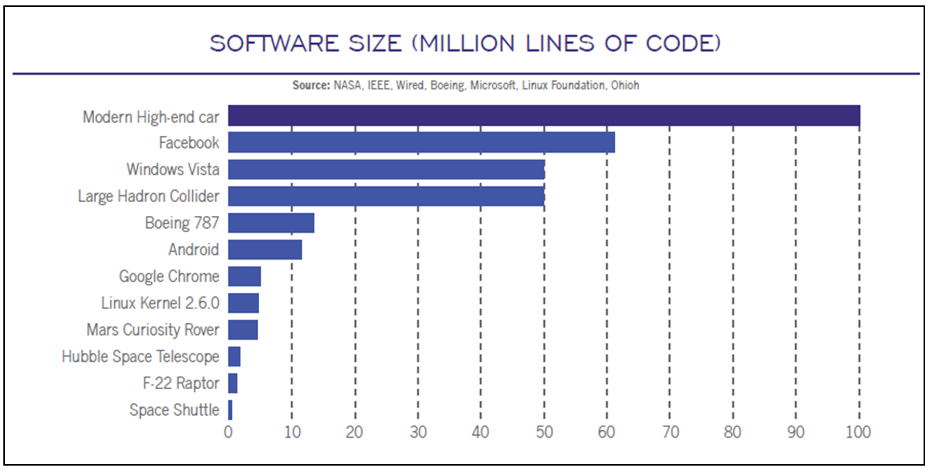


Figure 2.1: Software size difference in different systems [2].

The software is distributed and executed on multiple small computers that reside inside the car, called Electronic Control Units (ECUs). These ECUs are small computers that are interconnected with each other to perform certain functionalities, depending on their purpose. ECUs with similar tasks and functionalities are connected together in functional domains, and each domain is then connected with other domains through a central hub, called the Central Gateway (CGW), as shown in figure 2.5

2.1.1 Electronic Control Units

Today, the internal architecture of vehicles is quite complex and distributed over a hundred of ECUs. These ECUs are small computers that oversee and regulate certain functions inside a vehicle by processing information from sensors and actuators and by communicating with other ECUs through the network. These functionalities can range from simple automated operations to more complicated ones, such as controlling the windshield wipers when sensing rain, driver alertness monitoring, automatic parking system, and many others. Figure 2.2 shows a few ECU functionalities that can be found in modern vehicles.
The way that an ECU works is based on electronic signals received from sensors. Upon receiving a signal, the ECU then (based on how it was programmed) sends instructions to actuators to perform certain operations. ECUs inside the vehicle can also communicate with each other to perform collaborative tasks using the in-vehicle networks (to be discussed in subsection 2.1.3).
There are many architectures used by the industry when it comes to building ECUs, with each ECU having its protocols of communicating/interacting with the environment. However, most of them share the technical specifications shown in figure 2.3 to some extent.
These hardware differences between different ECU manufacturers have increased

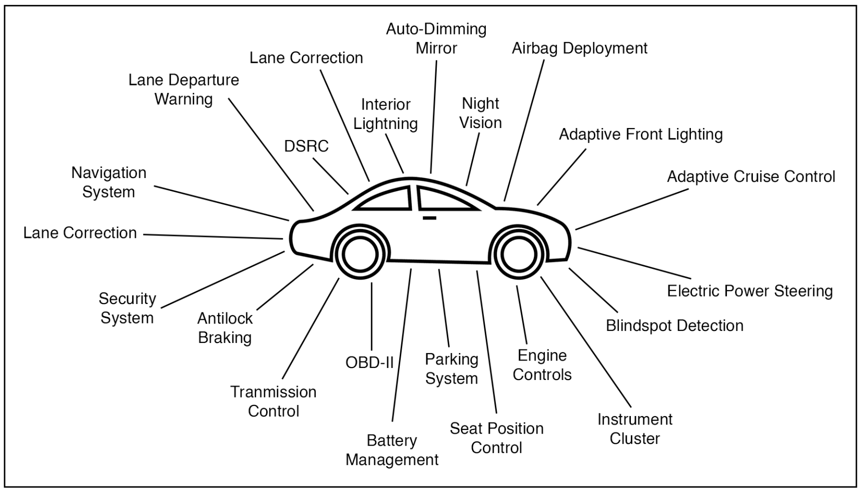


Figure 2.2: A few examples on the Electronic Control Unit functionalities that can be found in a modern vehicle.

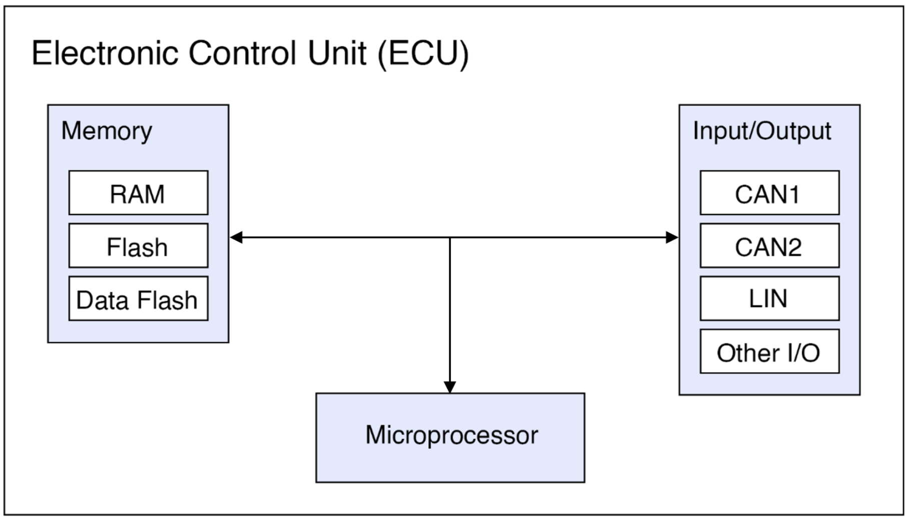


Figure 2.3: A simplified representation of an Electronic Control Unit. Adapted from [3].

the complexity of ECUs, making the process of software development harder. This led to the development of the Automotive Open System Architecture (AUTOSAR); an open standardized software architecture for automotive ECUs[1] which provides a standardized software architecture for ECUs that is independent of what hardware is being used.

2.1.2 Inter-Vehicle Communications

With the expansion of Internet connectivity in general and the implementation of 5G technology specifically, it was only logical that every electronic device could benefit from services that, before the Internet, were not possible. So is the case for the automotive domain, the term "Connected car" is used to identify a vehicle that has connections to the outside external networks, including the Internet and other networks, in order to improve the driving experience by providing better services and safety functions to its users.

Through connectivity, users can access features like traffic reports, service information regarding their vehicle, and to open up more services in the infotainment system; a combination of systems that enables the delivery of information and entertainment to the passengers. The infotainment system supports features like enabling the use of touch screen displays, watching live streaming videos, browsing the web and social media, the use of voice commands, and many other services.

That being said, modern vehicles can connect not only to the Internet but to their surroundings as well. Vehicle connectivity is often referred to as Vehicle to Everything Communications (V2X), which is considered to be the parent category. It is further divided into multiple subcategories, as shown in figure 2.4.

[1]https://www.fpt-software.com/what-is-autosar-and-why-is-it-important/

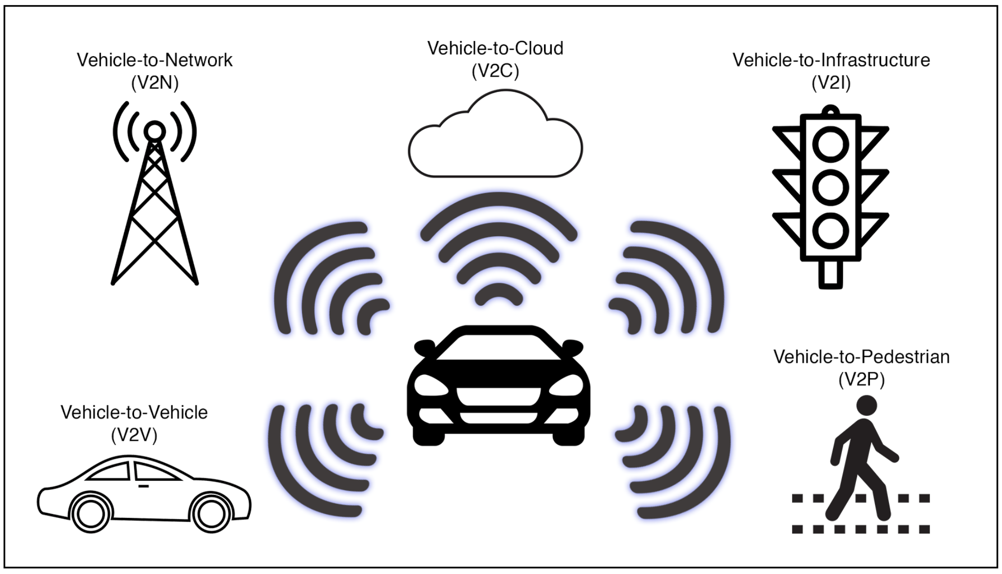


Figure 2.4: Different vehicle connectivity forms.

Vehicle-to-Network (V2N) communication enables the vehicle to use cellular networks for communicating with the V2X management system. It uses Dedicated Short-Range Communications (DSRC) standard to interact with other vehicles and infrastructures around it. This connectivity allows the vehicles to be considered as normal digital devices, like a smartphone or tablet. It also allows the vehicle to receive information like traffic alerts, weather information, and congestion status. By using 5G, LTE, and DSRC technologies, V2N allows the vehicle to interact reliably with other vehicles, devices, infrastructures, and even pedestrians around it.
Vehicle-to-Infrastructure (V2I) communication is used to exchange information bidirectionally between the vehicle and the road infrastructure, which is an integral part of the Intelligent Transportation Systems (ITS). This information can include traffic data that has been gathered from other vehicles or sensors on the road infrastructure. It could also include other information, such as speed limits, traffic jams, and weather conditions. V2I and ITS are considered to be the keys for autonomous vehicles, as they hold valuable information that vehicles could rely on.
Vehicle-to-Vehicle (V2V) communication allows vehicles to exchange information in real-time. This is performed wirelessly via DSRC, the same technology that is used in V2I communications. With V2V, vehicles can share data such as their speed, destination, location, and any other relevant information. When a vehicle starts V2V communications, it becomes a node inside a mesh network, in which it can capture, send, and re-transmit signals to/from vehicles.
Vehicle-to-Cloud (V2C) communication makes use of the access to broadband cellular mobile networks that are provided by V2N in order to offer data exchange with the cloud. Some applications that leverage this communication include Over-the-Air (OTA); a way to update the software of the vehicle remotely, and bidirectional communications with other IoT devices that are also connected to the cloud.

Vehicle-to-Pedestrian (V2P) communication can help vehicles detect and identify wheelchairs, strollers, and bicycles when a smart sensor is attached to them. With that, some level of awareness could be established. Some vehicles can also use LiDAR technology to detect pedestrians and expect possible collisions.

2.1.3 In-Vehicle Communications

In order to provide various services and functionalities to the driver/passengers, some collaboration of ECUs is likely to be required. It is also likely that some services require higher bandwidth than others in order to perform their desired tasks. For these reasons, different technologies are used within the vehicle in order to provide different types of connectivity, depending on the needs. Thus, the vehicle is divided into different functional domains that can perform certain operations, each requiring different properties depending on its applications.

That being said, it is not possible for different ECUs and functional domains that are using different communication protocols to properly communicate with each other. The reason being is that each functional domain has different network technologies implemented within. For that reason, all functional domains are connected to a central hub called the Central Gateway (CGW) [4]. The CGW's main task is to process data between the heterogeneous vehicle networks by translating the different protocols and routing data between different functional domains. It also provides an additional layer of security by physically separating functional domains from each other, as shown in figure 2.5.

As of today, there are five main types of networking technologies used inside the vehicle:

- **Controller Area Network (CAN):** introduced by Bosch in 1986, CAN is a bus system that is the dominant standard used by the industry for in-vehicle communications due to it being cost-efficient and more flexible when compared to other network technologies. In CAN, the data is segmented into frames that are labeled to determine the priority when being transmitted. Every segmented frame is then sent sequentially thereafter. The payload of each frame permits up to 8 bytes of data, with a maximum speed of 1 Mbps. That being said, there is an alternative to CAN, named CAN with flexible data-rate (CAN-FD), that supports up to 64 bytes of data and a speed that reaches 8 Mbps.
- **FlexRay:** developed by FlexRay Consortium, FlexRay was designed to provide faster and more reliable communications than what CAN has to offer. It is a bus system that is resilient to communication channel errors, coding/decoding, redundancy checks, and sampling signals. It can include static and dynamic segments; the static segment is based on time-division multiple access (TDMA), which is used in real-time communications, whereas the dynamic segment is based on an event-driven communication protocol, which is used in service-based messaging. FlexRay can support data rates of 10 Mbps as opposed to CAN that supports up to 1 Mbps. However, due to its higher cost and complex communication protocol, CAN is still the favourable choice in the industry. That being said, FlexRay is still used in safety-critical applications

where a strict time frame is required.

- **Media Oriented Systems Transport (MOST):** the MOST bus was introduced in order to meet the standard of applications that require higher data rates that other technologies could not offer. By providing low overhead and low-cost interfaces that can be implemented on speakers and microphones, making it ideal for multimedia and infotainment purposes.

- **Local Interconnect Network (LIN):** the first implemented specifications were released in 2002. LIN uses low-cost, flexible, and basic sensors to establish small networks. These subsystems can be established over a backbone network, like CAN. It has a very low speed of around 20 kbps when compared to other technologies. However, it is widely used in applications that do not require high data transfer speed, such applications as controlling the side mirrors, doors, and seats.

- **Automotive Ethernet (AE):** the AE is a growing technology that is capable of supporting a high bandwidth of 1 Gbps. It is more secure than CAN and LIN and implements IP-based routing schemes, which prevent ECUs from being hacked as well as hindering the attackers from taking full control over the whole Ethernet [22]. The AE can be easily adopted in the industry due to the existing common Ethernet standards.

Table 2.1 shows the technical specifications of the different networking technologies used for in-vehicle communications, followed by figure 2.5 that illustrates how they are connected through the CGW.

Table 2.1: Automotive Networks Overview [13].

	Max Speed	**Cable Type**	**Cost**	**Applications**
CAN	1 Mbps	Twisted pair, 5v	$$	ABS, Powertrain, Engine control
LIN	19.2 kbps	Single wire, 12v	$	Electric Seats, Mirrors, Tailgate
FlexRay	10 Mbps	2 or 4 wires	$$$	Steering, Traction control, Active suspension
MOST	23 Mbps	Fiber optic, Coax	$$$$	Media Players, Infotainment
Ethernet	1 Gbps	One or more twisted pairs	$$	IP Cameras, Radar, Infotainment

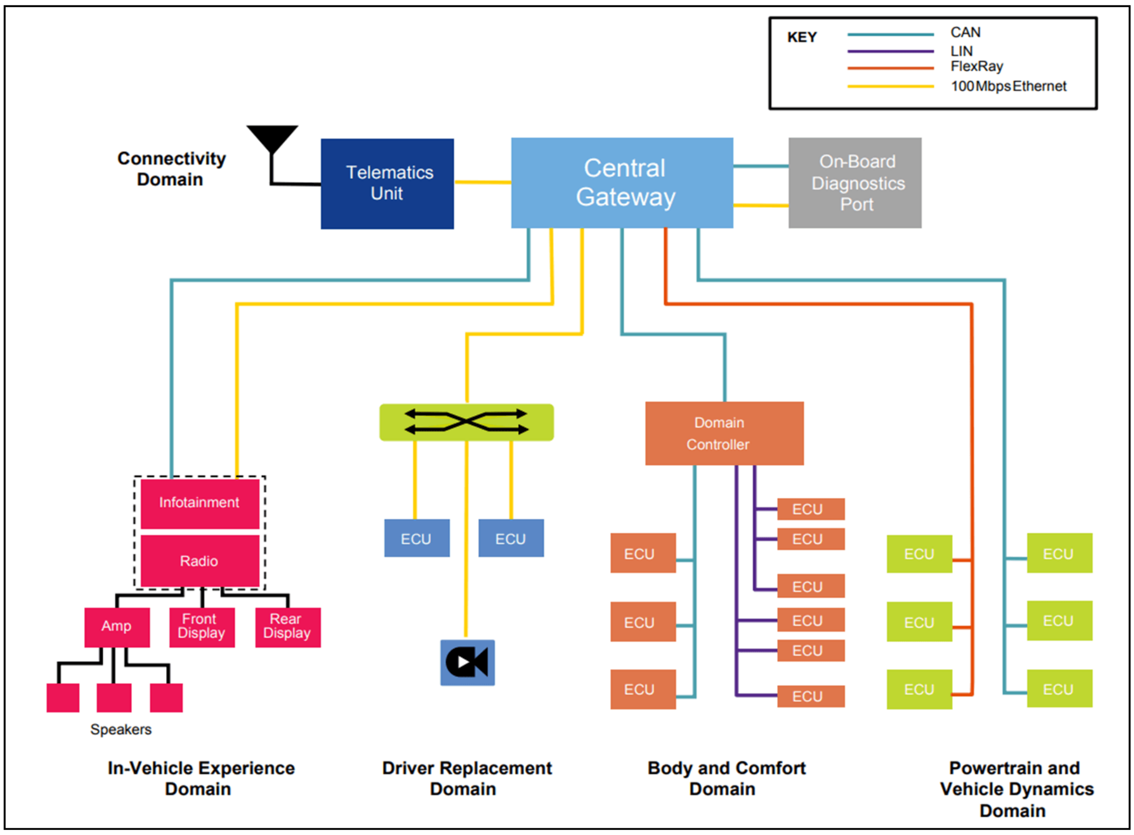


Figure 2.5: The Central Gateway bridging different functional domains and heterogeneous vehicle networks [4].

2.2 Vehicle Cyber Security Concerns

The automotive industry is growing every year through the huge developments of novel applications and services, with more than 90% of inventions lead to innovations in the vehicle's software and hardware [22]. Thanks to the modern advancements of electrical and electronics, specifically the ECUs, vehicles are now faster, more sophisticated, and more fuel-efficient.
The advancement in ECUs and the vast communication network that interconnects them enable new features to be achieved; remotely locking/unlocking, brakes, airbags, automatic parking, driving assistance, GPS, and many more. These achievements were possible because of the hundreds of megabytes of code written in the vehicle's ECUs [22].
However, as the vehicle becomes more advanced, the probability of cyber threats increases simultaneously. And since the ECUs are interconnected with each other through the CAN bus, any compromised ECU could potentially make the entire network in danger.
Today, vehicles can connect using multiple connection points to the outside networks, one of which is the Internet. This wide connectivity requires the application of cyber security principles to the various components of the vehicle in order to ensure the their safety and security from any type of malicious attacks. Rosenstatter et al. [15]

discuss four types of assets within a vehicle that can be compromised by an attacker; hardware, software, network/communication, and data storage.

- **Hardware:** can be divided into *ECUs*, *sensors*, and *actuators*. Each ECU can have different hardware and software, depending on the task it performs. An ECU can relate to the processing of one sensor signal, or it can deal with infotainment tasks that deal with many applications. Sensors can provide information, such as speed and temperature. Actuators can turn input from sensors into actions via the ECU, such as braking and steering.
  Attack example. Installing malicious hardware or tampering with one inside the vehicle. This can act as a mediator and enable the possibility to gain complete control over the vehicle [15].

- **Software:** can exist in different categories: *in-transit*, *at-rest* or *running*. In-transit can be related to software provisioning systems, like OTA or workshop updates. The running and at-rest categories can be related to software installation processes or software that runs in ECUs [15].
  Attack example. Software vulnerabilities could be exploited via a privilege escalation attack, which enables the ECU to be reprogrammed and include additional functionalities, such as enabling/adding remote access [15].

- **Network/Communication:** can be divided into *in-vehicle* and *inter-vehicle communications*. In-vehicle communications can be CAN, FlexRay, MOST, LIN, AE. Inter-vehicle communications can be Wi-Fi, Bluetooth, LTE, and V2X.
  Attack example. Denial-of-service attack.

- **Data Storage:** can be sensitive data, such as system information, cryptographic keys, forensics logs, and reports about the driver.
  Attack example. Secret keys can be exploited and used for sensitive diagnostics in order to disable the firewall.

Attackers typically exploit the assets mentioned above in order to achieve their goals through various vectors. Figure 2.6 shows examples of different faults and threats that could be performed against a connected vehicle.

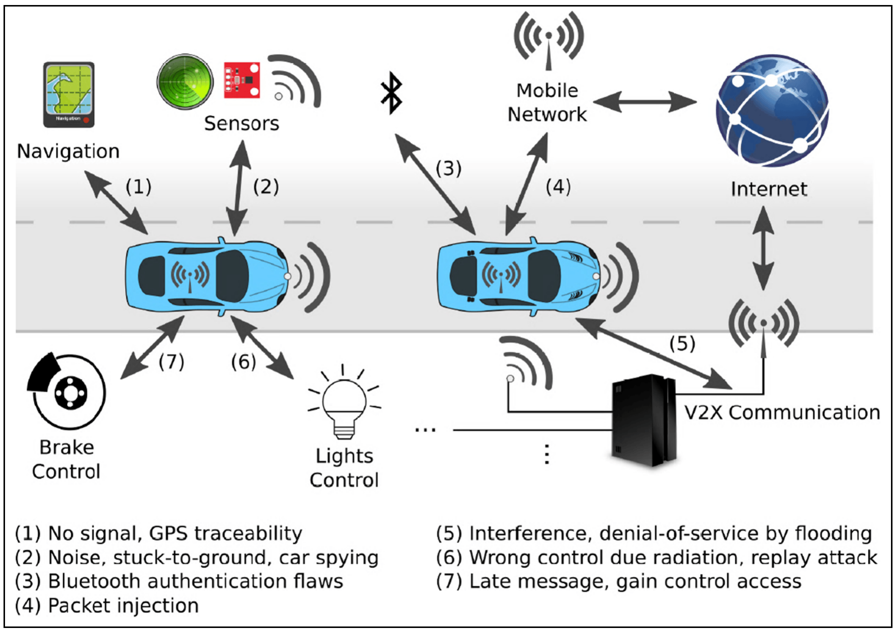

Navigation
Sensors
Mobile Network
Internet
(1)
(2)
(3)
(4)
(5)
(6)
(7)
Brake Control
Lights Control
V2X Communication
(1) No signal, GPS traceability
(2) Noise, stuck-to-ground, car spying
(3) Bluetooth authentication flaws
(4) Packet injection
(5) Interference, denial-of-service by flooding
(6) Wrong control due radiation, replay attack
(7) Late message, gain control access

Figure 2.6: Examples of threats and faults in a connected vehicle [5].

When it comes to securing a vehicle, it is definitely not a straight forward task. For that, Karahasanovic [23] discusses a few main difficulties that should be taken into account when securing a connected car:

- **Over-the-Air updates (OTA):** being referred to as "Computer on wheels" [24], the connected car has a very complex software architecture in order to support new enhanced features. For such systems, software updates are necessary to keep the products bug-free and continuously apply security patches. Some of these updates could have quite the challenge when applied, as they could be critical for the safety of the driver and passengers. Therefore, OTA ensures a connection between the vehicle and its manufacturer in order to apply software updates. However, this connectivity could potentially be abused by attackers in order to gain access to the vehicle.
- **Old vehicles technology:** because of the vehicle's long life-cycle, the decades years old vehicles would still be functional in the streets running old technologies. This could provide an advantage to attackers by overpowering the old vehicle through modern and more advanced computational power, making the process of exploiting the car easier.
- **Monitoring difficulty:** it is not easy to monitor the status of a vehicle by a certified authority because the vehicle might not always be connected to the Internet [23], [25].
- **Cost:** one of the major difficulties is to make all components of the vehicle secured. This requires the employment of more people as well as changing the entire development process to incorporate security [23].
- **No Safety without Security:** it requires only one malicious vehicle on the road to cause a potential hazard for the environment around it, such as other vehicles, pedestrians, and road infrastructures [23].

2.3 Continuous Practices

Continuous practices are techniques that are mainly used in the software industry to enable development, test, and deployment of software in a reliable and consistent manner. These techniques are Continuous Integration (CI), Continuous Deployment (CD), and Continuous Experimentation (CE).

Continuous Integration accelerates the development process by allowing immediate integration of the new software into the code base after being tested. This often happens in an automated fashion. Continuous Deployment, on the other hand, ensures that the newly integrated software is ready to be deployed into the final system. As for Continuous Experimentation, it allows the possibility to deploy and run different versions of a software alongside the official software in order to evaluate their respective performances [26].

The Continuous Experimentation was originated from the software-intensive web-based field. It guides the software evolution process by allowing the collection of real-world data from running experiments and use that to make informed decisions rather than opinions and past experience, i.e. closing the Open-Loop [27]. And since the automotive domain is growing rapidly and becoming more software-dependent, Continuous Experimentation might be suitable for implementation [26]. Some automotive domains have mechanisms to push updates into vehicles through Over-the-Air (OTA) technology. Hence, a link could be established between the vehicle and its manufacturer, and this is one of the prerequisites for Continuous Experimentation [26]. Figure 2.7 shows the Continuous Experimentation approach to the software evolution process.

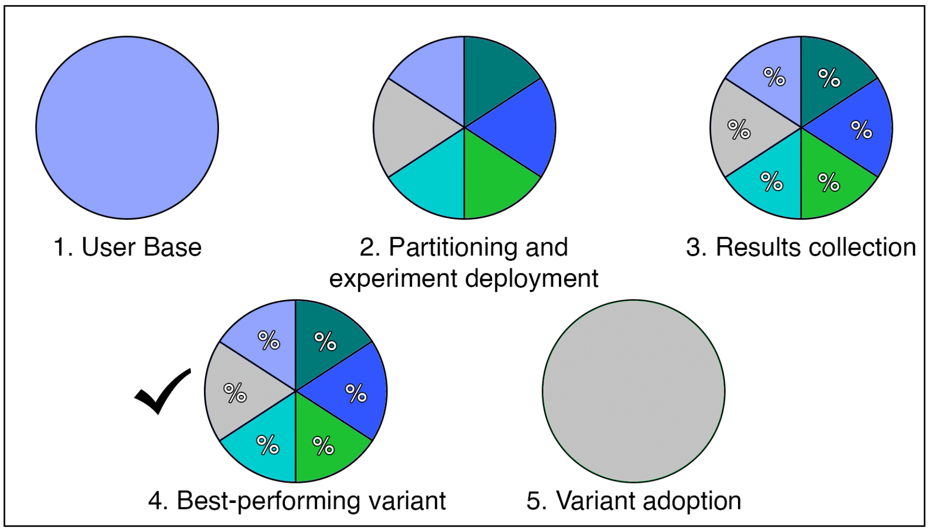

%
1. User Base
2. Partitioning and experiment deployment
3. Results collection
4. Best-performing variant
5. Variant adoption

Figure 2.7: The Continuous Experimentation process. Adapted from [6].

2.4 Related Work

This section discusses related work with the focus of a few main technologies that are closely related to this thesis study. This section will dive into work related to response techniques, followed by work related to vehicle collaboration in cyber security, and finally will discuss other work that focuses on the Continuous Experimentation in the automotive domain.

Response Techniques

For identifying cyber attack response techniques, Rosenstatter et al. [15] present a systematic literature review that proposes a framework "REMIND" to support the design of resilient automotive systems. The study provides a taxonomy of state-of-the-art techniques for cyber attack detection, mitigation, recovery, and endurance. It also discusses the trade-offs when using certain cyber attack response techniques and when applying the guidelines provided by the framework. This framework acts as multi-dimensional decision support that helps designers to make an informed and optimal selection of techniques when it comes to implementing them in the automotive systems.
On the other hand, Ratasich et al. [5] provide an overview of the state-of-the-art mechanisms to resilience for the IoT devices that require monitoring and controlling from a distance. It summarizes the state-of-the-art techniques on cyber attack detection, diagnosis, and recovery/mitigation by mainly focusing on non-intrusive methods, which act in the communication networks and on the face of IoT devices. This study also states the challenges when applying these techniques in the IoT and describes a road map on how to achieve resilience for these devices.

Vehicle Collaboration in Cyber Security

While there is almost no literature regarding collaborative cyber attack response of automotive systems that could be found, there are a few studies that are concerned with vehicle collaboration in other fields of cyber security.
Mousavinejad et al. [28] proposed a distributed attack detection and recovery mechanism to address the problem of detecting cyber attacks that target the vehicle platooning system through the shared communication network and employed on-board sensors. The proposed solution is mainly divided into two parts: *attack detection* and *recovery*. The attack detection focuses on attacks that compromise sensor measurements and/or control command data. The core of the distributed attack detection algorithm is the ellipsoidal filtering that provides state prediction and state estimation sets. The detection of the attack is determined when the two sets intersect each other. The recovery mechanisms provided by this study depend on reliable modifications of signals of the vehicle that are attacked.
Other related work regarding this matter is based on collaborative Intrusion Detection Systems (IDS). With that, Nandy et al. [29] proposed a trust-based collaborative IDS, in which each vehicle keeps a score table of other vehicles in order to identify their previous patterns of the network behavior. The neighboring vehicles

would then share their score tables with each other using the Vehicle ad-hoc network (VANET).
Another work has been conducted regarding collaborative IDS from Raja et al. [30], in which it proposes the use of distributed machine learning (DML) model that is based on the alternating direction method of multipliers. This to leverage the V2V collaboration in the learning processes in order to improve the accuracy, scalability, and storage efficiency. This work also targets privacy risks associated with the DML-based collaborative IDS.

Continuous Experimentation in the automotive domain

While Continuous Experimentation is mainly used for software-intensive web-based applications, it also found its way into the cyber-physical systems, as they grow to become more software-dependant. A study has been conducted by Giaimo et al. [31] in order to introduce continuous experimentation to the field of cyber-physical systems, on the example of the automotive domain. The study demonstrates and evaluates a prototype infrastructure that is implemented on a distributed computational system in a commercial truck that is used daily in public roads. The system contains units and sensors, and the software deployment and data retrieval processes are done remotely via a mobile data connection. The study showed that the development team was able to apply software deployments based on real-world data that is collected during the experiment. Hence, proving the applicability of Continuous Experimentation in the automotive domain.

3 Research Methodology

This chapter describes the research methods used for conducting this study. It begins with describing the design science research methodology that this study follows in 3.1, followed by a qualitative study in 3.2 and simulation in 3.3. Finally, a description of the thematic analysis used to analyze the qualitative data in 3.4.

3.1 Design Science Research

The design science research (DSR) methodology is used to produce interesting and true knowledge as it aims to further understand and improve human-made designs [32]. According to [33], DSR is "A research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence. The designed artifacts are both useful and fundamental in understanding that problem".
The design science research methodology consists of multiple cycles, with each cycle having the same iterations repeated in order to further develop the solution design. That said, the practice of design science research is not similar to all types of research, and the potential improvements are not the same for all practices. Therefore, it addresses general problems through studying specific problem instances in practice, which establishes the research context, and this is where the research activities of problem conceptualization, solution design, and validation take place [32], which can be instantiated in different ways [32]. These activities are necessary to provide a rigorous solution to the problem in question, especially validation, which should be done in a real-world context or an artificial one that resembles aspects of it [32]. And since iterations may include different stages that vary across literature [34], research questions for this study are designed to fit into the stages of DSR, which are answered subsequently through gaining knowledge and progressing into further iterations by performing one cycle. The problem conceptualization iteration would be conducted by performing a literature review that would provide knowledge in the automotive domain, specifically in the cyber security field, and to explore the available state-of-the-art response techniques for the software asset of the vehicle (RQ1), which is further discussed in 4.1. Completing the first iteration would pave the way for the second iteration, in which the knowledge gained is put to use into designing the solution for collaboratively evaluating different response techniques, followed by a simulation experiment to test the implementability of such solution (RQ2), which is discussed in detail in 4.2. Finally, an evaluation of safety of the designed solution is performed through a qualitative study with domain experts

(RQ3).
Figure 3.1 shows the research activities for this study when conducting DSR.

3.1.1 Problem Conceptualization

Problem conceptualization iteration seeks to understand the domain and problem using data collection methods. For this study, a full systematic literature review was not considered. However, the main taxonomies of cyber attack response techniques are elicited from the 'REMIND' research paper [15], as it is the most recent systematic literature review available in the literature regarding this matter. That being said, each cyber attack response technique regarding the software asset mentioned in that paper is further explored and discussed through snowballing.

3.1.2 Solution Design

Solution design iteration is the creative step that aims to create a potential solution for the problem using the knowledge gained from the previous iteration. Here, brainstorming and an investigation of a few technical topics were conducted in order to support the decisions of designing the proposed solution. Relevant keywords included: peer-to-peer architecture, client-server architecture, vehicle collaboration, connected vehicle, cyber response technique, V2V collaboration, automotive cyber response.

3.1.3 Design Evaluation

The main goal of design evaluation iteration is to evaluate the proposed solution. For this study, the proposed solution was continuously evaluated with the academic supervisor and domain experts through regular discussions. That being said, an evaluation of safety was performed through a questionnaire with security and/or safety experts.

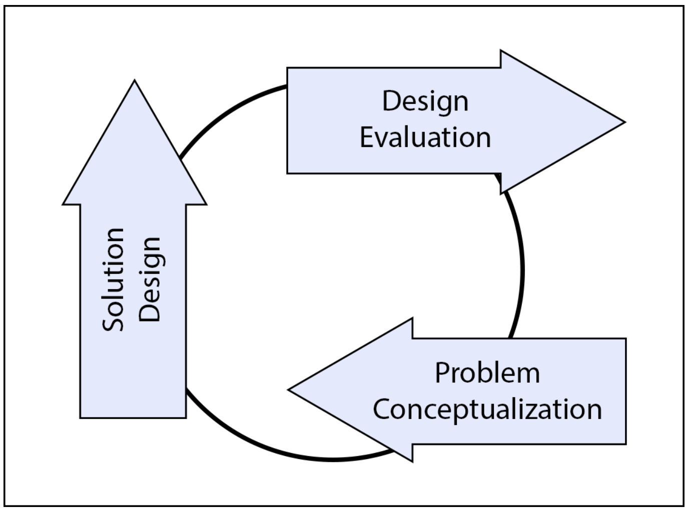


Figure 3.1: Research activities when conducting design science research.

3.2 Qualitative Study

A qualitative study has been conducted through the use of a questionnaire to collect viewpoints of domain experts in a structured way [35]. It included a combination of both closed and open-ended questions in order to gather qualitative data and a feel of intensity of the answers. The questionnaire was performed with security and/or safety experts from different organizations and occupations with the main purpose of evaluating the safety of the proposed solution based on the participants' perception.
In order to answer the questionnaire, emails were sent to all of the participants including the following:

- **Video demonstration:** an 8 - 9 minutes long video demonstration providing background information about the problem and what is the proposed solution for that problem[1]
- **Slides:** the slides used with the video demonstration is attached to the email for reference[2]
- **Questionnaire:** at the end of the email, a link to the questionnaire was placed. The questions are listed in A.1

Additionally, the phone number of the author was left on the slides, email, and the video demonstration to clarify any missing information or vague explanations. Furthermore, all the materials mentioned above were discussed and evaluated with the academic supervisor beforehand in order to determine whether some questions need modifications, missing or not to be included at all.
When constructing the questionnaire, the decision was to go with a few closed questions defined using the Likert scale [36] to measure the intensity of the answers for specific statements and a few open questions as well to further learn about the participants' personal experience. That being said, open questions usually require more effort to be analyzed. However, this was deemed acceptable due to the limited number of participants who participated in the questionnaire.
Furthermore, the response rate for this questionnaire was high because the participants were *captive* [35], as they have been personally requested to answer the questionnaire.

3.2.1 Participants

The participants of this study were selected following convenience sampling, which is a non-probabilistic type of sampling where the target population meets certain practical criteria [37]. The participants were chosen through the following criteria: knowledge, experience in the domain, and their willingness to participate. Table 3.1 lists the details of participants who answered the questionnaire.

[1]https://youtu.be/gGe9GLsfMtw
[2]https://www.slideshare.net/secret/5P1sngWFEiJnN2

Table 3.1: Participants of the questionnaire.

Participant	Highest Educational Degree	Domain of the Highest Educational Degree	Occupation	Organization	Experience in Systems Security and Safety	Rate your Expertise in Systems Security and Safety
Participant A	B.Sc.	Computer Science and Engineering	Principal engineer in cybersecurity	Volvo Group	84 months	Very High
Participant B	M.Sc.	Computer Science and Engineering	PhD student in networks and systems	Chalmers University of Technology	45 months	High
Participant C	P.hD.	Computer Science and Engineering	Researcher - verification and validation of systems with respect to safety and security requirements	RISE Research Institutes of Sweden	100 month: including the period of my Ph.D. education	High
Participant D	P.hD.	Vehicular Communication Security	Technical Specialist Automotive Cybersecurity	Volvo Cars Group	8 years in automotive cybersecurity, none in safety	High

3.3 Simulation

In order to evaluate the proposed solution, [38] suggests that the evaluation of the artefact (proposed solution) can be performed either before (ex-ante) the construction of the artefact, or after (ex-post). The paper also notes that the design science research is an iterative process, so the definition of the ex-post here refers to the artefact and not the final developed system [38].

Other considerations were on which method to use when evaluating the artefact. Veanble et al. [39] propose two methods, naturalistic (such as observation) and artificial (such as simulation and experiment). Naturalistic can be complex due to the nature of reality and the number of variables included. Thus, it can include the risk of misinterpretation. Artificial evaluation, on the other hand, can limit the risk of misinterpretation, but on the cost of applicability in a real scenario.

After designing the artefact, a simulation was conducted to test the final artefact. The choice of an artificial evaluation was made instead of naturalistic due to the limited resources available when conducting the study.

The simulation process started by implementing the framework on three Raspberry Pi 4 Model B[3] devices with identical specifications and a computer[4]. These Raspberry Pi's have the same hardware specifications in order to make sure that the process of evaluating different response techniques is performed in an optimal way. Furthermore, all Raspberry Pi's had Raspberry Pi OS installed and Raspberry Pi Desktop was installed on the computer. Then all the devices were connected to a local network through Ethernet cables into a switch, in which the Raspberry Pi's acted as clients (cars) that connect to the server (computer) to request/provide services.

3.4 Thematic Analysis

Thematic analysis is used in order to analyze qualitative data elicited from the questionnaire. Thematic analysis is a method that is used to identify, analyze, and report patterns (themes) within the data [7]. Many aspects of Thematic analysis are not unique, sharing many commonalities with other analysis methods used for qualitative data [40].

Figure 3.2 shows the process followed when conducting the thematic analysis.

[3]https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf, 4GB version.

[4]Laptop: PS63 MODERN 8SC-036NE

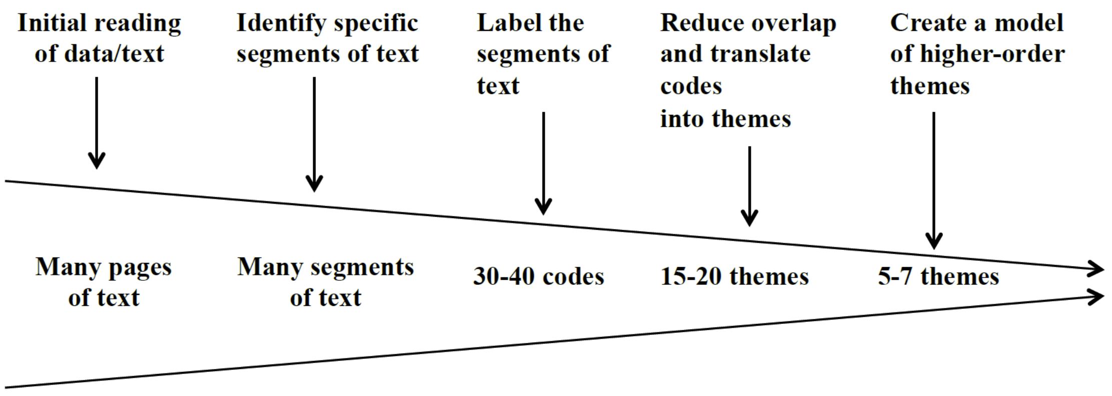

Initial reading of data/text
Identify specific segments of text
Label the segments of text
Reduce overlap and translate codes into themes
Create a model of higher-order themes
Many pages of text
Many segments of text
30-40 codes
15-20 themes
5-7 themes

Figure 3.2: Thematic synthesis process [7][8].

With the help of the open-ended questions generated in the questionnaire, a few higher-order themes were placed before the beginning of the process. Afterward, the participants' responses to the questionnaire have been well-read in order to get familiarized with the data. Then the coding process of the data has been performed by manually assigning sticky notes to the data and then assigning them to themes as seen fit. Finally, these themes were linked to a higher-order theme that is either already existed or a newly formed one without forcing any of them to fit in a non-relevant higher-order theme.

4 Results and Discussion

This chapter discusses the results and discussion of the thesis study. It starts with problem conceptualization in 4.1, followed by the solution design in 4.2, and finally the design evaluation in 4.3.

4.1 Problem Conceptualization

The automotive domain is rapidly growing, and the main force that is driving this growth is the increased use of electronics and software inside the vehicle [24]. This dependability on software increases the number of services provided to the customer but increases the complexity of the system at the same time. Such complexity could open up ways for attackers to conduct malicious activities against the customer and manufacturer of the vehicle alike.
There are many assets that could be targeted in a vehicle, such as hardware, software, network/communication, and data storage [15]. Therefore, different response techniques can be applied to eliminate these attacks or try to mitigate their effects. However, many of these techniques are adopted from different fields of IT, such as cloud computing and real-time systems. For that, evaluating these techniques in terms of effectiveness and efficiency is necessary.
As mentioned, many assets could be the target of an attack. However, this thesis will focus on the software asset alone. To further discuss that, this section will examine the state-of-the-art response techniques for threats against the software asset of the connected vehicle.

Software Asset

Software in automotive systems can exist in several categories: *in-transit*, *at-rest*, or *running*. In-transit may refer to software delivery systems, such as OTA or shop updates. The categories "running" and "at-rest" can refer to software installation processes or software running in ECUs [15].

In order to understand the different state-of-the-art cyber attack response techniques that are available at this time, [15] is a recent study that proposes a framework for cyber resilience in the automotive domain by providing strategies for attack detection, mitigation, recovery, and endurance. It describes different techniques regarding each strategy as well as the asset that can make use of these techniques. Each strategy then is broken down into different patterns and each pattern into different techniques. Some techniques and patterns can overlap with different strategies.

This thesis mainly aims at cyber attack response techniques. Thus, it will mainly make use of two strategies that are proposed in [15]; the *Mitigation* and *Recovery strategies*. The detection strategy will be partly considered as well; as it shares the redundancy pattern with the mitigation strategy.

The mitigation strategy aims at triggering mitigation techniques when an anomaly is detected and located. These techniques will keep the system operational, but could also result in a non-optimal system state. There are different patterns for the mitigation strategy: *Redundancy*, *Diversity*,*Adaptive Response*, *Runtime Enforcement*, and *Reconfiguration/Re-parameterization*. Moreover, the last pattern can overlap with the recovery strategy.

On the other hand, the recovery strategy's main goal is to use recovery techniques in order to transition back to the optimal state. It is also broken down into different patterns: *Reconfiguration/Re-parameterization* (can overlap with the mitigation strategy), *Migration*, *Checkpointing and Rollback*, and *Rollforward Actions.*

Some of these strategies could include techniques that are not concerned with the software asset. Therefore, they will be ignored for this thesis. The techniques that will be discussed are shown in figure 4.1.

4.1.1 Redundancy and Diversity

Redundancy's approach is mainly the use of information from multiple sources in order to detect faults and verify the output through plausibility check or majority voting [5], [41]. Redundancy is typically the last resort in providing dependability, because it is considered costly when added explicitly [5].

Purely redundant systems can also suffer from the same design vulnerabilities and faults. Therefore, diversity is combined with redundancy to overcome this issue [15].

There are several techniques that fall under these two patterns [15]: *Software Redundancy*, *Agreement/Voting*, *N-version Design*, *Recovery Blocks*, *N self-checking*, *N-variant Systems*, and *Replacement of Cold/Hot Spares.*

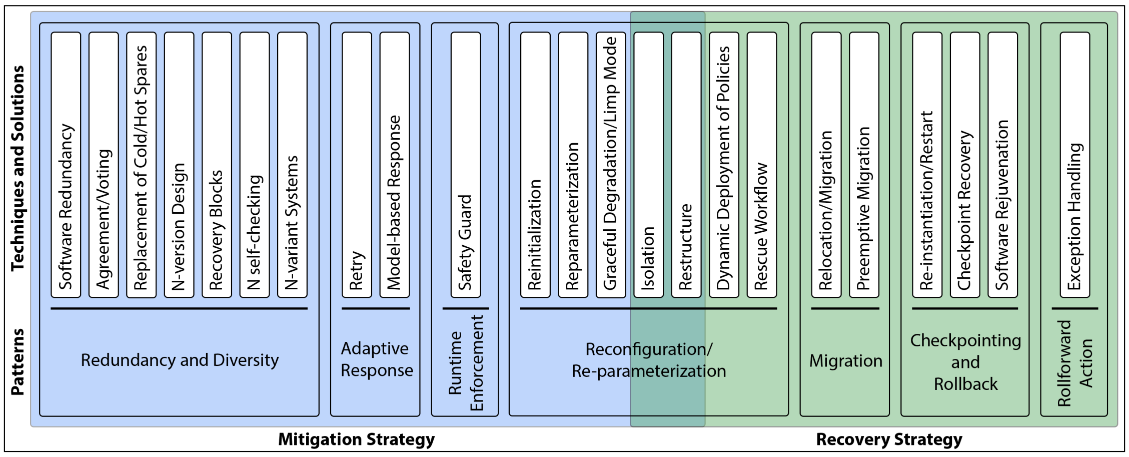


Figure 4.1: Response techniques for the software asset of a vehicle.

4.1.1.1 Software Redundancy

The approach of redundancy is the use of information from multiple sources to mask system failures [15]. This is performed by the voter through comparing different results of several independent executed software modules and, for example, selecting the majority vote [15], [42].

Software Redundancy can help contain and exclude malicious behavior, i.e. reduce the likelihood of harm. It can also overcome disruption by enabling restoration and enhances the availability in critical operations [15]. However, the main drawback of it being resource-demanding; as it mainly depends on multiple sources for information. Additionally, it suffers from the possibility of degrading; as configurations get updated or connectivity changes. Furthermore, it is often applied with other diversity techniques, which increases complexity and affects scalability [15]. Figure 4.2 shows a simple representation of software redundancy.

4.1.1.2 Agreement/Voting

The agreement/voting technique concerns with byzantine failures (inconsistent failures to different observers) that are typically caused by malicious attacks [5]. This can be detected and tolerated through using replicas or redundant sources by reaching an agreement on the outputs [5], [43]. The output of all the redundant components can then be combined or fused [5], or can be used for finding the average or median value of the results from these redundant sources [15], as demonstrated in figure 4.3.

That being said, the attention to ensure authentication and integrity of data is now shifting toward the use of blockchains [5], [44], [45], [46], [41]. However, blockchains suffer from complexity, latency, and energy consumption. Therefore, it is limited in functionality and cannot be applied on systems with low computational power [5], [41]. Additionally, attackers can exploit the voting process and force the system into a degraded mode [15], which can cause loss of performance.

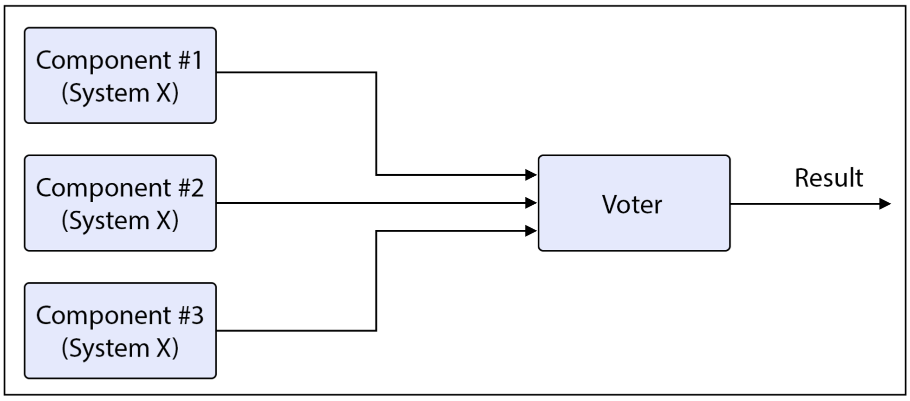


Figure 4.2: Software Redundancy.

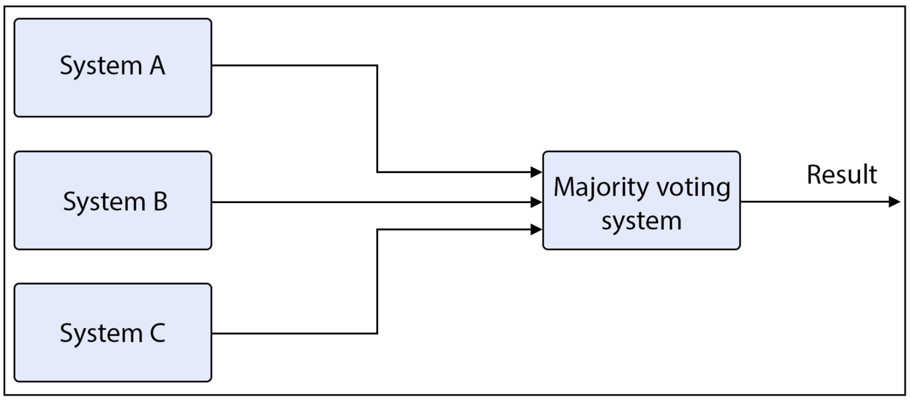


Figure 4.3: Agreement and Voting.

4.1.1.3 N-version Design

With this approach, several versions of software that satisfy the same requirements would be developed by different independent teams. This results in a diverse set of software components that have equal functionalities and fulfill the same specifications. These versions are executed concurrently, and a voter then decides the results based on the majority or some type of calculation, for instance, the average value [15], [47], as shown in figure 4.4.

While this design can help mitigate the impact of failures when the system configuration or design is introduced to risk, it requires much more effort and cost when it comes to design, implementation, test, and validation of the number of independent versions [15]. Additionally, some researchers argued that using different programming teams can make similar mistakes [48], but that is unlikely when the software gets more complex.

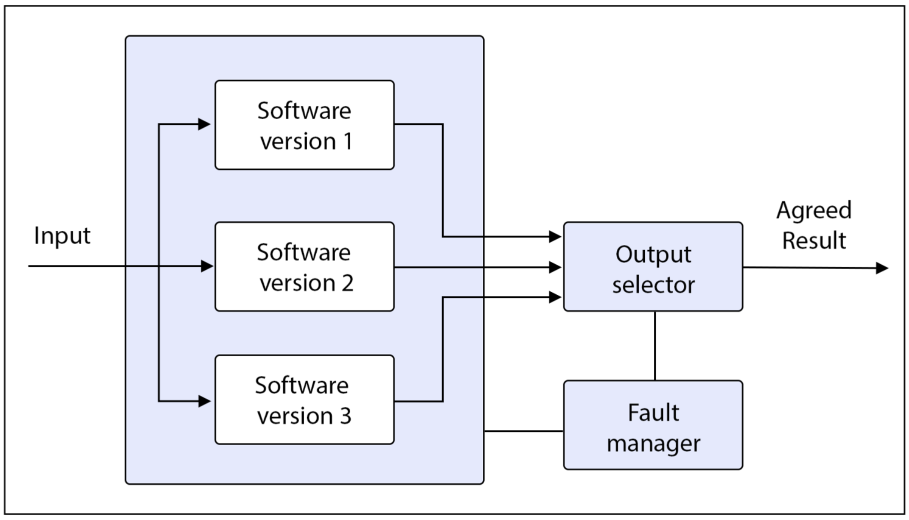


Figure 4.4: N-version Design.

4.1.1.4 Recovery Blocks

This approach is similar to the N-Version Design that is discussed in 4.1.1.3, where there are several versions of a software component that exist. However, only one of these versions is executed at a time. After the execution of that version, a common acceptance test will be performed and decides whether the results are accepted or not. In case of rejection, the subsequent version is then executed and evaluated as performed for the first version [15], [47], [49], as shown in figure 4.5

While the recovery blocks approach does not require all the versions of the software component to be executing at the same time, these versions still require extra validation and verification efforts. Therefore, it would still consume more resources [15].

4.1.1.5 N self-checking

This technique is a combination of the previously discussed techniques (N-Version Design in 4.1.1.3 and Recovery Blocks in 4.1.1.4). This technique requires at least two different versions of the software component with their own acceptance test. When the active version fails its acceptance test, the subsequent version would take over [15], [49].

The purpose of the N self-checking technique is to provide mitigation by creating several versions of the same software component, but each with its own acceptance test. All versions that pass their acceptance test would then be selected through an acceptance voting system [15], [9], or it can compare the outputs in order to detect errors [9], as shown in figure 4.6 and figure 4.7, respectively .

However, this technique causes an increase in required resources and time for execution [15].

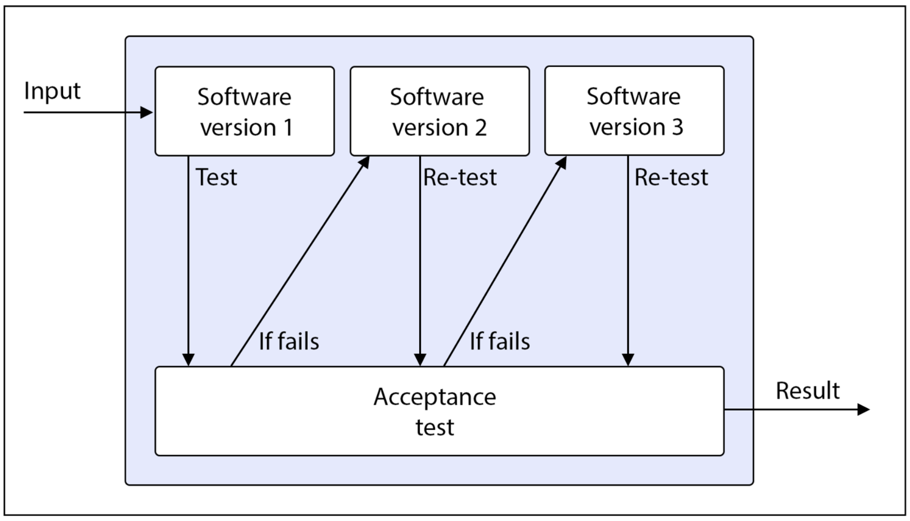


Figure 4.5: Recovery Blocks.

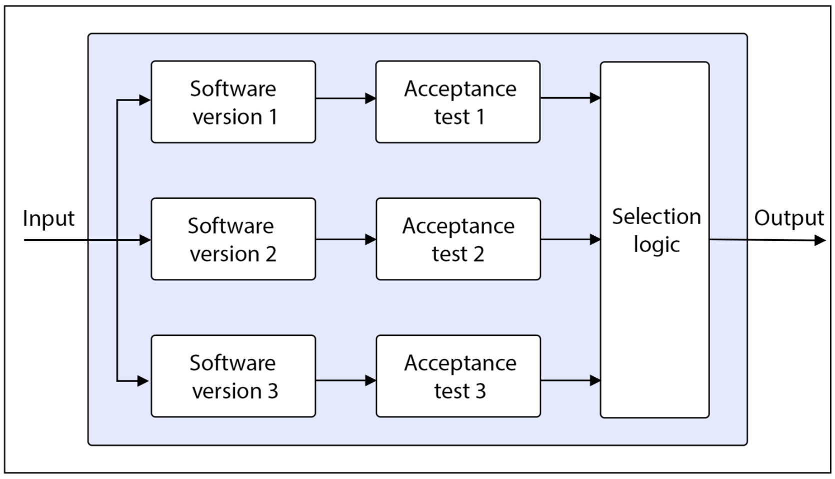


Figure 4.6: N self-checking using Acceptance Tests. Adapted from [9].

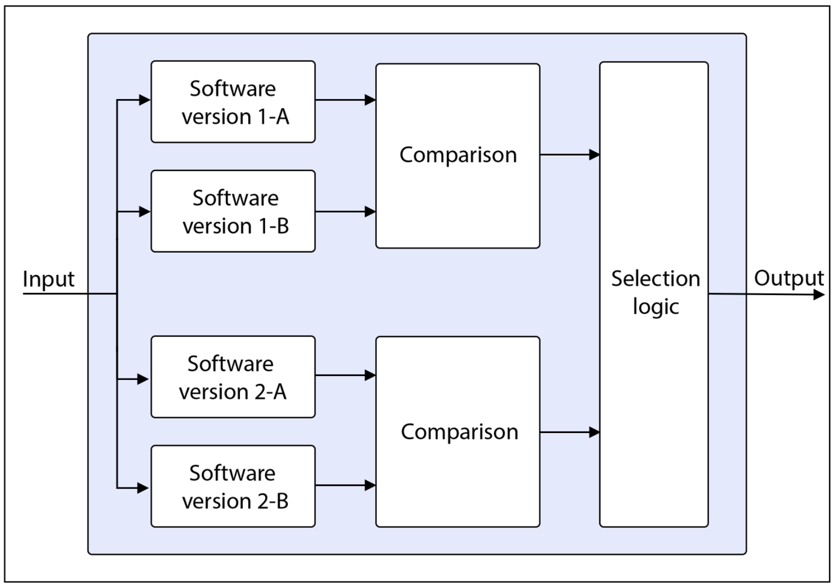


Figure 4.7: N self-checking using Comparison. Adapted from [9].

4.1.1.6 N-variant Systems

A multi-variant execution will automatically diversify the software and monitors the output of at least two variants in order to detect and mitigate attacks [15], [10]. Here, the client's input is put into whats called a *Polygrapher*, and then gets copied to all the variants inside the system [10]. These different variants are artificially diversified in a way that makes them behave in a different way when abnormal inputs are received [10]. The *Monitor* observes these behaviors in order to detect divergences, which could reveal attacks. When the monitor detects a divergence, it would restart the variants to known uncompromised states [10]. This is illustrated in figure 4.8.
While this approach can help mitigate attacks and detect them, it would require resources for all the different running variants as well as the monitor.

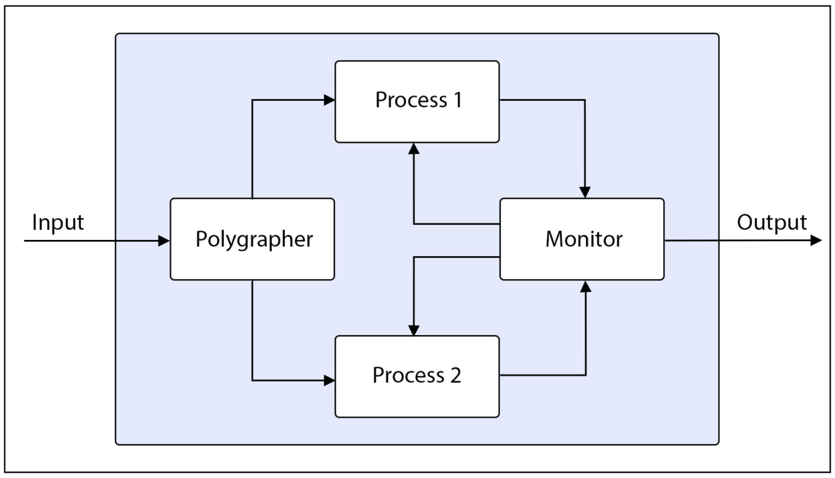


Figure 4.8: N-variant Systems. Adapted from [10].

4.1.1.7 Replacement of Cold/Hot Spares

Hot and cold spares strategy is used to recover from failed components [50] by having a redundant component taking over in case the main component fails.
The term *hot spare* means that the redundant component is running in the system at all times, and is automatically ready to take over if the main component fails [50]. On the other hand, a *cold spare* is also a redundant component that can be used to automatically replace the main component when a failure has occurred, but it might require some configuration before doing so, e.g. a reboot [50], which might cause loss of availability.
In the automotive domain, the concurrent or sequential execution of redundant software components is costly in both energy consumption and computational resources. Therefore, the introduction of cold/hot spares, such as N self-checking in 4.1.1.5, is found to be a viable alternative [15], [5].

4.1.2 Adaptive Response

The focus here is on techniques that adapt the response of a sub-system or a function to maintain its intended functionality [15]. The adaptive response pattern has two techniques: *Retry* and *Model-based Response.*

4.1.2.1 Retry

The Retry technique is used to deal with transient type of faults [51] that results in an error. This is done by performing the same computation that has already been done after a certain period of time [51]. The waiting period here is to give the transient fault a chance to disappear in order for the system to perform the same computation again and recover from that error state, as shown in figure 4.9.
Furthermore, it is important to make the waiting period long enough for the transient fault to disappear and, at the same time, short enough to avoid faults from overlapping [51], [52]. Additionally, this technique can mitigate the risk of replay attacks [15].

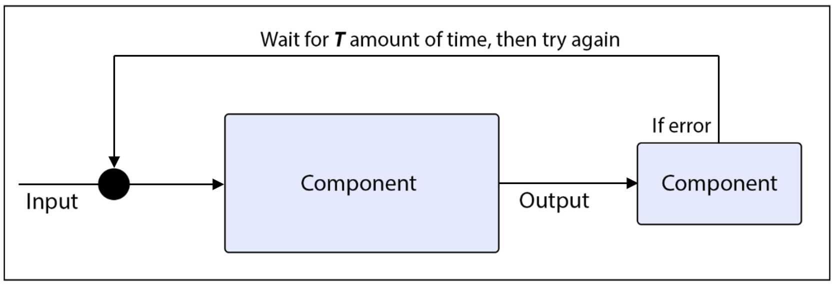


Figure 4.9: Retry technique.

4.1.2.2 Model-based Response

This technique uses a model to generate an approximation of the normal behavior of the system without introducing it to attacks [11]. It is mainly based on parameter estimation techniques, such as regression analysis [15]. With this technique, a comparison would be performed between the measurements and the estimations provided by the system model, and based on that comparison, a decision of an existing anomaly is taken [11].

The Model-based Response technique can be used not only as a temporary solution to mitigate attacks, such as masquerading and replay attacks, but also to alert the system and to log information that can be later used for forensics [15], [53].

Figure 4.10 shows the general scheme of the model-based response technique, where Y_k represents the true output value of the system, Y_k^a represents the value of results after a sensor attack, $\hat{Y}_k$ represents an estimation of Y_k that is based on a model of the plant (a model that describes the dynamic behavior of a physical process [11]), and the Anomaly Detection Module (ADM). The ADM takes two inputs ($\hat{Y}_k$ and Y_k^a), comparing the estimation provided by the system model and the measured signal to take a decision. Then the ADM sends a safe signal (Y_k^s) to the controller to perform the adequate control action (U_k) [11].

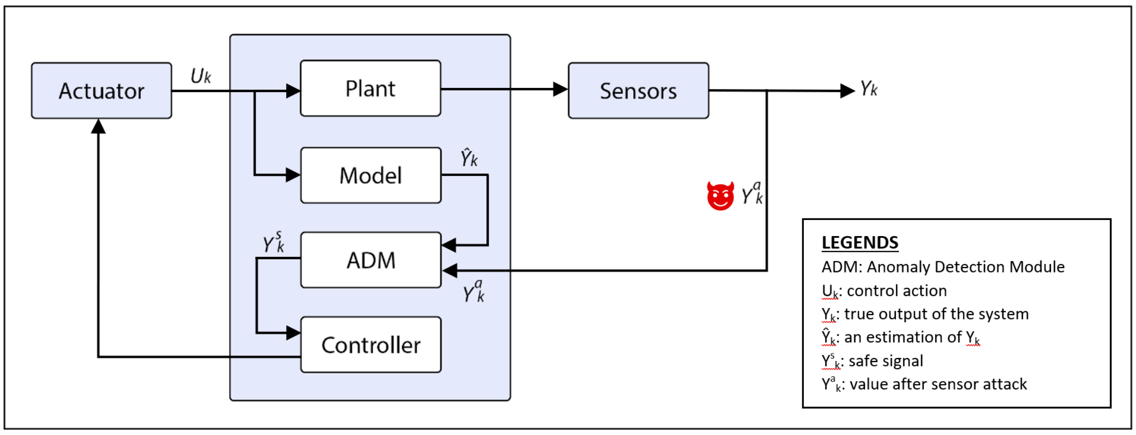

Actuator
U_k
Plant
Sensors
Y_k
Model
$\hat{Y}_k$
Y_k^a
Y_k^s
ADM
Y_k^a
Controller
LEGENDS
ADM: Anomaly Detection Module
U_k: control action
Y_k: true output of the system
$\hat{Y}_k$: an estimation of Y_k
Y^s_k: safe signal
Y^a_k: value after sensor attack

Figure 4.10: Model-based Response general scheme. Adapted from [11].

4.1.3 Runtime Enforcement

With the pattern, the system is observed by a monitor at runtime to verify the correctness of the execution and react to violations [15], [54]. An example on such techniques is *Safety Guard* [15].

4.1.3.1 Safety Guard

The Safety guard is a reactive component that can be attached to the system in order to protect it against catastrophic failures [12]. This is done by ensuring the satisfaction of a set of predefined safety-critical properties that are defined by the system even in the presence of an attack or unknown failures [12]. By treating the system as a black box, the guard monitors the input and output of that black box and corrects any property violations instantaneously, and minimizes the deviation from the original system [12]. Figure 4.11 illustrates Safety Guard.
The main advantage of a safety guard is that it can be suited for a wide variety of cyber-physical applications; because it does not require knowledge of the implementation details of the system [12].

4.1.4 Reconfiguration/Re-parameterization

The system adapts parameters to protect itself when an attack is detected. Reconfiguration and Migration patterns can be similar. However, the main distinction between them is that migration focuses on the relocation of functionality, whereas reconfiguration focuses on changing the parameters of the system [15].
This pattern is broken down into the following techniques: *Reinitialization*, *Reparameterization*, *Graceful Degradation/Limp Mode*, *Isolation*, *Restructure*, *Dynamic Deployment of Policies*, and *Rescue Workflow.*

4.1.4.1 Reinitialization

This approach aims to address transient faults and temporary attacks [15] by restoring the system to its initial state, which implicitly cleans up the system from effects caused by a fault, error or a failure [55], as illustrated in figure 4.12. However, this technique does not mitigate permanent faults and reoccurring attacks by restoring the system to its initial state [15].

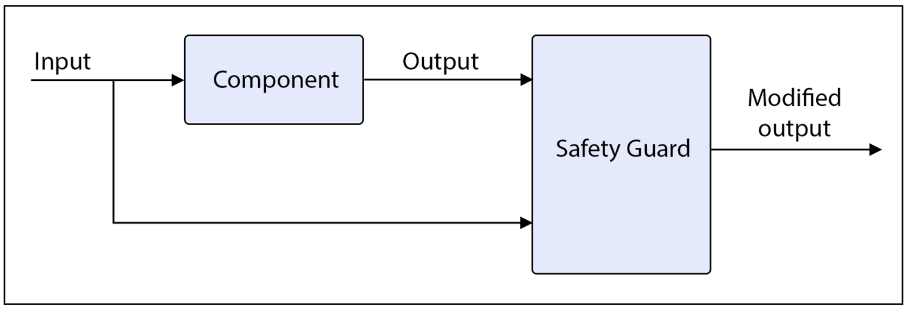


Figure 4.11: Safety Guard. Adapted from [12].

This technique is applied in conditions in which the mitigation or recovery operations deems to be impossible, or excessively expensive in terms of performance [55].
Though this technique can cause loss of work and disturbance to other components that rely on the affected component, it guarantees that the impact of the error is completely removed before the resumption of service [55].
Furthermore, this technique can be seen as Checkpoint Recovery (discussed in 4.1.6.2); the checkpoint being the initial state of the system or function [15].

4.1.4.2 Reparameterization

This technique is similar to Reinitialization, that was discussed previously in 4.1.4.1. However, the main distinction between them is that reparameterization dynamically adjusts the system configuration based on the situation [15]. This adaptation of parameters requires knowledge about the algorithm of the erroneous component [5]. Therefore, this technique is typically applied to the component itself or within a subsystem [5]. Additionally, this technique can result in a non-optimal state and leads to Graceful Degradation [5].
Figure 4.13 illustrates the basic functionality of the reparameterization technique.

4.1.4.3 Graceful Degradation/Limp Mode

Graceful degradation enables the ability of a system to continue functioning even when parts of the systems are damaged or compromised [56]. When this technique is applied, the system usually performs slower than the regular performance it is intended to function at and can decrease further as the number of failing components grows [56] or by shutting down non-critical functions in order to save resources for

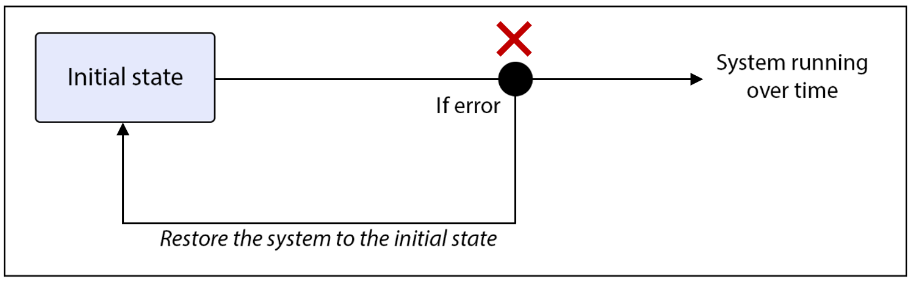


Figure 4.12: Reinitialization technique.

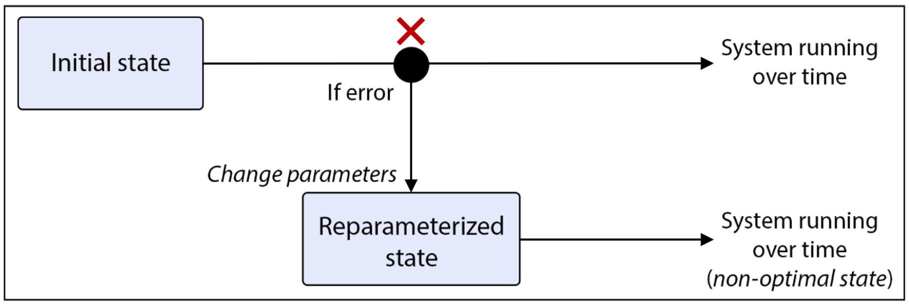


Figure 4.13: Reparameterization technique.

more critical ones [15], as shown in figure 4.14.
The main use of this technique is to prevent catastrophic failures of the system [56], especially when considering the safety of the driver and passengers when key components of the vehicle fail [15]. Modern vehicles have already introduced such functionality, called "Limp mode", which is triggered when major technical problems are detected by the vehicle [57].

4.1.4.4 Isolation

This technique is used to restrict access or completely isolate erroneous system components in order to limit the impact on the performance of the entire system [15].
While this approach enables the system to continue functioning by offsetting the effect of an attack and prevents the loss of functionality, it introduces extra resources by using replica modules in order to compensate for the affected component within the system [15], as shown in figure 4.15.

4.1.4.5 Restructure

This technique aims to address faults, errors, or failures that could affect the correctness of the interactions between subsystems [55]. The Restructure technique modifies the configuration between the interconnected subsystems in order to isolate the erroneous part [55]. Figure 4.16 illustrates the basic functionality of the

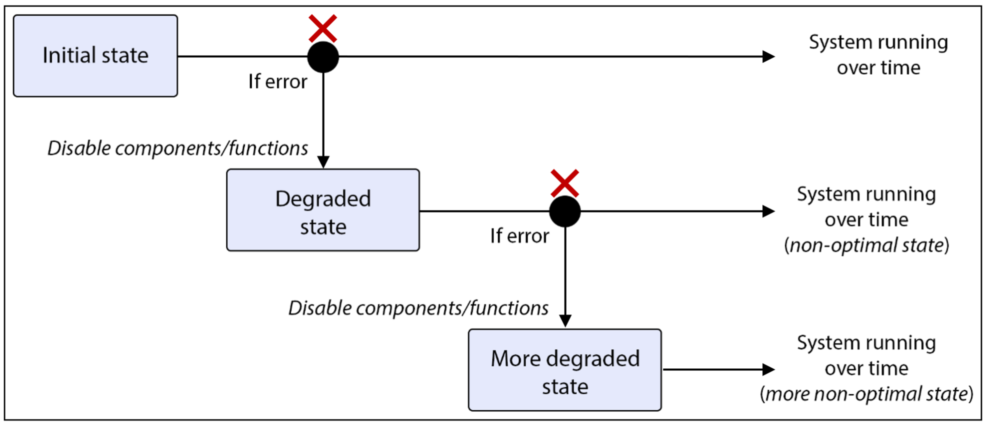

Initial state
If error
System running over time
Disable components/functions
Degraded state
If error
System running over time (non-optimal state)
Disable components/functions
More degraded state
System running over time (more non-optimal state)

Figure 4.14: Graceful Degradation/Limp Mode technique.

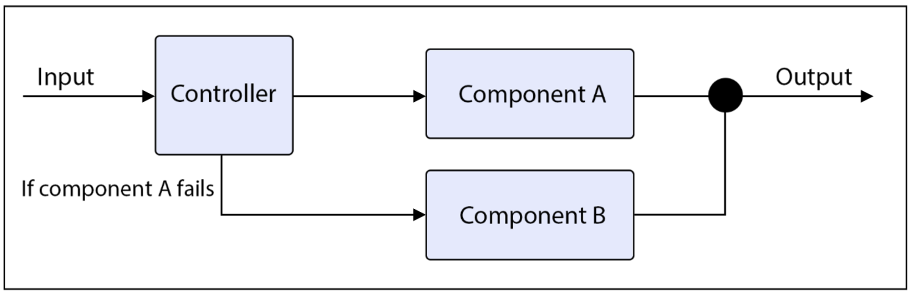


Figure 4.15: Isolation technique.

Restructure technique.
The reconfiguration technique changes the organization of the system by working around the erroneous subsystem or by excluding it from interacting with other subsystems [55]. This is done in order to maintain (sub-)system functionality that is equivalent to the original [55]. However, this technique may also result in a degraded condition and incurs additional time overhead to the system [15], [55].

4.1.4.6 Dynamic Deployment of Policies

This approach addresses the security or other policies to be applied dynamically based on the type of attack that is introduced, such as Denial of Service or masquerading attacks that are detected [15]. This technique takes dynamicity and the changing nature of attacks into account by deploying different defense policies based on the attack [15]. For example, this technique can modify the executed actions under an ongoing attack [15].
However, this approach leads to performance overhead and requires runtime permissions, which may not be present when running normally, leading to an increase in complexity [15].

4.1.4.7 Rescue Workflow

The Workflow technique uses a workflow to describe tasks with their dependencies to each other [15]. This allows the system to continue functioning even when some tasks fail because of an error or an intrusion until it becomes impossible to continue without this failed task [58]. This is done by dynamically adjusting the structure of the workflow [15].
That being said, tasks that are already executed do not require to be re-executed; which saves time and resources [15]. However, this technique can also lead to a decrease in the quality of service [15]. Additionally, it can cause a time penalty because of the re-computing and migration of the tasks that caused the problem [15].

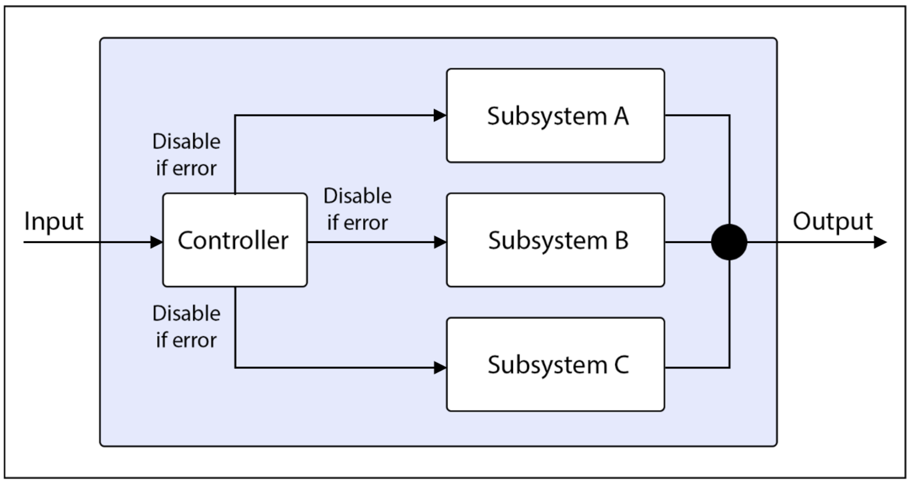


Figure 4.16: Restructure technique.

4.1.5 Migration

These techniques are similar to the Reconfiguration techniques (as discussed earlier in 4.1.4), but focuses on the relocation of the functionality instead of changing the parameters.
Migration techniques mainly originate from high-performance computing and cloud systems. And because the future of the automotive systems is moving toward a more centralized architecture, virtualization and service-oriented architectures can be more relevant in the coming years [15].
The Migration pattern is broken down into two techniques; *Relocation/Migration* and *Preemptive Migration*.

4.1.5.1 Relocation/Migration

This approach aims to prevent failing system components from affecting the performance of the system in a virtualized environments, such as hypervisor and container-based solutions [15], [58]. It provides mechanisms for migrating the execution of the programs from one component to another [58]. This is done by moving components away from failing nodes to more stable nodes in the vehicular network [15].
This technique helps in maintaining system functionality in an operational state as it was originally designed for before the fault or attack [15]. However, it might still cause the system to function in a degraded condition and decrease the performance of the system [15].

4.1.5.2 Preemptive Migration

This approach functions like the previously explained Relocation/Migration technique in 4.1.5.1. However, this technique continuously monitors and analyzes the system for failing components and relocates software and services to other more stable nodes in real-time [15], [58].

4.1.6 Checkpointing and Rollback

A checkpoint describes the system state at a specific point in time. And by design, recovery does not prevent the same attack from happening again [15].
This pattern is further subdivided into three techniques: *Re-instantiation/Restart*, *Checkpoint Recovery*, and *Software Rejuvenation*.

4.1.6.1 Re-instantiation/Restart

This technique aims to re-instantiate or restart the component that was affected by an intrusion in order to recover into a known, erroneous free state [15].
This technique helps in restoring the system to its initial state when the error or attack could not be dealt with in another manner [15]. It also guarantees the complete removal of that error effect or to completely remove the attack [15]. However, it may also cause loss of private data (such as location, speed, and driving behavior), and workshop data (such as vehicle health, engine data, and emissions) [15]. The impact of the lost data varies based on the type of data and the current need for it

[15]. Additionally, the re-instantiation of safety-critical functions could require the vehicle to be in standstill [15].
Since the re-instantiating technique does not prevent the same attack from happening, it can still be combined with Reparameterization (discussed in 4.1.4.2) to avoid the anomaly from happening again [15].

4.1.6.2 Checkpoint Recovery

This technique works by continuously saving the system state [58] by taking snapshots, which are either based on checkpoints or logs [15]. In the event of failure, the system then returns to the most recent state by using the checkpoint [58]. Figure 4.18 shows the basic implementation of this technique.
This technique helps to overcome intermittent faults that can be avoided when the system goes forward from the checkpointed state [59]. Furthermore, it is especially useful when the operation of starting from scratch wastes a lot of time and resources [59], such as when using the Re-instantiation/Restart technique (discussed in 4.1.6.1).
When dealing with a single-node system, checkpointing is relatively straightforward. However, when dealing with distributed systems, it is harder to keep the checkpoints consistent [59] due to the variation of the logical clock, parallel computations, and possibly different system status [15].
This technique can help the system in resuming its operation in a clean state; free from the effects of an attack or fault [15]. Additionally, frequent checkpointing can reduce the amount of lost work [15].
However, creating each checkpoint may require the interruption of the normal operations in order to save the current state of the system [15]. Furthermore, the creation of a checkpoint requires storage resources and can still contain an error or intrusion that has not been detected yet [15].

4.1.6.3 Software Rejuvenation

This technique carries out periodic graceful termination of the system and starting it again by using, for example, Re-instantiation/Restart or Reinitialization techniques (discussed in 4.1.6.1 and 4.1.4.1, respectively), in order to maintain a known, error-free state of the system [15], [58]. Figure 4.19 shows an illustration of the basic functionality of this technique.
This technique helps in avoiding the costs of failures that happen from software

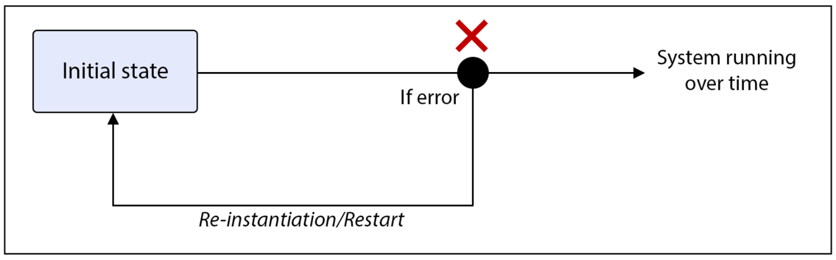


Figure 4.17: Re-instantiation/Restart technique.

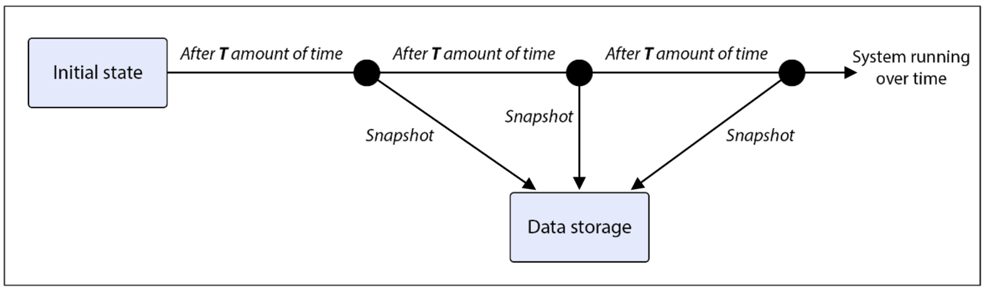


Figure 4.18: Checkpoint Recovery technique.

degradation by allowing the release and re-allocation of memory used by that software component and, thus, allowing it to operate in a clean state [15]. However, the main issue of this technique is the periodic shutting down of the system and starting it again, causing unavailability for the software during that duration [15]. Additionally, it is often a slow process that requires an extra overhead [15].

4.1.7 Rollforward Actions

This pattern aims to bring the system into a stable state immediately before an error or attack is detected. As in rollback, the system recovers by using checkpoint-based or log-based recovery [15], [55]. A technique for this pattern is *Exception Handling.*

4.1.7.1 Exception Handling

It is a structuring mechanism that separates normal from exceptional behavior [59]. While in general it can be used to activate the restoration of a checkpoint, it is more often used in order to roll forward [59].
When an exception is raised by a component after detecting an error, the exception handler gets activated. From there, it can either make another attempt at the task that was being performed or can forward the exception to a higher-level exception handler. This can be useful when the error needs to be dealt with in a broader scope [59].
The main drawback of roll forward exception handling is that it requires the designers of the system to have already anticipated such an error, and that is by realizing how to detect the error and how to recover from it [59]. On the other hand, this means that recovery can be tailored to some very specific error situations [59].

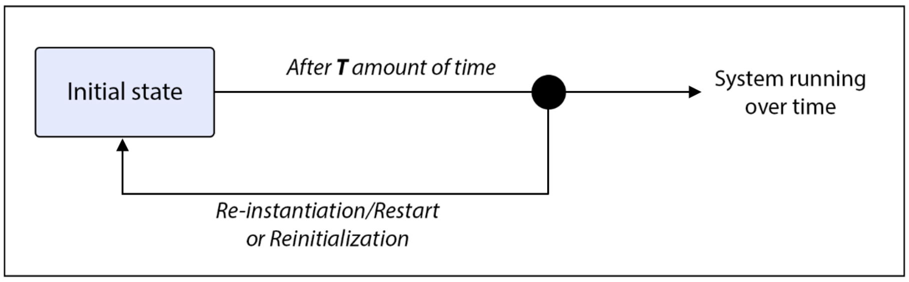


Figure 4.19: Software Rejuvenation technique.

4.2 Riposte: Solution Design

Having that many response techniques available could make the decision of choosing which technique to apply challenging. The solution proposed for this problem is Riposte[1], a collaborative framework between connected vehicles that helps in making decisions of what response technique to use against an attack. These decisions are based on metrics gathered from experiments that are conducted in real-time when an ongoing attack is happening.
As part of the design process, there were a few critical aspects that needed to be researched as they can directly affect the design of the solution. These aspects are the *network architecture of the framework* and the *Continuous Experimentation implementation*, in which both will be discussed in 4.2.1 and 4.2.2, respectively. Afterwards, the structure is discussed in 4.2.3, followed by a discussion of its behavior in 4.2.4. Finally, 4.2.5 discusses the implementation and testing of the framework.

4.2.1 Network Architecture of the Framework

Generally, there are two widely used variants of network architectures: **Peer-to-Peer (P2P)** architecture and **Client-Server** architecture [60].
In P2P, the nodes (peers) organize themselves in an overlay network, enabling them to connect and share information and resources using standard Internet protocols [61], with each peer having the same privileges and potential as others [62] and does not require central coordination by a server [63].
That said, the client-server architecture is based on the concept of services provided to the clients through servers [62]. The server is a host that provides services to the clients by sharing its resources and management. Additionally, the tasks and workloads requested by the client can be partitioned between multiple servers [62].
There are several advantages and disadvantages when it comes to choosing an architecture for the framework. Table 4.1 lists these differences.

Table 4.1: Key differences between Client-Server and Peer-to-Peer architectures.

Client-Server	**Peer-to-Peer (P2P)**
Dedicated server and specific clients	Each peer can act both server and client
Server provides services	Peers can provide/request services
Data stored in a centralized server	Each peer has its own data
Single server; more chance to bottleneck	Servers are distributed
More expensive to implement	Cheaper to implement
More stable	Less stable
More secured	Less secured

[1]Riposte is the act of making a quick, clever reply to an insult or criticism.

For the development of this framework, the client-server architecture is more suitable for different reasons. Since the focus of this thesis study is mainly concerned with the security aspect of the automotive systems, the client-server architecture is more secured than P2P, where security is mainly managed by a central authority instead of each peer individually. Some malicious vehicles could join in the P2P network and provide false data, hence disrupting the process of the framework. Additionally, the client-server architecture is more stable when clients join/leave without causing disruption to the network. On the downside, it is more expensive to implement a server due to the cost of hardware and management. However, in the case of the automotive systems, usually the manufacturer has a dedicated server and management that delivers OTA (Over-the-Air) updates to the vehicles without the need for the vehicle to be at a workshop, for example. This framework can be implemented on such a server that already has an established link between the car and the manufacturer and use it to evaluate cyber attack response techniques.

4.2.2 Continuous Experimentation

It is necessary to consider the experimentation settings when it comes to designing a framework that uses experimentation at its core. Some techniques may require different coding implementations, and adapting to such techniques at a later time could be costly.
There are two main variations when it comes to implementing experimentation, according to a survey study conducted by [64]: *code-level techniques*, where multiple versions of the same code exist within the same code base (known as *feature toggles*), and *deployment-based techniques*, where multiple instances of a service with different software versions are running in parallel (known as *traffic routing*). According to [64], there is no "one size fits all" solution. Instead, there are different pros and cons when it comes to implementing each of the techniques, and many companies combine both of the techniques to counterbalance the disadvantages of particular techniques in order to meet their requirements [64].
Feature toggles are a code-level experimentation technique [64], [65], allowing the modification of a system behavior without changing code [65]. In the simplest form, they can be conditional statements that decide which code to execute next [64]. However, the simplicity of it comes with a cost, mainly technical debt caused by dead code and additional maintenance for the developers [66]. Another challenge with feature toggles is to synchronize the experimentation state and setup and to make sure that multiple instances can be toggled to new versions simultaneously [64].
On the other hand, traffic routing is about the deployment of multiple versions of an application that run in parallel (e.g. in containers or multiple cloud instances) [64]. Depending on the filter criteria that are applied to user requests, dynamically configured components (e.g. network-level proxies) decide which version of the software should be forwarded [64]. The main advantage of this technique is that it is non-intrusive on the code level, avoiding technical debt [64]. However, deploying multiple instances can be costly (e.g. CPU and bandwidth usage) [64]. Moreover, the intermediate components that decide the routing path (e.g. proxies) introduce

overhead that needs to be taken into account [64].

When it comes to this thesis study, feature toggles seem more suitable for multiple reasons. By having all cyber attack response techniques implemented in the codebase cannot be considered as dead code by any means. Instead, they can be considered as tools that are ready to be activated when a certain threat occurs. Additionally, each threat can be best dealt with using a different cyber attack response technique. Thus, having them available at all times is necessary. For the reasons mentioned, the downside of technical debt can be avoided.

Furthermore, time is important when it comes to a vehicle's security, especially when a vehicle is under attack. The best cyber attack response technique for a specific cyber attack can be instantly updated in a system (e.g. via a command), while the traffic routing technique could require the system to be updated, which may take several minutes of time downloading and installing the concrete version and may also require a reboot afterward that may not be safe for some applications.

4.2.3 Framework Structure

One way used to describe the structural aspect of the framework is through a Component Diagram. This diagram shows the main components of the framework and describes their relationships, as seen in figure 4.20. The *Response Component* is the main component that handles the response system inside the vehicle's software. This component handles the application of the response technique after an attack is detected. The Response Component has a *Communication Module* that helps it connect to the Collaboration Management Component in order to ask for services and receive the most efficient response technique for a specific attack. The *Response Assessment Component* handles the assessment of different response techniques by applying them and logging information, such as CPU utilization and/or memory usage. These logs will then be sent to the Collaboration Management Component in order to be saved in the *Data Storage* and evaluated by the *Evaluation Component* by comparing them in order to find the most efficient response technique. Finally, the *Collaboration Management Component* manages all coordination and collaboration between these components.
Another way to discuss the structural aspect is through a Deployment Diagram. The diagram in figure 4.21 shows the main components of the framework when installed on physical devices. The *onRoadCar* device is the customer car that would ask for response technique evaluation from the server when it is under attack through the TCP/IP protocol. The *server* device then receives this request and looks for workshopCars to perform experiments on and evaluate response techniques. The *workshopCar* device is a testbed that has identical parameters to the onRoadCar and helps by running experiments to assess different response techniques. The workshopCar will generate logs and communicate with the server through the TCP/IP protocol.

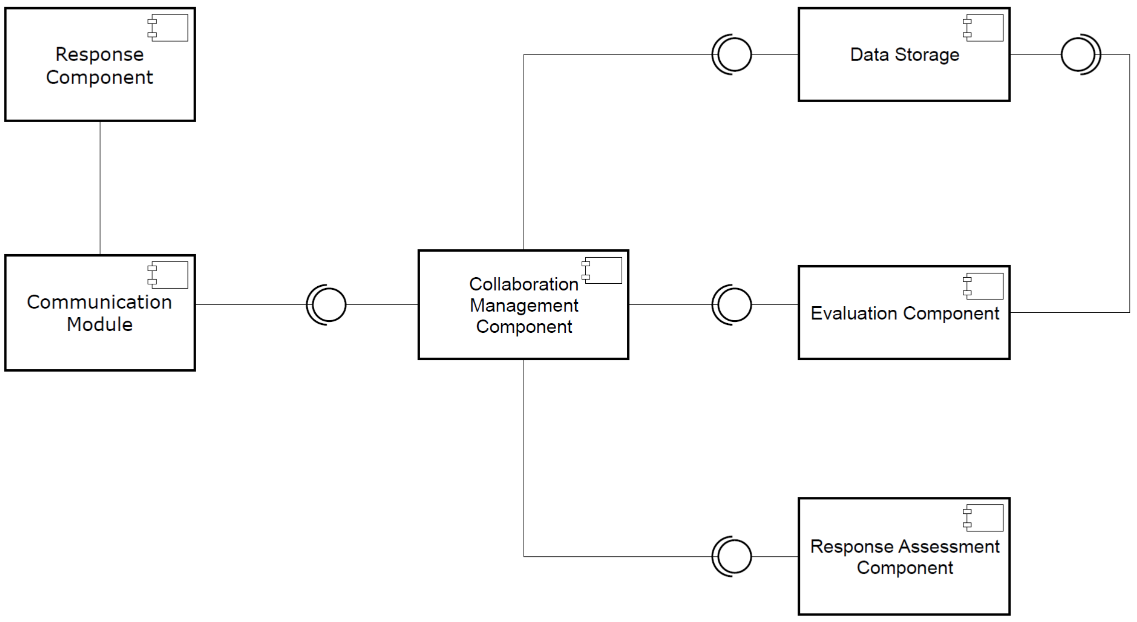


Figure 4.20: The Component Diagram of the "Riposte" framework.

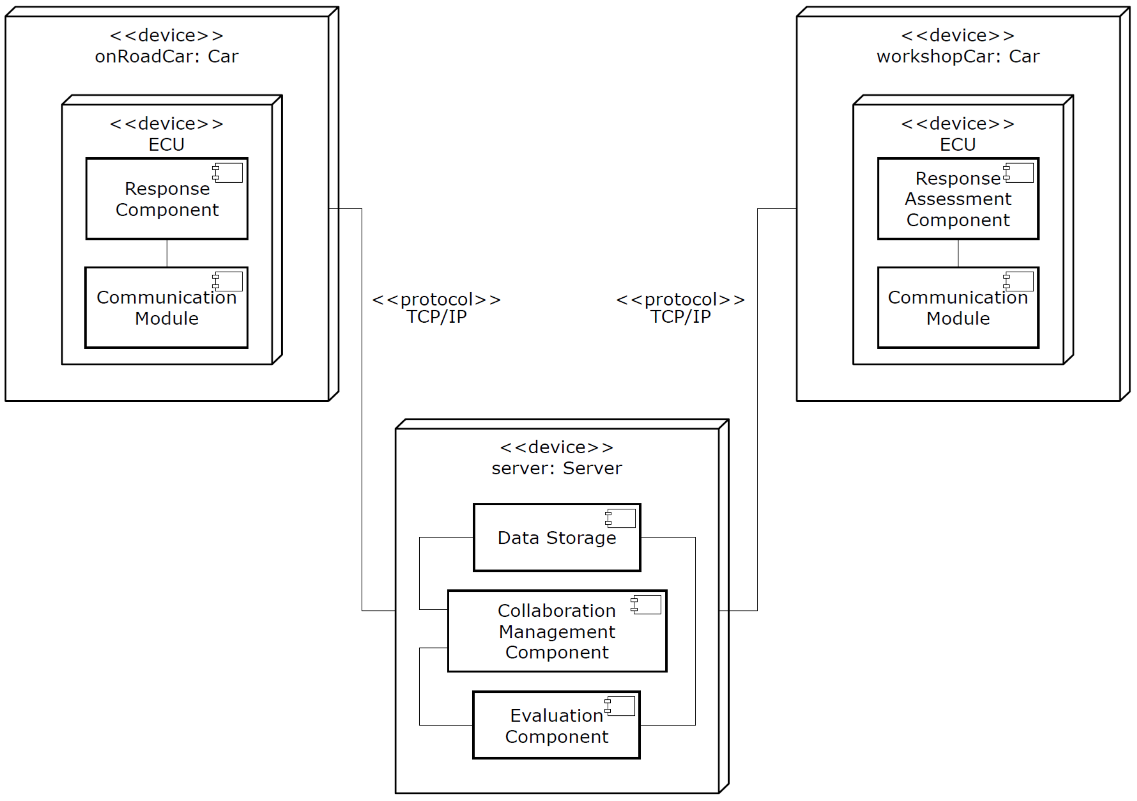


Figure 4.21: The Deployment Diagram of the "Riposte" framework.

4.2.4 Framework Behavior

To describe the behavioral aspect of the framework, figure 4.22 illustrates the behavior in a Sequence Diagram. The Sequence Diagram shows an onRoadCar, a server, and *n* workshopCars.
The onRoadCar and workshopCars will first enroll to the server, passing in their software version, hardware specifications, and the collaboration status. If the car is collaborative, the car then would act as a testbed and helps in conducting experiments. If the onRoadCar gets attacked after some time, it would send a request evaluation message to the server letting it know that it is under attack, and the response technique it is planning to apply. The server would check its database to look for previously assessed response techniques that match the software version and the hardware specifications of the onRoadCar. If it finds that all response techniques were previously assessed and evaluated, it will see whether the onRoadCar is using the best response technique (e.g. most efficient technique with least CPU utilization). If the car is using the same technique, it would send an acknowledgement message, letting the onRoadCar know that it is using the best technique. Otherwise, it will send the best response technique so that the onRoadCar can apply it instead. However, in the case of unassessed response techniques (e.g. due to added new techniques), the server would start looking for available workshopCars (i.e. testbeds) to help assess these response techniques. The server would update these workshopCars with the appropriate response technique and run an attack simulation in order to activate the newly applied response techniques. After the workshopCar is done applying the response technique that was assigned to it, it will revert the changes that happened and go back to its original state (i.e. before the start of the experiment) and be ready for more experiments. The operation of running experiments (i.e. finding the workshopCar, updating them, and running attack simulation) is done in parallel, to ensure less time is being used for each car and a faster response for the onRoadCar.
When all response techniques were assessed and all the logs were received from the workshopCars, an evaluation of these logs would start. The server would look for a specific metric from these logs to find the best response technique. Once found, it would update its database and then push the update to the onRoadCar. The onRoadCar would then apply the best response technique received from the server. If the onRoadCar gets attacked with the same attack later, it would contact the server for evaluation again. However, this time the onRoadCar is using the best response technique according to the server, so the server would quickly send an acknowledgement message letting the onRoadCar know that it is using the best response technique that it knows of.
Another behavioral scenario for the framework is that the onRoadCar can first apply the default response technique it has (such as rebooting) and then requests an evaluation of the technique it used. This way, the onRoadCar can instantly respond to an attack without having to wait for the server's evaluation, and still can get the best response technique to be used if the attack happens to be performed on it again at a later time. However, the first time the onRoadCar uses the default response technique could be not the best response technique to be used, a trade-off between instant response to an attack and the efficiency of the response.

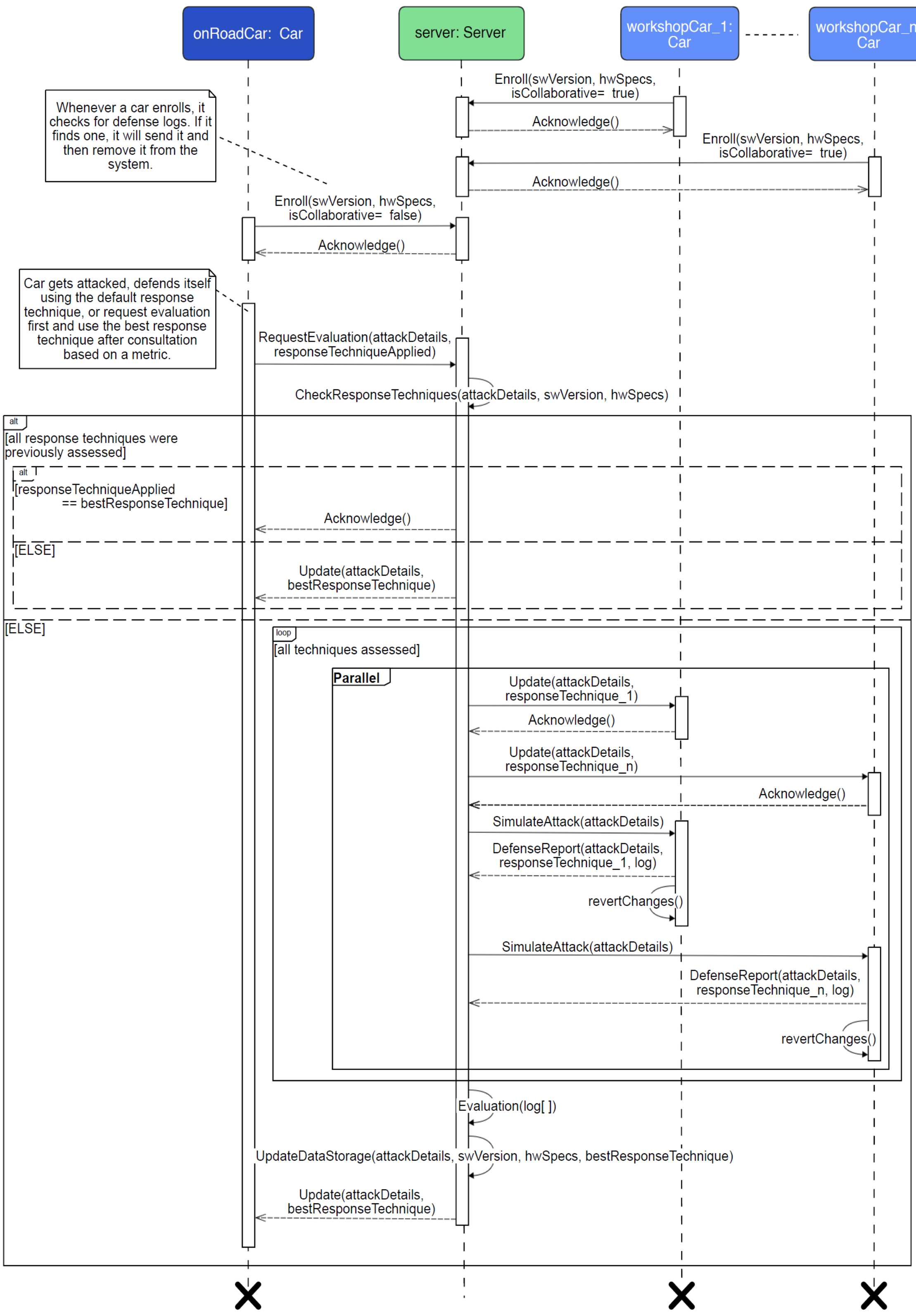

onRoadCar: Car
server: Server
workshopCar_1: Car
workshopCar_n: Car
Enroll(swVersion, hwSpecs, isCollaborative= true)
Acknowledge()
Enroll(swVersion, hwSpecs, isCollaborative= true)
Acknowledge()
Whenever a car enrolls, it checks for defense logs. If it finds one, it will send it and then remove it from the system.
Enroll(swVersion, hwSpecs, isCollaborative= false)
Acknowledge()
Car gets attacked, defends itself using the default response technique, or request evaluation first and use the best response technique after consultation based on a metric.
RequestEvaluation(attackDetails, responseTechniqueApplied)
CheckResponseTechniques(attackDetails, swVersion, hwSpecs)
alt
[all response techniques were previously assessed]
alt
[responseTechniqueApplied == bestResponseTechnique]
Acknowledge()
[ELSE]
Update(attackDetails, bestResponseTechnique)
[ELSE]
loop
[all techniques assessed]
Parallel
Update(attackDetails, responseTechnique_1)
Acknowledge()
Update(attackDetails, responseTechnique_n)
Acknowledge()
SimulateAttack(attackDetails)
DefenseReport(attackDetails, responseTechnique_1, log)
revertChanges()
SimulateAttack(attackDetails)
DefenseReport(attackDetails, responseTechnique_n, log)
revertChanges()
Evaluation(log[ ])
UpdateDataStorage(attackDetails, swVersion, hwSpecs, bestResponseTechnique)
Update(attackDetails, bestResponseTechnique)

Figure 4.22: The Sequence Diagram of the "Riposte" framework.

4.2.5 Implementation and Testing

To test the framework, a simulation has been conducted. The simulation process started by implementing the framework using Python on the Raspberry Pi 4 devices and the server. One of the Raspberry Pi's acted as the onRoadCar that requests an evaluation from the server when it is under attack. The other two Raspberry Pi's acted as workshopCars that are ready to perform assessments on different response techniques that are sent by the server.

In order to test the framework, an implementation of an attack and three different response techniques were placed. The attack selected was a simple representation of a *Privilege Escalation Attack*, which is further discussed in 4.2.5.1.

For defending against the attack, a selection of response techniques was performed. The selection was based on including response techniques that are suitable to deal with the privilege escalation attack and to use response techniques from both strategies (mitigation and recovery). Afterward, two of these nominated techniques were picked based on the lower complexity level of implementation, the Graceful Degradation/Limp Mode and Reparameterization. Finally, that last technique was the Re-instantiation/Restart response technique, which was picked manually due to it being an edge case (the car has to disconnect, reboot, and reconnect). These response techniques are further discussed in subsection 4.2.5.2.

The process starts by manually attacking the onRoadCar to trigger the framework, causing the onRoadCar to request an evaluation from the server with its applied response technique that it's planning to use. Afterward, the framework would run and send back the optimal response technique to the onRoadCar. This optimal technique is based on two main terms: it should be an effective technique (can prevent the attack from happening again) and is the most efficient technique out of the other effective techniques.

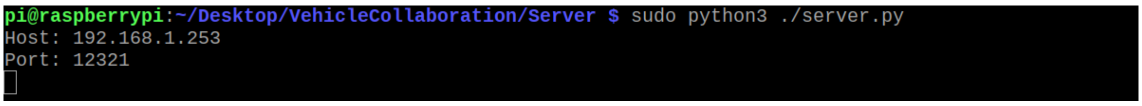
A response technique can either be *effective or not effective.* This is measured by running the attack simulation a second time on the workshopCars after finishing the execution of the response technique. If the attack did not succeed, then this technique would be considered effective against that attack. However, if the attack was successful, then the response technique used is not effective.

As for efficiency, the implementation focused on the *time duration when applying the technique.* Each response technique would be timed exactly before the technique is applied and exactly after it is done applying. Afterward, the framework would look for all effective techniques it has and evaluate them based on their duration, and the shortest duration technique would be marked as the most efficient.

The simulation process uses the implementation found on the author's GitHub repository[2], and this is how it was performed:

Step 1: The server started listening on the interface's IP address: 192.168.1.253, with port number: 12321. Figure 4.23 shows the server in the listening state.

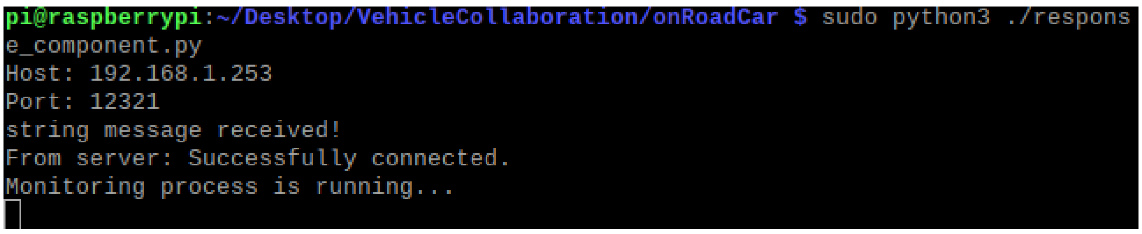
[2]https://github.com/SaifDaghistani/Riposte



```
pi@raspberrypi:~/Desktop/VehicleCollaboration/Server $ sudo python3 ./server.py
Host: 192.168.1.253
Port: 12321
```

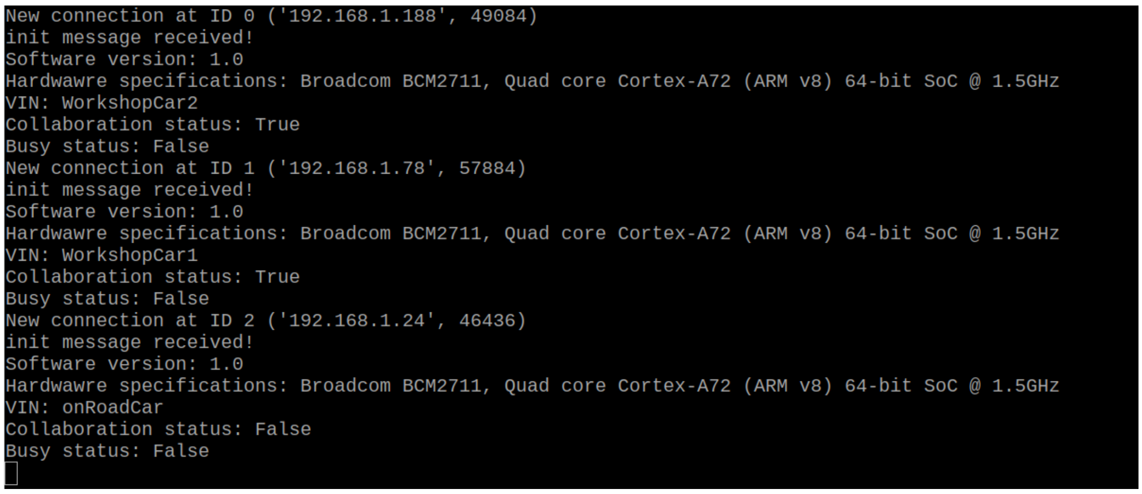
Figure 4.23: Server: listening state.

Step 2: Three clients connect to the server by passing in the server's IP address and the port number, sending their technical specifications, and will start the monitoring process, as seen in figure 4.24. The server then receives the connection request, sending an acknowledgment message to the client that the connection was successful. The server also receives the technical specifications sent from the client and prints them, as seen in figure 4.25. Furthermore, the last figure also shows that there two collaborative clients that connected to the server (WorkshopCar1 and WorkshopCar2) and one non-collaborative client (onRoadCar).



```
pi@raspberrypi:~/Desktop/VehicleCollaboration/onRoadCar $ sudo python3 ./respons
e_component.py
Host: 192.168.1.253
Port: 12321
string message received!
From server: Successfully connected.
Monitoring process is running...
```

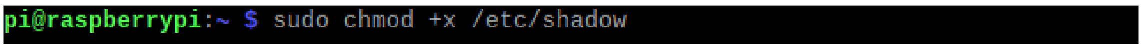
Figure 4.24: Client(onRoadCar): successfully connected.



```
New connection at ID 0 ('192.168.1.188', 49084)
init message received!
Software version: 1.0
Hardwawre specifications: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
VIN: WorkshopCar2
Collaboration status: True
Busy status: False
New connection at ID 1 ('192.168.1.78', 57884)
init message received!
Software version: 1.0
Hardwawre specifications: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
VIN: WorkshopCar1
Collaboration status: True
Busy status: False
New connection at ID 2 ('192.168.1.24', 46436)
init message received!
Software version: 1.0
Hardwawre specifications: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
VIN: onRoadCar
Collaboration status: False
Busy status: False
```

Figure 4.25: Server: several clients connected, listing their technical specifications and their collaboration status.

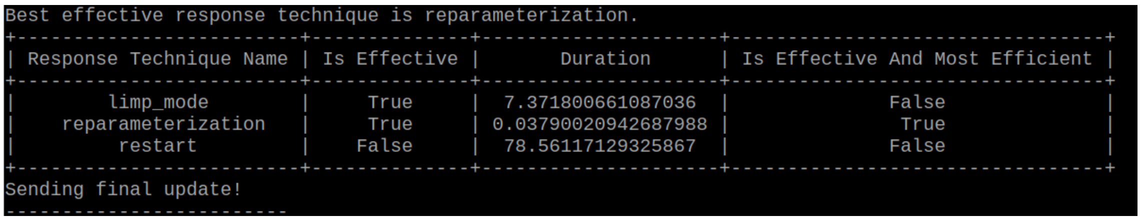
Step 3: The attack is manually triggered in the onRoadCar in order to start the framework, as seen in the figure 4.26.



```
pi@raspberrypi:~ $ sudo chmod +x /etc/shadow
```

Figure 4.26: Client(onRoadCar): manual execution of the attack script.

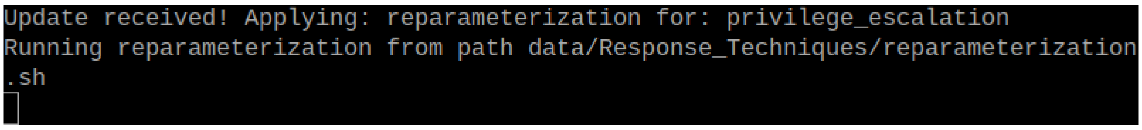
By now, the server would manage the collaboration operation with the available workshop cars. After the assessment of all available response techniques, the server lists the results of the evaluation and sends the effective and most efficient response technique to the onRoadCar, as seen in figure 4.27.



```
Best effective response technique is reparameterization.
+------------------------+-------------+--------------------+--------------------------------+
| Response Technique Name | Is Effective |      Duration      | Is Effective And Most Efficient |
+------------------------+-------------+--------------------+--------------------------------+
|        limp_mode        |     True     | 7.371800661087036  |             False              |
|    reparameterization   |     True     | 0.03790020942687988 |             True               |
|         restart         |    False     | 78.56117129325867  |             False              |
+------------------------+-------------+--------------------+--------------------------------+
Sending final update!
------------------------
```

Figure 4.27: Server: listing the results of the evaluation and the best effective response technique.

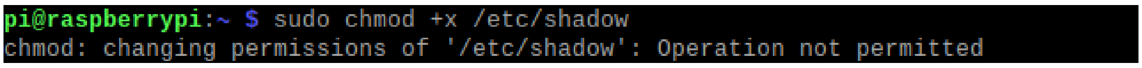
The onRoadCar receives the update message and applies the response technique suggested by the framework which, in this case, reparameterization, as shown in figure 4.28.



```
Update received! Applying: reparameterization for: privilege_escalation
Running reparameterization from path data/Response_Techniques/reparameterization
.sh
```

Figure 4.28: Client(onRoadCar): received the update from the server and applying the suggested response technique.

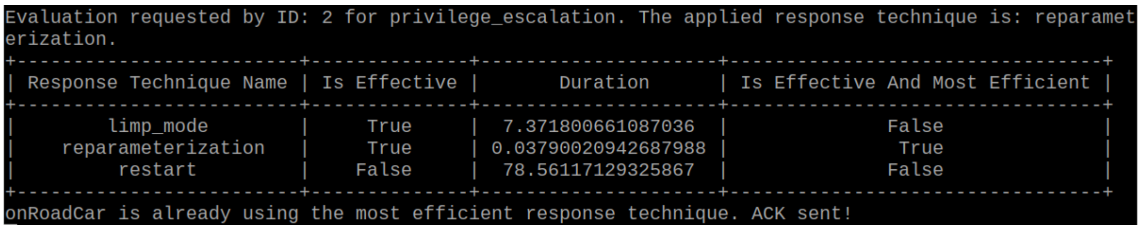
Step 4: A manual attack is executed again on onRoadCar to check whether the suggested response technique was applied, as shown in figure 4.29.



```
pi@raspberrypi:~ $ sudo chmod +x /etc/shadow
chmod: changing permissions of '/etc/shadow': Operation not permitted
```

Figure 4.29: Client(onRoadCar): checking that the response technique is applied.

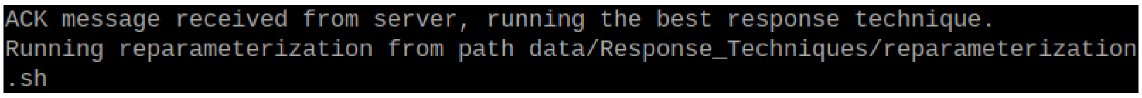
Step 5: A manual revert of the reparameterization technique was done (as discussed in 4.2.5.2), and an execution of the attack is performed again. This returned a response technique suggestion without the need for evaluation of any response technique because all response techniques were previously assessed and evaluated, as shown in figure 4.30.



```
Evaluation requested by ID: 2 for privilege_escalation. The applied response technique is: reparamet
erization.
+------------------------+-------------+--------------------+--------------------------------+
| Response Technique Name | Is Effective |      Duration      | Is Effective And Most Efficient |
+------------------------+-------------+--------------------+--------------------------------+
|        limp_mode        |     True     | 7.371800661087036  |             False              |
|    reparameterization   |     True     | 0.03790020942687988 |             True               |
|         restart         |    False     | 78.56117129325867  |             False              |
+------------------------+-------------+--------------------+--------------------------------+
onRoadCar is already using the most efficient response technique. ACK sent!
```

Figure 4.30: Server: The onRoadCar requests for evaluation, the server acknowledges that the onRoadCar is using the best response technique.

After the client receives the acknowledgement message, it will execute the response technique it already set up to use, as shown in figure 4.31



```
ACK message received from server, running the best response technique.
Running reparameterization from path data/Response_Techniques/reparameterization
.sh
```

Figure 4.31: Client(onRoadCar): The onRoadCar uses the already applied response technique after receiving an acknowledgement message from the server.

That being said, WorkshopCar1 has been picked to assess the restart response technique. After the execution of the technique, a manual running of the client was initiated in order to connect back to the server. And since the main purpose of the simulation was to test the framework and not to measure efficiency between response techniques, the extra time spent to manually connect back WorkshopCar1 to the server was not considered.

4.2.5.1 Attack Simulation

The attack used in the simulation was a simple representation of a privilege escalation attack. The car monitors the shadow file (/etc/shadow) that stores encrypted passwords of users on the system and can only be read by the root account. Any modification to the attributes of this file requires root privileges and thus, triggering the framework.
For the attack to happen, this simple script was used:

```bash
#!/bin/bash

sudo chmod +x /etc/shadow
```

That being said, more sophisticated attacks were planned to be used by using vulnerable Raspberry Pi OS's, such as RasPwn OS[3] or DV-PI[4]. Unfortunately, most resources were designed for Raspberry Pi 3 and/or earlier versions and were not compatible with Raspberry Pi 4.

4.2.5.2 Implemented Response Techniques

This subsection describes the selected response techniques and their implementation when applied in the simulation process.

Graceful Degradation/Limp Mode

As discussed in 4.1.4.3, this response technique allows the system to continue functioning and prevent a catastrophic failure in the case of a compromise. This is done by lowering the performance of the system or reducing the number of services provided.

[3]http://raspwn.org/
[4]https://whitedome.com.au/re4son/sticky-fingers-dv-pi/

It is important to mention that every system requires different sets of services to perform its core functionality, and selecting what functionalities/services to disable/enable can widely differ on each system.
For this study, the core functionality considered was to keep the operating system running with minimum losses of services. Table 4.2 shows the services that were available on the systems and table 4.3 lists the services that will be disabled in order to limit the privilege escalation attack.

Table 4.2: Available services on the system.

Available services			
alsa-utils	cups-browsed	lvm2	rpcbind
anacron	dbus	lvm2-lvmpolld	rsync
apparmor	dhcpcd	mdadm	rsyslog
auditd	dphys-swapfile	mdadm-waitidle	saned
avahi-daemon	exim4	networking	ssh
binfmt-support	haveged	nfs-common	sudo
bluetooth	hddtemp	ntp	triggerhappy
console-setup.sh	hwclock.sh	plymouth	udev
cron	keyboard-setup.sh	plymouth-log	unattended-upgrades
cryptdisks	kmod	procps	uuidd
cryptdisks-early	lightdm	raspi-config	x11-common
cups	live-tools	rng-tools	

Table 4.3: Services that are disabled in the system.

Disabled services
bluetooth
cron
exim4
nfs-common
ssh

The **SSH** and **bluetooth** services can allow for remote shell access to the system. Thus, disabling them limits the possibility for an attacker to have such access.
The **cron** service allows for scheduled jobs to be executed at a certain time. Keeping this enabled can let the attacker schedule an execution of a malicious script allowing him/her to re-enable other services in case they were disabled, and regain access.

The **exim4** is a mail transfer agent, and disabling it can limit any private or system information to be sent to the attacker via mail.
The **nfs-common** is used to enable the sharing of files between systems that reside within the same local area network. And since an attack can arrive from compromised local systems, malicious scripts can be transferred to the main system.
The following script was used to implement this technique:

```bash
#!/bin/bash

ARG=$1

if [ $ARG = "apply" ]
then
        echo "Stopping and disabling SSH"
        sudo systemctl stop ssh && sudo systemctl disable ssh
        echo "Stopping and disabling bluetooth"
        sudo systemctl stop bluetooth && sudo systemctl disable bluetooth
        echo "Stopping and disabling cron"
        sudo systemctl stop cron && sudo systemctl disable cron
        echo "Stopping and disabling exim4"
        sudo systemctl stop exim4 && sudo systemctl disable exim4
        echo "Stopping and disabling nfs-common"
        sudo systemctl stop nfs-common && sudo systemctl disable nfs-common
elif [ $ARG = "revert" ]
then
        echo "Starting and enabling SSH"
        sudo systemctl start ssh && sudo systemctl enable ssh
        echo "Starting and enabling bluetooth"
        sudo systemctl start bluetooth && sudo systemctl enable bluetooth
        echo "Starting and enabling cron"
        sudo systemctl start cron && sudo systemctl enable cron
        echo "Starting and enabling exim4"
        sudo systemctl start exim4 && sudo systemctl enable exim4
        echo "Starting and enabling nfs-common"
        sudo systemctl start nfs-common && sudo systemctl enable nfs-common
fi
```

Reparameterization

Reparameterization, as discussed in 4.1.4.2, is used to dynamically adjust the system based on the situation. The results of applying this technique can lead to a non-optimal state, which is similar to the Graceful Degradation/Limp mode technique. Therefore, the implementation of this technique focused on making the shadow file immutable; preventing any further attribute modifications on this file even by root.
The following script was used to implement this technique:

```bash
#!/bin/bash

ARG=$1

if [ $ARG = "apply" ]
then
```

```
        sudo chattr +i /etc/shadow
elif [ $ARG = "revert" ]
then
        sudo chattr -i /etc/shadow
fi
```

Re-instantiation/Restart

This is one of the interesting response techniques that was picked manually to better evaluate the framework. The framework should handle the effects of rebooting while successfully assessing the technique. The system first has to save the analysis log before it reboots, finishes rebooting, reconnects back to the server, and sends the log file with the analysis data.
This response technique also helped in improving the architecture of the framework. If a car loses its connection to the server during the evaluation process for some reason, it can still reconnect and send the analysis log to the server without an issue.
The following script was used to implement this technique:

```
#!/bin/bash

ARG=$1

if [ $ARG = "apply" ]
then
        sudo reboot
elif [ $ARG = "revert" ]
then
        echo "revert"
fi
```

4.3 Riposte: Design Evaluation

Design evaluation is an important step of design science research. With that, further cycles can be performed in order to enhance the quality of the artefact. These different quality attributes can be in terms of functionality, completeness, reliability, usability, and other relevant attributes [67].
For this study, the artefact is evaluated in terms of *safety* through conducting a questionnaire with domain experts. The responses of the questionnaire were then analyzed using thematic analysis following the steps recommended by [7]. Figure 4.32 shows the thematic model created for the analysis.

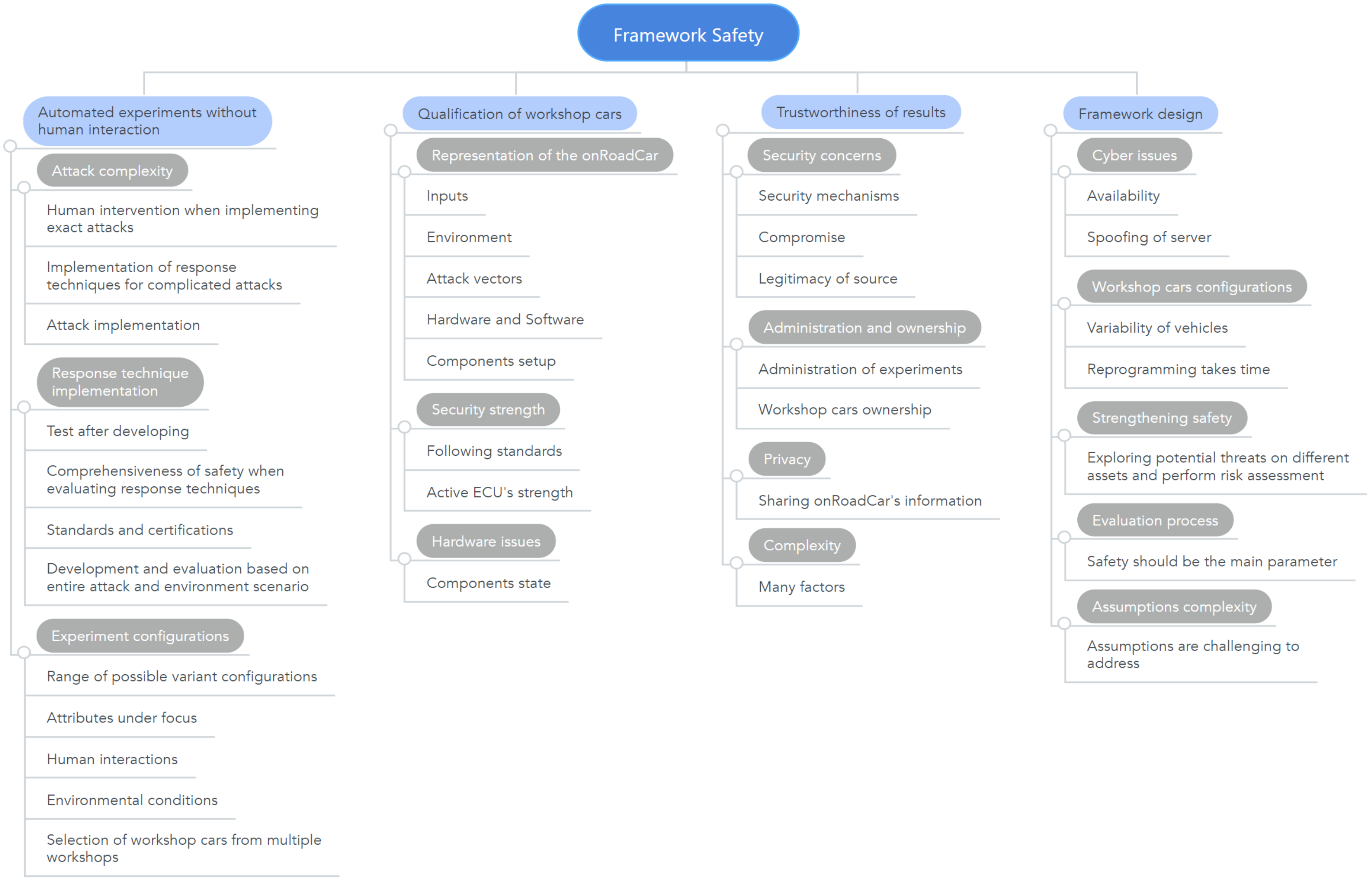

Framework Safety
Automated experiments without human interaction
Attack complexity
Human intervention when implementing exact attacks
Implementation of response techniques for complicated attacks
Attack implementation
Response technique implementation
Test after developing
Comprehensiveness of safety when evaluating response techniques
Standards and certifications
Development and evaluation based on entire attack and environment scenario
Experiment configurations
Range of possible variant configurations
Attributes under focus
Human interactions
Environmental conditions
Selection of workshop cars from multiple workshops
Qualification of workshop cars
Representation of the onRoadCar
Inputs
Environment
Attack vectors
Hardware and Software
Components setup
Security strength
Following standards
Active ECU's strength
Hardware issues
Components state
Trustworthiness of results
Security concerns
Security mechanisms
Compromise
Legitimacy of source
Administration and ownership
Administration of experiments
Workshop cars ownership
Privacy
Sharing onRoadCar's information
Complexity
Many factors
Framework design
Cyber issues
Availability
Spoofing of server
Workshop cars configurations
Variability of vehicles
Reprogramming takes time
Strengthening safety
Exploring potential threats on different assets and perform risk assessment
Evaluation process
Safety should be the main parameter
Assumptions complexity
Assumptions are challenging to address

Figure 4.32: Thematic model for the safety of the developed framework.

As seen in the previous figure, the participants have expressed some considerations and concerns that need to be taken into account for the proposed framework. The overall safety of the framework was rated by the participants as **slightly unsafe with a score of 2 out of 5 (Scale from 1(Extremely unsafe) to 5(Very safe); Median = 2.5; Interquartile Range = 3.25)**, as seen in figure 4.33.

The safety concerns and considerations that are shown in the thematic model are used to explain the obtained overall safety rate of the developed framework

4.3.1 Automated experiments without human interaction

One of the considerations was about the attack complexity when conducting automated experiments. Being able to replicate exact attacks from a source can require human intervention, including consultations with safety and security experts. That being said, the implementation of response techniques that target complicated attacks can be hard to perform, especially in an automated manner. For that, one of the participants suggested the use of emulators within simulators in order to overcome environment replication challenges and get as close as possible to the conditions that the on road car is in.

Another consideration was the implementation of response techniques. Standards and certifications should be followed when developing the response techniques. Afterward, the response techniques should be tested and evaluated in terms of safety by running an entire attack scenario along with the environmental input to achieve optimal results.

Experiment configurations was another consideration. Participants mentioned that environmental conditions and human interactions should be considered when evaluating response techniques, as the environment for a moving vehicle is quite different than a stationary vehicle, it can lead to varying results from successful to devastating. Moreover, the attributes under focus can play an important role when judging the safety of automated experiments. A participant also mentioned that experiments should be kept inside a variability of ranges of the possible configurations. This in order to avoid any unconsidered states that might be left unnoticed or not anticipated in the design analysis. Lastly, a participant mentioned that the selection process of the workshop cars, when there are multiple workshops available, should be addressed.

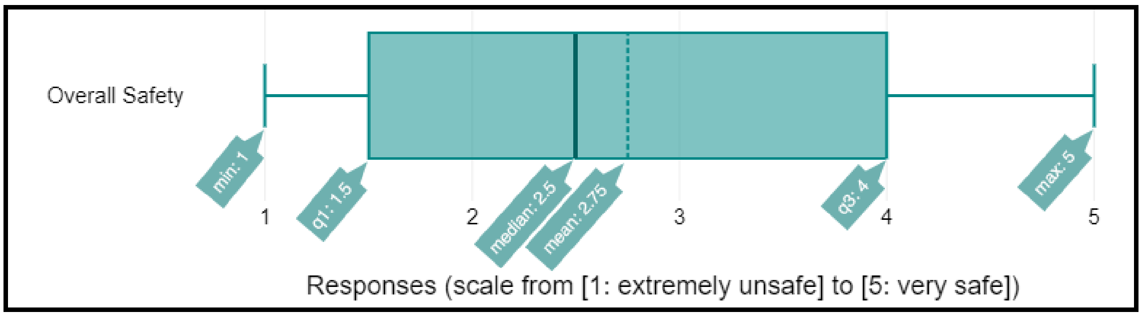


Figure 4.33: Questionnaire box plot: overall safety of the developed framework.

4.3.2 Qualification of workshop cars

The qualification of workshop cars was another safety concern as it was rated **severe with a score of 5 out of 5 (Scale from 1(Insignificant) to 5(Severe); Median = 5; Interquartile Range = 1.5)**, as shown in figure 4.34.
The participants focused on how well the workshop car represents the on road car that is under attack. This included the representation of inputs, environment conditions, and attack vectors. Participants also pointed out that technical specifications (hardware and software) and component setup should be identical in order to get optimal evaluation results.
Security strength should also be considered as following the standards and increasing the security of active safety ECUs could potentially reduce unintended errors during the attack simulation process.
One of the participants also pointed out that the components state should be seriously considered. As components age, workshop cars can provide wrong data to the server, making the evaluation process faulty. However, the participant also pointed out that this can be resolved through using majority voting with many cars running the experiments simultaneously.

4.3.3 Trustworthiness of results

The trustworthiness of results was addressed by the participants and rated as **severe with a score of 5 out of 5 (Scale from 1(Insignificant) to 5(Severe); Median = 5; Interquartile Range = 2.25)**, as shown in figure 4.35.
The security aspect of the workshop cars should be considered by implementing different security mechanisms, such as mutual authentication and secured communication channel. Additionally, workshop cars need to be ensured that they are not compromised through malware, for example. Also, the legitimacy of the source should be taken into account and to make sure that no tampering was introduced.
One of the participants also pointed out that the administrative authority of the experiments should be considered, whether it was the trustful OEM or another third party that could raise concerns. The participant also pointed out that the ownership of the workshop cars should be taken into account, whether the workshop cars are owned by the OEM (i.e. testbeds) or individuals who agreed on participating in the

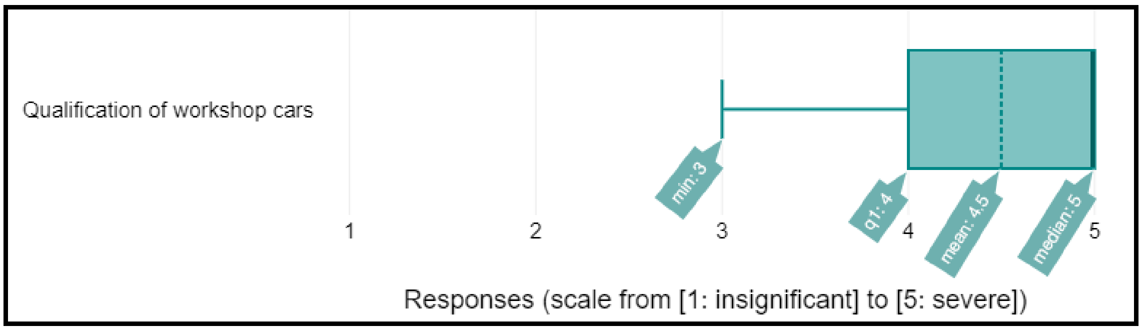


Figure 4.34: Questionnaire box plot: qualification of workshop cars as a safety concern.

experimentation process.
Privacy was a concern as well when sharing the information of the on road car with the server and other workshop cars. Lastly, for the on road car to trust the solution can be a complex process, as it includes many factors that need to be considered and taken care of.

4.3.4 Framework Design

Lastly, there were some considerations for the framework design that should be considered. One of the main topics was the cyber issues, where one of the participants pointed out that besides safety, availability can in some cases become a factor. An example of that would be when a driver is using a compromised infotainment system and gets automatically shut off in response to a cyber attack, the unavailability could cause the driver to be irritated or nervous, which could lead to an unfortunate accident. Another cyber issue is spoofing the server's identity and suggesting inefficient response techniques to be used by the on road car. Although the participant points out that this applies to all centralized communications, but it is a concern that should be taken into account.
Workshop car configurations were also touched by the participants, as one of the participants mentioning that the variability between vehicles is a major factor that needs to be considered because it is hard to find two identical cars with the same parameters and configurations. This participant also pointed out the workshop cars are likely going to be reprogrammed, which could delay the process of responding to the on road car and could delay the consultation of other requests as well.
One participant mentioned the strengthening of safety through looking into threats that could potentially be explored on different assets of the vehicle and perform a risk assessment on the potential of exploiting these threats and influencing system safety. The participant also pointed out that, when evaluating different response techniques, the main parameter should be safety.
Finally, one of the participants pointed out that the framework relies on a set of assumptions that could be challenging to address, such as attack simulation.

From the analysis of the questionnaire's data, these are the main takeaways on how to improve the safety of the framework based on the participants' perception:

- Complex attacks can be challenging to implement

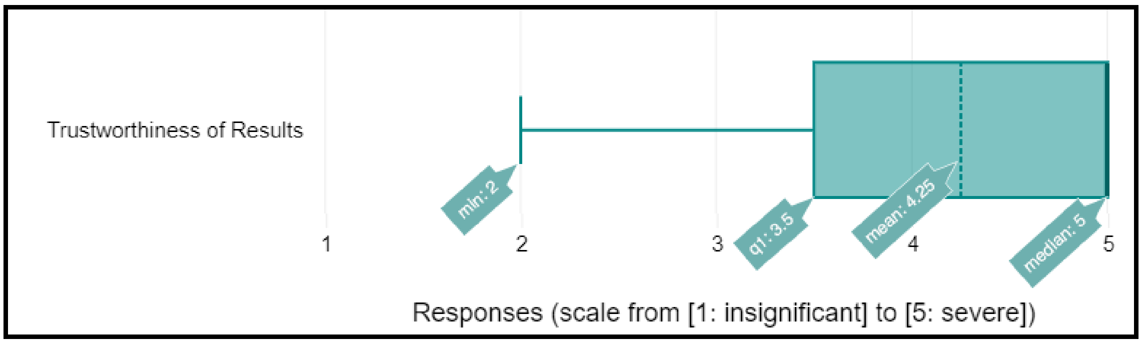


Figure 4.35: Questionnaire box plot: trustworthiness of results as a safety concern.

- Response techniques should follow standards and certifications
- Response techniques should be tested before implementation
- Selection process of workshop cars from different workshops
- Workshop car should represent the on road car from all aspects
- Components state should be regularly addressed
- Implementation of security mechanisms should be considered
- Administration of experiments should be performed by a trusted party
- Privacy of the on road car should be considered
- Cyber issues should be counted for, such as availability is mitigated by adding redundant servers

5 Threats to Validity

This chapter discusses possible threats to validity for this thesis study. The internal validity is discussed in 5.1, followed by external validity in 5.2 and construct validity in 5.3, and finally the conclusion validity in 5.4.

5.1 Internal Validity

Since this thesis work is done individually, the main threat could be the experimenter bias. However, this can be mitigated using the design science research methodology. Moreover, since this study aims to perform a qualitative study with domain experts, a convenience sampling of participants has been done and this could include selection bias. While this threat can be mitigated through randomizing the selection of participants, finding participants that are expert on the safety automotive field was challenging due to the overall pandemic situation and time constraints. For that, this risk has had to be taken.
Furthermore, a few participants were specialized in the automotive security domain rather than safety, even though the evaluation was performed on the safety of the framework. For that, it can be argued that ensuring the functional safety of components (the correct functionality of a safety component in a system [68]) is a security matter as well. Thus, securing safety devices from cyber attacks can in the end improve safety.
Another possible threat is bias in the coding process of the qualitative data by existing hypotheses as well as the interpretation process that could be subjective. While it is true that higher-order themes were already identified based on the questions provided in the questionnaire, codes and themes were not forcefully assigned to the already assigned themes. Instead, a new higher-order theme emerged. Moreover, a discussion of the questions with the supervisor was performed before handing in the questionnaire in order to minimize this threat.
Additional possible threat is the variation of the experience and expertise of the participants. While it is beneficial to have varying participants in order to elicit different viewpoints and concerns, this variation could influence the safety assessment of the framework, as different levels of experience/expertise could play a major role in the assessment. That being said, due to the limited number of participants, this threat cannot be easily dealt with and the risk has to be taken.
Finally, the hardware and software used for the experiment could be a possible threat for this study. Using different hardware or different implementation of the framework could lead to different results than this study has showed. Therefore,

the hardware specifications used for the experiment are explained in detail and the software implementation can be found in the report itself and an online repository mentioned earlier.

5.2 External Validity

This thesis focuses on simulated attacks for experimenting and comparing the different response techniques, some real-life factors are not to be considered within this study. However, the simulation and experimentation processes, together with the used resources in this thesis, including the software and hardware, are reported in detail, and thus enable replication and increases the reliability and generalizability of this study.
Another possible external threat is the sample size of the questionnaire. It was not possible to find more participants due to time constraints and the overall pandemic situation. To partially deal with this risk, the selected participants were diverse in occupation, education, experience, and work organizations.

5.3 Construct Validity

To prevent the limited scope that the participants could face when answering the questionnaire, a mix of closed and open-ended questions was provided. This in order to allow the participants to freely express what they think of the proposed solution without limiting them to scale answers only.

5.4 Conclusion Validity

As the thesis study relies on subjective measurements in order to answer a few research questions, there could have been occasions where the author has influenced a participant's response before answering the questionnaire. During the call/meeting, the author tried to not affect the participants' answers. This means that the call/meeting was not exactly the same for every participant, even if it was planned to be. However, this threat was taken into account and a video demonstration of the proposed solution, as well as a thorough explanation of the context, has been provided to each participant before they answer the questionnaire. Additionally, it can be argued that curiosity can support the conclusion by providing better feedback from participants.

6 Conclusion

The aim of this thesis was to investigate and find potential solution to the raising cyber security concerns of the automotive sector due to the wide connectivity of modern vehicles. The purpose was to design a solution that could help connected vehicles to collaborate in finding out the most suitable cyber attack response technique for a particular attack that is taking place in real time.
The contributions of this thesis are the following:

- **A dataset of cyber attack response techniques for the software asset of the vehicle:** The study first identified and explored the available state-of-the-art cyber attack response techniques that target the software asset of the vehicle through conducting a literature review, resulting in a list of the latest state-of-the-art cyber attack response techniques for the software asset along with their description. This rich description of the attack response techniques helps practitioners to have a better understanding of the functionality of each response technique. Moreover, it provides a means for supporting the selection process thereof in order to let practitioners adopt the most suitable response techniques for the tasks at hand.
- **A collaborative framework to evaluate and pick an effective and most efficient cyber attack response technique against an ongoing cyber attack:** A design of a collaborative framework has been proposed to help vehicles under attack to collaborate to find out an effective and most efficient response technique for that on-going attack, described in structural and behavioral UML designs. Additionally, this design was implemented and tested by conducting a simulation experiment. Furthermore, the proposed framework is the first of its kind in the literature that proposes the leverage of collaboration between vehicles to enhance the efficiency and effectiveness of the response to cyber security attacks, as previous works focused on collaborative IDS systems and collaborative attack detection mechanisms. Therefore, this framework could serve as a stepping stone toward supporting the collaboration of vehicles in regard to response techniques. Moreover, further studies can be conducted toward either enhancing the framework itself or adapting as well as evaluating other collaborative attack response alternatives.
- **An evaluation of the framework in terms of safety:** An evaluation of safety was performed through running a qualitative study with experts from the automotive security and/or safety domain through a questionnaire to know the perception of these experts on the safety of the proposed solution. The results of the safety evaluation indicate that there are some safety concerns that need to be addressed such as the qualification of workshop cars, trustworthiness

of results, automated experiments without human interaction, and the overall framework design. Moreover, the safety evaluation shows that the interplay of safety and security should be one of the top priorities when it comes to developing a solution in the automotive domain. The evaluation also provided insight into how the safety concerns could be addressed in the future.

The outcome of this study could benefit the researchers by providing a dataset and discussion of the available state-of-the-art cyber attack response techniques for the software asset of the vehicle. It provides a basis for future research whether to improve the framework itself or learn about the domain area.

Furthermore, this study could benefit the practitioners as well through the use of the Riposte framework to collaboratively evaluate different response techniques for an ongoing cyber attack. This could enhance the cyber security of on road vehicles by increasing the efficiency and effectiveness of cyber response techniques against cyber attacks.

6.1 Future Work

Due to the lack of related work regarding the collaboration of vehicles in evaluating different response techniques in real time, this proposed solution can serve as a starting point in this direction. Therefore, different implementation alternatives could be sought.

That being said, further design science research cycles should be conducted to improve the proposed solution. This to provide a safer and more efficient solution than the current one.

In addition, further modifications and improvements could be added, such as the process of replicating the environmental inputs from the on-road car to the workshop cars, as pointed out by experts, in order to make better decisions when evaluating the response techniques. Another suggestion is to add more security mechanisms such as encryption and mutual authentication to minimize spoofing threats and man-in-the-middle attacks.

Bibliography

[1] "Security threat: About 63% vehicle makers don't test half of hardware and software technologies: Report, Auto News, ET Auto." Available at: `https://tinyurl.com/benppzju` (Accessed: 2021-02-11).

[2] "QNX Auto Blog: Holistic Security for the Software-Defined Car." Available at: `https://tinyurl.com/m47afnxf` (Accessed: 2021-04-04).

[3] A. Lautenbach, "On Cyber-Security for In-Vehicle Software," tech. rep., Chalmers University of Technology, Göteborg, 2017.

[4] NXP, "Automotive Gateway: A Key Component to Securing the Connected Car," tech. rep.

[5] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, "A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems," *IEEE Access*, vol. 7, pp. 13260–13283, 2019.

[6] F. Giaimo, H. Andrade, and C. Berger, "Continuous experimentation and the cyber–physical systems challenge: An overview of the literature and the industrial perspective," *Journal of Systems and Software*, vol. 170, p. 110781, 12 2020.

[7] D. S. Cruzes and T. Dyba, "Recommended Steps for Thematic Synthesis in Software Engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*, no. 7491, pp. 275–284, IEEE, 9 2011.

[8] J. W. Creswell, *Educational research: Planning, conducting, and evaluating quantitative.* 2002.

[9] M. N. Adnan, A. Kabir, L. Karim, and N. Khan, "Performance Comparison of Different Software Fault Tolerance Methods," tech. rep., 2011.

[10] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-variant systems a secretless framework for security through diversity," *15th USENIX Security Symposium*, no. August, pp. 105–120, 2006.

[11] L. F. Combita, J. Giraldo, A. A. Cardenas, and N. Quijano, "Response and reconfiguration of cyber-physical control systems: A survey," in *2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC)*, pp. 1–6, IEEE, 10 2015.

[12] M. Wu, H. Zeng, C. Wang, and H. Yu, "Safety Guard," in *Proceedings of the 54th Annual Design Automation Conference 2017*, vol. Part 12828, (New York, NY, USA), pp. 1–6, ACM, 6 2017.

[13] "PROTOCOLS – AUTOMOTIVE BASICS." Available at: `https://automotivetechis.wordpress.com/protocols/` (Accessed: 2021-02-09).

[14] H. Martin, Z. Ma, C. Schmittner, B. Winkler, M. Krammer, D. Schneider, T. Amorim, G. Macher, and C. Kreiner, “Combined automotive safety and security pattern engineering approach,” *Reliability Engineering & System Safety*, vol. 198, p. 106773, 6 2020.

[15] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson, “REMIND: A Framework for the Resilient Design of Automotive Systems,” in *2020 IEEE Secure Development (SecDev)*, pp. 81–95, IEEE, 9 2020.

[16] European Union Agency for Network and Information Security (ENISA), *Cyber security and resilience of smart cars. Good practices and recommendations.* No. December, 2017.

[17] “ISO - ISO/SAE DIS 21434 - Road vehicles — Cybersecurity engineering.” Available at: `https://www.iso.org/standard/70918.html` (Accessed: 2020-11-03).

[18] M. D. Stojanović and S. V. Boštjančič Rakas, eds., *Cyber Security of Industrial Control Systems in the Future Internet Environment.* Advances in Information Security, Privacy, and Ethics, IGI Global, 2020.

[19] “Cyber Threat Analysis - Cyber Experts.” Available at: `https://cyberexperts.com/cyber-threat-analysis-a-complete-overview/` (Accessed: 2020-11-09).

[20] “Sony’s CEO Stringer: No Network ’100% Secure’ | Newsmax.com.” Available at: `https://tinyurl.com/45j7a6b9` (Accessed: 2020-11-27).

[21] “Highly dependable automotive software.” Available at: `https://tinyurl.com/45frpzct` (Accessed: 2021-04-04).

[22] S. Kim and R. Shrestha, “Introduction to Automotive Cybersecurity,” in *Automotive Cyber Security*, pp. 1–13, Singapore: Springer Singapore, 2020.

[23] A. Karahasanovic, “Automotive Cyber Security Threat modeling of the AUTOSAR standard,” Master’s thesis, Chalmers University of Technology, 2017. Available at: `https://hdl.handle.net/20.500.12380/247979`.

[24] “The car will become a computer on wheels | Roland Berger.” Available at: `https://www.rolandberger.com/en/Insights/Publications/The-car-will-become-a-computer-on-wheels.html` (Accessed: 2021-02-09).

[25] C. Czosseck, R. Ottis, and K. Ziolkowski, *2012 4th International Conference on Cyber Conflict (CyCon 2012).* Tallinn, Estonia: NATO CCD COE, 2012.

[26] F. Giaimo, *Bridging the Experimental Gap: Applying Continuous Experimentation to the Field of Cyber-Physical Systems, in the Example of the Automotive Domain.* PhD thesis, Chalmers University of Technology and Gothenburg University, 2020.

[27] F. Auer and M. Felderer, “Current State of Research on Continuous Experimentation: A Systematic Mapping Study,” in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 335–344, IEEE, 8 2018.

[28] E. Mousavinejad, F. Yang, Q.-L. Han, X. Ge, and L. Vlacic, “Distributed Cyber Attacks Detection and Recovery Mechanism for Vehicle Platooning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, pp. 3821–3834, 9 2020.

[29] T. Nandy, R. M. Noor, M. Yamani Idna Bin Idris, and S. Bhattacharyya, "T-BCIDS: Trust-Based Collaborative Intrusion Detection System for VANET," in *2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA)*, pp. 1–5, IEEE, 2 2020.

[30] G. Raja, S. Anbalagan, G. Vijayaraghavan, S. Theerthagiri, S. V. Suryanarayan, and X.-W. Wu, "SP-CIDS: Secure and Private Collaborative IDS for VANETs," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2020.

[31] F. Giaimo and C. Berger, "Continuous Experimentation for Automotive Software on the Example of a Heavy Commercial Vehicle in Daily Operation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12292 LNCS, pp. 73–88, 3 2020.

[32] P. Runeson, E. Engström, and M.-A. Storey, "The Design Science Paradigm as a Frame for Empirical Software Engineering," in *Contemporary Empirical Methods in Software Engineering*, pp. 127–147, Cham: Springer International Publishing, 2020.

[33] A. Hevner and S. Chatterjee, *Design Research in Information Systems*, vol. 22 of *Integrated Series in Information Systems*. Boston, MA: Springer US, 2010.

[34] A. Dresch, D. P. Lacerda, and J. A. V. Antunes, *Design science research: A method for science and technology advancement*. Springer International Publishing, 1 2015.

[35] B. Gillham, *Developing a Questionnaire*. Bloomsbury Publishing Plc, 2008.

[36] N. Salkind, "A Technique for the Measurement of Attitudes," in *Encyclopedia of Research Design*, 2455 Teller Road, Thousand Oaks California 91320 United States: SAGE Publications, Inc., 2012.

[37] I. Etikan, "Comparison of Convenience Sampling and Purposive Sampling," *American Journal of Theoretical and Applied Statistics*, vol. 5, no. 1, p. 1, 2016.

[38] J. Pries-Heje, R. Baskerville, and J. Venable, "Strategies for design science research evaluation," *16th European Conference on Information Systems, ECIS 2008*, 2008.

[39] J. R. Venable, J. Pries-heje, and R. Baskerville, "Design Science Research in Information Systems. Advances in Theory and Practice," vol. 7286, 2012.

[40] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, pp. 77–101, 1 2006.

[41] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," *IEEE Access*, vol. 6, pp. 10179–10188, 1 2018.

[42] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Real-Time Systems Series, Boston, MA: Springer Science & Business Media, 2011.

[43] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, pp. 398–461, 11 2002.

[44] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in *2016 IEEE/ACS 13th International*

Conference of Computer Systems and Applications (AICCSA), pp. 1–6, IEEE, 11 2016.

[45] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 11 2018.

[46] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 5 2018.

[47] L. Chen and A. Avizienis, "N-version programming : a fault-tolerance approach to reliability," vol. 1, pp. 3–9, 1996.

[48] L. Nagy, R. Ford, and W. Allen, "FIT Computer Science Technical Report: CS-2006-04 N-Version Programming for the Detection of Zero-day Exploits N-Version Programming for the Detection of Zero-day Exploits," tech. rep., 2006.

[49] J. C. Laprie, J. Arlat, C. Beounes, and K. Kanoun, "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," *Computer*, vol. 23, no. 7, pp. 39–51, 1990.

[50] "Hot and Cold Spares : Networking." Available at: `https://www.brainbell.com/tutorials/Networking/Hot_Spare_And_Hot_Swapping.html` (Accessed: 2021-02-18).

[51] A. Kumar and D. Malhotra, "Study of Various Proactive Fault Tolerance Techniques in Cloud Computing," *International Journal of Computer Sciences and Engineering*, vol. 06, pp. 81–87, 4 2018.

[52] A. Saleh and J. Patel, "Transient-fault analysis for retry techniques," *IEEE Transactions on Reliability*, vol. 37, no. 3, pp. 323–330, 1988.

[53] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229–252, 2008.

[54] E. Bartocci, J. Deshmukh, and A. Donz, *Lectures on Runtime Verification*, vol. 10457 of *Lecture Notes in Computer Science.* Cham: Springer International Publishing, 2018.

[55] S. Hukerikar and C. Engelmann, "Resilience Design Patterns: A Structured Approach to Resilience at Extreme Scale," *Supercomputing Frontiers and Innovations*, vol. 4, pp. 4–42, 8 2017.

[56] M. Segovia, A. R. Cavalli, N. Cuppens, and J. Garcia-Alfaro, "A Study on Mitigation Techniques for SCADA-Driven Cyber-Physical Systems (Position Paper)," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11358 LNCS, pp. 257–264, 2019.

[57] "Limp Mode: Causes and what to do about it." Available at: `https://tinyurl.com/3pjbfh44` (Accessed: 2021-02-06).

[58] M. A. Mukwevho and T. Celik, "Toward a Smart Cloud: A Review of Fault-tolerance Methods in Cloud Systems," *IEEE Transactions on Services Computing*, pp. 1–1, 3 2018.

[59] V. Slåtten, P. Herrmann, and F. A. Kraemer, "Model-Driven Engineering of Reliable Fault-Tolerant Systems—A State-of-the-Art Survey," in *Advances in Computers*, vol. 91, pp. 119–205, 2013.

[60] "Computer Network Architecture - javatpoint." Available at: https://www.javatpoint.com/computer-network-architecture (Accessed: 2021-05-26).

[61] P. Kisembe and W. Jeberson, "Future of Peer-To-Peer Technology with the Rise of Cloud Computing," *International Journal of Peer to Peer Networks*, vol. 8, pp. 45–54, 8 2017.

[62] "Difference between client server and peer to peer - propatel." Available at: https://www.propatel.com/peer-to-peer-and-client-server/ (Accessed: 2021-03-17).

[63] "Understanding the differences between client/server and peer-to-peer networks - TechRepublic." Available at: https://tinyurl.com/65puxdyh (Accessed: 2021-03-17).

[64] G. Schermann, J. Cito, and P. Leitner, "Continuous experimentation: Challenges, implementation techniques, and current research," *IEEE Software*, vol. 35, pp. 26–31, 3 2018.

[65] "Feature toggles (aka feature flags)." Available at: https://martinfowler.com/articles/feature-toggles.html (Accessed: 2021-03-17).

[66] M. T. Rahman, L. P. Querel, P. C. Rigby, and B. Adams, "Feature toggles: Practitioner practices and a case study," *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, pp. 201–211, 2016.

[67] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly: Management Information Systems*, vol. 28, no. 1, pp. 75–105, 2004.

[68] "Functional safety | RISE." Available at: https://www.ri.se/en/what-we-do/expertises/functional-safety (Accessed: 2021-05-17).

A

Appendix 1 - Questionnaire

A.1 Questions about the safety of the framework

1. What are your first and last names?
2. What is your highest educational degree?
   (a) B.Sc.
   (b) M.Sc.
   (c) P.hD.
   (d) Other (insert text)
3. What is the domain of your highest educational degree?
   (a) Computer Science and Engineering
   (b) Software Engineering
   (c) Networks and Systems
   (d) Other (insert text)
4. What is the name of the organization where you currently study or work in?
5. What is your current occupation?
   Description: e.g., security expert, system architect, researcher in security, PhD student in computer science.
6. How many months of working experience do you have in systems security and safety?
7. How do you rate your expertise in systems security and safety?
   (a) 1 very low
   (b) 2 low
   (c) 3 average
   (d) 4 high
   (e) 5 very high
8. How do you rate the safety of the developed collaborative attack response system?
   (a) 1 extremely unsafe
   (b) 2 slightly unsafe
   (c) 3 neither unsafe or safe
   (d) 4 slightly safe
   (e) 5 very safe
9. How safe is it to make decisions based on automated experiments with almost no human interactions? Please elaborate.
   Description: automated experiments are performed to evaluate the response techniques using the workshop cars.

10. How do you rate the severity of "trustworthiness" of the collaborating vehicles as a safety concern when using the developed collaborative attack response system in practice?
    Description: trustworthiness: how much can the on-road car (the car that requests the evaluation) trust the results of experiments conducted on the workshop cars (the cars being experimented on).
    (a) 1 insignificant
    (b) 2 minor
    (c) 3 moderate
    (d) 4 major
    (e) 5 severe
11. In your opinion, how can the safety concern related to "trustworthiness" be addressed?
12. How do you rate the severity of "qualification" of the collaborating vehicles as a safety concern when using the developed collaborative attack response system in practice?
    Description: qualification: how qualified are the workshop cars to be experimented on.
    (a) 1 insignificant
    (b) 2 minor
    (c) 3 moderate
    (d) 4 major
    (e) 5 severe
13. In your opinion, how can the safety concern related to "qualification" be addressed?
14. In your opinion, what other aspects can be considered as safety concerns when using the developed collaborative attack response system in practice?
    Description: Please mention the concern, rate its severity (1 to 5), and elaborate (if possible) on how to address it.
15. Any other comments that you would like to mention?