



UniBa

UNIVERSITÀ
DEGLI STUDI
DI BARI
ALDO MORO

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

COMPUTER SCIENCE DEPARTMENT

PHD PROGRAMME IN COMPUTER SCIENCE AND MATHEMATICS

XXXVII CICLO

SCIENTIFIC DISCIPLINARY AREA IINF-05/A

SECURITY MANAGEMENT IN AUTOMOTIVE ENVIRONMENT

Coordinator:

Prof. Francesca MAZZIA

Firmato digitalmente da: Francesca Mazzia
Organizzazione: UNIVERSITA' DEGLI STUDI DI BARI/80002170720
Limitazioni d'uso: Esclusivo. Il titolare fa uso del presente certificato solo per
le finalità di lavoro e non può utilizzarlo per altri scopi. Il titolare deve usare
il certificato solo per le finalità per le quali è stato emesso.
Data: 30/01/2025 13:36:37

Tutor:

Prof. Danilo CAIVA



PhD Candidate:

Mirko DE VINCENTIIS

Firmato digitalmente da:

CAIVANO DANILO

Firmato il 30/01/2025 13:52

Seriale Certificato: 2034561

Valido dal 27/12/2022 al 27/12/2025

InfoCamere Qualified Electronic Signature CA

Co-Tutor:

Dr. Vita Santa BARLETTA



**Digitally signed
by BARLETTA
VITA SANTA
C=IT**

FINAL EXAM 2025

Abstract

In the automotive industry, Electronic Control Units (ECUs) equipped with internal communication protocols have revolutionized vehicle functionality, allowing for the management of various operations, starting from basic tasks such as controlling windshield wipers to more intricate safety-related functions. However, an advanced system such as this also introduces vulnerabilities to cyber-attacks. Such attacks often exploit software weaknesses within the communication protocol components the ECUs utilize. Recent research has also underscored the susceptibility of modern vehicles to these cyber threats.

It is crucial to note that the methodologies and techniques proposed by automotive researchers primarily address only one of the three essential phases of cybersecurity: Detection, Response, and Prevention, highlighting the necessity for developing new techniques and methodologies that encompass these phases and enhance the security of modern vehicles. The literature review in this regard discovered diverse critical concerns. Firstly, there is a need to compare identification techniques between traditional and quantum algorithms regarding detection time and model accuracy. Secondly, there is a notable absence of comprehensive risk assessment methodologies utilizing international standards or reports. Consequently, the absence of comprehensive methodologies or frameworks for managing cybersecurity in the automotive context renders this imperative.

In response to these critical issues, this thesis introduces ANDURIL (Automotive Network Defense Unified Response Intrusion Limitation), a model distinguished by five dimensions and three operational units. Within each dimension, various techniques and methodologies have been proposed, considering the three phases of cybersecurity: Detection, Response, and Prevention. This approach enables automotive organizations to select one or more techniques from each domain to carry out detection, response, and prevention activities. These techniques involve the use of Machine Learning (ML) algorithms, a novel approach based on the Security Operation Center (SOC), and the use of Quantum ML algorithm to identify threats on the Controller Area Network (CAN) bus. In the response and prevention phase, the new approaches include a pattern designed to assess the risk of attack and propose security controls and a web application to support automotive companies in the event of an attack.

However, it is important to note that the experiments conducted so far have been *in vitro*, and the proposed techniques need to be validated on real-world

automotive embedded components (in vivo). Moreover, validating such techniques with automotive companies is a critical area that needs future exploration.

Contents

Abstract	i
List of Figures	vi
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.2 Main contribution	2
1.3 Thesis Outline	4
2 Background and Threat Model	5
2.1 In-vehicle Network	5
2.1.1 Controller Area Network	6
2.1.1.1 Data Frame	8
2.1.1.2 Remote Frame	9
2.1.1.3 Error Frame	9
2.1.1.4 Overload Frame	10
2.1.2 Local Interconnect Network	10
2.1.2.1 LIN Network Topology	11
2.1.2.2 LIN Frame Format	11
2.1.3 Media Oriented System Transport	12
2.1.3.1 MOST Network Topology	13
2.1.3.2 MOST Frame Format	13
2.1.4 FlexRay	14
2.1.4.1 FlexRay Frame Format	14
2.2 Automotive Standards	16
2.3 Attack Model	17
2.3.1 Eavesdropping Attack	20
2.3.2 Denial of Service Attack	20
2.3.3 Fuzzing Attack	21
2.3.4 Replay Attack	21

3	Related Work	22
3.1	Security Solutions based on the IT	22
3.2	Security Solutions for In-Vehicle Networks	23
3.2.1	Intrusion Detection System	23
3.2.1.1	Classical Intrusion Detect System	24
3.2.1.2	Quantum Intrusion Detection System	25
3.2.2	Encryption and Authentication Security Measures	26
3.3	Security Solutions of Automotive Industry	26
3.4	Limitation and Improvement	27
4	Proposed cybersecurity model: ANDURIL	30
4.1	Detection	33
4.2	Response	35
4.3	Prevention	36
5	Detection techniques in ANDURIL	38
5.1	Datasets description	39
5.1.1	Car-Hacking Dataset	40
5.1.2	Survival Analysis Dataset	41
5.1.3	SA-Hacking Dataset	42
5.1.4	TON_IoT	45
5.1.5	CICIoT2023	45
5.2	Detection techniques using SOC	46
5.2.1	Experimental Results	50
5.2.1.1	DoS Attack	50
5.2.1.2	Fuzzing Attack	53
5.2.2	Comparison with the state-of-the-art	56
5.3	Detection techniques using QML Algorithms	57
5.3.1	Binary classifiers	57
5.3.2	Multi-class classifiers	63
5.3.3	Comparison with the state-of-the-art	65
5.4	Detection techniques using ML algorithms	66
5.4.1	MaREA IDS	66
5.4.1.1	Experimental phase	67
5.4.1.2	Evaluation results	69
5.4.2	Cybersecurity educational framework	69
5.4.2.1	Proposed methodology	70
5.4.2.2	Multi-class IDS solution	72
5.4.3	Cybersecurity technology transfer	73
5.4.3.1	Methodology	73
5.4.4	Comparison with the state-of-the-art	76
5.5	Detection technique using a Hybrid approach	77
5.5.1	Smart City Data Source	77

5.5.2	Smart City Security Dashboard	78
5.5.3	Principal Application	79
5.5.4	Database	81
5.5.5	Quantum Cloud services	81
5.5.6	QRadar - SIEM	81
5.5.7	Experiments	81
5.5.7.1	QBoost validation	82
5.5.7.2	IBM QRadar SIEM validation	84
5.5.7.3	Evaluation results	88
5.5.8	Comparison with the state-of-the-art	89
6	Response techniques in ANDURIL	90
6.1	Auto-CyberKC	91
6.2	Response techniques combining V-SOC	95
6.3	Response techniques combining IDS algorithms	100
6.3.1	Background to NPL	100
6.3.2	Methodology	101
6.4	Comparison with the state-of-the-art	103
7	Prevention techniques in ANDURIL	105
7.1	MARISMA-CAR	107
7.1.1	Automotive and Smart Car Standards for MARISMA-CAR	108
7.1.2	Cybersecurity Engineering Process for Automotive and Smart Cars	109
7.1.3	Domain, objective and controls	110
7.1.4	Type of assets	111
7.1.5	Dimensions	112
7.1.6	Family of threats and types of assets	113
7.1.7	Objective-domain-threats matrix	114
7.1.8	T-assets/T-threats/dimensions matrix	115
7.2	ICSEQAI: Industrial Cyber Security Education with Quantum Ar- tificial Intelligence	116
7.3	Comparison with the state-of-the-art	118
8	ANDURIL Case Study	119
8.1	Pachamama Methodology	120
9	Conclusion, Limitation, and Future Work	126
9.1	Conclusion	126
9.2	Limitation	127
9.3	Future Work	128
	Bibliography	130

List of Figures

2.1	Arbitration mechanism that involves three nodes. Node 1 wins the arbitration.	7
2.2	Data Frame comparison. On the top, 2.0 A, and on the bottom, 2.0 B.	8
2.3	CAN error handling.	10
2.4	LIN master-slave model architecture with one master node and two slave nodes [1].	11
2.5	MOST ring topology [2].	13
2.6	Structure of MOST frame [3].	13
2.7	FlexRay frame and communication cycle [4].	15
2.8	FlexRay frame format [4].	15
2.9	Example of attack surfaces in a modern car.	18
4.1	The Hack-Space organizational functions [5].	31
4.2	Proposed life cycle in which there are the operational units (principles) horizontally and vertically the domains.	31
4.3	Techniques proposed for the detection phase.	35
4.4	Techniques proposed for the response phase.	36
4.5	Techniques proposed for the prevention phase.	37
5.1	Proposed techniques regarding the detection phase considering the <i>dimensions</i> (vertical) and the <i>operational units</i> (horizontal). . . .	39
5.2	Syntax of the message and signal used in the DBC file.	43
5.3	V-SOC4AS architecture.	46
5.4	Messages generated by ICSim and transformed into JSON format by the TCU component.	48
5.5	The figure on the top shows the CAN messages generated by ICSim that are sent on the vcan0. Instead, the figure on the bottom shows the messages generated by ICSim and transformed into JSON format by the TCU component.	49
5.6	Result after creating all properties. On the console, the payload can be seen as correctly formatted.	50
5.7	DoS rule in QRadar.	51

5.8	Rule Wizard with the field used to identify the DoS attack.	51
5.9	Example of JSON messages that correspond to DoS attack.	52
5.10	Generated offenses when a DoS attack is detected.	53
5.11	Generated offenses when a Fuzzing attack is detected.	53
5.12	Fuzzing rule in QRadar.	54
5.13	The actions chosen for the fuzzing attack.	55
5.14	Generated offenses when a fuzzing attack is detected. (a) Shows the offense regarding the vehicle model OpelAstra with five identified attacks (red rectangle). In (b), the figure refers to the RenaultClio with five identified attacks.	56
5.15	Confusion matrix for SVC and QBoost on DoS & Fuzzy datasets. "0" malicious; "1" normal messages. TN-True Negative, FP-False Positive, FN-False Negative, TP-True Positive.	61
5.16	The figure on the top shows the quality of the ML models. Instead, the figure on the bottom shows the time performance.	62
5.17	An overview of the proposed methodology.	67
5.18	Goal and Strategy of Cyber Security Education in Automotive Industry.	70
5.19	Adaptation of NICE framework.	71
5.20	Proposed methodology for Cyber Security education.	71
5.21	Automotive Technology Transfer.	73
5.22	Methodology phase for the Technology Transfer.	74
5.23	Laboratory setup for Technology Transfer.	75
5.24	Quantum Hybrid Architecture of Smart City Security.	78
5.25	Integrated Smart City Security Dashboard.	79
5.26	IDS Pipeline.	80
5.27	The pictures show the rules to identify an HTTP Flood attack with the category DDoS. Figure (a) indicates the main rule. If one of the rules indicated in (a) is positive, it will raise an offense. Figures (c) and (d) show the BB rules.	84
5.28	Offense when a DDoS attack is identified.	85
5.29	Rule to identify an XSS attack.	85
5.30	Figure (a) shows information about the offense when an XSS attack is detected. Figure (b) shows more information about the offense.	86
5.31	Rule to identify a SQL Injection attack.	86
5.32	SQL Injection Offense generated.	86
5.33	SQL Injection with more detail obtaining the destination IPs involved in the attack.	87
5.34	The pictures show the rules to identify a port scan. Figure (a) indicates the main rule. If one of the rules indicated in (a) is positive, it will raise an offense. Figure (c) indicates a BB rule.	87
5.35	Quality metrics of Classifiers.	88

6.1	Proposed techniques regarding the response phase considering the <i>dimensions</i> (vertical) and the <i>operational units</i> (horizontal). . . .	91
6.2	Figure (a) shows the single step of the knowledge graph. Instead, figure (b) represents the graph for the multi-stage attack path. In red is the root node.	93
6.3	Auto-CyberKC Knowledge Base for attack path with the ID four that contains a multi-stage attack.	94
6.4	Auto-CyberKC Knowledge Base.	94
6.5	Communication between V-SOC and Auto-CyberKC.	95
6.6	Auto-CyberKC filtered attack path for the vehicle model Ateca and Bad Content.	96
6.7	Auto-CyberKC Dashboard.	96
6.8	Overview of proposed model to respond to an attack using a mobile application.	98
6.9	Mobile app: on the left the real-time data tab, on the right the attacks tab.	99
6.10	The top three sentences that are similar to the description obtained from the taxonomy. CWE and CWE descriptions have been omitted for space reasons from the last sentence. Similar words can be seen underlined in yellow and green.	102
6.11	A graphical improvement of Figure 6.10. The figure shows an example of a suggestion regarding the DoS attack.	103
7.1	Proposed techniques regarding the prevention phase considering the <i>dimensions</i> (vertical) and the <i>operational units</i> (horizontal). .	105
7.2	Components of MARISMA-CAR pattern.	107
7.3	Overview of MARISMA-CAR pattern mapping according to the NIST Cybersecurity Framework.	109
7.4	ICSEQAI Vision Model to support Developers and Testers during SDLC.	117
8.1	ANDURIL considering the horizontal dimension (Operational Units) by taking techniques and methodologies from the vertical dimension (Dimensions).	119
8.2	An overview of the Pachamama methodology phase.	120
8.3	Example of DoS attack processed according to the Pachamama methodology.	121
8.4	Log Activity in IBM QRadar after the SIEM receives messages from the IDS component.	121
8.5	Simulation process in which the Arduino Uno (a malicious ECU) sends malicious CAN data frame, and MaREA (deployed on a Raspberry Pi 4) identifies if there are attacks.	122
8.6	Generated DoS offense.	122

8.7	Interaction between the Auto-CyberKC and IBM QRadar SIEM using REST API.	123
8.8	The figure above (a) shows a portion of the categories taken from the AAD and suggested. The figure below shows one of the top three similar sentences obtained with the Sentence Transformer. In the red rectangle, the <i>Description</i> extracted from the AAD. In the yellow rectangle, the information obtained from the NVD. . .	124

List of Tables

3.1	Related works mapped according to the <i>domains</i> and <i>operational units</i> . The last row, proposed , shows the works mapped into the three cybersecurity phases.	29
4.1	Related works mapped according to the <i>domains</i> and <i>operational units</i> . The proposed techniques and methodologies are in bold. . .	33
5.1	Overview of the Car-Hacking Dataset for Intrusion Detection. . .	41
5.2	Overview of the Survival Analysis Dataset.	41
5.3	Number of normal and fabrication messages contained in the SA-Hacking Dataset.	44
5.4	Results obtained with QBoost.	58
5.5	Comparison of the results between SVC and QBoost.	58
5.6	Classification performance of binary classifiers: Train & Test sets for DoS & Fuzzy dataset. Bold shows best performance.	60
5.7	Average time performance of the Binary classifiers on DoS and Fuzzy datasets. Bold shows best performance.	61
5.8	Number of messages in the QUANTICAR Dataset.	64
5.9	Results obtained with QBoost in a multi-class classification considering the QUANTICAR Dataset.	65
5.10	Car-Hacking Dataset after the under-sampling.	68
5.11	Concatenated dataset (Survival Analysis Dataset and SA-Hacking Dataset) after the under-sampling.	68
5.12	Hyperparameters considered and the values selected by the grid search for MaREA considering the Car-Hacking and Concatenated dataset.	69
5.13	Comparison with other approaches that use Random Forest and the Car-Hacking Dataset as a testbed.	69
5.14	Comparison with a multi-class approach using the Survival Analysis Dataset as a testbed. The fabrication attack is reported only for MaREA. The precision score is not included in [6].	70
5.15	Results obtained using the Random Forest classifier for the concatenated dataset in multi-class approach.	72

5.16	Results obtained with Random Forest and Arduino Elegoo Uno R3 (Injected Messages).	76
5.17	Time performance analysis between QBoost and Random Forest. .	88
7.1	Domains, objective, and controls of MARISMA-CAR pattern. . .	111
7.3	Matrix objective-domain-family of threats of MARISMA-CAR pat- tern.	114
7.4	Matrix of T-Assets/T-Threats/Dimensions for MARISMA-CAR pat- tern.	116

Chapter 1

Introduction

1.1 Motivation

Years of research and development of electronic components have enabled engineers to develop embedded systems for diverse applications. One of the most prominent applications of these components is the automotive sector. Electronic Control Units (ECUs) are inherently embedded components that control mechanical parts programmed to perform several tasks in the automotive system. These tasks vary from triggering simple vehicle features such as windshield wipers or power windows through driver assistance to automated tasks such as stability control or anti-collision systems, the latter two enabling safety mechanisms to protect drivers from accidents or make driving easier and safer.

With the advent of artificial intelligence systems, ECUs are gradually turning into small computers capable of running artificial intelligence algorithms to generate decisions based on environmental input. In addition, it is common for modern vehicles to integrate Bluetooth and Internet connectivity.

However, integrating sophisticated connectivity gateways like Bluetooth and Internet into vehicles has inadvertently opened avenues for cyber-attacks. Specifically, the ECUs communicate with each other utilizing standard protocols designed for reliability and electromagnetic resistance but not cyber security. Over the years, researchers have shown that such vulnerabilities can be exploited at the design level to intercept, collect, and manipulate messages exchanged by the ECUs.

Among the several solutions proposed to manage these vulnerabilities, one approach involves using anomaly detection techniques by leveraging artificial intelligence models and Machine Learning (ML) or Deep Learning (DL) algorithms to safeguard the internal network of vehicles. While these techniques facilitate the detection of attacks, developing methodologies to reconstruct the attack chain is essential for identifying the vulnerable ECU(s) and addressing the associated vulnerabilities.

In a different aspect, other researchers have proposed frameworks or method-

ologies using international standards (e.g., documents published by the International Organization for Standardization (ISO)) regarding automotive security to introduce cyber-security during the development phase. However, these documents can be very high-level and complex to comprehend.

Having said that, a critical problem in this field is generalizing vulnerabilities into well-defined and recognizable classes and deriving attack strategies, allowing cybersecurity experts to understand attack causation. The sole work discussed in the literature is an attack taxonomy presenting attack composition and decomposition to identify possible attacks that did not occur in practice [7]. Despite its utility, this work faces notable challenges as the attacks outlined in the taxonomy are now outdated, and updating the taxonomy with emerging threats requires substantial effort.

1.2 Main contribution

This thesis proposes multiple solutions to improve the communication network security of modern vehicles. It presents a new methodology to prevent and respond to cyberattacks, including using machine learning algorithms to identify attacks that can occur in the in-vehicle network. The cyberattack detection results are compared with those of state-of-the-art studies while proposing in vitro components. As mentioned earlier in this chapter, attack detection should not be the only choice or option. Therefore, a novel methodology has been proposed consisting of identifying in-vehicle attacks and reconstructing the attack path based on the information retrieved to understand how to fix the vulnerability leading to the cyberattack. Additionally, another methodology has been proposed to address the challenge of identifying new vulnerabilities in the automotive field by utilizing Natural Language Processing (NLP) techniques. This methodology has been validated by developing a web interface that can support developers or cyber-security experts in understanding the risk of an attack and how to fix it.

These contributions are based on two goals:

1. **Goal 1:** Analyze the attacks that can be conducted on in-vehicle and out-vehicle with the purpose to evaluate the security of critical components.
2. **Goal 2:** Investigate, implement, and evaluate new methodologies to increase security and robustness of vehicles.

In particular, the following research questions are extracted from the *Goal 1*:

- **RQ1.1:** What are the standard protocols used in modern vehicles?
- **RQ1.2:** What are, at the state-of-the-art, the attacks that can be conducted inside the vehicle?

- **RQ1.3:** What are, at the state-of-the-art, the attacks that can be conducted outside the vehicle?
- **RQ1.4:** What techniques are researchers using in conducting automotive attacks?
- **RQ1.5:** What methodologies currently exist to ensure the vehicle's safety?

Instead, to address the *Goal 2*, the following research questions are considered:

- **RQ2.1:** What dimensions should be considered to ensure vehicle safety throughout their life cycles?
- **RQ2.2:** What operational units should be considered to ensure vehicle safety throughout their life cycles?
- **RQ2.3:** Do current methodologies in the automotive context adequately cover the entire Detection-Response-Prevention cycle?
- **RQ2.4:** Can Detection techniques interact across different dimensions to improve attack identification?
- **RQ2.5:** Can Response techniques work across various dimensions to minimize response damage from attacks?
- **RQ2.6:** Can Prevention techniques interact across multiple dimensions to mitigate attacks on vehicles?

Based on the goals and the research questions, this thesis contributes by proposing several techniques and methodologies divided into Automotive Functional Dimensions to perform *Detection*, *Response*, and *Prevention* phases (Operational Units) to create an Automotive Cybersecurity Life Cycle. The techniques and methodologies can be summarized as per the following phases:

- **Detection:** Proposes a V-SOC to identify in-vehicle attacks. A cybersecurity educational framework was developed to enhance automotive risk awareness by educating and training industry resources by implementing Intrusion Detection System (IDS) algorithms utilizing Quantum Machine Learning and traditional Machine Learning techniques for binary and multi-class classification. Based on this, a traditional Machine Learning algorithm was tested using a dedicated testbed to facilitate the transfer of this technology to the automotive industry. Finally, a hybrid classical architecture using Quantum Machine Learning algorithms and an SIEM was proposed to boost Smart City security.

- **Response:** Combines the proposed V-SOC with a web application called Auto-CyberKC to present a response phase using automotive standards to help cybersecurity analysts understand automotive attacks. In addition, a prototype of an Android application is proposed to inform and help the driver manage any attack.
- **Prevention:** Proposes an automotive pattern called MARISMA-CAR to manage risks during the automotive design phase using automotive standards and reports. Eventually, a vision model called Industrial Cyber Security Education with Quantum Artificial Intelligence (ICSEQAI) has been proposed as a high level of abstraction of a learning framework to support developers and testers during the Software Development Life Cycle (SDLC).

1.3 Thesis Outline

The following thesis is organized as follows:

- **Chapter 2:** discusses the background related to the protocol used for the in-vehicle networks and the threat regarding the automotive context.
- **Chapter 3:** shows the state-of-the-art countermeasures to identify, respond, and protect against automotive threats.
- **Chapter 4:** discusses the automotive cybersecurity model used as a reference to create a cyber-kill chain for the automotive sector.
- **Chapter 5:** discusses the detection techniques first analyzing the used datasets, then the proposed solutions.
- **Chapter 6:** illustrates the response techniques combining the detection solutions.
- **Chapter 7:** shows the prevention techniques proposing a novel pattern for the risk assessment in the automotive context and a vision model for cybersecurity education with Quantum Artificial Intelligence.
- **Chapter 8:** illustrates the life-cycle of the automotive cybersecurity model by combining the solutions to validate the detection, response, and prevention.
- **Chapter 9:** presents the conclusions, limitations, and future works.

Chapter 2

Background and Threat Model

This chapter introduces the background related to the most used in-vehicle protocols and standards, which are fundamental to the understanding of this thesis. In particular, this chapter addresses the first goal of this research work discussed in Section 1.1 related to the state-of-the-art regarding the automotive context.

Section 2.1 provides a brief introduction to the most used protocol in modern vehicles, while the following sections describe in the details these in-vehicle protocols. Section 2.2 shows the standards used in the automotive context that refer to risk assessment and good practices in component design. Finally, Section 2.3 points out the most popular threat models proposed by the researchers.

2.1 In-vehicle Network

To answer RQ1.1: "What are the standard protocols used in modern vehicles?" an analysis of the literature was conducted to identify the most used protocols for the in-vehicle network. In this manner, it is possible to investigate the threats.

Modern vehicles are controlled by embedded components called Electronic Control Units (ECUs). The role of ECUs is to make decisions based on input received from sensors. Usually, these decisions are regulated by software and can include steering and breaking. ECUs can also manage many other less-critical features like air conditioning and infotainment systems. The embedded components are physically connected to the vehicle and communicated between them using specialized protocols that must satisfy safety, security, real-time capabilities, and message prioritization. In a classic in-vehicle network, the ECUs can be connected using a single network in which there are critical components that exchange messages with less-critical components like infotainment system ECUs. Other in-vehicle architectures are more complex and divide the safety-critical from the less-critical ECUs to increase security. For example, a typical architecture contains a gateway ECU whose role is to handle translation between the different sub-networks and ensure smooth, safe, and correct communication between all

ECUs.

Several protocols characterize the in-vehicle network:

- **Controller Area Network (CAN)**: one of the most used protocols for the automotive field. CAN is a broadcast protocol, and the bus is composed only of a twisted pair, thus increasing electromagnetic resilience and containing deployment costs. Since CAN is the most deployed protocol in modern and old vehicles, it is one of the most targeted for cyber-attacks [8, 9]. Since the CAN protocol is the preliminary focus of this thesis, Section 2.3 describes in detail its operation.
- **Media Oriented System Transport (MOST)**: high-speed multi-media fiber optic network used for in-vehicle multimedia networking and infotainment. Basic applications supported by MOST are audio and video transfers.
- **Local Interconnect Area (LIN)**: is used for non-critical functions inside a vehicle, such as doors module and air conditioning.
- **FlexRay**: featuring error detection and correction, redundancy, and safety, FlexRay is used for high-end applications inside vehicles, such as powertrain and safety functions like Adaptive Cruise Control.

In this thesis, only the CAN protocol has been considered because it is the most used protocol for the in-vehicle network and analyzed in the literature.

2.1.1 Controller Area Network

Controller Area Network (CAN) is a serial communication protocol that supports distributed real-time control with a very high level of protection [10]. The protocol was developed by Robert Bosch in 1980 GmbH, and it is used to connect several components such as sensors, ECUs, and controllers. In general, the CAN bus is a multi-master and message broadcast system that specifies a maximum signaling rate of 1 MBps, despite common deployment uses 500 kbps (for high-speed CAN) and 125 kbps (for low-speed CAN). The CAN messages are typically short and transmitted to the entire network, which provides for data consistency in every node of the system.

CAN specification is composed of two parts: *2.0A* is called *standard* because the identifier uses 11-bits, and *2.0B* is called *extended* because the identifier has 29-bits. To send messages, the CAN bus uses a non-destructive process of arbitration, which means that to avoid collisions, the bus uses this arbitration process. The CAN specification defines each bit sent on the network as either *dominant* (logical value 0) and *recessive* (logical value 1). If a node in the network sends a dominant bit and another node sends a recessive bit, the node that sent the dominant bit wins the arbitration and can send the message. Instead, the node that sends the recessive bit stops the transmission and attempts re-transmission of

the message after the sender concludes the current transmission. This mechanism is called "arbitration" and uses the high-priority concept where the dominant bit has higher priority than the recessive bit. Figure 2.1 shows the three nodes that send messages on the CAN bus.

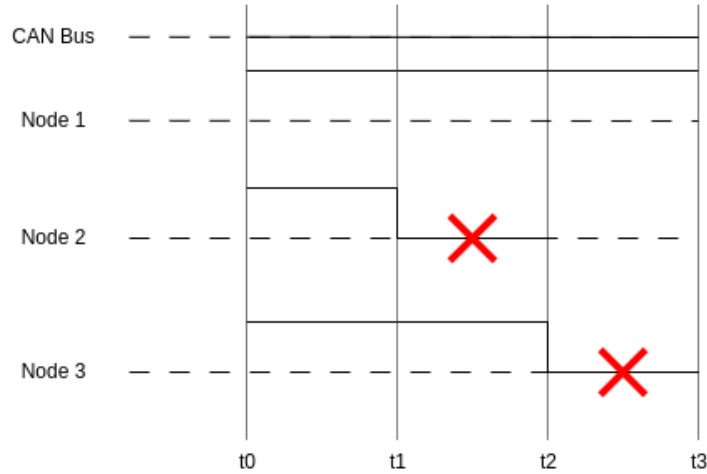


Figure 2.1: Arbitration mechanism that involves three nodes. Node 1 wins the arbitration.

At time t_0 , all the nodes send a dominant bit. At time t_1 , node 2 lost arbitration and stopped the transmission because it sent a recessive message. Instead, nodes 1 and 3 continue the transmission. At time t_2 , node 3 lost the arbitration and stopped the transmission. At this time, node 1 wins the arbitration and sends the message on the bus, while the other nodes re-attempt any transmission until node 1 has finished. Data transmission on the CAN bus is defined as a message-based communication between different microcontrollers. The CAN protocol defines four different message frames:

- **Data Frame:** the most common message type. It is used to carry the message from one node to another.
- **Error Frame:** a special message that is transmitted when a node detects an error in a message.
- **Remote Frame:** it is used to solicit the transmission of data from another node.
- **Overload Frame:** it is used to add an extra delay between two consecutive data or remote frames.

The description of the frames is provided in the following sections.

2.1.1.1 Data Frame

The *Data Frame* is the principal frame used to make an attack. This is because the Data Frame contains information about a message used by other ECUs to make decisions. Figure 2.2 shows two different structure of Data Frame used for the standard CAN (2.0 A Figure 2.2a) and extended (2.0 B Figure 2.2b).

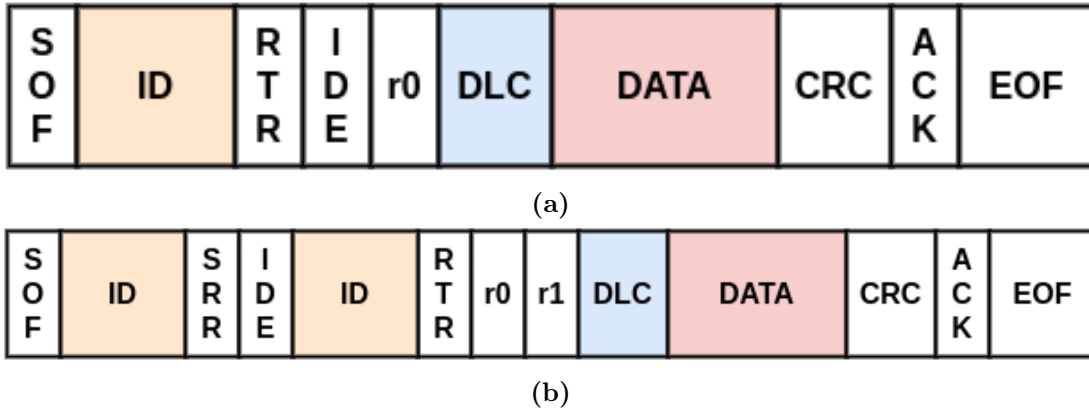


Figure 2.2: Data Frame comparison. On the top, 2.0 A, and on the bottom, 2.0 B.

The main fields composing the CAN Data Field are:

- **Start of Frame (SOF):** a single dominant bit marks the start of the message, and is used to synchronize the node on a bus after being idle.
- **Identifier (ID):** the standard 11-bit identifier is assigned to a message, and it is used to interpret the data contained in the *Data* field. For the extended CAN message, the second identifier has 18-bit encoded separately for compatibility purposes. The identifier is also used for the arbitration.
- **Remote Transmission Request (RTR):** is a single dominant bit and is used when information is required from another node. All the nodes received the data, but the identifier specified the node. In the extended format, the RTR is substituted by the **Substitute Remote Request (SRR)** and must be recessive.
- **Identifier Extension (IDE):** is a dominant bit that means that a standard CAN identifier with no extension is being transmitted. It is recessive in the extension format.
- **Reserved bit (r0):** a single reserved bit in the standard format while two reserved bits (r0 and r1) are in the extended.
- **Data Length Code (DLC):** a sequence of 4 bits that contains the length of the information encoded into the Data field.

- **Data:** information codified as a sequence of bits with a variable length (from 0 to 64 bits).
- **Cyclic Redundancy Check (CRC):** 16-bit that contains the checksum of the preceding application data for error detection. 1 bit is used for the delimiter.
- **Acknowledgment slot (ACK):** a single bit used for the acknowledgment of the message and a single bit for the delimiter. This bit is sent by the transmitter in recessive form. Every node receiving an accurate message overwrites the recessive bit with a dominant bite to indicate that an error-free message has been transmitted.
- **End-of-Field (EOF):** 7-bit that specifies the end of the frame.

2.1.1.2 Remote Frame

When a data frame is required for its functioning, the CAN remote frame is sent in broadcast and received by all nodes. The node creates a remote frame message with the ID sets with the ID of the required message. The frame of the remote frame is the same as Figure 2.2a. The difference between the data frame and the remote field is that the remote frame does not contain the data field, and the RTR in the data frame is dominant instead of the remote frame, which is recessive. The different values of the RTR field allow data frames to win arbitration in case a data frame and a remote frame with the same arbitration field are sent on the network concurrently.

2.1.1.3 Error Frame

Is a special message that is sent when a CAN message violates the framing rules. A node that detects an error in the message sends an error frame and causes all other nodes to detect a fault. The transmitter then will automatically try to resent the message. The error frame is composed of two different frames:

- **Error Flag:** 6 bits are of equal value (either dominant or recessive).
- **Error Delimiter:** 8 recessive bits used to delimit the end of the error frame.

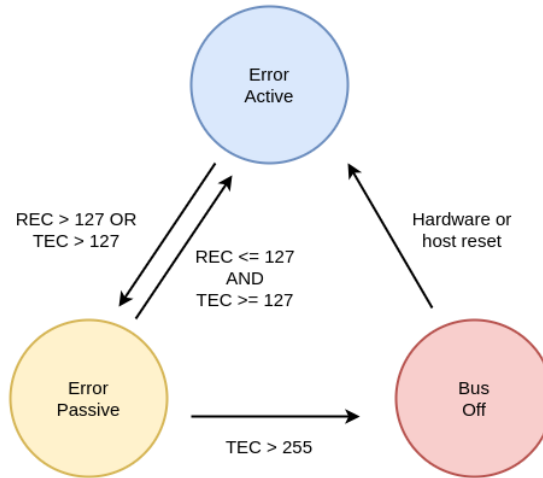


Figure 2.3: CAN error handling.

Each node on the network has two different error counters, namely **Transmit Error Counter (TEC)** or **Recessive Error Counter (REC)**, which are increased by the node on the network if an error is identified. When at least one between TEC or REC is greater than 127 and lower than 255, the node is in an error passive state, and when the error is detected, the relative error flag is sent. When both TEC and REC are smaller or equal to 127, the node is in an error active state and will send an active error flag upon error detection. If TEC is greater than 255, the node enters the *buss-off state*, which prevents the node from sending any other frame on the network.

A graphical representation of the error frame is shown in Figure 2.3.

2.1.1.4 Overload Frame

The overload frame is used when the delay between two consecutive data frames and a remote frame must be increased. It is very similar to the remote frame since all the nodes in the network receive the message and the format. It consists of two different fields:

- **Overload Flag:** 6 dominant bits, which correspond to the *active error flag*.
- **Overload Delimiter:** 8 recessive bits, used to denote the end of an overload flag.

2.1.2 Local Interconnect Network

Local Interconnect Network (LIN) protocol is used for low-cost vehicular applications that have a low bit rate and asynchronous data requirements [11]. It is usually used for controlling non-critical applications such as car comfort elements (like electric window lifts or windshield wipers), door modules, and air condition

systems [12]. LIN is used where the reliability and robustness of the network are not critical, and instead, cost-efficient communication is required. LIN bus enables a hierarchical vehicle network implementation to enhance quality and reduce cost reduction [1]. LIN can be implemented using a single wire and is commonly used as a sub-bus for CAN and FlexRay [13]. The LIN protocol has a bit rate from 1 Kbit/s to 20 Kbit/s and a single master with multiple slave nodes, with support for up to 16 nodes (without arbitration ID).

2.1.2.1 LIN Network Topology

LIN (Local Interconnect Network) uses a master-slave model, where a master node shares a bus with up to 16 slave nodes. The master device contains both a master task and a slave task, and each slave only contains a slave task. A slave can only send a message if previously asked to by the master, so the master task decides when and which frame shall be transferred on the bus [1]. A master node loops through each of the slave nodes, sending a request for information, and each slave responds with data when polled, as shown in Figure 2.4.

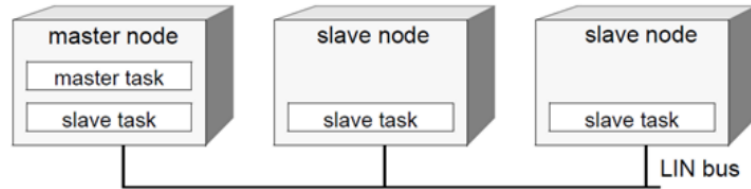


Figure 2.4: LIN master-slave model architecture with one master node and two slave nodes [1].

2.1.2.2 LIN Frame Format

The entities transferred on the LIN bus are frames, which consist of a header (provided by the master task) and a response (provided by a slave task): (i) the master node always transmits the header and consists of three distinct fields: break, synchronization (sync), and identifier (ID); (ii) a slave task transmits the response and can reside in either the master node or a slave node. It consists of a data payload and a checksum.

The master broadcasts the identifier to the network, and only one slave responds with a data payload, meaning there is zero collision risk (and hence no need for arbitration). At the start, the master task loops through each of the slave tasks by transmitting a header, which consists of a break-sync-ID sequence. Before starting the LIN, each slave task is configured to publish data to the bus or subscribe to data in response to each header ID received. So, when it receives the header, each slave task checks the ID to determine whether it needs to publish or

subscribe [22]. If it needs to publish a response, it will transmit 1 to 8 data bytes to the bus; otherwise, in the case of a subscription, it will read the data payload from the bus and take appropriate action.

The LIN frame is composed of a header and a response:

- **Header:** it is transmitted by the master task. It is composed by:
 - *Break field:* made by 13 dominant bits + 1 recessive bit, it acts as a “start of frame” indicator for all the bus’s LIN nodes.
 - *Sync field:* made by 8 bit and defined as the hex character x55, the sync field allows each of the slave nodes to synchronize with the bus.
 - *ID:* consists of two sub-fields: frame identifier 6 bits, followed by two parity bits. It identifies each message on the network and determines which nodes should receive or respond to each transmission.
- **Response:** it is transmitted when the master polls a LIN slave. It is composed by:
 - *Data field:* it contains 1 to 8 bytes of payload data bytes (data length can be 1/2/4/8 bytes).
 - *Checksum:* an 8-bit checksum field ensures the validity of the LIN frame, using a checksum algorithm to calculate its value.

2.1.3 Media Oriented System Transport

MOST (Media Oriented Systems Transport) is a multimedia plastic optical fiber (POF) network developed in 1998 and is the standard for multimedia and infotainment networking in the automobile industry (like radio, GPS navigation, and video entertainment) [13]. MOST is a synchronous network offering multiple data channels and a control channel to set up which data channels the sender and receiver will use. Data channels can be synchronous and asynchronous: synchronous channels are used to transfer streaming data, such as sound and movie data; instead, asynchronous channels are used to transfer asynchronous data and internet packet traffic [3].

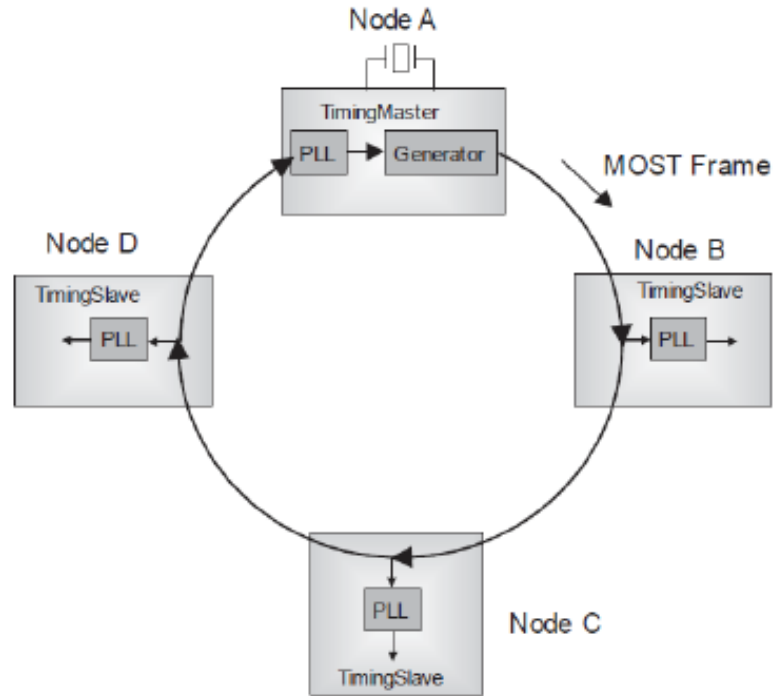


Figure 2.5: MOST ring topology [2].

2.1.3.1 MOST Network Topology

The MOST bus uses a ring topology, as shown in Figure 2.5, with one node working as a TimingMaster that generates the system clock, allowing the synchronization of all the other nodes, the TimingSlaves [2], to avoid collisions. The slave nodes will receive the message from their previous node in the ring and send it to the following one.

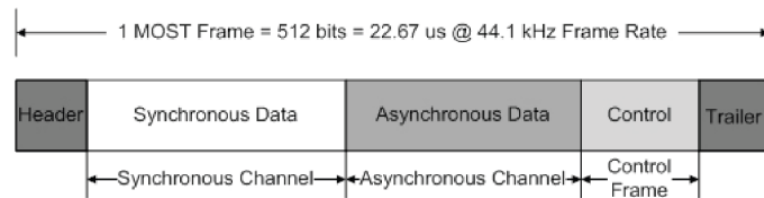


Figure 2.6: Structure of MOST frame [3].

2.1.3.2 MOST Frame Format

The MOST frame, as shown in Figure 2.6, contains one channel for the synchronous transmission of streaming data, one channel for the asynchronous trans-

mission of packet data, one channel for the transmission of control data [2], together with a header and a trailer of the frame.

Considering the most recent MOST150, it can reach 150 Mb/s and adds support to the Ethernet Packet Channel. A MOST150 frame contains:

- **Control Channel:** used for hardware control data and for turning on/off the device in the MOST network [3].
- **Synchronous Channel:** used to exchange audio-video synchronously.
- **Asynchronous Channel:** used to transmit packages asynchronously, like Internet traffic and information from the navigation system.
- **Packet Data Channel:** it can transport pure Ethernet Data in MOST150, together with configuration data of the navigation [2].

2.1.4 FlexRay

FlexRay is a communication protocol often used in safety-critical applications, such as electrical control of mechanical systems, like steering-by-wire or braking-by-wire [12], ensuring high data rates, fault tolerance, and operating on a time cycle. In FlexRay, message transmissions must follow a precise timing schedule [14]. FlexRay network supports redundancy and is very flexible regarding the topology; it can be configured as a bus, a star, or a multi-stars topology [13]. FlexRay was conceived to be a successor to CAN, offering a bit rate of up to 10 Mbps with half-duplex bus access, a data payload length of 254 bytes, and support for fault-tolerant mechanisms. However, this protocol is more expensive and complex than CAN or LIN, so FlexRay was not inserted in all vehicles, and CAN-FD (Controller Area Network with Flexible Data-rate) emerged as the successor of CAN instead.

2.1.4.1 FlexRay Frame Format

FlexRay frames can be either time-triggered (static frames) or event-triggered (dynamic frames) [14]. Time-triggered frames provide efficiency and determinism but at the price of flexibility. Instead, event-triggered frames are used to respond to stimuli from the outside [13]. As shown in Figure 2.7, FlexRay defines a communication cycle as the concatenation of time-triggered static segments and event-triggered dynamic segments [13], adding a Network Idle Time (NIT) at the end of each communication cycle to maintain synchronization.

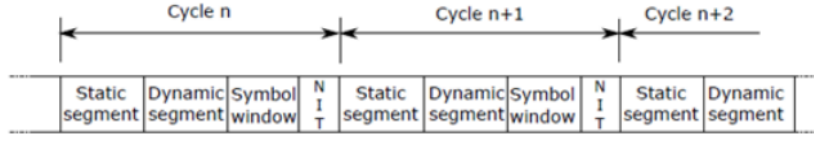


Figure 2.7: FlexRay frame and communication cycle [4].

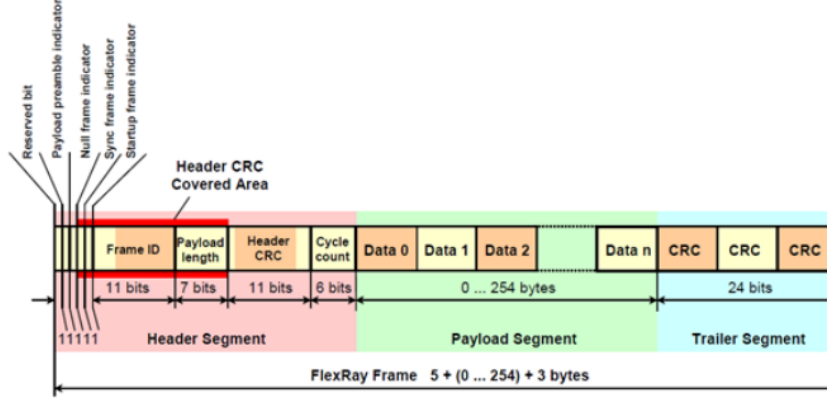


Figure 2.8: FlexRay frame format [4].

A FlexRay frame consists of three segments: header, payload, and trailer segment, as seen in Figure 2.8:

- **Header:** the FlexRay header segment consists of 5 bytes and can be divided into:
 - *Reserved bit* (1 bit): the reserved bit is reserved for future protocol use.
 - *Payload preamble indicator* (1 bit): the payload preamble indicator indicates whether or not an optional vector is contained within the payload segment of the frame transmitted.
 - *Null frame indicator* (1 bit): The null frame indicator indicates whether or not the frame is a null frame, for example, a frame that contains no usable data in the payload segment of the frame.
 - *Sync frame indicator* (1 bit): it indicates whether or not the frame is a sync frame, for example, a frame that is utilized for system-wide synchronization of communication.
 - *Startup frame indicator* (1 bit): it indicates whether or not a frame is a startup frame, and they are used in the startup mechanism.
 - *Frame ID* (11 bits): it defines the slot in which the frame should be transmitted. A frame ID is used no more than once on each channel in a communication cycle.

- *Payload length* (7 bits): used to indicate the size of the payload segment.
 - *Header CRC* (11 bits): contains a cyclic redundancy check code (CRC) computed over the sync frame indicator, the startup frame indicator, the frame ID, and the payload length.
 - *Cycle count* (6 bits): it indicates the transmitting node's value of the cycle counter at the time of frame transmission
- **Payload:** contains 0 to 127 two-byte words of data.
 - **Trailer:** contains a single field, a 24-bit cyclic redundancy check code (CRC) for the frame computed over the header segment and the payload segment of the frame.

The node will transmit the network frame by transmitting the header segment, then the payload segment, and lastly, the trailer segment.

2.2 Automotive Standards

In August 2021, the International Organization for Standardization (ISO) published the ISO/SAE 21434 "Road vehicles - Cyber security engineering" [15]. The document addresses the cyber security perspective in engineering electrical and electronic (E/E) systems within road vehicles. This standard will be mandatory for the next vehicles. It covers every stage of the vehicle life cycle, from design, secure engineering, and product development to decommissioning. Cybersecurity engineering applies to connected vehicles, electronic systems, software, etc. In addition, the automotive security standard gives developers a thorough road map for incorporating safety measures in development processes across the supply chain. This ISO aims to reduce potential attack threats and minimize potential losses by designing security from the start. It is widely adopted across the automotive industry and is aligned with regulatory requirements like UNECE WP.29 [16]. Adherence to this standard helps ensure that cybersecurity is considered throughout the vehicle development process. On the other hand, UNECE WP.29 requires vehicle manufacturers to implement a cybersecurity management system (CSMS) and demonstrate compliance to gain vehicle-type approval. UNECE WP.29 effectively makes cybersecurity a regulatory requirement, driving widespread adoption of good practices in the automotive industry.

The ISO/SAE 21434 defines a Threat Analysis and Risk Assessment (TARA) approach as a key to identifying and mitigating attacks inside vehicles. TARA covers risk evaluation and assessment, as well as the treatment and planning of identified risks. When combined with the ISO/SAE 21434, it is important to identify assets inside the vehicles, the threats, and the associated risks in order to mitigate them. The combination of ISO/SAE 21434 and the TARA helps the

automotive industry identify assets and threats, protect the identified assets by providing strategies, and respond to attacks using proactive and reactive methodologies.

Another important standard is the AUTOSAR¹ (Automotive Open System Architecture), which provides a framework for developing automotive software architectures. Its security specifications are important because they offer standardized modules and guidelines that help manufacturers implement consistent and robust cybersecurity measures across different vehicle platforms and components.

2.3 Attack Model

The analysis conducted to answer to the RQ1.1, highlighted that the most used in-vehicle network protocol is the **CAN**. In addition, most of the solutions proposed by the researchers to avoid attacks are focused on it. The CAN was proposed to make the messages exchanged between the different ECUs fast without losing robustness and security. The main problem is that the engineers do not protect the protocol against cyber attacks because it was developed in 1980. For this reason, the protocol is exposed to real cyber-attacks. In general, automotive companies develop ECUs in order to make them send messages with a specific pool of CAN IDs. Since the CAN is a broadcast protocol, other ECUs are programmed to read the messages from this pool of IDs. When an ECU receives a broadcast message, it decides whether to elaborate or ignore it based on the pool. If an attack has access to an ECU, it can send arbitrary CAN messages because the protocol does not implement mechanisms to prevent or identify them. For this reason, researchers proposed several techniques to identify and protect the CAN bus from malicious messages. In the following, several attack mechanisms are discussed.

To answer RQ1.2 "What are, at the state-of-the-art, the attacks that can be conducted inside the vehicle?" and RQ1.3 "What are, at the state-of-the-art, the attacks that can be conducted outside the vehicle?" an analysis of the threats are been conducted. In this manner, it is possible to understand the most common attack surface to propose defense approaches. The goal of a cyber-attack is to modify the behavior of the vehicle by targeting safety-critical ECUs.

¹<https://www.autosar.org/about>

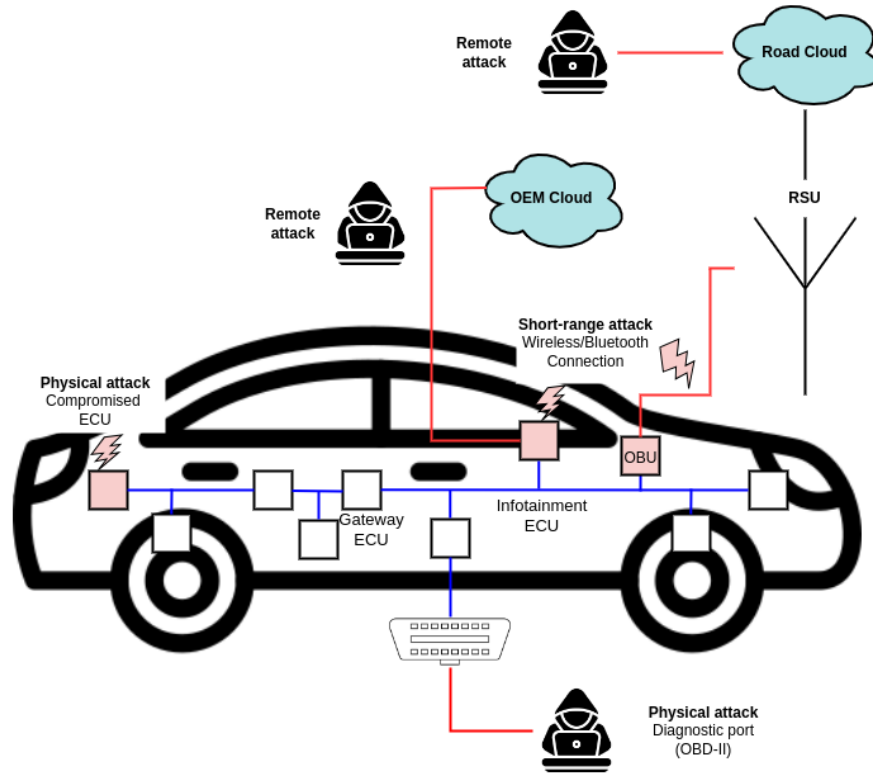


Figure 2.9: Example of attack surfaces in a modern car.

Figure 2.9 shows an example of the architecture of modern and smart cars. It is possible to see that an attacker has several entry points that he/she can use to gain access to the in-vehicle network. Based on the point of entry, there are several attack vectors that an adversarial can exploit to attack a vehicle internally (RQ1.4 "What techniques are researchers using in conducting automotive attacks?"). Among these, we have:

- **Physical attack:** in order to make this attack, an adversarial must have physical access inside the vehicle. He/She can use this attack to reverse engineer the ECUs and obtain the firmware or access to the OnBoard Diagnostic (OBD-II) port in order to collect CAN messages and carry out several attacks. For example, after collecting the data, the attacker can understand the in-vehicle network and the CAN IDs and fabricate a malicious message. Finally, physical attacks also involve damage to physical components like RSU.
- **Short-range attack:** the vehicles to communicate between them or to send

messages to RSU use short-range radio transmitters. Figure 2.9 shows that the On-Board Unit communicates with the RSU using this kind of protocol but also Bluetooth and Wi-Fi. If these components are vulnerable or do not perform any authentication and encryption mechanism, an attacker could use, for example, a smartphone to establish communication with the *Infotainment ECU* via Bluetooth or Wi-Fi and with a malicious application, he/she could access to the in-vehicle network and stole information or change the behavior of the vehicle. An attacker could also jam the communication between the OBU/RSU to send malicious messages between the vehicles in order to physically damage or prevent other vehicles from communicating [17].

- **Remote attack:** remote attacks involve the use of vulnerabilities that affect the *cloud* provider (OEM or road) in order to gain access inside the vehicle. For luxury vehicles, automotive companies release mobile applications that provide several features like opening/closing and turning on/off the car. In general, these mobile applications use the Application Programming Interface (API) to communicate with the cloud provider and execute several functionalities. If the communication between the smartphone and the API is not correctly implemented, an attacker could, for example, impersonate a vehicle's user, open the vehicle, and steal it. According to Upstream [18], in 2023, the remote attacks were the most performed attack (95%) compared with the physical attack (5%).

Once the attacker has exploited an attack surface and obtained internal access from the vehicle network, it is possible to define two grades of compromised ECUs: fully and weakly compromised ECUs [19].

In particular, Cho and Shin [19] defined these two types of attackers because the probability of disrupting the behavior of the vehicle depends on several aspects, such as ECU hardware and software, network topology, and so on. In a weakly compromised ECU, the attacker can interrupt/suspend the transmission of particular messages from the ECU or listen to the messages but cannot inject any fabricated messages. In a fully-compromised ECU, on the other hand, the attacker has full control of the component and/or access to memory data. Researchers have shown many types of vectors through which it is possible to conduct attacks, including attacks on the CAN protocol [20], [7],

As mentioned before, the CAN protocol vulnerabilities are mainly due to the fact that the CAN bus lacks encryption and authentication mechanisms [20]. Given the characteristics of the CAN network, malicious actors can perform attacks such as *eavesdropping*, *DoS*, *fuzzing*, and *replay* against CAN communications.

2.3.1 Eavesdropping Attack

Eavesdropping attacks are passive attacks that occur when unauthorized users are able to gain access to vehicular messages. This happens because the CAN protocol does not implement encryption mechanisms to avoid an adversarial to read legitimate CAN IDs [21]. For example, Enev et al. [22] show that it is possible to identify the behavior of the driver based on the sensory data travel through the CAN bus using the OBD-II port. The authors were able to identify 15 drivers with 100% accuracy. Hoppe and Dittman [23] implemented a frame sniffing in a simulated environment of a simplified in-vehicle: they controlled a malicious ECU that was deployed on the low-speed CAN bus to obtain the information transmitted from the high-speed CAN bus.

Eavesdropping attacks can be used to carry out other attacks. As mentioned before, an attacker can exploit this attack to identify the CAN frame in order to understand the network topology and create anomalous messages. Considering this, Palanca et al. [24] captured CAN frames in order to identify the ID of the parking sensor that they planned to attack a second time using a DoS attack. Another sophisticated attack that involves this attack vector was performed in 2015 by Samy Kamkar [25], in which he created a physical component called *RollJam* that consists of recording and blocking the radio signal from the key fob. The role of RollJam is to jam the wireless communication between the key fob and receiver in the car. First, the attacker saves the signal received from the key fob and then breaks the communication in such a way it does not reach the vehicle. At this point, the car owner sends another signal to open the car. This creates a second signal that will also be recorded and blocked, but this time, the attacker replays the first code and unlocks the car. Now, the attacker knows the next code in the sequence (which has not yet expired), and he/she can use it to unlock the car in a second moment.

2.3.2 Denial of Service Attack

A Denial of Service (DoS) attack involves sending high-priority messages to prevent all other messages from being transmitted on the bus. Since the CAN protocol uses a bit-wise mechanism for the arbitration, a DoS attack can occur by sending high-priority messages, thus preventing any other nodes from receiving messages.

The researchers have proposed several types of DoS attacks. Miller and Valasek [9] demonstrated how few messages sent remotely in the form of CAN packets through the Jeep Cherokee's potential entry points (e.g., TPMS and Cellular) resulted in disabling some of the car's critical components like braking system and the steering wheel while the passenger was driving.

Mukherjee et al. [26] implemented a DoS attack on SAE J1939, a standard used in vehicles implemented on top of the CAN physical layer. They demon-

strated three DoS attacks: (i) sending too many request messages for a supported Parameter Group Number (PGN) to overload the ECU, (ii) sending manipulated False Request to Send (RTS), causing overflow of the buffer, keeping connections open via Clear to Send (CTS) messages, and (iii) occupying the whole network. In general, the DoS attack for in-vehicle networks consists of injecting high-priority messages by setting the arbitration field with all zeros.

2.3.3 Fuzzing Attack

Fuzzing attacks (known in the literature as Fuzzy attacks) in automotive networks consist of injecting forged messages in order to understand the network topology. The term fuzzy derives from software testing in which invalid, malformed, or unexpected data are sent to the software in order to identify vulnerabilities or defects. The same idea is adopted for fuzzing attacks where an adversarial could send random messages to identify the behavior of such components. Usually, an attacker uses a physical attack exploiting the OBD-II port to understand the network topology. The different typologies of fuzzing attacks can be subdivided into two categories:

- **Fuzzing on the arbitration message:** in this case, an attacker creates random CAN IDs to understand the IDs pool, and then he/she can make an injection message with the identified IDs.
- **Fuzzing on the payload:** the attacker creates a random payload using the *data* field with a specific ID to change the behavior of such components.

2.3.4 Replay Attack

Replay attacks on automotive internal networks are conducted by injecting messages based on a specific ID. First, an attacker conducts an eavesdropping attack to obtain all the identifiers and payload and then reverse engineers the payload, thus injecting messages to obtain control of a specific component.

Replay attacks can be subdivided into two types: a single message with a specific ID is sent on the bus with a particular frequency, and a sequence of messages is sent on the bus as a flow with a specific frequency.

The first type of replay attack is used by Cho and Shin [19] called *masquerading* in which the two researchers explain that to perform this attack, an adversary should have the control of two ECUs (one as stronger and the other as weak). An example of a masquerading attack is performed by Miller and Valasek [9]. They remotely stop a Jeep Cherokee by controlling the ABS mounting a masquerading attack. The second type of replay attack is reported by Cho and Shin as a fabrication attack. In this case, if an attacker has strong access to an ECU, he/she can send fabricated messages with forged ID, DLC, and data.

Chapter 3

Related Work

This Chapter describes the related works on the methodologies and techniques proposed by the researchers to improve the security of modern vehicles. In particular, different categories are analyzed to improve:

- **Security Solutions based on the IT:** this category regards methodologies that use a centralized function called Security Operation Center (SOC) that is consolidated in the Information Technology (IT) area but not in the automotive field. This is important for identifying in-vehicle attacks and understanding and developing prevention and response techniques regarding automotive threats.
- **Security Solutions for In-Vehicle Networks:** this category regards solutions for Intrusion Detection Systems (IDSs), cryptographically, and authentication techniques used to identify CAN threats or to make the protocol secure.
- **Security Solutions for Automotive Industry:** this category regards methodologies proposed to help the automotive industry categorize attacks to understand them.

Finally, the related work chapter answers RQ1.5 "What methodologies currently exist to ensure the vehicle's safety?", which points out the categories explained before.

3.1 Security Solutions based on the IT

IT companies usually use the Security Operation Center (SOC) to identify and respond to threats. There is a gap regarding the proposition of a SOC in the automotive field because the analyzed works focussed on attack identification using standard techniques like Machine Learning (ML) or Deep Learning (DL)

algorithms. The problem with these algorithms is that they require high computational power, making it challenging for the Original Equipment Manufacturer (OEM) to deploy them into an ECU. The implementation of a SOC has several advantages. For example, it helps to understand the identified attack and to reconstruct the attack kill chain.

As mentioned before, there is a lack of work on the proposition of methodology that exploits the SOC approach. Langer et al. [27] proposed an automotive cyber defense center aimed at creating a SOC to analyze and react to incidents. Meyer et al. [28] presented a demo of a security infrastructure that uses a software-defined network (SDN), anomaly detection (AD), and a cyber defense center. The researchers built a prototype of a Seat Ateca consisting of ECUs grouped into five functional domains with one CAN bus per domain. However, a limitation of this work is that the authors presented a demo without a real implementation. Considering the work of [27] and [28], the researchers proposed preliminary works that address only the detection without considering response and prevention units.

Upstream has a solution called *AutoThreat Intelligence Cyber Incident Repository*¹, that aggregates and normalizes multiple property data feeds and has a complete and contextual view of the vehicle owing to digital twin profiling of the connected vehicle environment and powerful AI-powered detection modeling [29]. However, since this platform is close-source, it is difficult to assess the operability of this platform.

Considering the limitation explained before, to address the detection (Section 5.2) and response (Section 6.2) units, a V-SOC has been proposed to increase the state-of-the-art in the automotive field. The proposed solution can help the automotive industry identify CAN threats in the detection and response phases using the Security Information and Event Management (SIEM) discussed in Section 5.2.

3.2 Security Solutions for In-Vehicle Networks

This section describes the techniques found in the literature about identification techniques regarding IDSs that use standard and Quantum algorithms. Then, the section will describe authentication and encryption techniques for the CAN protocol.

3.2.1 Intrusion Detection System

Many studies have been conducted to detect attacks and propose solutions to increase the security of CAN protocol. Furthermore, Ye et al. [30] highlight the trend of increasing research interest in applying machine learning in vehicular networks as it provides a versatile set of tools to exploit and mine multiple sources of data generated.

¹<https://upstream.auto/research/automotive-cybersecurity/?id=null>

Machine learning methods can be divided into three categories:

- **Supervised Learning:** for intrusion, fault, anomaly detection applications.
- **Unsupervised Learning:** for clustering and dimension reduction tasks in congestion control, hierarchical routing, and data aggregation applications.
- **Reinforcement Learning:** instead, for policy learning tasks in resource management and routing applications.

In addition to this, there are several solutions for IDSs that can be host-based (HIDS) or network-based (NIDS) [31]. HIDS is capable of monitoring and identifying malicious activities inside an individual system. Instead, NIDS monitors the activity that comes from a network. From the point of view of the automotive sector, a HIDS requires modification of the ECU and, therefore, additional costs. Instead, a NIDS can be a simple node (such as an OBD-II dongle) that monitors the CAN bus without modifying the bus [11]. The following subsections present classical and Quantum IDS algorithms identified in the literature to detect CAN attacks.

3.2.1.1 Classical Intrusion Detect System

The concept of Intrusion Detection has been consolidated in the literature in the field of ML. Researchers use classical ML or DL algorithms, such as random forest (RF), support vector machine (SVM), neural networks, and so on, to identify attacks that can be conducted on in-vehicle networks.

Seo et al. [32] proposed a generative adversarial network (GAN)-based intrusion detection system using a deep-learning model. This model uses a generative model (G) and a discriminative model (D). The model encoded the CAN message into a one-hot vector and then transformed it into a CAN image. The authors also used a threshold to detect whether the real-time CAN message is an attack. IDSs may be limited to detecting specific attacks; if exposed, an attacker could use this information to avoid detection.

Sami et al. [33] proposed a supervised Deep Neural Network (DNN) that allows contrasting three attacks on the CAN bus: DoS, Fuzzy, and impersonation. The implementation proposed by the researchers consists of collecting CAN data from an online dataset Hacking and Countermeasure Research Lab (HCRL) [34] and real-time CAN data extracted from a Mercedes-Benz ML350 using CL2000 (an external component to log the CAN bus). The authors use these datasets to train and test a deep Feed-Forward Neural Network (FFNN) with two hidden layers.

Hanselmann et al. [35] presented an unsupervised IDS to detect attacks on the CAN bus. Their method consists of an LSTM neural network used to capture the temporal dynamics of each ID. The result is then aggregated and inserted into a fully connected sub-network with an autoencoder structure. The researchers

tested their method using SynCAN² as a dataset containing signals that can be used for unsupervised environments.

Longari et al. [36] proposed CANnolo, an unsupervised learning model consisting of an autoencoder-based LSTM, a model trained to import anomalous signals that may occur on the CAN bus, and an anomaly detector to analyze and reconstruct errors. In addition, the researchers used ReCAN [37] as a dataset whose data was obtained in a real environment.

The works analyzed before have used deep learning algorithms to identify attacks on the CAN bus. These models require high computational resources, making them difficult to implement within low-resource devices, typically microcontrollers. In addition, the researchers do not address time performance, which is a constraint for real-time detection in the automotive sector.

A different approach was proposed by Cho et al. [19] in which the researchers presented an anomaly-based IDS that measures and then exploits the interval of periodic in-vehicle messages for fingerprinting ECUs. Lokman et al. [38] have suggested deploying IDS on a gateway component inside the vehicle. Moreover, the risk of an attacker compromising the CAN network and gateway is greater than compromising the ECU.

Tian et al. [39] proposed an IDS based on the Gradient Boosting Decision Tree (GBDT) to detect CAN attacks. The authors artificially construct an entropy-based feature that considers the CAN ID and time of the data. The researchers changed the payload data values of a real dataset to create abnormal messages. Instead, in [40], the authors presented their comparison of traditional ML algorithms (k-Nearest Neighbors, Random Forest, Support Vector Machine, and Multi-Layer Perceptron) to devise an IDS to detect DoS, Fuzzy, and Impersonation CAN attack messages. The researchers tested the algorithms using the Car-Hacking Dataset.

3.2.1.2 Quantum Intrusion Detection System

In the context of Quantum Computing, there is a lack in the proposition of Quantum, or hybrid quantum algorithms, regarding the identification of the attack for in-vehicle networks. For example, Salek et al. [41] proposed a framework using a classical neural network and a quantum-restricted Boltzmann machine (RBM). The classical neural network extracts the features from CAN images generated from the CAN bus. Instead, the RBM is used to reconstruct the CAN image and determine if the traffic is normal or if an attacker inject malicious messages on the bus.

To decrease this gap, this thesis proposes several IDSs using the principle of Quantum Annealing and compares them with traditional Machine Learning algorithms considering classification metrics and training and testing time.

²<https://github.com/etas/SynCAN>

3.2.2 Encryption and Authentication Security Measures

As mentioned in Section 2.3, the CAN protocol lacks encryption and authentication mechanisms. Usually, in cybersecurity, it is possible to use symmetric or asymmetric algorithms to secure communication between two hosts. To be more specific, in this context, secure means that an unauthorized user can not access sensitive information by scrambling data, so it is only read by using a key. Instead, authentication is a standard security designed to guarantee data integrity and authenticity, usually using a piece of information called a Message Authentication Code (MAC). This means that if an attacker tries to modify the data exchanged, he/she can be detected. However, since the CAN protocol uses fixed-small packets, these solutions make implementation on the bus unfeasible.

One of the most common approaches proposed by the researchers is to authenticate the different ECUs before sending a message [42, 43, 44]. The main disadvantage of these techniques is that they require sending many messages to authenticate. The second problem is that the techniques proposed in [43, 44] may affect hardware and protocol design changes. In [45], the authors proposed a "lightweight" protocol that uses symmetric encryption to authenticate the node. The problem with this approach is that the latency is reasonable for the Automotive Ethernet and CAN high-speed but not for the CAN Low-speed. However, the Automotive Ethernet standard is not yet widely used.

3.3 Security Solutions of Automotive Industry

The vehicle life-cycle is composed of several actors that, in general, interact between them. These are:

- **Original Equipment Manufacturer (OEM):** Is the company that produce their product. In this case, it can also be called *vehicle manufacturer*.
- **Supplier:** A company that produces vehicle components. It is also providing maintenance, assistance, and replacement.
- **Owner:** A person that buy the vehicle.

Nowadays, it is necessary to introduce the concept of cybersecurity between these actors (in particular between OEMs and suppliers) to improve the security of software and hardware components. Researchers from universities and the automotive industry have proposed risk assessment methodologies and reports to improve vehicles' security, starting from the design of the components. As mentioned in Section 2.2, one of the most important International documents that address cybersecurity in the automotive is the ISO/SAE 21434. A downgrade of this standard is that it does not address practice guidelines, making it difficult

to understand. To improve the quality of the development process, Automotive SPICE (ASPICE) [46] became mandatory for automotive organizations.

Several works tried to improve the weaknesses of the ISO/SAE 21434 by proposing a life-cycle model [47] or improving the risk assessment [48, 49]. Although these approaches help fill the gaps in ISO/SAE 21434, they require much effort in identifying attacks (in the case of [48]) and understanding and identifying the assets to be protected. In [50], the authors proposed SHARA in order to merge together safety and security analysis in one approach. In addition, no counter-measures are presented once the risk assessment phase is done. For this reason, there is a need to introduce databases of vulnerabilities or platforms to obtain the last identified attacks and vulnerabilities in the automotive sector. Upstream proposed an example of a platform that contains automotive vulnerabilities to reduce this gap. However, as discussed in Section 3.1, this platform is closed.

Another platform that can be used to obtain the last automotive vulnerabilities is the National Vulnerability Database (NVD). The NVD is a centralized and authoritative source that enables the automation of vulnerability management, security measurement, and compliance [51]. This database contains the vulnerabilities identified by security researchers using a unique Common Vulnerability and Exposure (CVE) identifier, for example, CVE-2023-1234. For each vulnerability, the NVD provides several fields like a description, a score using the Common Vulnerability Scoring System (CVSS) that provides a way to capture the principal characteristics of a vulnerability and produce a numerical score [52], and the references about a Proof of Concept (PoC). However, since this database contains IT and automotive vulnerabilities, it is necessary to propose approaches to obtain only current context vulnerabilities based on the identified attack.

Finally, Sommer et al. [7] proposed a taxonomy to classify and analyze automotive attacks. The main contribution was to publish an Automotive Attack Database (AAD) consisting of 361 attacks composed of 612 attack steps. This taxonomy is subdivided into three levels of abstraction. The last two levels describe technical details about the attacks. However, as reported by the authors, mapping these attacks required high effort and time. In addition to this, the attacks present in the AAD can be old.

3.4 Limitation and Improvement

This section analyzes the limitations of the analyzed works and explains how this thesis tries to improve the state-of-the-art in the automotive context.

As shown in Section 3.1, the majority of these works have predominantly utilized deep learning algorithms for CAN bus attack identification. However, these models are resource-intensive, posing challenges for implementation on low-resource devices such as microcontrollers. Furthermore, the issue of time performance for real-time detection is often overlooked. In contrast, this thesis proposes

the use of more practical ML models like Random Forest, which requires fewer computational resources than DL algorithms, thereby enhancing feasibility. Another criticality identified in the literature is that those works propose IDS models that address only the detection. To overcome this gap, in addition to proposing IDS algorithms (with ML models and SIEM), these were also applied in a context to be used for prevention and response (see Sections 5.2, 5.5.2, and 6.3). Finally, one notable gap in the literature is the absence of proposals for testbeds to evaluate detection algorithms. Recognizing this need, this thesis introduces a testbed that simulates a CAN communication between two components (Raspberry Pi 4 and an Arduino Elegoo Uno R3). The best-performing model from the experiments was then deployed on the Raspberry Pi 4 for real-time detection testing (see Section 5.4.3).

Since the CAN messages have a maximum payload size of 8 bytes, implementing authentication and encryption mechanisms becomes difficult. This size problem limits the space available for adding cryptographic elements like message authentication codes (MACs) or encrypted data without fragmenting messages. Another challenge is securely storing cryptographic keys in resource-constrained devices and securely distributing keys across nodes without exposing them to attackers. The network lacks a built-in mechanism for key exchange or management. This problem is also due to the nature of the CAN protocol, which is a broadcast system. A message authenticated or encrypted for one node would still be received by others, introducing inefficiencies and potential attack points. For these reasons, instead of proposing authentication or encryption algorithms, several techniques are proposed in this thesis, which also aim to propose strategies to increase safety from the component design stage. For example, a risk assessment pattern was presented to suggest security controls and improve the security of the in-vehicle network. In addition to identifying attacks, the V-SOC4AS can help the security analyst detect the vulnerability and resolve the problem. In this manner, it is possible to cover the detection, response, and prevention phases, unlike the authentication and encryption algorithms that only cover prevention.

The majority of the studies analyzed in Section 3.3, have focused on risk rating methods, but the overall risk process and a concrete systematic assessment are rarely proposed. In addition, these research works do not propose security solutions validating the methodology or framework. To overcome these limits, Auto-CyberKC was proposed, which integrates the taxonomy proposed by Sommer et al. [7] and the NVD to help developers or security analysts fix vulnerabilities based on the identified attacks. In addition, a pattern called MARISMA-CAR was proposed, which considers the functions of detection, response, and prevention.

Table 3.1 shows the summarization of related works mapped based on the *detection*, *response*, and *prevention*. Most of the proposed techniques belong to all three operational units, which can support automotive security. The next chapter discusses in details the proposed model.

	Detection	Response	Prevention
IT	[27], [28]	[27]	[27]
IDS	[32], [33], [34], [35], [36], [19], [39], [40], [41]		
Encryption			[42], [43], [44]
Standards		[48], [49], [7]	[47], [48], [49], [50], [7]
Proposed	[53], [54], [55], [56], [57], [58], [59], [60], [61], [62]	[53], [59], [60], [61], [63], [62], [64], [65]	[53], [56], [59], [61], [62], [66], [63], [67]

Table 3.1: Related works mapped according to the *domains* and *operational units*. The last row, **proposed**, shows the works mapped into the three cybersecurity phases.

Chapter 4

Proposed cybersecurity model: ANDURIL

Considering the limitations identified in the Related Work section, this Chapter proposes a model designed to systematically address the complexities associated with the embedded components of advanced automotive and safety-critical systems to answer RQ2.1: "What dimensions should be considered to ensure vehicle safety throughout their life cycles?". The model consists of various *dimensions* and *operational units*.

Dimensions are functional areas of security that refer to organizational, legal, and other technological aspects of security:

- **Software & Hardware components:** Includes security measures that protect internal or external hardware components like Electronic Control Units (ECUs), Road Side Units (RSUs), and their software.
- **Automotive Security Architecture:** Involves security measures regarding the communication between the in-vehicle components.
- **Supply chain security automotive:** Includes security measures regarding the use of standards, reports, etc. An example is the ISO/SAE 21434.
- **Automotive & Smart Mobility Ecosystem:** Includes security measures that protect assets regarding the interaction between in-vehicle and out-vehicle (for example, Vehicle-to-Everything (V2X), Cloud, ECUs, and TCU).
- **Automotive Threat Intelligence:** Includes security measures regarding the real-time detection, response, and investigation of threats for modern vehicles and Smart Mobility, such as Smart Cars.

On the other hand, the operational units are based on the concept of cybersecurity proposed by the National Institute of Standards and Technology (NIST)

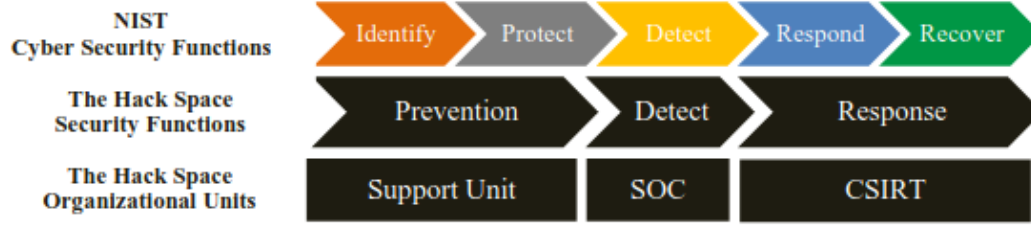


Figure 4.1: The Hack-Space organizational functions [5].

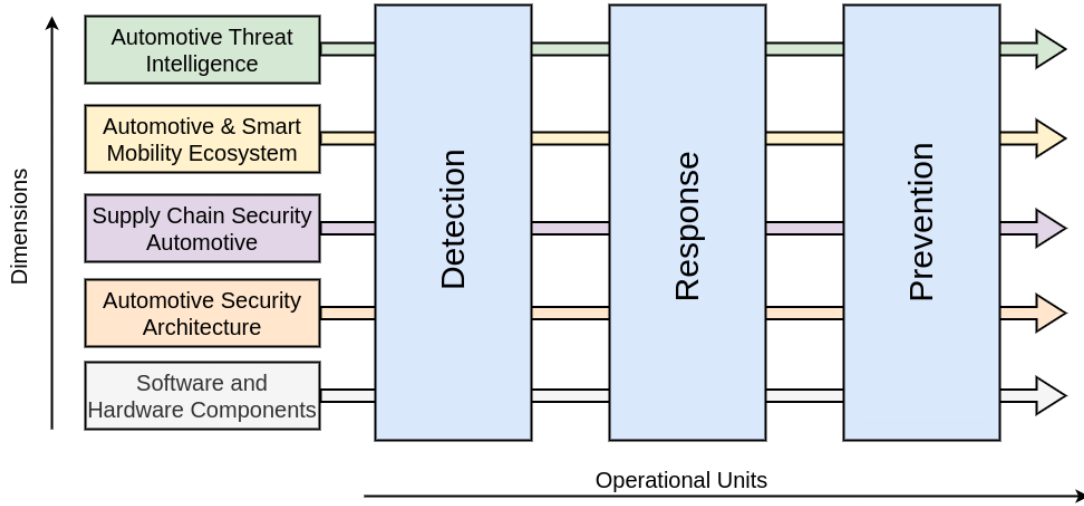


Figure 4.2: Proposed life cycle in which there are the operational units (principles) horizontally and vertically the domains.

[68], which are: (i) *Identify*, (ii) *Protect*, (iii) *Detect*, (iv) *Respond*, (v) *Recover*. In [5], the authors used the five principles of the NIST framework to redefine them on three functions: (i) Prevention (Identify and Protect), (ii) Detect, (iii) Response (Respond and Recover) (Figure 4.1). To address RQ2.2, "What operational units should be considered to ensure vehicle safety throughout their life cycles?" this analysis adopts the principles of *detection*, *response*, and *prevention* as foundational elements of the cybersecurity life cycle.

Section 3.4 shows that the works analyzed in the literature do not entirely answer RQ2.3: "Do current methodologies in the automotive context adequately cover the entire Detection-Response-Prevention cycle?" since several techniques propose only one of these phases (see 3.1). Therefore, addressing the absence of comprehensive methodologies or frameworks to manage cybersecurity in the automotive sector becomes essential.

Based on RQ2.1, RQ2.2, and RQ2.3, this thesis proposes a model called ANDURIL (Automotive Network Defense Unified Response Intrusion Limitation), which is subdivided into dimensions and operational units (Figure 4.2). AN-

DURIL focuses on detection (*Network Defense*), response (*Unified Response*), and prevention (*Intrusion Limitation*). The first operational unit aims to propose IDSs and a V-SOC to defend the in-vehicle network (Network Defense). The second operational unit proposes unified response techniques to support security analysts in understanding vulnerabilities using open-source vulnerability databases. The last operational unit presents risk assessment methods to suggest security controls that the automotive industry could apply during the components design phase to limit intrusion attacks. Table 4.1 shows the related works mapped according to ANDURIL.

As described before, the subsequent Chapters (detection, response, and prevention) achieve the verticals (Dimensions) by proposing various techniques or methodologies. Chapter 8 shows the combination of several techniques or methodologies (taken from the dimensions) to validate the automotive cybersecurity life cycle considering the horizontal Operational Units.

The following sections present both the description and experimentation of the proposed techniques.

	Detection	Response	Prevention
Automotive Threat Intelligence	[27], [32], [33], [34], [35], [36], [19], [39], [40], [41], [53] , [54] , [55] , [56] , [57] , [58] , [59] , [60] , [61] , [62] , [63]	[7], [27], [53] , [64] , [60] , [66]	[7], [27], [53] , [67] , [63]
Automotive & Smart Mobility Ecosystem	[62]	[7], [62]	[7], [62] , [67] , [63]
Supply Chain Security Automotive	[53] , [54] , [55] , [56] , [57] , [58] , [59] , [60] , [61] , [62]	[48], [49], [7], [66] , [61] , [59] , [65]	[47], [48], [49], [50], [7], [67] , [59] , [61] , [63]
Automotive Security Architecture	[27], [32], [33], [34], [35], [36], [19], [39], [40], [41], [53] , [54] , [55] , [56] , [57] , [58] , [59] , [60] , [61] , [62]	[27], [7], [53] , [61] , [66] , [59] , [64] , [65] , [60]	[42], [43], [44], [27], [47], [48], [49], [50], [7], [61] , [67] , [63] , [59]
Software and Hardware Components	[27], [32], [33], [34], [35], [36], [19], [39], [40], [41], [53] , [54] , [55] , [56] , [57] , [58] , [59] , [60] , [61] , [62]	[27], [7], [53] , [61] , [66] , [59] , [64] , [65] , [60]	[27], [47], [48], [49], [7], [67] , [61] , [63] , [59]

Table 4.1: Related works mapped according to the *domains* and *operational units*. The proposed techniques and methodologies are in bold.

4.1 Detection

The first operational unit specifies the development and implementation of appropriate activities to identify the occurrence of a cybersecurity event [68]. This activity involves, for example, continuous monitoring systems and anomalies and events methodology. In The Hack Space, the detection function is coordinated with the SOC, which analyzes and monitors activities on the networks, servers, and other relevant locations. For this unit, the SOC methodology has been repurposed to meet the requirements of the automotive field in detecting threats on

the CAN protocol using SIEM [53]. Additionally, the proposition of this approach belongs to the *Automotive Threat Intelligence* domain, which includes *Response* and *Prevention*. Further, the other security domains are the *Automotive & Smart Mobility Ecosystem*, and *Automotive Security Architecture*, which involves attack detection for in-vehicle networks and V2X communication like Smart Mobility.

Modern vehicles rely on Intrusion Detection Systems (IDS), anomaly detection, and onboard monitoring for overall vehicle safety and security. However, the diversity and complexity of attacks (e.g., physical, network-based, or software-based attacks) present a considerable challenge for comprehensive coverage. Several detection methodologies using classical and quantum ML algorithms based on the CAN standard have been proposed [54, 55, 56, 57, 58].

The ISO/SAE 21434 emphasizes identifying potential cybersecurity threats and vulnerabilities throughout the vehicle lifecycle. Therefore, most proposed algorithms are developed to detect CAN attacks and support the automotive industry. For example, in [57], a mock-up of a real-time microcontroller-based intrusion detection system was built to boost the safety and security of vehicles, drivers, and passengers, highlighting how technology transfer can be adopted in this domain. In [56], a methodology to improve cybersecurity education has been proposed, introducing cybersecurity education right from the university using innovative research methodologies and subsequently proposing strategies that could help the automotive industry. The proposed approach adopts a multi-class Intrusion Detection System to identify CAN attacks. A hybrid approach was also presented to detect attacks in the Smart City environment [59], which belongs to the Automotive and Smart Mobility Ecosystem domain, to protect assets involving the Smart City.

Figure 4.3 shows the techniques proposed in ANDURIL for the detection phase.

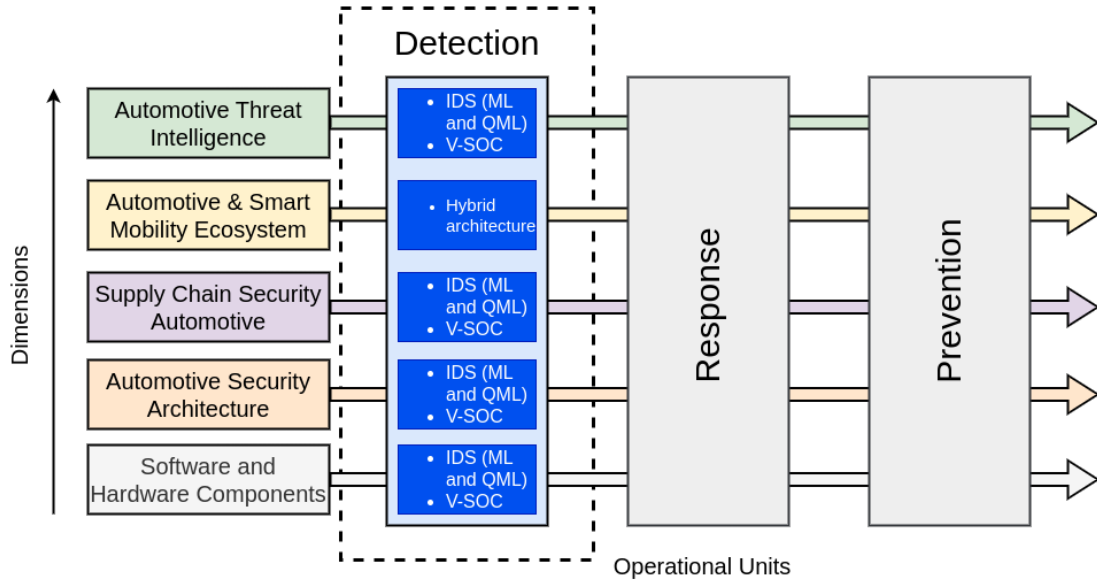


Figure 4.3: Techniques proposed for the detection phase.

4.2 Response

Response and recovery strategies are critical in automotive cybersecurity because the embedded components are crucial systems that usually can not be stopped when the vehicle is in motion. For example, if an IDS installed in a gateway unit or another part of the onboard network detects an attack, stopping the vulnerable component as a response technique is impossible since it could affect the vehicle's critical functionality.

Response strategies are outlined in the ISO/SAE 21434 standard through incident response planning, ensuring that there are effective containment and mitigation processes once a threat is detected. However, automotive response mechanisms are still developing and, therefore, are not robust. For this reason, several techniques have been proposed to reduce this gap.

The proposed solutions include, for example, informing the driver about the attack [60] or supporting the security analysts in understanding the attack by integrating information from the vulnerability databases to support decision-making processes (see Chapter 7) [59, 61].

Figure 4.4 shows the techniques proposed for the response phase. These techniques belong to all the domains since they aim to suggest solutions to detect and fix a vulnerability when the detection techniques identify an attack.

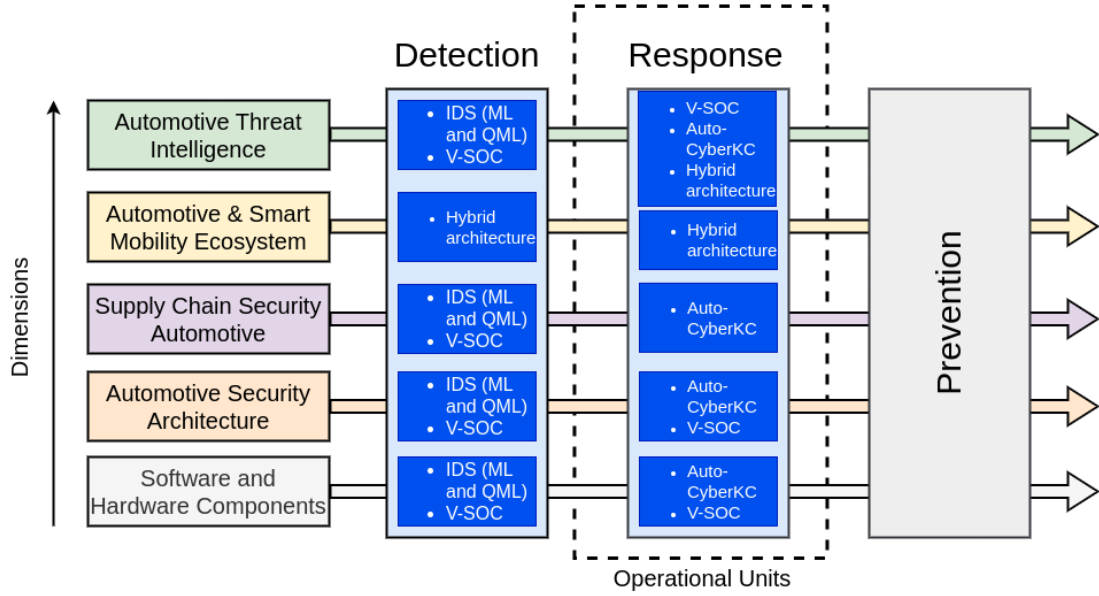


Figure 4.4: Techniques proposed for the response phase.

4.3 Prevention

This third function encloses the functionalities that focus on identifying critical resources that support critical functions and the related cybersecurity risks that enable an organization operating in the automotive sector to prioritize these risks and implement risk management strategies. In order to do so, first, it is important to identify the assets that need to be protected, a critical task in the automotive field because the assets (for example, embedded components, cloud providers, and in-vehicle protocols) may differ from one vehicle to another. An example of modern architecture is provided in Section 2.3, Figure 2.9. For example, an automotive organization may identify the *Gateway Unit* as a critical access point since it enables the other embedded components to communicate with each other using the CAN protocol. Once critical assets are identified, an IDS can be deployed in the Gateway component to identify attacks that exploit the vulnerabilities in the CAN protocol as a risk assessment strategy and prevention approach.

In *The Hack Space Model* [5], the prevention function consists of an organizational unit called the Support Unit (Figure 4.1), in which all the activities for monitoring the assets of the organization are concentrated, as well as the internal training activities for the prevention and improvement of all the processes and procedures for managing internal security. For example, risk assessment techniques can be used to evaluate the impact of a threat on one or more critical assets and propose security controls to reduce this risk.

This thesis proposes three approaches that consider all the domains to protect in-vehicle assets (for example, the CAN bus, ECU components, etc.). In brief,

these approaches are as follows:

- A web application using an automotive taxonomy that contains attacks.
- A semi-automatic risk assessment methodology.
- A vision model based on Quantum Artificial Intelligence to educate and train developers and testers during different phases of the Software Development Life Cycle (SDLC).

Figure 4.5 shows the proposed techniques for the prevention phase. In the figure, MARISMA-CAR spans all domains as it introduces security controls to protect various components, such as hardware, software, and networks (in-vehicle and V2X). Whereas, *AutoCyber-KC* applies to all dimensions except the *Automotive & Smart Mobility Ecosystem* because the hybrid architecture is used as a prevention technique for the V2X network and Smart City.

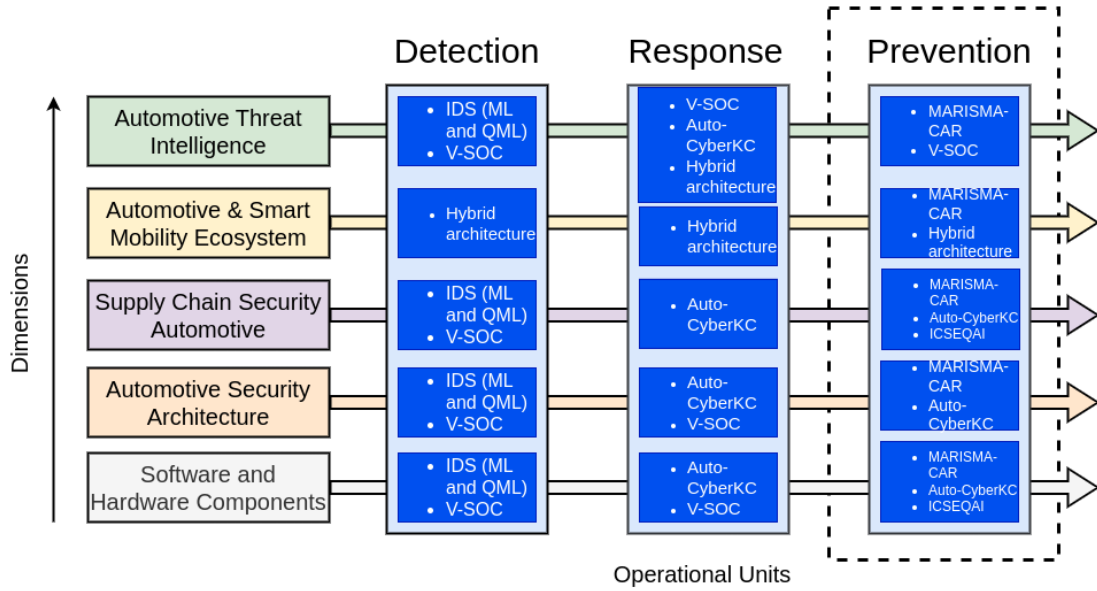


Figure 4.5: Techniques proposed for the prevention phase.

Chapter 5

Detection techniques in ANDURIL

As shown in Chapter 4, several works consider only the detection phase without the possibility of interacting with other domains. For this reason, to answer the RQ2.4 "Can Detection techniques interact across different dimensions to improve attack identification?", this Chapter analyzes and experiments with several techniques, considering the domains shown in Chapter 4. In particular, the proposed techniques belong to different domains based on the technical aspect of security. Machine Learning algorithms are proposed that use traditional and Quantum mechanisms (see Section 5.3 and Section 5.4). In light of the constraints outlined in Section 3.4, traditional ML algorithms were selected due to their lower computational complexity compared to DL models. Specifically, the SVM and Random Forest (RF) algorithms were chosen. The initial experiments involved using the SVM model on a cutting-edge dataset comprising raw CAN normal and attack messages. Subsequent experiments were conducted, incorporating the RF algorithm. The findings indicated that RF outperformed SVM in terms of training and testing time and performance metrics. For this reason, the RF was chosen to be deployed into the Raspberry Pi 4 (Section 5.4.3.1).

Regarding Quantum ML algorithms, QBoost was chosen because traditional Quantum algorithms, like Quantum SVM, take a long time to train since they are run in a simulation environment. The results show that the RF is more efficient than QBoost in terms of time and metrics performance. However, since these algorithms can be executed in hybrid Quantum Computers, it is possible to send CAN raw messages to be analyzed instead of deploying the model inside the embedded component.

In the automotive context, the IDS analyzes the CAN bus traffic to protect the connected ECUs. In the chapters 6 and 7, these techniques will be adapted for the response and prevention operational units based on the identified threat.

Due to computational limits, an IDS could be difficult to implement in an ECU. Therefore, it is also important to propose and experiment with alternative detection methods. For this reason, a Vehicle-SOC has been proposed to detect, respond, and prevent attacks that can occur in the automotive context (see Sec-

tion 5.2). In particular, the V-SOC belongs to all the dimensions because it can help not only detect a threat but also help cybersecurity experts obtain new Common Vulnerabilities and Exposures (CVEs) to understand and identify software vulnerabilities.

Finally, the last methodology involves using a hybrid classical architecture, using Quantum Machine Learning algorithms, with the support of SIEM for attack detection (see Section 5.5). This proposed architecture can be enclosed within the domain *Automotive & Smart Mobility Ecosystem*) since it aims to mitigate potential damage and minimize their impact on interconnected systems like Smart Cities or Smart Cars.

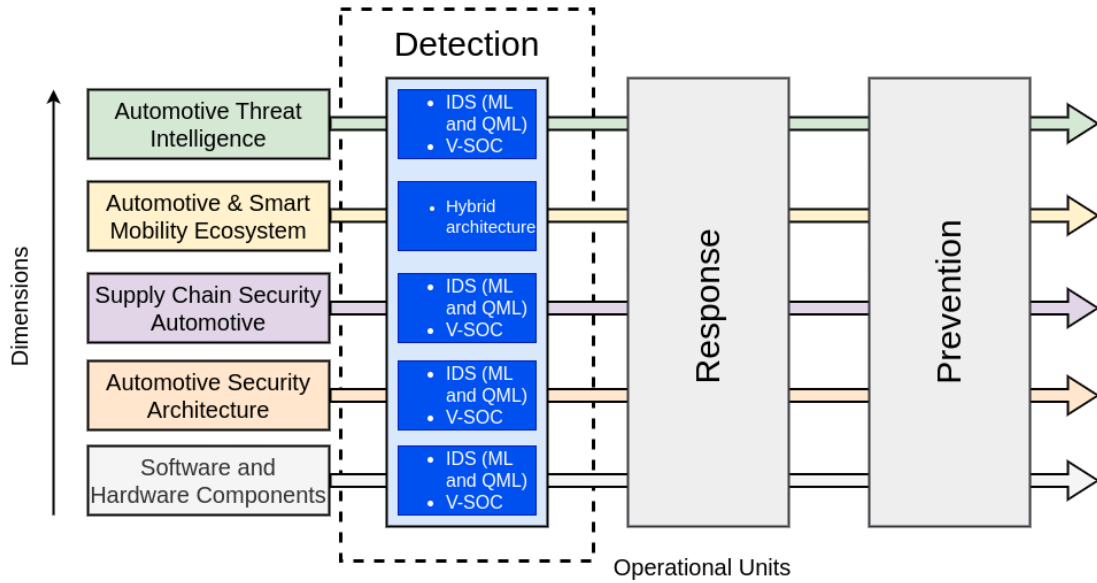


Figure 5.1: Proposed techniques regarding the detection phase considering the *dimensions* (vertical) and the *operational units* (horizontal).

Figure 5.1 shows the proposed techniques considering the domain and the operation unit that will be analyzed in the following sections. First, a brief description of the used dataset is shown.

5.1 Datasets description

This section describes the datasets used to train and validate the proposed algorithms to identify CAN attacks and for Smart City. First, two state-of-the-art datasets, the Car-Hacking Dataset and the Survival Analysis Dataset, are explained. These contain raw CAN messages and are used for the automotive context. Then, the third dataset, called SA-Hacking, is shown. The SA-Hacking Dataset is a proposed dataset that contains synthetic raw CAN messages, which

consists of normal and attack messages considering the threats discussed in Section 2.3. Finally, the last two datasets are used in the IoT field to consider attacks that can be performed in a Smart City.

5.1.1 Car-Hacking Dataset

The Car-Hacking Dataset for the Intrusion Detection [32], [69] is an open-source dataset in which the researchers obtained traffic data via the OBD-II from a real vehicle. In particular, these data were obtained from a Hyundai's YF Sonata using two custom Raspberry Pi devices, one for logging the traffic from the OBD-II port and the other for injecting fabricated messages as attack node [69].

The dataset is divided into five subsets depending on the type of attack: (i) DoS, (ii) Fuzzy, (iii) Gear Spoofing, (iv) RPM Spoofing, and (v) Attack-Free. Each subset contained 300 intrusions by injected messages. Each intrusion was performed for 3 to 5 seconds, and each subset contains CAN traffic collected for thirty to forty minutes [69]. The Car-Hacking dataset includes:

1. **DoS Attack:** injecting messages with ID "0x0000" every 0.3 milliseconds.
2. **Fuzzy Attack:** injecting random ID and payload every 0.5 milliseconds.
3. **Gear Spoofing Attack:** injecting messages of certain CAN ID related to gear information every one millisecond.
4. **RPM Spoofing Attack:** injecting messages of certain CAN ID related to RPM information every one millisecond.
5. **Attack-free:** normal traffic data without attack messages.

Each record in the dataset is identified by the following attributes:

- **Timestamp:** recorded time(s).
- **CAN ID:** identifier of CAN message expressed in HEX.
- **DLC:** number of data bytes, from 0 to 8.
- **DATA[0-7]:** data value (byte).
- **Flag:** T or R, T represents an injected message while R represents a normal message.

Table 5.1 summarizes the distribution of each of the four different subsets, with each row representing a single subset containing a specific attack except for normal messages that contain only attack-free messages.

In the context of this thesis, the subset *Normal messages* is not used since the other subsets contain normal and attack messages.

Attack Type	Total number of messages	Number normal messages	Number of injected messages
DoS attacks	3,665,771	3,078,250	587,521
Fuzzy Attacks	3,838,860	3,347,013	491,847
Gear Spoofing	4,443,142	3,845,890	597,252
RPM Spoofing	4,621,702	3,966,805	654,897
Normal messages	988,871	988,871	-

Table 5.1: Overview of the Car-Hacking Dataset for Intrusion Detection.

Attack Type	Total number of messages (Sonata)	Number normal messages (Soul)	Number of injected messages (Spark)
Flooding Attacks	149,547	181,901	120,570
Fuzzy Attacks	135,670	249,990	65,665
Malfunction Attacks	132,651	173,436	79,787
Attack free state	117,173	192,516	136,934

Table 5.2: Overview of the Survival Analysis Dataset.

5.1.2 Survival Analysis Dataset

The second dataset is called Survival Analysis Dataset¹ [70] consisting of Flooding, Fuzzy, and Malfunction attacks conducted on three vehicles: HYUNDAI YF Sonata, KIA Soul, and CHEVROLET Spark. The raw CAN messages are obtained through an in-vehicle OBD-II port. The researchers collected the data using a Raspberry Pi3² and the PiCAN2³ devices connected by a serial peripheral interface, and the DB9 port of PiCAN2 was connected to the vehicle's OBD-II port. The attack was performed using a Python tool by sending abnormal messages to the *can0* interface of the Raspberry Pi3.

- **Flooding attack:** the attack was performed by injecting messages with CAN ID set to 0x000 into the in-vehicle networks.
- **Fuzzy attack:** For the fuzzy attack, the authors generated random numbers

¹<https://ocslab.hksecurity.net/Datasets/survival-ids>

²A series of small single-board computers.

³CAN bus board for Raspberry.

with the "randint" function, a generation module for generating random integer numbers within a specified range. Messages were sent to the vehicle once every 0.0003 seconds. This process was conducted for both the ID field and the Data field. The randomly generated CAN ID ranged from 0x000 to 0x7FF and included both CAN IDs originally extracted from the vehicle and CAN IDs which were not.

- **Malfunction attack:** The malfunction attack targets a selected CAN ID from among the extractable CAN IDs of a specific vehicle. As CAN IDs for the malfunction attack, the authors have chosen the IDs 0x316, 0x153, and 0x18E from the HYUNDAI YF Sonata, KIA Soul, and CHEVROLET Spark vehicles, respectively.
- **Attack free state:** CAN messages that occurred during the driving.

Table 5.2 shows an overview of the subset of datasets contained in the Survival Analysis Dataset.

5.1.3 SA-Hacking Dataset

The third proposed dataset is the SA-Hacking Dataset, which contains raw synthetic CAN messages. Automotive classification models are fitted with datasets [71] that contain the same attacks (such as denial of service, fuzzing, and spoofing or malfunction). Furthermore, without the Database for CAN (DBC) file, the raw messages cannot be interpreted to understand the type of attack, except for reverse engineering techniques [72, 73]. In addition, with the DBC file, researchers can create a custom CAN network to conduct experiments. For these reasons, the SA-Hacking Dataset is generated with a custom DBC file. This dataset is also available for research purposes⁴ [58].

The DBC file [74] describes how a single CAN network must communicate. This information can be used to monitor and analyze the network and to simulate nodes. The information in the DBC file can also be used to encode a message into a CAN frame or decode it into a human-readable format. The keywords used in the DBC file are: **BU_** identifies a network node, **BO_** identifies a message, **SG_** identifies a signal, and **EV_** identifies the environment variable.

Figure 5.2 shows an example of a message contained in the DBC file used to generate the synthetic data. The green rectangles correspond to the message definition. The **message_id** identifies the CAN ID and must be unique and in decimal format; the **message_name** must be unique; the **message_size** specifies the message length (DLC); the **transmitter** identifies the node that sends this message.

The second row (blue rectangles) corresponds to the signal definition. The **signal_name** indicates the name of the signal and must be unique; **start_byte**

⁴<https://github.com/SERLABUniba/MaREA>

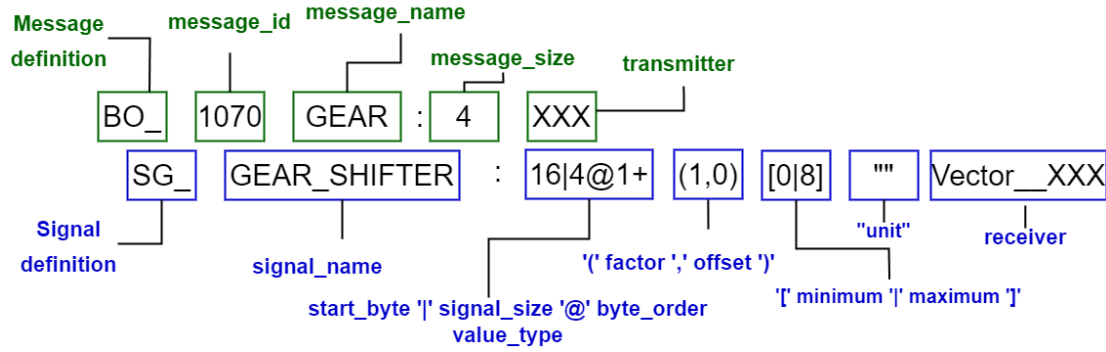


Figure 5.2: Syntax of the message and signal used in the DBC file.

start from zero and specifies the position of the signal within the data field of the frame; **signal_size** specifies the length of the signal in bits; **byte_order** is zero if the signal is big-endian or one if is little-endian; **value_type** defines the signal as unsigned (-) or signed (+); **factor** and **offset** define the linear conversion rule to convert the signal raw into a physical value; **[minimum|maximum]** and **unit** are optional and define the range and the unit of valid physical values of the signal; **receiver** the name of the receiving node (Vector__XXX is used as default).

The attack messages are generated by considering the methodology proposed by Cho and Shin [19] described in Section 2.3.4. To briefly recap, Cho and Shin considered three attack scenarios on in-vehicle networks: Fabrication, Suspension, and Masquerade. In the fabrication attack, an adversary forged a specific CAN identifier message's identifier, DLC, and data. The objective of the suspension attack is to stop/suspend the weakly compromised ECU's message transmissions, thus preventing the delivery/propagation of information it acquired to other ECUs. Finally, the objective of the masquerade attack is to manipulate an ECU while shielding the fact that an ECU is compromised.

The SA-Hacking Dataset was generated on Ubuntu 20.04 with a virtual CAN interface using the SocketCAN [75], a set of open-source CAN drivers.

To generate the raw CAN messages, a DBC file was created considering seven different message structures:

- **ENGINE:** BO_ message corresponding to the Revolutions Per Minute (RPM). The values are randomly generated in a fixed range from 1 to 3000, and the messages are sent every 0.08 seconds using the Python sleep function⁵.
- **ENGINE_TORQUE:** BO_ message corresponding to the engine torque. The values are randomly in a fixed range from -200 to 200 and sent every 0.02 seconds.

⁵<https://docs.python.org/3/library/time.html>

- **WHEEL_SPEED**: BO_ message corresponding to the vehicle speed. The values are randomly generated in a fixed range from 0 to 200. The messages are sent every 0.02 seconds.
- **STEER**: BO_ message corresponding to the steering angle. The values are randomly generated in a fixed range from 0 to 20 and sent every 0.04 seconds.
- **GEAR**: message corresponding to the gear. The values of the gear shift can be *D* for **d**rive; *S* for **s**port; *N* for **n**eutral; *R* for **r**everse; and *P* for **p**ark. Only *D* and *P* are used in this first approach. If the actual value of the encoded WHEEL_SPEED CAN message is greater than zero, the GEAR message will be D, and otherwise, if the same is zero, the GEAR message will be P. The messages generated by the GEAR are sent every 0.1 seconds.
- **BRAKE_PEDAL**: BO_ message containing the break information. If the value of WHEEL_SPEED is greater than zero, the SG_ NOT_BRAKE_PEDAL_PRESSED is codified with value '1' and BRAKE_PEDAL_PRESSED '0'. Otherwise, NOT_BRAKE_PEDAL_PRESSED will be '0' and BRAKE_PEDAL_PRESSED '1'. The messages regarding the BRAKE_PEDAL are sent every 0.03 seconds.

A Python 3 script was created using the *cantools*⁶ and *python-can*⁷ libraries. The *cantools* library decodes and encodes values using a DBC file. The *python-can* library provides CAN support for Python.

The attack phase was simulated based on the fabrication attack of Cho and Shin [19]. Six threads generate genuine data regarding the aforementioned bullet list of messages. Instead, one thread is used to generate attack messages. The WHEEL_SPEED and the STEER messages are chosen to create the fabrication attack. Every 10 seconds, 100 injected messages are sent without using the sleep function. In addition, each of these 100 messages is randomly generated using the *urandom* function⁸. Table 5.3 shows the normal and attack messages generated in about three hours.

Label	Number messages	%
Normal	1,322,823	82.82
Fabrication	274,400	17.18

Table 5.3: Number of normal and fabrication messages contained in the SA-Hacking Dataset.

⁶<https://pypi.org/project/cantools/>

⁷<https://python-can.readthedocs.io/en/stable/>

⁸<https://docs.python.org/3/library/os.html>

5.1.4 TON_IoT

The TON_IOT dataset [76] includes heterogeneous data cataloged from Telemetry datasets of IoT and IIoT sensors. The datasets were collected from a realistic, large-scale network designed at the Cyber Range and IoT Labs, School of Engineering and Information Technology (SEIT), UNSW Canberra Australian Defence Force Academy (ADFA). The dataset consists of 7 features and 587.077 rows (Message ID, Date, Time, Fridge Temperature, Temperature Condition, Attack or Normal Label, and Attack Type).

5.1.5 CICIoT2023

The CICIoT2023 dataset [77] is an IoT dataset that consists of seven attack categories: **DDoS**, **DoS**, **Recon**, **Web-Based**, **Brute-Force**, **Spoofing**, and **Mirai**. For each of these categories, the authors perform different attacks. For example, for DDoS, they performed an *UDP Flood*, a *HTTP Flood*, and so on. These attacks are generated with different tools like nmap, hping3, and golang-httpflood.

To perform these attacks, the researchers deployed an IoT platform that consists of 105 IoT devices. Of these, 67 IoT devices were directly involved in the attacks, and 38 other Zigbee and Z-Wave devices were connected to five hubs [77]. The typology is divided into two parts. In the first part, an ASUS router connects the network to the Internet, and a Windows 10 machine shares this connectivity. A Cisco switch is placed between this computer and a VeraPlus access point connecting 7 Raspberry Pi devices responsible for the attacks in the experiments. The traffic is collected using Wireshark with the Cisco switch that is connected to a second part of the network through a Gigamon Network Tap.

The CICIoT2023 has 46.686.579 rows (considering attacks and normal messages) of data with 47 independent variables (features). The dataset is 13.8 GB (Gigabytes) in size. The dataset is made available as a set of 169 different CSV files, which need to be concatenated before experimentation with the full dataset. The data are labeled with 33 attack types: *DDoS-ICMP_Flood*, *DDoS-UDP_Flood*, *DDoS-TCP_Flood*, *DDoS-PSHACK_Flood*, *DDoS-SYN_Flood*, *DDoS-RSTFINFlood*, *DDoS-SynonymousIP-Flood*, *DoS-UDP_Flood*, *DoS-TCP_Flood*, *DoS-SYN_Flood*, *Mirai-greeth_flood*, *Mirai-udpplain*, *Mirai-greip_flood*, *DDoS-ICMP_Fragmentation*, *MITM-ArpSpoofing*, *DDoS-ACK_Fragmentation*, *DDoS-UDP_Fragmentation*, *DNS_Spoofing*, *Recon-HostDiscovery*, *Recon-OSScan*, *Recon-PortScan*, *DoS-HTTP_Flood*, *VulnerabilityScan*, *DDoS-HTTP_Flood*, *DDoS-SlowLoris*, *DictionaryBruteForce*, *Browser-Hijacking*, *SqlInjection*, *CommandInjection*, *XSS*, *Backdoor_Malware*, *Recon-PingSweep*, *Uploading_Attack*; and one normal data label, *BenignTraffic*.

In the context of ML algorithms, the dataset is extremely unbalanced, which means that the distribution of classes or categories is significantly skewed or imbal-

anced. For example, at a more granular level, the dataset has about 15.5% records labeled as *DDoS-ICMP_Flood*, 11.6% of records labeled as *DDoS-UDP_Flood*, and 9.7% records labeled as *DDoS-TCP_Flood*. In contrast, there are only 0.002%, 0.005% and 0.006% records labeled as *Uploading Attack*, *Recon-PingSweep*, and *Backdoor_Malware*. Further, normal messages (*BenignTraffic*) consist of only 2.3% of the dataset, which makes prediction difficult using ML classification algorithms.

5.2 Detection techniques using SOC

One of the proposed detection techniques is to use the SOC approach inside the automotive field to make a Vehicle-SOC. Generally, the SOC architecture covers all three cybersecurity phases: *Detection* implements the set of security controls and analysis of events that have occurred to detect possible offenses; *Response* defines both cyber and physical structured response plans to contain the damage from a cyber offense; *Prevention* implements security controls to prevent the possibility of cyber-attacks by using, among others, cognitive technologies, and threat intelligence tools. The prevention phase also allows updating the ECU to fix the vulnerability that causes a specific attack identified in the detection phase. V-SOC4AS (vehicle-security operation center for improving automotive security) [53] has been proposed to detect CAN attacks. In particular, based on the threats described in Section 2.3, the proposed approach has been validated on two attack models: *DoS*, which consists of preventing legitimate access to services of a target machine by overwhelming its resources, and *fuzzing*, which involves sending a series of messages with minor randomized changes.

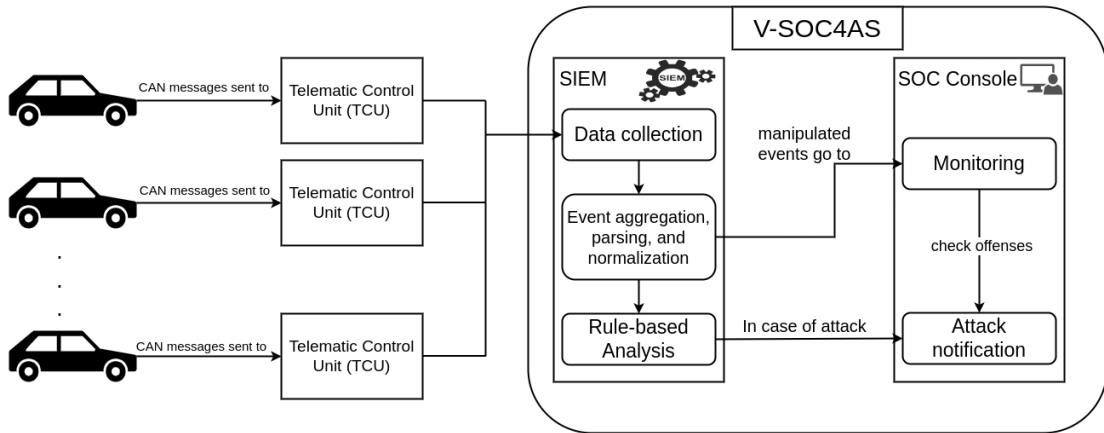


Figure 5.3: V-SOC4AS architecture.

In information technology (IT), a security operation center (SOC) is a centralized solution that an organization uses to continuously monitor critical assets to

prevent, detect, respond, and analyze cybersecurity attacks. A SOC is formed by people, processes, and technology in order to fulfill the previous goals [5]. Madani et al. [78] subdivided the architecture into four subsystems: sensors to collect traffic data; log management to normalize, classify, prioritize, and collect these data; correlation engine to analyze these events; and response system to react to security incidents. Based on this, the V-SOC4AS architecture can be represented in Figure 5.3.

The two main components of the proposed architecture are:

- **SIEM:** collects the raw events that will be aggregated, parsed, normalized, and analyzed to be displayed correctly. It can analyze data from different types of vehicles. This component allows us to parse these data without adding further physical components to the car. After the events are parsed and normalized, the rule-based component provides control over whether the events sent to SIEM correspond to attacks through custom rules. For example, in the information technology (IT) field, rules could detect if there are multiple login failed attempts on a login page and excessive firewall denies.
- **SOC Console:** the parsed events will be shown on the SOC console component and monitored by a cybersecurity specialist. If a rule-based mechanism triggers an offense, it will appear on this component so that the analyst can analyze it and potentially move on to the response phase. An advantage of the SOC console is that if an attack is detected, the cybersecurity specialist can check if it is a false positive or a real attack. Furthermore, the SOC analyst (a person in the organization who has knowledge of the automotive domain and automotive attack), when monitoring the data flow from the vehicle, could identify anomalies or attacks that the SIEM has not identified and subsequently add or modify a rule.

To provide an example of the proposed architecture, when an ECU generates a CAN frame, it sends the CAN frame to the telematic control unit (TCU). The TCU converts the messages into JSON format and sends them to the SIEM. When the SIEM receives these events, it aggregates and displays them on the SOC console. Simultaneously, the SIEM, with a rule-based mechanism, checks whether these events can be considered as attacks or not. Events that are considered attacks will be displayed on the SOC console. A tool called ICSim⁹ has been used to validate this communication. This tool emulates an instrument cluster, creates the data, and sends them to a virtual channel using the SocketCAN interface. In particular, the instrument cluster represents a vehicle's dashboard and includes a speedometer, door lock indicators, turn signal indicators, and a control panel. The ICSim creates the data generated by the instrument cluster. The vcan (virtual CAN) interface allowed communication between the ICSim and the CAN bus.

⁹<https://github.com/zombieCraig/ICSim>

Ubuntu 20.04 was used to run ICSim. The TCU is a Python module that captures the messages from the vcan0 and sends them to the SIEM in JSON format.

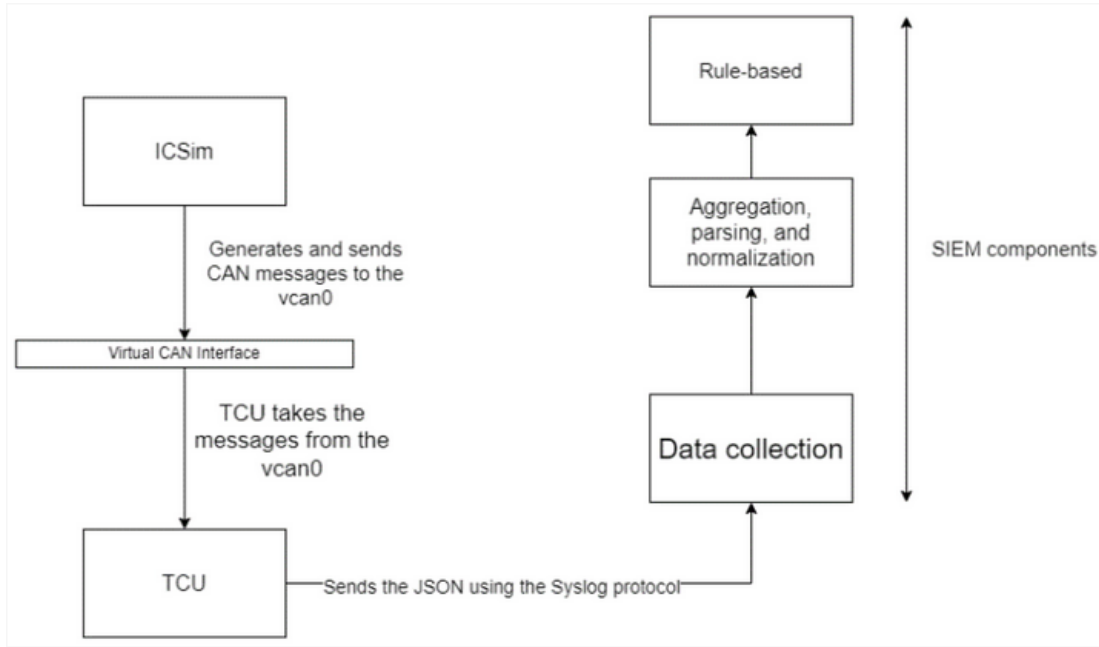


Figure 5.4: Messages generated by ICSim and transformed into JSON format by the TCU component.

Starting from the proposed architecture, IBM QRadar SIEM was used to identify attacks in an automotive environment. IBM QRadar uses the Syslog protocol to receive generic events. For this reason, the CAN messages are sent using this protocol in JSON format in the proposed architecture. When IBM QRadar obtains data from a generic log source (for example, the TCU or other vehicle components in the context of automotive) using the Syslog protocol, it manipulates them using a three-tier architecture.

This architecture consists of a *data collection* layer that collects data such as events or flows from a specific asset; *data preprocessing* in which the data are run through the Custom Rule Engine (CRE), which generates offenses and alerts, and then are written to the storage; *data search* in which the data collected and processed by QRadar are available to users for searches, analysis, reporting, and alerts or offense investigation.

The first approach is to simulate the communication between the simulator (ICSim) and the SIEM (QRadar) to validate the possibility of sending and processing data to an external component. Figure 5.4 shows the communication process between ICSim and IBM QRadar. In particular, the *SIEM Component* shows the three-layer architecture of QRadar and how the CAN messages received are analyzed.

```

vcan0 333 [7] 00 00 00 00 00 00 1E
vcan0 095 [8] 80 00 07 F4 00 00 00 35
vcan0 1A4 [8] 00 00 00 08 00 00 00 2F
vcan0 1AA [8] 7F FF 00 00 00 00 67 20
vcan0 1B0 [7] 00 0F 00 00 00 01 66
vcan0 1D0 [8] 00 00 00 00 00 00 00 0A
vcan0 166 [4] D0 32 00 09
vcan0 244 [5] 00 00 00 01 D3
vcan0 158 [8] 00 00 00 00 00 00 00 0A
vcan0 161 [8] 00 00 05 50 01 08 00 0D
vcan0 191 [7] 01 00 90 A1 41 00 30
vcan0 133 [5] 00 00 00 00 98
vcan0 136 [8] 00 02 00 00 00 00 00 1B
vcan0 13A [8] 00 00 00 00 00 00 00 19
vcan0 13F [8] 00 00 00 05 00 00 00 1F
vcan0 164 [8] 00 00 C0 1A A8 00 00 31
vcan0 17C [8] 00 00 00 00 10 00 00 12
vcan0 18E [3] 00 00 5C

```

(a)

```

{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.935275", "ID": "0333", "DATA": "00 00 00 00 00 00 1E"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.936347", "ID": "0095", "DATA": "80 00 07 F4 00 00 00 35"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937420", "ID": "01a4", "DATA": "00 00 00 08 00 00 00 2F"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937436", "ID": "01aa", "DATA": "7F FF 00 00 00 00 67 20"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937439", "ID": "01b0", "DATA": "00 0F 00 00 00 01 66"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937441", "ID": "01d0", "DATA": "00 00 00 00 00 00 00 0A"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.938495", "ID": "0166", "DATA": "D0 32 00 09"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940095", "ID": "0244", "DATA": "00 00 00 01 D3"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940616", "ID": "0158", "DATA": "00 00 00 00 00 00 00 0A"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940624", "ID": "0161", "DATA": "00 00 05 50 01 08 00 0D"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940625", "ID": "0191", "DATA": "01 00 90 A1 41 00 30"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940627", "ID": "0133", "DATA": "00 00 00 00 98"}

```

(b)

Figure 5.5: The figure on the top shows the CAN messages generated by ICSim that are sent on the vcan0. Instead, the figure on the bottom shows the messages generated by ICSim and transformed into JSON format by the TCU component.

ICSim generates the CAN messages and sends them to the vcan0, and a component called TCU transforms these messages into a JSON format. After this, the TCU sends the data to IBM Qradar using the Syslog protocol. The data generated by ICSim are shown in Figure 5.5a. When the ICSim generates and sends the messages, TCU remains listening on the vcan0 channel, waiting for new messages. If a new message is received, the component transforms it into JSON format (Figure 5.5b) and sends it to the SIEM (Figure 5.6).

Event Name	EventID (custom)	Log Source Time	Source IP	Vehicle ID (custom)	ID CAN (custom)	Data CAN (custom)
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:40 PM	192.168.1.231	53cd3082a0	01aa	7F FF 00 00 00 00 67 20
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:37 PM	192.168.1.231	53cd3082a0	01a4	00 00 00 08 00 00 00 2F
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:34 PM	192.168.1.231	53cd3082a0	0095	80 00 07 F4 00 00 00 35
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:31 PM	192.168.1.231	53cd3082a0	0333	00 00 00 00 00 00 1E

Figure 5.6: Result after creating all properties. On the console, the payload can be seen as correctly formatted.

5.2.1 Experimental Results

Two attacks, DoS and fuzzing, were considered in the experimental setting to validate the proposed architecture. A state-of-the-art dataset was used for this evaluation called *Intrusion Dataset V2* [79]. For the DoS dataset considering the OpelAstra vehicle, the authors inject messages with CAN ID “000” for 10 s to simulate a flood of a 500 Kbps on the CAN bus. The packets injected are in the form "000#0000000000000000" with a total number of messages of 827555. Instead, the fuzzing dataset comprised an injection of 10 messages in total with illegitimate CAN ID values. Instead, the data field is populated with hexadecimal values that are set to "FF".

For the experimental analysis, the JSON sent to IBM Qradar is changed from the one depicted in Figure 5.5b. Two new attributes are added into the new JSON message:

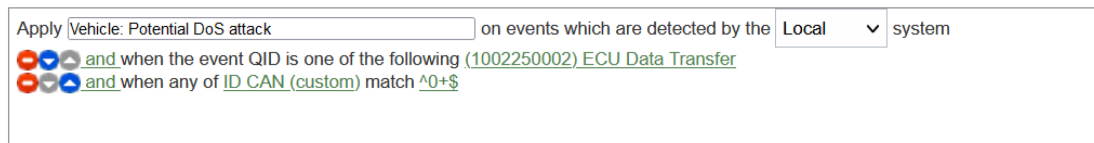
- **UUID_ECU:** A unique identifier for each ECU. It can be used to check which ECU is under attack. For this experiment, two unique identifiers will be used: one for the DoS attack and another for the Fuzzing.
- **vehicleModel:** To represent the vehicle model. Two different models were used to simulate two different vehicles: OpelAstra and RenaultClio.

The attribute UUID will be modified based on the vehicleModel. For the OpelAstra, the UUID *53cd3082a0* will be used. Instead, the UUID *6fab142fe3* will be used for the RenaultClio. The fuzzing dataset was divided into two splits to prove that V-SOC4AS can also be used to identify threats in multiple vehicles. In the first five messages (first split), the attribute vehicleModel was set to "OpelAstra", and for the last five messages (second split), the attribute vehicleModel was set to "RenaultClio". The attribute UUID_ECU was the same for both splits to simulate the case in which two different vehicles have the same ECU produced by a specific OEM.

5.2.1.1 DoS Attack

Since CAN does not have inherent authentication mechanisms, an attacker can send high-priority messages (for example, "000") to manipulate the bus. To detect this attack, a rule was created in Qradar that checks if the CAN ID of a specific

event contains all zeros. Figure 5.7 shows the created rule to detect the DoS attack.

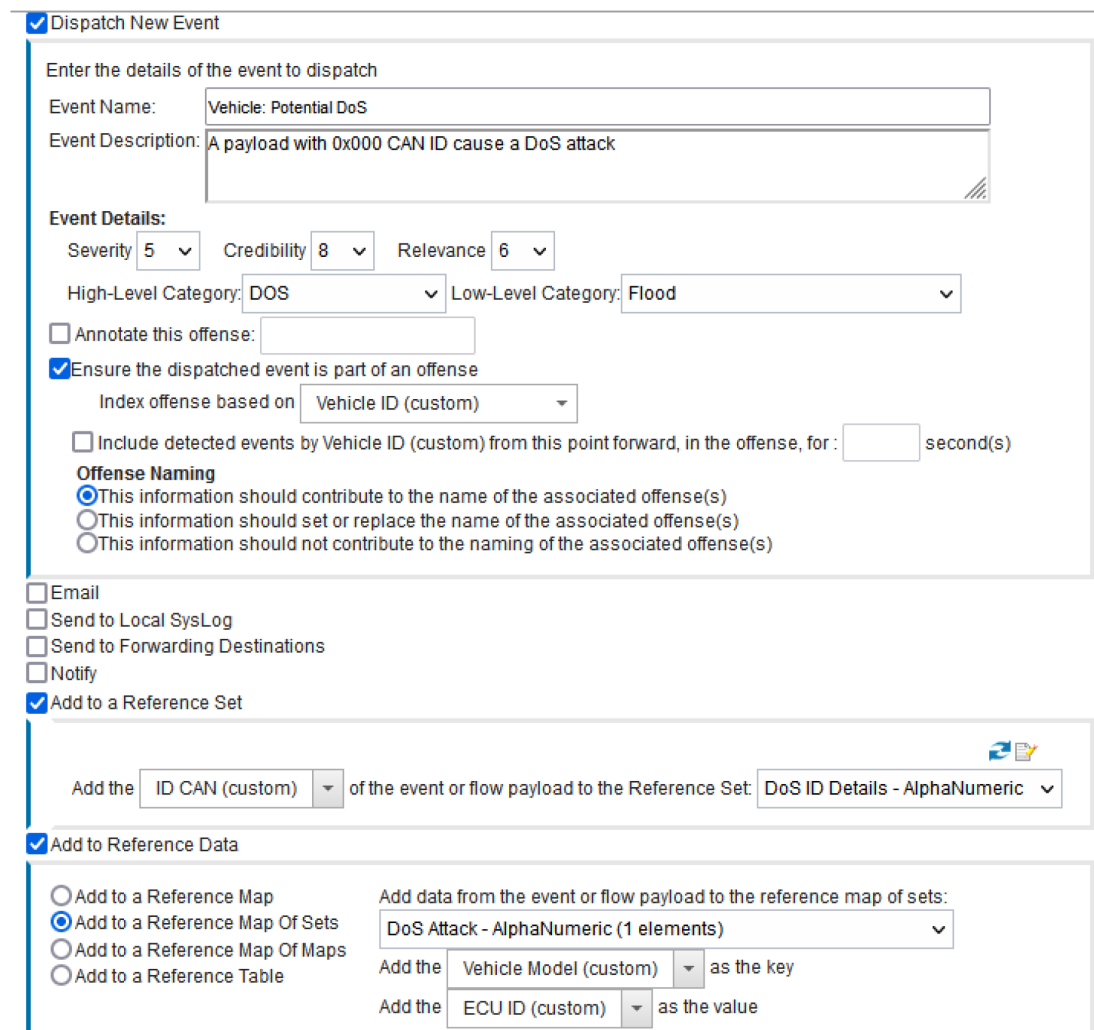


Apply **Vehicle: Potential DoS attack** on events which are detected by the **Local** system

and when the event QID is one of the following (1002250002) ECU Data Transfer

and when any of ID CAN (custom) match ^0+\$

Figure 5.7: DoS rule in QRadar.



☒ Dispatch New Event

Enter the details of the event to dispatch

Event Name: **Vehicle: Potential DoS**

Event Description: **A payload with 0x000 CAN ID cause a DoS attack**

Event Details:

Severity **5** Credibility **8** Relevance **6**

High-Level Category: **DOS** Low-Level Category: **Flood**

☐ Annotate this offense:

☒ Ensure the dispatched event is part of an offense

Index offense based on **Vehicle ID (custom)**

☐ Include detected events by Vehicle ID (custom) from this point forward, in the offense, for : **0** second(s)

Offense Naming

☒ This information should contribute to the name of the associated offense(s)

☐ This information should set or replace the name of the associated offense(s)

☐ This information should not contribute to the naming of the associated offense(s)

☐ Email

☐ Send to Local SysLog

☐ Send to Forwarding Destinations

☐ Notify

☒ Add to a Reference Set

Add the **ID CAN (custom)** of the event or flow payload to the Reference Set: **DoS ID Details - AlphaNumeric**

☒ Add to Reference Data

☐ Add to a Reference Map

☒ Add to a Reference Map Of Sets

☐ Add to a Reference Map Of Maps

☐ Add to a Reference Table

Add data from the event or flow payload to the reference map of sets:

DoS Attack - AlphaNumeric (1 elements)

Add the **Vehicle Model (custom)** as the key

Add the **ECU ID (custom)** as the value

Figure 5.8: Rule Wizard with the field used to identify the DoS attack.

The first row is used to filter out the other events not related to DoS. For example, this rule filters messages with QIDs belonging to ECU data transfer for

further analysis. The second row indicates a regular expression, a string of text that allows searching for a pattern in the text. When both conditions are true, an offense is raised.

To test the DoS rule, the dataset "dosattack.log" contained in the OpelAsstra dataset was used. The messages were transformed into JSON format with additional fields: `UUID_ECU` and `vehicleModel`. Figure 5.9 shows some of the messages that were sent to IBM QRadar.

```
{ "UUID": "53cd3082a0", "UUID_ECU": "f1b186f8fc", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "eventCategory": "CAN Payload", "Timestamp": "1536574995.000345", "ID": "000", "DATA": "0000000000000000" }
{ "UUID": "53cd3082a0", "UUID_ECU": "f1b186f8fc", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "eventCategory": "CAN Payload", "Timestamp": "1536574995.000577", "ID": "000", "DATA": "0000000000000000" }
{ "UUID": "53cd3082a0", "UUID_ECU": "f1b186f8fc", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "eventCategory": "CAN Payload", "Timestamp": "1536574995.000828", "ID": "000", "DATA": "0000000000000000" }
{ "UUID": "53cd3082a0", "UUID_ECU": "f1b186f8fc", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "eventCategory": "CAN Payload", "Timestamp": "1536574995.001122", "ID": "000", "DATA": "0000000000000000" }
{ "UUID": "53cd3082a0", "UUID_ECU": "f1b186f8fc", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "eventCategory": "CAN Payload", "Timestamp": "1536574995.001392", "ID": "000", "DATA": "0000000000000000" }
```

Figure 5.9: Example of JSON messages that correspond to DoS attack.

To determine what action should be taken once both conditions are valid, QRadar uses a mechanism called the Rule Wizard. Figure 5.8 shows the actions chosen to raise the offense regarding the DoS attack. *Dispatch New Event* defines the field used to generate the offense, such as the Event Name and Event Description. By choosing the action *Add to a Reference Set* and *Add to a Reference Data*, the values regarding the attack will be saved to understand where the attack occurs and what payload has been sent by the attacker. The role of the Reference Set is to save the CAN ID(s) sent as an attack on the bus. Therefore, it is possible to resolve the problem by fixing the ECU source code.

The reference data is important for two reasons. First, since V-SOC can analyze multiple vehicles, it is possible to determine, based on the vehicle model, if the attacker is attacking multiple vehicles or if the attack has already been identified in other cars. If the attack has been identified in different vehicles, we have the knowledge about the attack and, hence know how to thwart it. Second, with the ECU ID, we know where the attack began and if other vehicles are mounted with the same ECU. With this information, the developers can fix the ECU source code and extend the fix to the attacked vehicle and others using the same ECU. In this way, both a detection and prevention phase are carried out to prevent an adversary from attacking other similar vehicles and ECUs. Finally, with the *vehicleModel* and the *IP*, it is possible to perform a response phase. This aligns with the domain *Automotive Threat Intelligence* based on response and prevention (discussed in the following chapters).

The SIEM allows the addition of information about the attack's severity, credibility, and relevance. The severity responds to the question: "How high is the potential damage to the destination?"; the credibility responds to the question: "How valid is the information from that source?"; and the relevance responds

to the question: "How important is the destination?". This information can be used to understand the threat level and required response time. For example, the fuzzing attack has a severity level higher than the DoS attack because it could activate anomalous behavior in the vehicle. For this reason, the fuzzing attack is of higher priority than the DoS attack.

Offense 500				
Magnitude		Status	Relevance 5	Severity 5
		Offense Type	Vehicle ID (custom)	
Description	Vehicle: Potential DoS		Event/Flow count: 40,015 events and 0 flows in 1 categories	

Figure 5.10: Generated offenses when a DoS attack is detected.

When a payload with CAN ID 000 is sent to IBM QRadar, an offense is generated (Figure 5.10).

5.2.1.2 Fuzzing Attack

To detect the fuzzing attack with SIEM, the following approach was considered: each ECU's identifier is determined at the design phase, and consequently, the SOC analyst could create the database with ECU identifiers. For this reason, a reference set was created on QRadar by putting CAN IDs without attacks. These IDs have been taken from "full_data_capture.log" present in the OpelAstra and RenaultClio dataset that contains 2.690.069 packets captured with a total of 85 unique CAN IDs for the first dataset, and 386.567 packets captured with 55 unique CAN IDs for the second dataset.

```
{
  "UUID": "53cd3082a0",
  "UUID_ECU": "8b470861",
  "vehicleModel": "OpelAstra",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536574998.963000",
  "ID": "111",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "53cd3082a0",
  "UUID_ECU": "8b470861",
  "vehicleModel": "OpelAstra",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575008.043000",
  "ID": "111",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "53cd3082a0",
  "UUID_ECU": "8b470861",
  "vehicleModel": "OpelAstra",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575036.002000",
  "ID": "111",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "53cd3082a0",
  "UUID_ECU": "8b470861",
  "vehicleModel": "OpelAstra",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575061.281000",
  "ID": "222",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "53cd3082a0",
  "UUID_ECU": "8b470861",
  "vehicleModel": "OpelAstra",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575081.616000",
  "ID": "222",
  "DATA": "FFFFFFFFFFFFFFFF"
}

{
  "UUID": "6fab142fe3",
  "UUID_ECU": "8b470861",
  "vehicleModel": "RenaultClio",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575109.908500",
  "ID": "222",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "6fab142fe3",
  "UUID_ECU": "8b470861",
  "vehicleModel": "RenaultClio",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575133.263300",
  "ID": "333",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "6fab142fe3",
  "UUID_ECU": "8b470861",
  "vehicleModel": "RenaultClio",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575165.597000",
  "ID": "333",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "6fab142fe3",
  "UUID_ECU": "8b470861",
  "vehicleModel": "RenaultClio",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575181.757100",
  "ID": "333",
  "DATA": "FFFFFFFFFFFFFFFF"
}
{
  "UUID": "6fab142fe3",
  "UUID_ECU": "8b470861",
  "vehicleModel": "RenaultClio",
  "eventID": "CAN Payload",
  "eventCategory": "Flow",
  "Timestamp": "1536575200.992100",
  "ID": "444",
  "DATA": "FFFFFFFFFFFFFFFF"
}
```

Figure 5.11: Generated offenses when a Fuzzing attack is detected.

In Figure 5.11, the JSON messages regarding the Fuzzing attack were created in this way:

- The first five messages are used for the OpelAstra vehicle. The last five for the RenaultClio vehicle. The UUID related to the OpelAstra will not be changed, while a new one is generated for the RenaultClio.

- The UUID_ECU is the same for both vehicles. In this way, it is possible to simulate two vehicles that mount the same ECU and are attacked at the same time.

Apply	Anomaly: ID Not In Reference Set	on events which are detected by the	Local	▼	system
			and when the event QID is one of the following (1002250002) ECU Data Transfer		
			and NOT when an event matches any of the following Vehicle: Potential DoS attack		
			and NOT when any of ID CAN (custom) are contained in any of OpelAstra - AlphaNumeric, RenaultClio - AlphaNumeric		

Figure 5.12: Fuzzing rule in QRadar.

Figure 5.12 shows the rule created to identify the fuzzing attacks. The first row works similarly to the DoS rule. The second row is used to avoid being recognized as an event of a DoS attack. In the third row, the offense will be raised if the ID of the current payload sent to QRadar does not exist in the table.

☒ Dispatch New Event

Enter the details of the event to dispatch

Event Name:

Event Description:

Event Details:

Severity Credibility Relevance

High-Level Category: Low-Level Category:

☐ Annotate this offense:

☒ Ensure the dispatched event is part of an offense

Index offense based on

☐ Include detected events by Vehicle ID (custom) from this point forward, in the offense, for : second(s)

Offense Naming

☒ This information should contribute to the name of the associated offense(s)

☐ This information should set or replace the name of the associated offense(s)

☐ This information should not contribute to the naming of the associated offense(s)

☐ Email

☐ Send to Local SysLog

☐ Send to Forwarding Destinations

☐ Notify

☒ Add to a Reference Set

Add the of the event or flow payload to the Reference Set:

☒ Add to Reference Data

☐ Add to a Reference Map

☒ Add to a Reference Map Of Sets

☐ Add to a Reference Map Of Maps

☐ Add to a Reference Table

Add data from the event or flow payload to the reference map of sets:

Add the as the key

Add the as the value

Figure 5.13: The actions chosen for the fuzzing attack.

Figure 5.13 shows the actions chosen for the fuzzing attack.

Offense 495				Summary Display ▼ Events Flows Actions ▼ Print							
Magnitude				Status		Relevance	5	Severity	8	Credibility	2
Description	Anomaly: The ID is not in reference set			Offense Type	Vehicle ID (custom)						
Source IP(s)	192.168.1.9			Event/Flow count	5 events and 0 flows in 1 categories						
Destination IP(s)	192.168.1.9			Start	Oct 8, 2022, 2:02:12 AM						
Network(s)	Net-10-172-192_Net_192_168_0_0			Duration	7s						
				Assigned to	Unassigned						
Offense Source Summary											
Custom property value	53cd3082a0										

(a)

Offense 496

Magnitude

<

(b)

Figure 5.14: Generated offenses when a fuzzing attack is detected.

(a) Shows the offense regarding the vehicle model OpelAstra with five identified attacks (red rectangle). In (b), the figure refers to the RenaultClio with five identified attacks.

Figure 5.14a shows the generated offense when the SIEM sends and analyzes the five fuzzing messages. This offense corresponds to the OpelAstra (green rectangle corresponds to the UUID). Instead, Figure 5.14b shows the offense generated for the RenaultClio. The reference set contains the CAN_IDs detected by the SIEM as a fuzzing attack. In addition, the reference *map of set* contains the vehicleModel as a key and the UUID_ECU as a value. In this approach, using the reference map of set, if the attack occurs in multiple vehicles, the information about the attack happened and whether multiple cars are involved are available. For example, with the UUID of the ECU, the SOC analyst could control if other vehicles mount the same ECU and proceed to implement security measures before the attack is carried out in these vehicles.

Finally, the SIEM successfully recognized all the received attacks for the detection phase, as it is possible to see in Figure 5.14 and Figure 5.10.

5.2.2 Comparison with the state-of-the-art

The current state of detection using IT solutions has not been widely explored. Few studies have discussed the implementation of a Security Operations Center (SOC) to identify attacks in modern vehicles, but these papers lack experiments

and valid methodologies. Since the SOC is crucial for not only detecting threats but also for response and prevention (to be discussed in Chapter 6 and Chapter 7), there is a need to explore this field further. Therefore, the methodology proposed in this section aims to identify CAN attacks using IBM QRadar SIEM with a rule-based approach. Additionally, while the current state of the art focuses on attack detection using machine learning (ML) or deep learning (DL) algorithms (see Table 3), which require high computational resources, the SIEM can address this issue by analyzing messages without using the resources of the ECU. However, as discussed in Section 9.2 and Section 9.3, it is important to consider the feasibility of handling a large number of messages that need to be sent.

5.3 Detection techniques using QML Algorithms

Quantum Machine Learning algorithms are being widely used in cybersecurity. In the automotive context, identifying an attack as early as possible could prevent unexpected behavior in vehicles and even save the driver's life. For this reason, fast and quick response times for attack identification are crucial, especially for CAN, as it is the most used protocol in the in-vehicle network. For this reason, this thesis proposes several approaches using QML algorithms, first in binary and then multi-class, to improve the detection phase (Figure 5.1).

5.3.1 Binary classifiers

The first work [54] analyzes the use of a QML algorithm called QBoost in the Internet of Things (IoT) field. The IoT is extensively used in the automotive field, transforming the industry in various ways, such as connected vehicles using telematics and infotainment units, V2X, autonomous driving, and remote driving. In this way, it was possible to analyze the behavior of QBoost using an IoT dataset and then translate it to the automotive sector.

In classical machine learning, ensemble algorithms combine predictions of multiple base estimators to increase system performance, leading to better results when there is high diversity among each and every single estimator. QBoost [80, 81] is derived from one of the classical ensemble learning techniques and is specifically designed for quantum annealing, whose main goal is to optimize the best combination of learners for a given set of training data by minimizing the quadratic deviation between observed and predicted labels. This problem can be resolved on the Leap Quantum Cloud¹⁰ environment that provides a series of hybrid solvers located in different geographical regions.

This first approach aims to compare QBoost with a traditional ML algorithm, Support Vector Machine (SVM). This comparison is in terms of metrics (Accuracy, Precision, Recall, and F1-Score) and training time and validation. The *IoT Fridge*

¹⁰<https://www.dwavesys.com/solutions-and-products/cloud-platform/>

has been considered from the TON_IoT [76]. The methodology consists of three main stages:

- *Data Selection*: loads the dataset and makes a preprocessing phase. For example, for the QBoost model, the Label (0 for normal message and 1 for attack) is converted into +1 for normal and -1 since it involves operations of the qubits.
- *Training Phase*: consists of training QBoost and Support Vector Classifier (SVC) after the preprocessing phase.
- *Validation Phase*: validates the result with the testing dataset using traditional Accuracy, Recall, and F1-score metrics.

	Precision (%)	Recall (%)	F1-Score (%)	Support
0	0.82	1.00	0.90	10.000
1	1.00	0.78	0.88	10.000
Accuracy			0.89	20.000
Macro avg	0.91	0.89	0.89	20.000
Weighted avg	0.91	0.89	0.89	20.000

Table 5.4: Results obtained with QBoost.

Attacks/ normal classes	SVC accuracy (%)	Qboost accuracy (%)	SVC time (s)	QBoost time (s)
100, 100	1.00	1.00	0.00059	0.00043
200, 200	1.00	1.00	0.00066	0.00037
300, 300	1.00	1.00	0.00039	0.00082
400, 400	1.00	1.00	0.00099	0.00044
400, 500	1.00	1.00	0.00106	0.00039
500, 400	1.00	1.00	0.00125	0.00052
500, 500	1.00	1.00	0.00111	0.00042
1000, 1000	1.00	1.00	0.00181	0.00044
400, 5000	1.00	1.00	0.00425	0.00087
10,000, 300	1.00	0.80	0.00803	0.00125
300, 10,000	1.00	1.00	0.00993	0.00205
5000, 5000	1.00	1.00	0.00128	0.00074
10,000, 10,000	1.00	0.89	0.02348	0.00923
55,000, 55,000	0.25	0.29	0.05606	0.01944
60,000, 60,000	0.23	0.27	0.05947	0.01618

Table 5.5: Comparison of the results between SVC and QBoost.

Table 5.4 shows the results obtained with QBoost considering 20.000 samples of data. Instead, Table 5.5 compares the results between Support Vector Classifier (SVC) and QBoost. In particular, it is possible to see that the prediction time for QBoost is less than that of the SVC model. In the last two experiments, where there are more examples than the others, even if the accuracy of both is clearly decreasing, the accuracy of the quantum classifier is higher.

The same classifiers are compared considering in-vehicle attacks based on the results obtained in the IoT field. In particular, the second proposed technique in the detection phase consists of the use of SVC and QBoost to identify CAN attacks [55]. The two models are validated using the Car-Hacking Dataset (see Section 5.1.1) that contains raw CAN messages considering DoS and Fuzzy attacks. As the previously analyzed approach, a generic IDS pipeline has been considered that can be summarized in the following:

- *Pre-processing & Feature extraction*: Removal of messages with data field shorter than 8 bytes ('DLC' != 8) since classifiers can not handle missing values. The *Frequency* feature was derived from the timestamp, representing the time in milliseconds between two successive messages on the CAN bus with the same CAN ID. To handle categorical data, two types of encoding techniques: One-Hot-Encoding and binary encoding on *CAN ID* and string *Data* obtained by merging data bytes Data[0-7]. Finally, each feature was individually scaled to the range [min, max] with MinMax scaler. Labels used for classification were also mapped from "R": non-malicious, "T": malicious to "+1": non-malicious, "-1": malicious.
- *Feature Selection*: For each experiment, a subset of N features, consisting of "Frequency" and attributes derived from encoding "CAN ID" and "Data," were selected.
- *Model Training*: For each dataset two binary classifiers, SVC and QBoost were trained.
- *Prediction*: Test data was provided as input to the two trained models separately to make predictions.

The datasets were split into 80% training and 20% testing set before training. In order to reduce bias and balance the ratio of CAN IDs between the two splits [82], every dataset was shuffled.

Data in DoS dataset has a simple structure formed by 25 unique CAN IDs messages and malicious messages sent with CAN ID "0000" and null Data payload. One-Hot-Encoding¹¹ strategy was applied on *CAN ID* and binary encoding¹²

¹¹<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>

¹²https://contrib.scikit-learn.org/category_encoders/binary.html

Dataset	Set	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
DoS	Train	SVC	1.0	1.0	1.0	1.0
		QBoost	1.0	1.0	1.0	1.0
	Test	SVC	1.0	1.0	1.0	1.0
		QBoost	1.0	1.0	1.0	1.0
Fuzzy	Train	SVC	0.9858	0.9945	0.9891	0.9918
		QBoost	0.9778	0.9988	0.9757	0.9871
	Test	SVC	0.9716	0.9785	0.9892	0.9838
		QBoost	0.9756	0.9959	0.9760	0.9858

Table 5.6: Classification performance of binary classifiers: Train & Test sets for DoS & Fuzzy dataset. Bold shows best performance.

strategy on *Data*. The one-hot-encoding technique creates a new binary variable for each unique category, therefore extracting N features out of N categories, where 0 and 1 indicate the absence or presence of that category in each row. The total number of features were 43 including *Frequency*.

The Accuracy, Precision, Recall, and F1-score were computed to compare the classification performance. The results of two binary classifiers are shown in Table 5.6. Both classifiers were able to distinguish between malicious and non-malicious messages, reaching 100% Accuracy, Precision, Recall, and F1-score in each of the ten rounds of testing. This is because the DoS attack has a simple pattern and both classifiers are able to distinguish between normal and attack. In addition, as data shuffling before every round did not affect the metrics, and the training and testing set metrics were the always the same, it is safe to assume there is a low overfitting. In fact, the confusion matrices of SVC and QBoost in Figure 5.15 shows that each attack (label 0) and normal message (label 1) in the testing set was correctly classified. The confusion matrices shown in Figure 5.15 refer to the results from the last run of the test set.

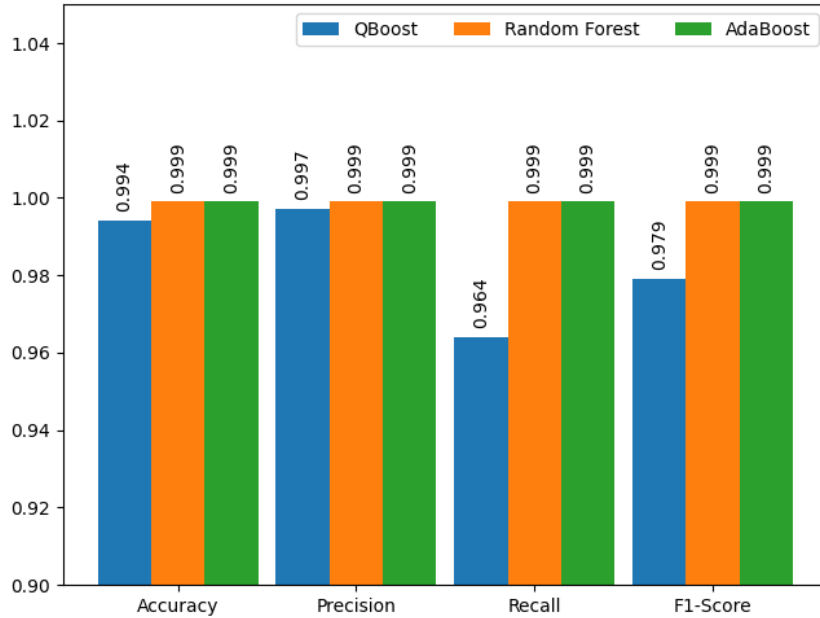
SVC - DoS Dataset		Qboost - DoS Dataset		
0	TN 16.16%	FP 0%	TN 16.16%	FP 0%
1	FN 0%	TP 83.84%	FN 0%	TP 83.84%
	0	1	0	1
0	TN 12.27%	FP 0.85%	TN 12.92%	FP 0.2%
1	FN 0.13%	TP 86.74%	FN 1.59%	TP 85.29%
	0	1	0	1
SVC - Fuzzy Dataset		Qboost - Fuzzy Dataset		

Figure 5.15: Confusion matrix for SVC and QBoost on DoS & Fuzzy datasets. "0" malicious; "1" normal messages. TN-True Negative, FP-False Positive, FN-False Negative, TP-True Positive.

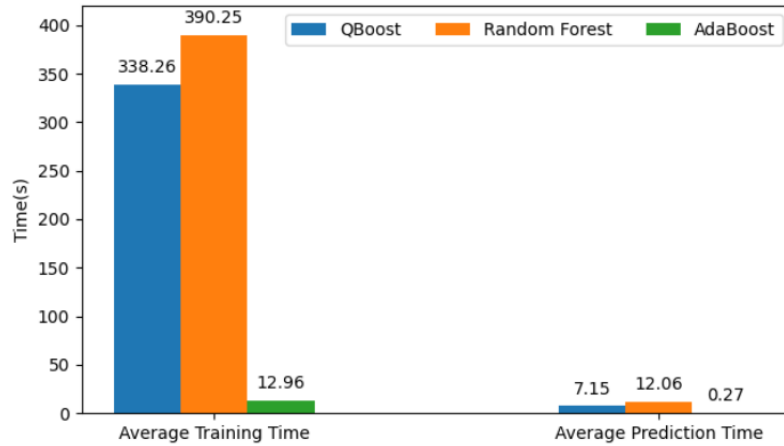
Table 5.7 compares the performance in seconds between the two classifiers. It is possible to see that there is a high gain in execution time in both the training and prediction phases, with gains of 62% and 99%, respectively.

Classifier	Attack Label	Average training time (s)	Average prediction time (s)
SVC	DoS	68.6735	2.5609
QBoost		26.1415	0.0275
		-42.532	-2.5334
SVC	Fuzzy	5933.2840	431.4609
QBoost		25.7134	0.1462
		-5907.5705	-431.3146

Table 5.7: Average time performance of the Binary classifiers on DoS and Fuzzy datasets. Bold shows best performance.



(a)



(b)

Figure 5.16: The figure on the top shows the quality of the ML models. Instead, the figure on the bottom shows the time performance.

The Fuzzy database has a more complex structure compared to the DoS since malicious messages are randomly sent to the CAN bus using all the 2.017 unique CAN IDs, 37 of which also transmit normal traffic. Binary encoding was applied to both *CAN ID* and *Data*, which returned 11 and 20 features, respectively, making 32 features in total with *Frequency*. The results of two binary classifiers are shown in Table 5.6. Considering the recall metric, the SVC performed slightly better than QBoost.

From the viewpoint of time performance, Table 5.7 shows the average results

obtained from 10 runs and it can be seen that QBoost clearly outperformed SVC in terms of execution time, during both training and prediction phases, with 99% gain in both cases. The confusion matrices for SVC and QBoost shown in Figure 5.15 confirm the precision and recall metrics presented in Table 5.6. For the fuzzy attack, the confusion matrices also refer to the last run of the testing set. The training time for SVC was closer to that of QBoost for DoS dataset results. On the other hand, the Fuzzy set being more complex in comparison to DoS, the acquired results prove that quantum annealing based algorithm successfully managed to optimize a complex problem.

5.3.2 Multi-class classifiers

The works discussed above are considered only as binary classifications. In the context of cybersecurity, it is important to distinguish the attack typology to employ the correct countermeasures (discussed in Section 6.2 related to the response). For this reason, the following proposed works take into consideration multi-class algorithms. The first approach [63] proposed traditional and QML algorithms to improve the identification of in-vehicle attacks and redefine the security life cycle in this context, serving as a baseline to define a vision model based on Artificial Intelligence that supports developers' decisions to integrate concrete design methods in the automotive development life-cycle. This second step will be discussed in the next Chapter regarding the response phase.

The detection phase aimed to test QBoost in a multi-class classification against traditional ML algorithms, such as Random Forest (RF) and AdaBoost. Since QBoost allows only a binary classification, the One-vs-Rest (OvR) strategy allows the classifier to make a multi-class classification. In this strategy, one class is fitted against all the other classes to determine its efficiency. The Car-Hacking Dataset was used to evaluate the three classifiers. First, the dataset was pre-processed by transforming the CAN ID and DATA from hexadecimal to decimal. The timestamp and DLC were removed as they did not contribute to the classification. A random under-sampling for the majority class was performed to account for the unbalanced dataset. The features were scaled using the standard deviation (MinMaxScaler), and the Principal Component Analysis (PCA) was used to reduce feature dimensionality using four components. Finally, the dataset was split in 70% for training and 30% for testing. Since a multi-class classification is considered, the final labels will be changed from 0 and 1 *Normal*, *DoS*, *Fuzzy*, *Spoofing RPM*, and *Spoofing Gear*. Model Accuracy, Precision, Recall, and F1-Score were used to evaluate the machine learning models used for multi-class classification experiments.

The quality metrics of all the models are almost similar, with both AdaBoost and RF showing the same measures, whereas QBoost performed marginally lower, differing only in the thousandth place. Figure 5.16 clearly shows the quality metrics for the three ML algorithms on the Car-Hacking dataset. Although AdaBoost

and RF perform slightly better than QBoost, the quality metrics for all these ML algorithms are very much similar. The accuracy scores for QBoost, RF, and AdaBoost are 0.994, 0.999, and 0.999, respectively. The scores for the other metrics are almost similar, for example, 0.997, 0.999, and 0.999 for Precision. Instead, considering the time efficiency for training and prediction time, AdaBoost outperforms both RF and QBoost in training and prediction times. AdaBoost generally outperforms RF since it consists of a forest of trees stumps rather than full-sized trees. Again, QBoost, inherently being a binary classifier, may not achieve the same effectiveness when it is implemented as a multi-class classifier.

Based on the results obtained from previous work, QBoost was used in multi-class ambient by modifying the preprocessing and the dataset. In this proposed work [59], called QUANTICAR (Quantum-based Automotive Threat Intelligence and Countermeasures), the SA-Hacking Dataset and the Survival Analysis Dataset were used. Only the Flooding and Malfunction attacks were considered for the Survival Analysis Dataset since they are the most analyzed attacks in the literature. This dataset was extended by considering the Fabrication attack taken from the SA-Hacking Dataset.

The QUANTICAR dataset consists of several attributes that are the same of the other state-of-the-art datasets. Briefly are:

1. **Timestamp**: recorded time(s).
2. **CAN ID**: the message identifier.
3. **DLC**: number of data bytes, from 0 to 8.
4. **DATA [0~7]**: data value.
5. **Flag**: 0 represents the normal messages, while 1 for flooding, 2 for fabrication, and 3 for malfunction.

Table 5.8 shows the total number of messages in the QUANTICAR dataset.

Label	Number of messages
Normal	809,660
Flooding	33,141
Fabrication	61,800
Malfunction	7,401

Table 5.8: Number of messages in the QUANTICAR Dataset.

The pre-processing was performed based on the work of Alfardus et al. [40]. In particular, the *CAN ID* and *DATA* were transformed from hexadecimal to decimal form. Then, the data fields are concatenated and transformed from decimal to float. Finally, the *Timestamp* and the *DLC* were removed. For the training and

Label	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Normal	0.9790	0.9997	0.9776	0.9885
Flooding		0.6427	1.0	0.7825
Fabrication		0.9878	1.0	0.9938
Malfunction		0.9927	0.8679	0.9261

Table 5.9: Results obtained with QBoost in a multi-class classification considering the QUANTICAR Dataset.

test phase, the concatenated dataset was split into 70% for the training and 30% for the testing.

Table 5.9 shows the results obtained from QBoost. The IDS was evaluated using the Accuracy, Precision, Recall, and F1-Score metrics. The results show that there is a decrease in the precision and F1-Score metrics for the Flooding compared with the other three classes (Normal, Fabrication, and Malfunction). Although it should be simple for a Machine Learning algorithm to identify this pattern, the precision metric reaches a score of 0.6427. In addition, some of the DoS samples were predicted as no attack. Instead, this rarely happens for the samples belonging to the Fabrication and Malfunction. This is probably because two features (CAN ID and DATA) are few to generalize correctly, and also, there are a high number of normal messages compared to the attack messages.

5.3.3 Comparison with the state-of-the-art

The state-of-the-art related to the proposition of Quantum ML algorithms is fairly novel. To propose new techniques that exploit hybrid Quantum computers, this Section proposed the identification of CAN threats using a Quantum Annealing algorithm called QBoost. This Section, with its unique perspective, also compares the time and identification performance of QBoost with traditional ML algorithms to analyze the impact of Quantum ML algorithms. Moreover, previous works have never analyzed those metrics. As a novelty, Section 5.3.2 proposes several works that use QBoost in a multi-class classification, which is a novel approach as QBoost originally provides binary classification. By making QBoost in multi-class, it is possible to exploit the power of Quantum Annealing to identify CAN attacks and then use the classification result to propose response and prevention approaches (see Chapter 6 and Chapter 7).

5.4 Detection techniques using ML algorithms

This section shows the techniques for the detection phase using traditional ML algorithms. The first method explores the use of the RF validated with the Car-Hacking Dataset. Then, it was tested by concatenating the SA-Hacking Dataset with the Survival Analysis Dataset using a multi-class classification [58]. Since the obtained results are prominent, the classification model is used for the other two proposed methodologies. The second proposed work aims to explain the importance of increasing cybersecurity education, which starts with the university using innovative research methodologies and then proposing strategies that could help the automotive industry [56]. Then, the proposed strategies are used to create a technology transfer model to increase the security and safety of vehicles, drivers, and passengers [57].

5.4.1 MaREA IDS

Starting from the first proposed work, called MaREA (Multi-class Random Forest for Automotive Intrusion Detection) [58], the model was validated with two works, [83] and [84], that performed a multi-class classification with the Random Forest using the Car-Hacking Dataset to prove that the proposed model outperformed state-of-the-art works. Then, the model was tested with a new dataset with new attacks by concatenating the Survival Analysis and the SA-Hacking Dataset (Table 5.11).

5.4.1.1 Experimental phase

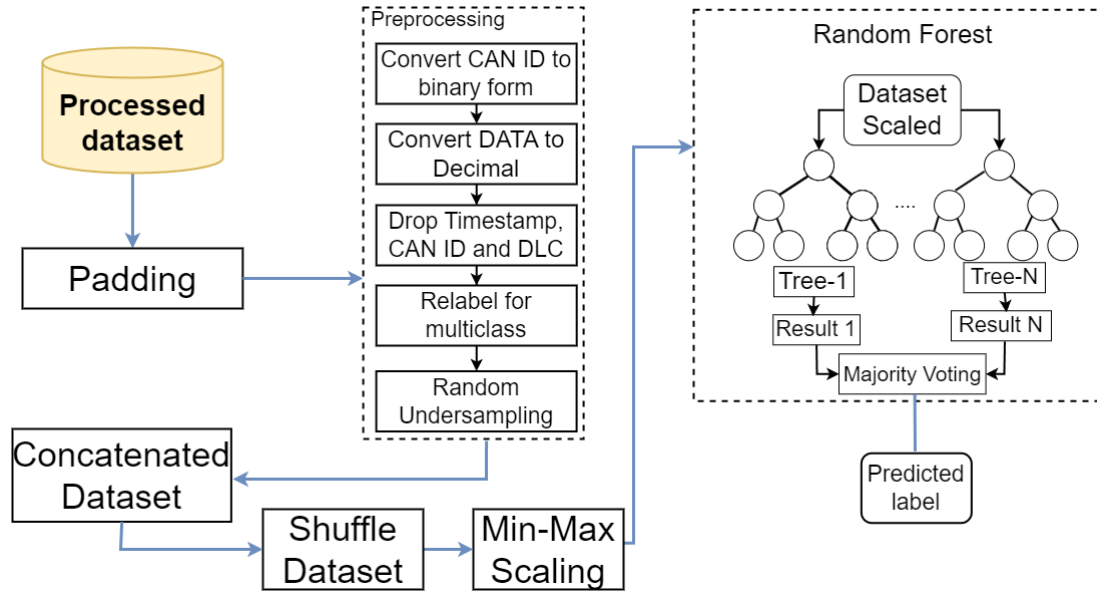


Figure 5.17: An overview of the proposed methodology.

Figure 5.17 shows the proposed methodology for MaREA IDS. The processed data set consists of the Car-Hacking, Survival Analysis, and SA-Hacking data set, where each of them is considered and processed individually. The padding process adds a "00" value at the end of the DATA attribute to remove any "Not a Number" (NaN) entries since all DATA attributes do not have a length of eight bytes. The padding algorithm takes the value contained in the DLC attribute and, if less than eight, adds the "00" values until it does not reach the length of eight. For example, if the DLC is four, to reach the length of eight, the algorithm will add '00' in the attribute from D4 to D7.

After padding, the preprocessing phase is performed, where the CAN IDs are transformed from hexadecimal to binary representation, obtaining eleven features. To clarify, the binary representation indicates that each CAN ID is first converted from hexadecimal to decimal form and then transformed into 0 and 1 because, for the standard identifier, it is a fixed 11-bit binary value. The DATA features were transformed from hexadecimal to decimal representation. After these phases, the following attributes were removed: *Timestamp* because MaREA does not perform a time series analysis; *CAN ID* was also removed because the model uses the binary form of the CAN ID and not the hexadecimal values; *DLC*, as the features are in a specific range from 0 to 8, it does not provide useful information for the model.

Label	Number of messages	%
Normal	2,331,517	50.00
DoS	587,521	12.60
Fuzzy	491,847	10.55
Spoofing Gear	597,252	12.81
Spoofing RPM	654,897	14.04

Table 5.10: Car-Hacking Dataset after the under-sampling.

Label	Number of messages	%
Normal	354,754	50.00
Flooding	33,141	4.67
Fuzzy	39,812	5.61
Malfunction	7,401	1.04
Fabrication	274,400	38.67

Table 5.11: Concatenated dataset (Survival Analysis Dataset and SA-Hacking Dataset) after the under-sampling.

As a multi-class classification is performed, the flag data are relabeled. For the Car Hacking Dataset, the labels were transformed by mapping T with 1 for DoS, 2 for Fuzzy, 3 for Spoofing Gear, 4 for Spoofing RPM, and R with 0 for normal messages. The labels of the Survival Analysis Dataset are transformed by mapping T with 1 for Flooding, 2 for Fuzzy, 3 for Malfunction, and R with 0 for Normal messages. The label for the normal message does not change in the SA-Hacking Dataset. Instead, the attack label is transformed into 4. For each dataset, a random under-sampling was performed to resolve the class imbalance (Table 5.10 and Table 5.11), and finally, all processed datasets were concatenated. Finally, since the features are not in a specific range, they are scaled in a range of $[0, 1]$ using the MinMaxScaler function. The hyperparameters of the RF were evaluated using a grid search method by considering every combination of the values. Grid search was performed with 5-fold cross-validation using GridSearchCV¹³.

Table 5.12 shows the values considered for the hyperparameters tuning for MaREA considering all datasets and the selected values for the Car-Hacking and Concatenated datasets. After hyperparameters were adjusted, the concatenated dataset was split into 70% for the train and 30% for the test.

¹³https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Hyperparameters	Considered	Car-Hacking dataset	Concatenated dataset
n_estimators	9, 20, 50, 100	9	100
max_features	sqrt, log2, None	sqrt	sqrt
max_depth	10, 15, 20, 25, 30, 40, None	20	40
min_samples_split	2, 5, 10	2	5
min_samples_leaf	1, 2, 4	1	1

Table 5.12: Hyperparameters considered and the values selected by the grid search for MaREA considering the Car-Hacking and Concatenated dataset.

5.4.1.2 Evaluation results

Reference	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[83]	0.94	0.93	0.93	0.93
[84]	0.97	0.90	0.99	0.94
MaREA	0.99	0.99	0.99	0.99

Table 5.13: Comparison with other approaches that use Random Forest and the Car-Hacking Dataset as a testbed.

Precision, recall, F1 score, and accuracy metrics were used to compare the results. Table 5.13 shows that MaREA outperforms [83] and [84] in terms of the aforementioned metrics using a different pre-processing phase. Instead, with [84], MaREA obtains similar recall values.

Table 5.14 shows the results comparing MaREA with [6] using the Survival Analysis Dataset. MaREA outperformed the LSTM model proposed by [6] for all types: normal, flood, fuzzy, and malfunction messages. However, both models obtained the same results for the flooding attacks, probably because the attack pattern for the flooding is easy to identify. MaREA obtained the best results for the Fuzzy and Malfunction attacks for the Recall and F1-Score metrics. In particular, the fuzzy attack is the most challenging, as the CAN IDs are randomly generated. Moreover, we can argue that the model is appropriately generalized by obtaining a low misclassification since MaREA is also trained in combining the Survival Analysis Dataset and the SA-Hacking Dataset.

5.4.2 Cybersecurity educational framework

Cybersecurity education plays an important role in helping data protection in different industries and public offices by training and preparing security specialists. Although industries conduct cybersecurity training from time to time, this

Reference	Model	Label	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[6]	LSTM	Normal	0.9971	N/A	1.0	0.9994
		Flooding		N/A	1.0	1.0
		Fuzzy		N/A	0.9469	0.9701
		Malfunction		N/A	0.9697	0.9087
MaREA	Random Forest	Normal	0.9997	0.9999	1.0	0.9999
		Flooding		1.0	1.0	1.0
		Fuzzy		1.0	0.9956	0.9978
		Malfunction		0.9959	1.0	0.9979
		Fabrication		0.9994	1.0	0.9997

Table 5.14: Comparison with a multi-class approach using the Survival Analysis Dataset as a testbed. The fabrication attack is reported only for MaREA. The precision score is not included in [6].

is primarily provided by the Universities through research and collaborations. Figure 5.18 presents the overall goal and the strategies proposed in this work regarding cybersecurity education.

5.4.2.1 Proposed methodology

Goal	Objective	Strategy
1. Raise awareness about risks in automotive	1.1 Improve knowledge of risks and vulnerabilities in automotive	1.1.1 Promote cyber security awareness through collaboration between academia and industry
		1.1.2. Promote strategies for training the company to manage cyber risk
	1.2 Promote the use of cyber security resources and tools	1.2.1 Integration of different stakeholders to understand the risks of different components involved in automotive industry
		1.2.2 Improve cyber security-focused activities

Figure 5.18: Goal and Strategy of Cyber Security Education in Automotive Industry.

The principal goal is to educate and train industry resources to raise automotive risk awareness. To achieve this, the goal is divided into objectives and subsequently into actionable strategies. To implement the strategies, the National Initiative for Cybersecurity Education (NICE) framework [85] was used. The principal building blocks of the NICE framework are the Tasks, Knowledge, and Skills (TKS) statements, which incorporate agility, flexibility, interoperability, and

modularity as important attributes. In this proposed work, the NICE framework (Figure 5.19) can be utilized to impart **Knowledge**, education and training on automotive cybersecurity domain, and **Skill**, technical and business skills to remediate automotive vulnerabilities, to perform the **Task** of implementing effective and accurate security measures to remediate automotive cyber-attacks.

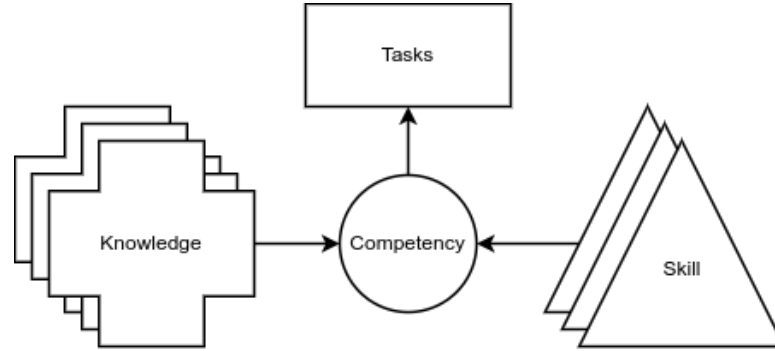


Figure 5.19: Adaptation of NICE framework.

With reference to the proposed educational strategies and the NICE framework, Figure 5.20 shows the proposed methodology, where the essential elements for cyber security education start from the University and terminate in the automotive industry with a possible stakeholder involved to manage the automotive cyber-attacks. Firstly, the university investigates the possibilities of new automotive cyber attacks on next-generation vehicles. For example, information about new attack typologies that may be used to attack vehicles can be used as necessary domain knowledge (Figure 5.18, **Strategy - 1.1.1**) and can be used in education and training. In particular, understanding what kind of cyber attack is occurring in the vehicle is essential to increase the competence and the skills of the stakeholders that operate in the automotive field. The use of Machine Learning algorithms could be a solution for automotive companies to manage cyber risk and attacks (Figure 5.18, **Strategy - 1.1.2**).

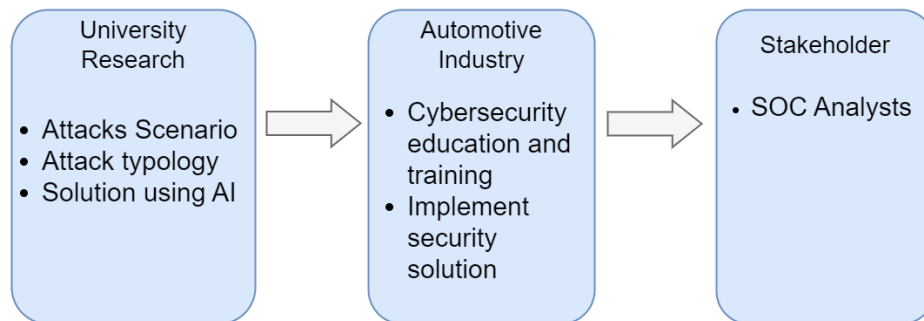


Figure 5.20: Proposed methodology for Cyber Security education.

A multi-class IDS has been proposed as a solution to identify the attacks and

protect the in-vehicle network. The proposed solution could be extremely useful in identifying the typology of the attacks and helping the automotive industry to adapt response solutions based on the attack type. Considering Figure 5.20, for example, the SOC Analyst, based on the attack that occurred in the vehicle, can make decisions on how to respond to the attack based on his knowledge and skill. Imparting the knowledge and understanding of these outcomes across the organization could ensure cybersecurity awareness among resources like developers, testers, and architects (Figure 5.18, **Strategy - 1.1.2 and 1.2.1**). Furthermore, the industry can use the knowledge and skills necessary for cybersecurity to develop new competencies in the employees to spread and equip specialists for specific cybersecurity tasks (Figure 5.18, **Strategy - 1.2.2**). The automotive industry, apart from the University led research and development programs, also should strictly adhere to the ISO/SAE 21434 regarding the secure development process and hardware component. Thus, the industry can assimilate innovative cybersecurity solutions to learn to protect particular components from cyber attacks by continuous monitoring, surveys, and deployment of novel onboard security systems.

5.4.2.2 Multi-class IDS solution

In the preprocessing phase, the DATA field was padded because it contains NaN values. After this phase, the DATA and CAN ID were transformed from hexadecimal to decimal. Then, the Car-Hacking Dataset was merged and relabelled based on multi-class classification. The dataset was scaled using the Min-Max strategy, and the PCA was applied. The default parameters were used for Random Forest. Finally, the concatenated dataset was split into 70% for training and 30% for testing. Accuracy, Precision, Recall, and F1-Score were used to evaluate the Random Forest model, and the results are shown in Table 5.15. The proposed approach obtained 100% for the DoS, Spoofing GEAR, and Spoofing RPM. For the Fuzzy attack, instead, the model reached 99.98% due to the randomness of the CAN ID and DATA, which leads the model to classify the normal messages as Fuzzy sometimes.

Model	Attack	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	Normal	99.99	99.99	99.99	99.99
	DoS		100	100	100
	Fuzzy		99.98	99.99	99.98
	Spoofing GEAR		100	100	100
	Spoofing RPM		100	100	100

Table 5.15: Results obtained using the Random Forest classifier for the concatenated dataset in multi-class approach.

5.4.3 Cybersecurity technology transfer

The propagation of ML-based IDS approaches from laboratory to market is low due to the complex algorithms and implementation difficulties. Therefore, we require better solutions to accommodate the advanced research in IDS using ML in the Technology Transfer (TT) process. TT has been ideally used to help the technology flow from the laboratory to industry, developed to developing countries, or put the information to diverse usages. TT processes encompass diverse phases of technological innovation and its dissemination until commercialization, for example, developing a commercialized technology developed by university-based research in collaboration with an external company. The next step from the previously analyzed approach is to transfer the technology, specifically the IDS, by testing it using a testbed [57].

5.4.3.1 Methodology

Technology Transfer refers to the process of conveying results stemming from scientific and technological research to the marketplace and to wider society, along with associated skills and procedures¹⁴. To achieve this goal, Figure 5.21 shows the essential elements of the technology transfer process in automotive industry.



Figure 5.21: Automotive Technology Transfer.

In the first phase (University Research), the idea to increase automotive security using Artificial Intelligence (AI) based on a machine learning multi-class IDS was presented. Then, regarding the channels to transfer the proposed solution (Technology Transfer), the following were identified: research projects, doctoral research, consultancy, publications, and meetings. Finally, the impacted stakeholders were identified as the driver, automotive company and workshop, insurance, and all technology, electronics, and telecommunications companies involved in hardware and software integration [86].

¹⁴https://knowledge4policy.ec.europa.eu/technology-transfer/what-technology-transfer_en

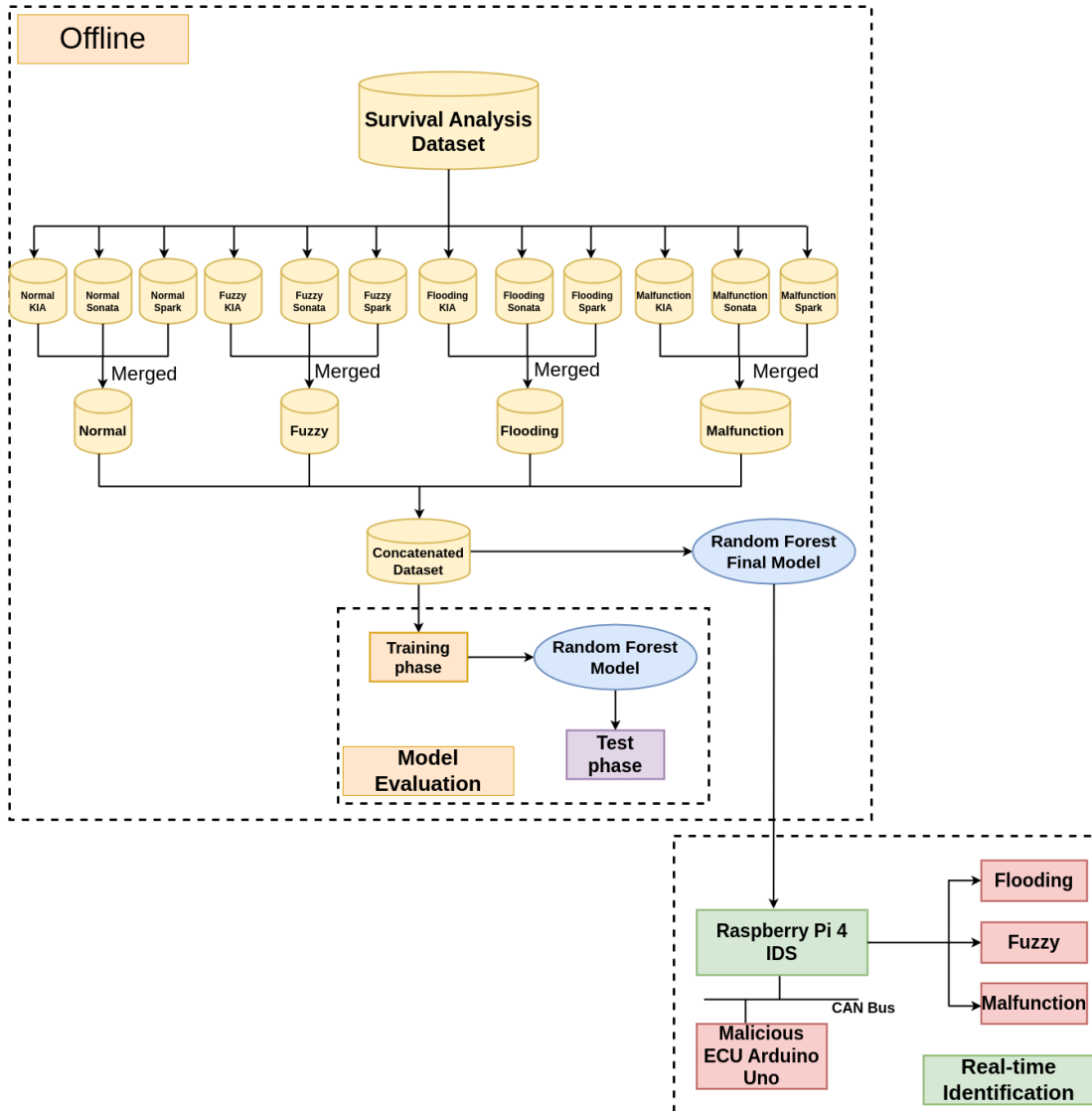


Figure 5.22: Methodology phase for the Technology Transfer.

The proposed IDS aims to design an approach to disseminate a laboratory setup for identifying CAN attacks to be available for commercial use in an automotive company. This is also in accordance with the ISO/SAE 21434 regarding the secure development process and hardware components in the automotive field. To do this, a model using the RF algorithm was trained and tested using the Survival Analysis Dataset, and then, the final model was deployed into a Raspberry Pi 4 to be tested in real-time. The training and testing phase was performed offline because, in general, an ECU has limited computation power. The entire concatenated dataset (Figure 5.22) will be used with the parameters found in the evaluation model to create a final RF model since using the single training set will lose additional information. This final model will be put into the IDS ECU,

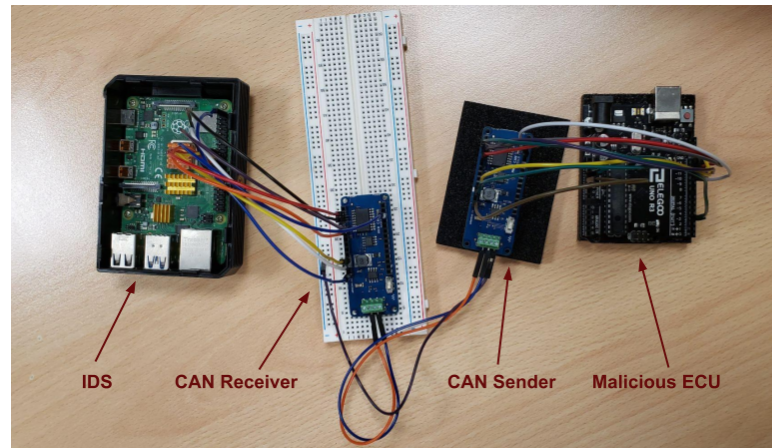


Figure 5.23: Laboratory setup for Technology Transfer.

whose purpose is to perform the identification phase, capturing the messages from the CAN bus and verifying if the messages sent are anomalous.

To simulate an in-vehicle network, Raspberry Pi 4 Model B¹⁵ with 4 GigaByte of RAM, Arduino MKR CAN Shield¹⁶, and Arduino Elegoo Uno R3¹⁷ were used. The Arduino Elegoo Uno R3 simulated an attacker who injects malicious messages on the CAN bus. Two Arduino MKR CAN Shields were used wherein one sent the messages, and the other received those and transmitted them to the IDS. The Raspberry Pi 4 Model B was used to simulate an IDS, which receives messages from the CAN bus (Receiver Arduino MKR CAN Shield) and performs the classification of messages according to the attacks defined in the Survival Analysis Dataset. Figure 5.22 shows the real-time analysis in which the malicious ECU (Arduino Elegoo Uno R3) sent messages on the CAN bus using the MKR CAN Shield. The Raspberry Pi 4 Model B checked these messages and predicted the attack category. Figure 5.23 shows the laboratory setup used to emulate the in-vehicle CAN Bus IDS. The MKR CAN Shield consists of a microchip MCP2515, a second-generation stand-alone CAN Bus capable of transmitting and receiving the CAN message [87]. The Raspberry Pi 4 Model B allows a connection with a can0 interface that receives data from a physical component emulating an actual CAN protocol. The bitrate of the can0 interface was set to 500000 because it is the speed supported by the hardware.

Prediction, Recall, and F1-Score were used as metrics to evaluate the Random Forest and the injected malicious messages. Table 5.16 shows the results using the traditional machine learning metrics for the Random Forest model and injected messages. Mostly, the predictions about the attack types were correct, apart from tiny aberrations that appeared in the results because of the randomness of the

¹⁵<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>

¹⁶<https://store.arduino.cc/products/arduino-mkr-can-shield>

¹⁷<https://www.elegoo.com/en-it/products/elegoo-uno-project-super-starter-kit>

DATA field. For instance, the Random Forest misclassified 0.014% of Normal messages as Fuzzy, 0.01% of Malfunction messages as Fuzzy, and 0.016% and 0.052% of Fuzzy messages as Normal and Malfunction, respectively. On the other hand, the IDS always classified the flooding messages correctly.

Technique	Class	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	Normal	99.98	99.99	99.98	99.99
	Flooding		100	100	100
	Fuzzy		99.79	99.93	99.86
	Malfunction		99.90	99.98	99.94
Injected Messages	Flooding	85.71	100	100	100
	Fuzzy		57.14	100	72.72
	Malfunction		100	25	40

Table 5.16: Results obtained with Random Forest and Arduino Elegoo Uno R3 (Injected Messages).

In this case, the Arduino Elegoo Uno R3 only injected malicious messages to simulate the attack and validate the IDS when a vehicle is under attack. As seen in Table 5.16, 75% of Malfunction messages were misclassified as Fuzzy. In the case of Malfunction messages, the CAN IDs are reused from the existing ones in the dataset, and the DATA field was randomly generated. In addition, the Malfunction DATA field was randomized to avoid model overfitting. Since the Fuzzy DATA field is also generated randomly, the model probably misclassifies a Malfunction attack as Fuzzy.

5.4.4 Comparison with the state-of-the-art

As discussed in Section 3.4, several research papers use DL algorithms to identify CAN attacks. However, since these models are computationally expansive, more lightweight models are necessary. In addition, no work attempts to implement the proposed model within embedded components or that can be compared to ECUs. A cybersecurity educational framework proposes to improve the state-of-the-art and help the organization correctly define cybersecurity inside the software and hardware components by also proposing an IDS to detect attacks. The technology transfer approach defines a testbed that can be used as an example for researchers to conduct experiments but also to prove the possibility of deploying an ML model inside a component like the Raspberry Pi.

5.5 Detection technique using a Hybrid approach

This section introduces a novel approach and innovative architecture that leverages QML to expedite the prediction of security incidents and intrusions, thereby mitigating potential damage and minimizing their impact on interconnected systems [62]. Consequently, this represents a significant advancement in addressing security and privacy concerns for Smart City residents and re-engineering traditional systems that integrate quantum components.

The envisioned architecture is characterized as a Hybrid-Quantum Computing system, bridging the realms of classical and quantum computing components. Within this framework, QML algorithms serve as a key component, accessible as a service from an external classical component. This proposed architecture amalgamates two distinct security implementation systems—QRadar and the DWave Leap Quantum Computing platform.

The envisioned system consists of a robust three-layer architecture meticulously organized into five core components for optimal functionality. These components include: (a) *Smart City Data Source*, (b) *Dashboard*, (c) *Principal Application*, (d) *Database*, (e) *Quantum Cloud Service* and (f) *QRadar - SIEM*. In this context, the presentation, application, and data tiers were developed as distinct infrastructures, offering increased security and modularity. Each of these components contributes uniquely to the overall functionality of our system, ensuring a robust and comprehensive approach to data processing and security in the Smart City environment. The details of each of the components are presented in the following sections.

5.5.1 Smart City Data Source

The Smart City data source, as depicted in Figure 5.24.a, represents the intelligence hub within the urban landscape. It serves as the nexus for seamless information exchange among various interconnected Smart City components, all aimed at enhancing the quality of life for city residents. These data sources are distributed throughout the city, forming a vital network that fuels the system's operations. Despite their physical location external to the system, they are extremely vital to the entire ecosystem. Data emanating from different facets of the Smart City arrives in various formats, which depend on the type of log source or data source integrated into the system. Generally the data is received in diverse formats like Common Event Format (CEF), commonly used for network and security-related logs, Syslog (protocol for sending event messages and log data), SON (JavaScript Object Notation) and XML (eXtensible Markup Language), SNMP (Simple Network Management Protocol), generated by network devices, database logs, and flat files such as CSV (comma-separated values) files among other.

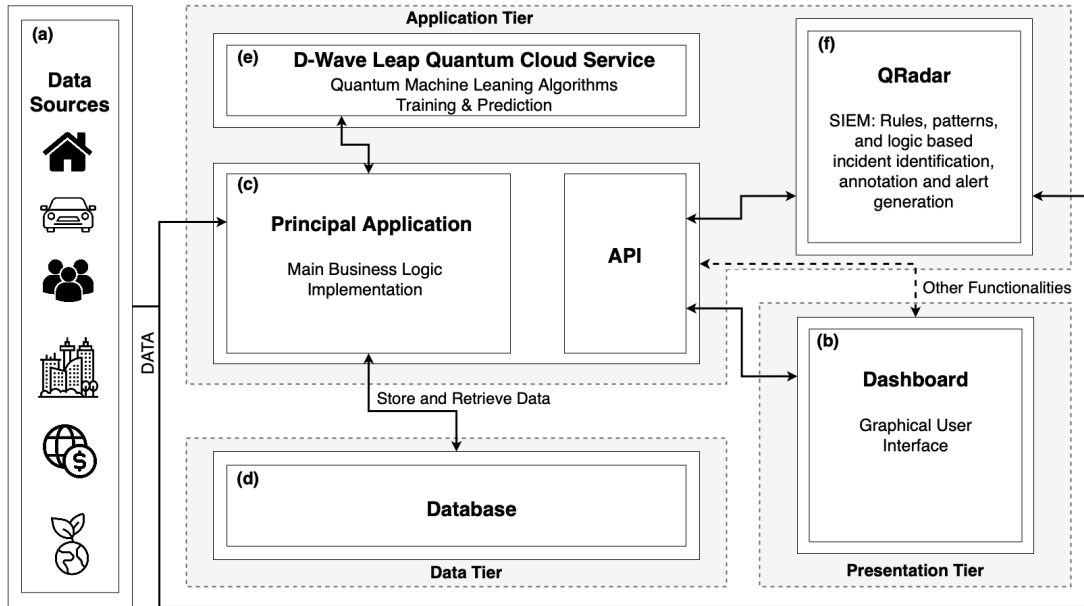


Figure 5.24: Quantum Hybrid Architecture of Smart City Security.

5.5.2 Smart City Security Dashboard

The proposed system introduces a dynamic graphical user interface (GUI), as exemplified by the interactive dashboard featured in Figure 5.24.b. This GUI serves as a real-time platform for monitoring and responding to a wide range of security incidents and attacks unfolding across various sectors within the Smart City infrastructure. The dashboard is intelligently structured so that different incidents associated with each Smart City sector can be accessed through filters on the respective headers, presenting them in an accessible tabular format. In the spirit of immediate responsiveness, the dashboard promptly updates upon incident registration, furnishing comprehensive details regarding the encountered threat.

Each threat is either assigned a unique identifier if they do not come assigned with one, or else they carry forward their own unique identifiers. Further, the threats or offenses are also characterized by their timestamps. The dashboard's data stream is directly sourced from a RESTful API, which, in turn, retrieves data from two different sources: the database tables and the QRadar data output API. This API functions as the GUI's primary data conduit, presenting information in a format that allows security experts to swiftly evaluate and counter threats.

The Integrated Security Dashboard for Smart City displays various offenses/intrusions encountered in Smart City data. Offense list for the data analyzed by DWave and QRadar are displayed below in a tabular manner for ready reference. The dashboard receives the data from both DWave QML interface and QRadar and displays the data below. The provided filters can be used to view the data easily. The data columns can also be sorted in order arrange the data in ascending or descending order for ease of viewing.

Incident ID	Offense Source	Incident Date & Time	Severity (1-10)	Attack Type	Data Source
20	192.168.137.64	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
19	192.168.137.89	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
18	192.168.137.225	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
17	192.168.137.90	2023-09-21 19:54:00	5	IoT: DDoS Attack	QRadar
16	192.168.137.139	2023-09-21 19:54:00	5	DDoS Attack Detected preceded by IoT: DDoS Attack	QRadar
15	192.168.137.60	2023-09-21 19:54:00	5	IoT: DDoS Attack	QRadar
190428092820WcgMa	IoT_Weather	2019-04-28 09:28:20	No Info	Attack State	DWave
190428092820PLWVP	IoT_Weather	2019-04-28 09:28:20	No Info	Attack State	DWave
190428092823zeIxa	IoT_Weather	2019-04-28 09:28:23	No Info	Attack State	DWave
190428092825uvahM7	IoT_Weather	2019-04-28 09:28:25	No Info	Attack State	DWave

Items per page: 10 1 - 10 of 16

Refresh Table Reset Filter

Figure 5.25: Integrated Smart City Security Dashboard.

Figure 5.25 serves as an illustrative sample of the dashboard, a pivotal component within our proposed architecture, spotlighting real-time alerts stemming from incoming Smart City data. The dashboard systematically organizes attack incidents in rows, each tagged with a distinct incident ID and supplemented with relevant particulars such as incident timestamp, severity (when available), category, and attack type (also when available). In addition to this, the GUI allows security experts to make decisions based on the attacks they receive. For example, considering the dashboard in Figure 5.25, the *Incident IDs* from 15 to 20 correspond to the threats identified by IBM QRadar SIEM. In this case, a security analyst can use the **Offense Source** IP to block the attacker and avoid the Distributed-Denial of Service (DDoS) attack (Attack Type column). On the other hand, for the attacks identified by the IDS, the security analysts could obtain the source IP from DWave (based on the prediction) and reconstruct the attack chain to identify the attacker IP and block the ones sending malicious content. In this preliminary work, the dashboard only shows information about the attack prediction with the **Offense Source** using a group-based typology of the type of attack, for instance, *IoT_Weather* in the dashboard screenshot (Figure 5.25), owing to the limitation of the dataset in consideration.

5.5.3 Principal Application

At the core of the proposed architecture, as depicted in Figure 5.24.c, lies the main application, a critical component responsible for processing incoming incidents from various Smart City sectors. Employing a well-structured IDS pipeline, the application methodically sifts through this influx of data to pinpoint potential

security incidents, a process illustrated in Figure 5.26. Subsequently, a sequence of operations extracts and prioritizes the most pertinent features for further analysis. The system then embarks on the training of machine learning models, subjecting them to rigorous validation before entering the prediction phase. During this predictive stage, test data is meticulously processed and subjected to classification by the trained ML models. The outcomes of these predictions are then compared against existing data to gauge the efficiency of the models, a task often assessed through popular quality metrics such as accuracy, precision, recall, or the F1-score.

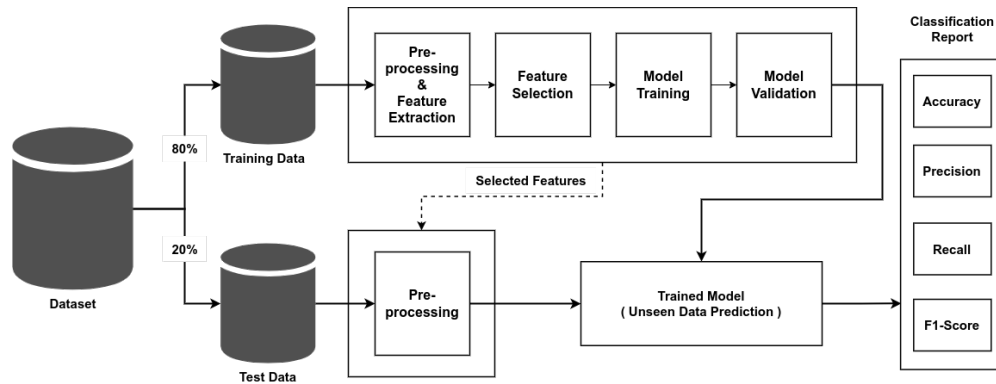


Figure 5.26: IDS Pipeline.

Within the architectural framework, the training and testing of data are conducted within the D-Wave Leap Quantum Cloud (DLQC) (Figure 5.24.e), which is external. After the training and validation stages, QML models are preserved locally ready for use in real-time predictions. When the main application initiates real-time predictions, the prediction model is loaded into the Quantum Cloud.

The main application is capable of processing single or batch incident predictions received from data sources. Quantum Cloud assumes responsibility for incident-type predictions, leveraging QML algorithms to distinguish between normal and malicious incidents. Classification results are relayed back to the main application via the token-based API, subsequently updating the database with the most recent attacks, threats, or security incidents.

To maintain the model's effectiveness, the proposed system consistently trains QML models using a diverse array of datasets, including those sourced from reputable organizations or institutions. The best-performing model, determined through multiple runs, is retained for real-time predictions. Ensuring secure and efficient data access, our system employs a RESTful API interface, enabling seamless communication with the database instance. This API features distinct endpoints for data retrieval from various database tables and sources, converting the data into the JSON format for easy data fetch by the dashboard.

5.5.4 Database

Situated at the core of the proposed architecture, the database (DB) depicted in Figure 5.24.d serves as the basis of our data management infrastructure. It acts as the repository for the information streamed from our diverse data sources. Each sector of the Smart City contributes its unique data, which is meticulously organized into dedicated DB tables. These tables, akin to specialized vaults, are tailored to accommodate their respective sectors' distinctive features and information. This structured approach ensures not only efficient data management but also facilitates quick retrieval.

5.5.5 Quantum Cloud services

The D-Wave Leap Quantum Cloud (DLQC) Service, as delineated in Figure 5.24.e, occupies a pivotal role within the proposed architectural framework, affording access to quantum computing capabilities via a secure native API mechanism, fortified by robust token-based encryption for authentication. DLQC harnesses the computational prowess of Quantum Unconstrained Binary Optimization (QUBO) algorithms, a cornerstone of quantum computing. This utilization extends not only to the training of quantum models but also to predicting outcomes, thus harnessing the computational supremacy intrinsic to quantum paradigms. The foundation of the architectural design is supported by the strategic deployment of QBoost.

5.5.6 QRadar - SIEM

IBM QRadar (Figure 5.24.f) forms another vital part of the main application, where it effectively incorporates external threat intelligence data to bolster its threat detection and incident response capabilities. Custom correlation rules trigger alerts when QRadar detects a potential threat or matches it with external threat indicators. Risk scores and severity levels are assigned to incidents based on this data, helping security teams prioritize their responses effectively. The pattern and rules-based risk and threat identification are shared with the users over a secure API local to the QRadar implementation site.

5.5.7 Experiments

To validate the proposed classical Hybrid solution, IBM QRadar SIEM was used by creating rules to identify the attacks and an IDS using QBoost with a new state-of-the-art dataset called CICIoT2023 [77]. The CICIoT2023 dataset consists of 34 labels (See Section 5.1.5), among which 33 are different types of IoT attacks executed in an IoT topology of 105 devices, and one label for *Benign-Traffic* (normal data). These attacks are classified into seven categories, namely, DDoS, DoS, Recon, Web-based, brute force, spoofing, and Mirai.

Since the total number of labels is 33, only a part of these attacks to create the rules in IBM QRadar SIEM was taken. In particular, the DDoS, DoS, Recon, and Web-Based were considered. For the category DDoS and DoS, the HTTP-Flood has been considered. For the Recon, the nmap port scanning. Finally, for the Web-Based, XSS and SQL Inject attacks were considered. The aim of this proposed approach is not to scrutinize the functionality of the involved systems but rather to propose a hybrid quantum-classical architecture integrating a quantum-based IDS and a classical SIEM. Additionally, the selected attacks are commonly observed in real-world contexts.

On the other hand, in the case of DLQC, as QBoost happens to be a binary classifier, the 33 different kinds of attacks are considered as a single *attack* class, and the normal data is paired against the attack data as the normal class for training and resting the IDS.

5.5.7.1 QBoost validation

To validate the functionality of the proposed architecture, the CICIoT2023 dataset was subject to the following steps to prepare it for machine learning analysis, using the QBoost QML algorithm from the DLQC:

1. **Categorical Feature Removal:** Categorical features representing non-numeric data were removed from the dataset. This was done to reduce the dimensionality of the data, as dealing with high-dimensional data can potentially reduce the quality of prediction models.
2. **Redundant Feature Removal:** Features with the same values across all instances were removed from the dataset. These features do not contribute to the prediction process and were thus eliminated to simplify the data.
3. **Handling Missing Values:** Any instances containing Infinity or NaN (Not-a-Number) values were processed by replacing them with the median values. This step ensures that the data is complete and suitable for analysis.
4. **Target Variable Transformation:** The dependent feature, which is the target variable, underwent a transformation. Normal or non-attack messages were assigned a value of -1, while attack messages were assigned a value of 1. This transformation is essential for compatibility with quantum machine learning algorithms.
5. **Data Normalization:** The dataset's independent variables, or features, were normalized using the *MinMaxScaler*.
6. **Balancing the Dataset:** To balance the dataset, a combination of techniques was applied, including Synthetic Minority Over-sampling Technique

(SMOTE) and Random Under-Sampling, and the best parameters for balancing were decided by *CrossValidation* of the mean recall scores for the **DecisionTree** classifier. These methods were employed to achieve a more equitable distribution of class instances, ensuring that both attack and non-attack classes are adequately represented. After balancing, the resulting dataset was reduced to 10.257.385 rows.

7. **Dimensionality Reduction:** To mitigate the challenges posed by a high-dimensional dataset (consisting of 46 features), PCA was applied. The threshold method was used to retain the top 13 most relevant features.
8. **Data Splitting:** Finally, the processed and scaled data were split into training and testing datasets in an 80% to 20% ratio [77]. This division is essential for evaluating the model's performance and ensuring that it generalizes well to unseen data.

To this end, the performance of QBoost was compared with that of RF. This comparison aimed to provide a holistic view of the strengths and weaknesses of both quantum and classical approaches in the context of Smart City Security.

For the sake of comprehensiveness and reproducibility, the hyperparameters are provided herewith:

1. **SMOTE Hyperparameters:**

- (a) *sampling_strategy*: 0.1, indicating the desired ratio of the number of synthetic samples to be generated for the minority class compared to the majority class.
- (b) *k*: 7, determining the number of nearest neighbors considered during the synthetic sample generation process.

2. **Random Undersampling Hyperparameters:**

- (a) *sampling_strategy*: 0.8, presenting the desired ratio of the number of instances to be retained for the majority class compared to the minority class.

3. **RepeatedStratifiedKFold Configuration:**

- (a) *n_splits*: 10, indicating the number of splits or folds in the cross-validation process.
- (b) *n_repeats*: 3, denoting the number of times the cross-validation process was repeated.
- (c) *n_jobs*: -1, enabling parallel processing when calculating cross-validation scores.

4. PCA Configuration:

- (a) *Threshold method*: Relies on the explained variance ratio.
 - (b) *Threshold value*: 0.99, indicating the proportion of variance that the selected principal components should collectively account for.
5. **QBoost λ value (lam)**: 0.085, introduced to control the complexity of the model and reduce the risk of overfitting.

5.5.7.2 IBM QRadar SIEM validation

DoS Attack

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal functioning of a network, service, or website by overwhelming it with a flood of traffic. The goal of a DDoS attack is to make the target system or network unavailable to its intended users. In the CICIoT23 dataset, the authors utilized multiple Raspberry Pi devices in an SSH-based master-client configuration to execute these attacks.

A series of building blocks were created to create the rules that identify a DDoS attack, followed by the main rule that aggregates the building blocks to identify the attack. A building block (BB) group commonly uses tests to build complex logic so that it can be used in rules.

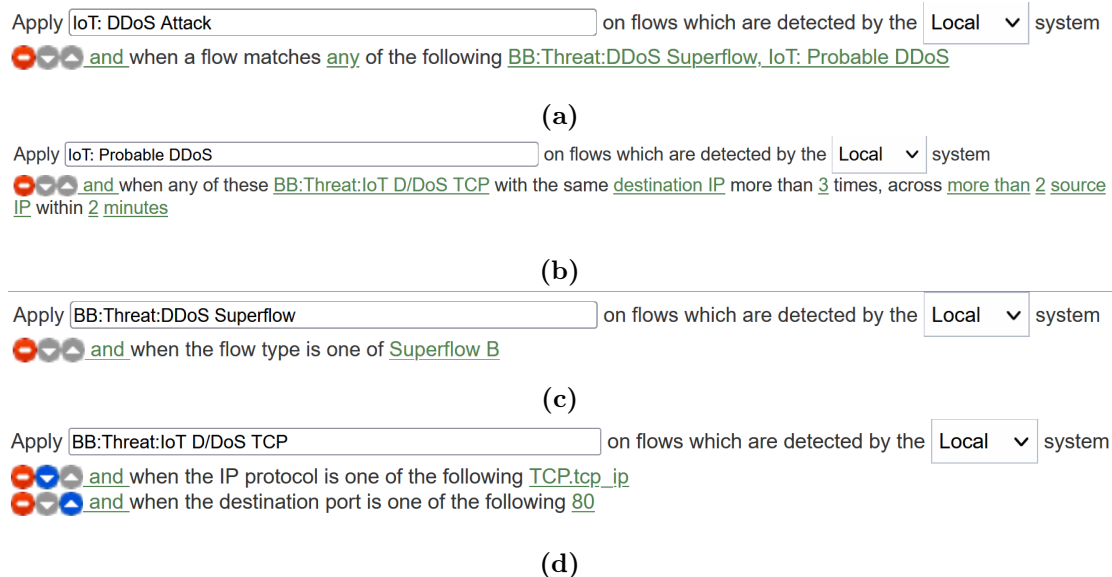


Figure 5.27: The pictures show the rules to identify an HTTP Flood attack with the category DDoS. Figure (a) indicates the main rule. If one of the rules indicated in (a) is positive, it will raise an offense. Figures (c) and (d) show the BB rules.

Description	Offense Type	Offense Source	Magnitude	Source IPs	Destination IPs	Us	Lo So	Events
IoT: DDoS Attack	Source IP	192.168.137.90		192.168.137.90	192.168.137.139	N/A	Cus	113
IoT: DDoS Attack	Source IP	192.168.137.225		192.168.137.225	192.168.137.139	N/A	Cus	104
IoT: DDoS Attack	Source IP	192.168.137.89		192.168.137.89	192.168.137.139	N/A	Cus	118
IoT: DDoS Attack	Source IP	192.168.137.64		192.168.137.64	192.168.137.139	N/A	Cus	102

Figure 5.28: Offense when a DDoS attack is identified.

Figure 5.27 shows the rules created to identify an HTTP Flood attack with the category DDoS. Figure 5.27d and Figure 5.27c show the BB rules. In particular, Figure 5.27d is used to filter the flow network based on the TCP protocol and port 80 since the HTTP Flood is performed by sending a GET request on port 80. Figure 5.27c shows the Superflow characteristics. IBM QRadar groups the individual flows into a Superflow. There are three types of Superflow: A (Network Scan), B (DDoS), and C (Port Scan) [88]. QRadar looks for flows where many hosts send data to one destination host and flags this activity as Superflow B.

Figure 5.27b is used to identify two or more source IPs that send a high number of data to a single destination IP in two minutes. Finally, the rule *IoT: DDoS Attack* (Figure 5.27a) aggregates the Superflow and the *Probable DDoS*.

Figure 5.28 shows the offense generated when IBM QRadar SIEM identifies an HTTP Flood attack with a DDoS category. In the generated offense, it is possible to see that there are have multiple *Offense Source* and *Source IPs* that correspond to the Raspberry Pi that performs the attack to one *Destination IP*. The events column indicates the number of event flows where a DDoS attack was identified. Since the CICIOT2023 does not provide the list of IP devices, only the Media Access Control (MAC) addresses, this information is used to understand the attack messages by analyzing the PCAP file.

Cross-Site Scripting (XSS) Attack

Cross-Site Scripting (XSS) is a type of web security vulnerability that occurs when a web application allows users to inject malicious scripts into web pages viewed by other users. XSS attacks are one of the most common security vulnerabilities found in web applications. They can have serious consequences, including data theft, session hijacking, and defacement of websites [89]. To perform this attack, the researchers used DVWA [90], a vulnerable PHP/MySQL web application usually employed to test penetration testing techniques or for education purposes.

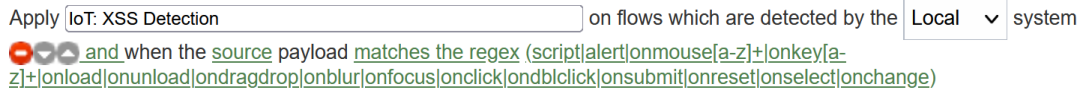


Figure 5.29: Rule to identify an XSS attack.

To create the rule that identifies an XSS attack, the PCAP file was first analyzed. Most messages relate to basic XSS injection techniques, such as using

the tag *script* with *alert*. Figure 5.29 shows the rule that matches several tags generally used to perform an XSS attack with a *regular expression*.

Description	IoT: XSS Detection	Offense Type	Source IP
		Event/Flow count	<u>20 events</u> and <u>0 flows</u> in 1 categories
Source IP(s)	<u>192.168.137.147</u>	Start	Sep 20, 2023, 5:36:03 PM
Destination IP(s)	<u>Local (3)</u>	Duration	39m 47s
Network(s)	<u>Net-10-172-192.Net 192.168.0.0</u>	Assigned to	<u>Unassigned</u>

(a)

Description	Offense Type	Offense Source	Magnitude	Source IPs	Destination IPs
IoT: XSS Detection	Source IP	192.168.137.147	■■■	192.168.137.147	Local (3)

(b)

Figure 5.30: Figure (a) shows information about the offense when an XSS attack is detected. Figure (b) shows more information about the offense.

Figure 5.30a and Figure 5.30b show the generated offense when an XSS attack is identified by QRadar using the rule in Figure 5.29.

SQL Injection Attack

A SQL injection is a web security vulnerability that occurs when an attacker is able to manipulate the input that is sent to a web application's SQL database. This manipulation can lead to unauthorized access, modification, or extraction of data from the database [91]. As with the XSS attack, the researchers used the DVWA application to perform the SQL Injection attack.

Apply	IoT: SQL Injection	on flows which are detected by the	Local	▼	system
and	when the	source	payload	matches the regex	(UNION ALL SELECT OR AND CONCAT WHEN SELECT FROM NULL SLEEP)
and	when the	destination	port	is one of the following	80

Figure 5.31: Rule to identify a SQL Injection attack.

First, the PCAP file regarding the SQL Injection was analyzed, noticing that this attack had been performed using basic SQL Injection with UNION, SLEEP, and so on. So, based on this, a rule was created that contains the most common SQL fields used in an SQL Injection attack. Figure 5.31 shows the rule with the regular expression created.

Instead, Figure 5.32 shows the generated offense. In particular, the field *Local (4)* is related to the number of the destination IPs involved in the attack.

Description	Offense Type	Offense Source	Magr	Source IPs	Destination IPs	Use	Log Sou	Events
IoT: SQL Injection	Source IP	192.168.137.178	■■■	192.168.137.178	Local (4)	N/A	C...	170

Figure 5.32: SQL Injection Offense generated.

Description	IoT SQL Injection	Offense Type	Source IP
Source IP(s)	192.168.137.178 (192.168.137.178)	Event/Flow count	170 events and 0 flows in 1 categories
Destination IP(s)	Local (4)	Start	Sep 29, 2023, 5:09:00 PM
Network(s)	Net-10-172-192-Net_192_168_0_0	Duration	48m 59s
		Assigned to	Unassigned

List of Local Destination IPs											
	Destination IP	Magnitude	Location	Vulnerability	User	MAC	Weight	Offenses	Source(s)	Last Event/Flow	Events/Flows
	192.168.137.4		Net-10-172-192...	No	?	?	0	4	1	51m 10s	209
	192.168.137.217		Net-10-172-192...	No	?	?	0	2	1	4m 11s	103
	192.168.137.154		Net-10-172-192...	No	?	?	0	2	1	4m 19s	4
	192.168.137.86		Net-10-172-192...	No	?	?	0	2	1	32m 10s	44

Figure 5.33: SQL Injection with more detail obtaining the destination IPs involved in the attack.

Port Scan

A port scan is a reconnaissance technique to discover which network ports on a target system are open and listening for incoming connections [92]. Network administrators often employ port scanning for legitimate purposes such as network troubleshooting and security assessments. However, attackers can also use it maliciously to identify potential vulnerabilities in a target system.

Apply on flows which are detected by the system
   and when a flow matches any of the following Port Scanning: Over 10 Seconds

(a)

Apply on flows which are detected by the system
   and when all of these BB:Threat:IoT Port Scanning, in any order, from any source IP to the same destination IP, over 10 seconds

(b)

Apply on flows which are detected by the system
   and when the source byte/packet ratio is equal to 64 bytes/packet
   and when the IP protocol is one of the following TCP:tcp_ip

(c)

Figure 5.34: The pictures show the rules to identify a port scan. Figure (a) indicates the main rule. If one of the rules indicated in (a) is positive, it will raise an offense. Figure (c) indicates a BB rule.

Analyzing the PCAP file, the packets' length sent by Nmap is 64 bytes. So, with this information, the first rule was created (Figure 5.34c). In addition, the rule will match only if a port scan is performed on the TCP protocol. Then, the second rule (Figure 5.34b) matches the flow traffic if any device sends a scanning packet (SYN, SYN-ACK, RST) at the same device (destination IP) for over 10 seconds. Finally, the SIEM (Figure 5.34a) will raise an offense if the rule *Port Scanning: Over 10 Seconds* is true.

5.5.7.3 Evaluation results

This section presents and analyzes the results obtained from our comprehensive experimentation, shedding light on the performance and efficacy of the proposed architecture. Figure 5.35 provides an overview of the experimental results on the dataset, focusing on the performance of RF and QBoost classifiers. Notably, both exhibited comparable and commendable classification quality. While RF performs excellently, QBoost, a quantum machine learning algorithm, also demonstrates competitive classification quality. This hints at the potential of quantum computing in enhancing data analysis and predictive tasks in Smart Cities. These results are also in line with the other shown in Section 5.3 related to the detection phase with QML algorithms. As quantum computing technology advances and becomes more accessible, it may open up new avenues for tackling complex Smart City challenges.

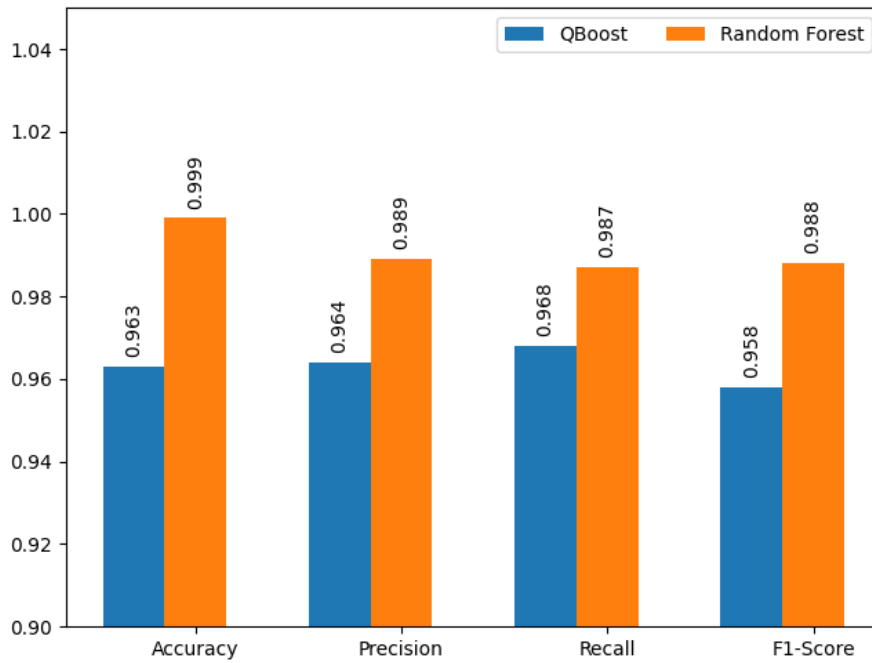


Figure 5.35: Quality metrics of Classifiers.

Algorithm	Training Time (s)	Prediction Time (s)
QBoost	554.5	2.24
Random Forest	2667.5	7.83

Table 5.17: Time performance analysis between QBoost and Random Forest.

Table 5.17 shows the training and prediction times associated with the mentioned algorithms. Notably, these times provide valuable insights into the compu-

tational efficiency of each approach. While the primary objective of this study does not center on a direct comparison between QBoost and RF, the results substantiate the feasibility of considering QBoost as a viable candidate for implementation within a Smart City framework. It is important to underscore that both classifiers exhibited remarkably similar results regarding quality metrics, suggesting that QBoost can be a viable alternative to RF. Where the accuracy of RF was 99%, QBoost attained 96% accuracy in classifying the attacks. In other words, RF was ahead of QBoost, with a margin of a mere 3% for all quality metrics.

However, what sets QBoost apart is its notable advantage in terms of computational efficiency. Specifically, it is possible to note an impressive reduction of approximately 80% in training time and a substantial 71% decrease in prediction times when comparing QBoost with RF. This substantial improvement in efficiency is particularly evident in the case of prediction times, where QBoost achieved a remarkable processing speed of just 2.24 seconds. Even when dealing with datasets containing over a million data entries, the rapidity of QBoost's predictions remains a standout feature.

On the other hand, with IBM QRadar SIEM, the time between the identification of an attack and the offense alert varies from about 2 to 3 minutes. This depends on whether it was used in a version deployed on a virtual machine and not in a cloud environment. By using more efficient hardware, this time could decrease.

5.5.8 Comparison with the state-of-the-art

This section introduces a hybrid approach that has the potential to significantly advance the state-of-the-art in attack detection for Smart Cities in the automotive context because there are no works that propose the combination of Quantum ML algorithms with IT security methods. By leveraging a Quantum ML algorithm and a SIEM, this approach enhances detection capabilities. If QBoost fails to identify a threat, IBM QRadar can serve as a backup and vice-versa, ensuring the system's effectiveness and the safety of Smart Cities.

Chapter 6

Response techniques in ANDURIL

As shown in Chapter 4 Table 4.1, few response techniques in the literature can answer the RQ2.5 "Can Response techniques work across various dimensions to minimize response damage from attacks?". In addition, no work has combined detection or prevention techniques that can be used for the response. The only work considering detection, response, and prevention is [27]. However, it proposes only guidelines. To answer RQ2.5, this Chapter discusses the proposed response techniques, considering the approaches discussed in Chapter 5.

The aim is to combine detection and prevention techniques (discussed in Chapter 7) to help the automotive industry's stakeholders understand the identified attack. In addition, the response techniques align with the second goal of this thesis that regards the *implementation and evaluation of new methodology* but also regards the first goal *analyze the attacks that can be conducted on in-vehicle and out-vehicle*. This is because, with the proposed approaches, the automotive industry can be updated with new vulnerabilities in the automotive context.

The proposed methods are based on two vision models [64, 63] that explore the use of Quantum algorithms to help the automotive industry support developers' decisions to integrate concrete design methods in the automotive development life-cycle. In particular, in the first proposed work [64], the aim is to propose a model that can suggest to developers steps to develop secure software during the design phase using *graphical notification* or *suggestion overlay* taking into consideration taxonomies or standards like ISO/SAE 21434. This suggestion can be used both in the prevention and response phases. Then, based on this idea, the second proposition is to use QML models that, exploiting the attack prediction, propose additional information about the impact of the attack so that developers can still devise a plan to fix the vulnerability at a more holistic level. This idea is extended in the following sections.

Finally, Figure 6.1 shows those techniques that belong to the domains and the operational unit *response*. In particular, most of these can be used singly or in combination with other techniques explained. For example, as explained in the following sections, *Auto-CyberKC* can be used singly to help cybersecurity experts

understand the identified attack and respond to it. It can also be combined with the V-SOC in the domain *Automotive Threat Intelligence* since it receives the offense from IBM QRadar. Then, using an attack taxonomy, it is possible to respond to the attack.

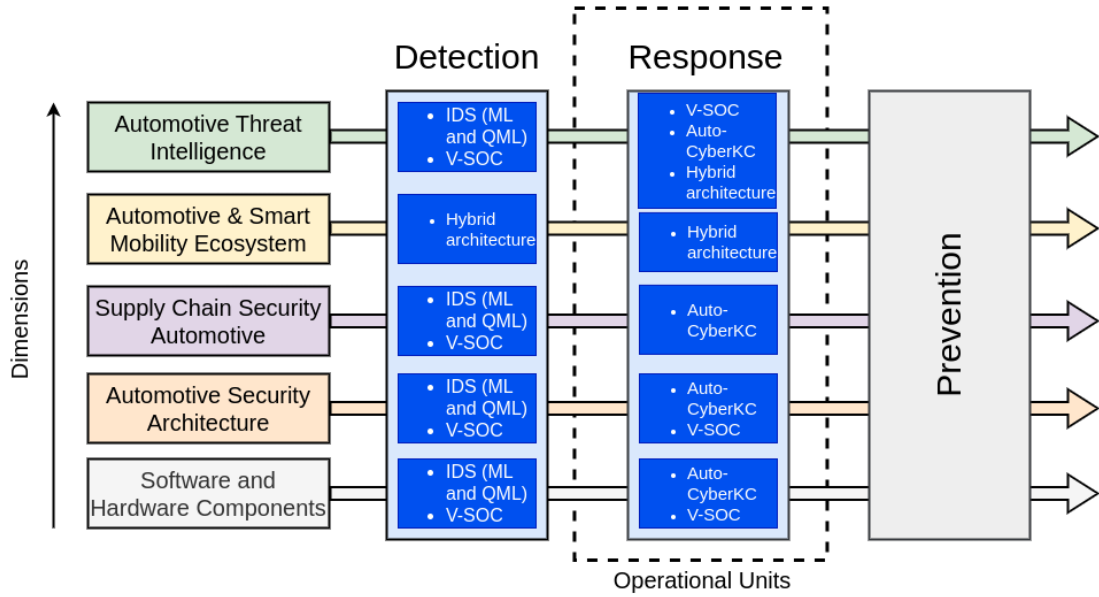


Figure 6.1: Proposed techniques regarding the response phase considering the *dimensions* (vertical) and the *operational units* (horizontal).

The Chapter is organized as follows: first, a proposed web application, Auto-CyberKC, is explained. Then, the combination of Auto-CyberKC and the V-SOC (shown in the detection phase) is proposed. Finally, the combination of the presented IDSs with the Natural Language Processing (NLP) algorithm is shown.

6.1 Auto-CyberKC

As described in Section 3.3, the taxonomy proposed by Sommer et al. [7] required high effort and time to map security incidents. Instead, Upstream released a Threat Intelligence platform containing cyber-attacks addressing automotive vulnerabilities. However, this is a proprietary platform and requires the purchase of a license for usage. The integration of the taxonomy proposed by Sommer et al. can be used to create an Automotive Knowledge Base (AKB) integrating the database released by the authors.

The taxonomy utilized in the methodology, proposed by Sommer et al. [7], consists of two separate files, one including 162 published security automotive incidents, and the other, a split version consisting of 412 multi-stage attacks published

online as Automotive Security Attack Classification (ASAC) [7]. In a following update [93], the authors published an Automotive Attack Database (AAD) file (SQLite DB file) based on the past published datasets containing 361 security attacks (621 attack steps) from public sources. The taxonomy supports 23 categories containing several levels of abstraction. All the categories have been used, but the most important categories will be described below.

The categories *Description*, *Reference*, *Consequence*, *Exploitability*, and *Attack Class* from the taxonomy [7] are used to show different information about the incident that can be used to fix the vulnerability. The description category provides brief information about the attack. The reference category is the publication source of an attack that is described. The consequence category describes the attack's effect on a vehicle or system. The exploitability category is the rating value considering the Common Vulnerability Scoring System (CVSS). Finally, the attack class category defines the automotive threats using the STRIDE model [94]. The STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege.

To create the knowledge base about an attack, the **Attack Path** category was used. For each security incident, the authors described the steps of an incident as precisely as possible, where the multi-stage attacks are described based on the taxonomy and split into individual steps.

An **attack path** provides a visual representation of a chain of attack techniques that an attacker with initial access to a target network can undertake to gain access to other assets in the network. Knowledge about the attack path can help reconstruct the kill chain and identify the vulnerability inside a vehicle's component. For example, a threat actor who has access to a vehicle can obtain the Bluetooth Message Authentication Code (MAC) address. Having acquired this information, the attacker can perform a brute-force attack to get the Bluetooth PIN and pair other Bluetooth devices. Consequently, the attacker can send malicious messages to the in-vehicle network from any connected Bluetooth device. The aforementioned attack is described in the attack path contained in the AAD as a multi-stage attack.

All the attack paths in the AAD are identified as a graph. After this, the graph is crossed to find all simple paths from the root node to the leaf node. The process can be summarized as follows (Figure 6.2b):

- If an attack does not have several steps (Figure 6.2a), then a direct degree with the single edge is created between the root node R (the attack) and the ID. The ID is a string present in the AAD and is characterized by the year of publication of the article, the author's initials, and the attack number. The attack number identifies if the attack path is single or multi-stage.
- If an attack has several steps (Figure 6.2b), then all the IDs that are part of the attack group are identified to build more complex graphs.

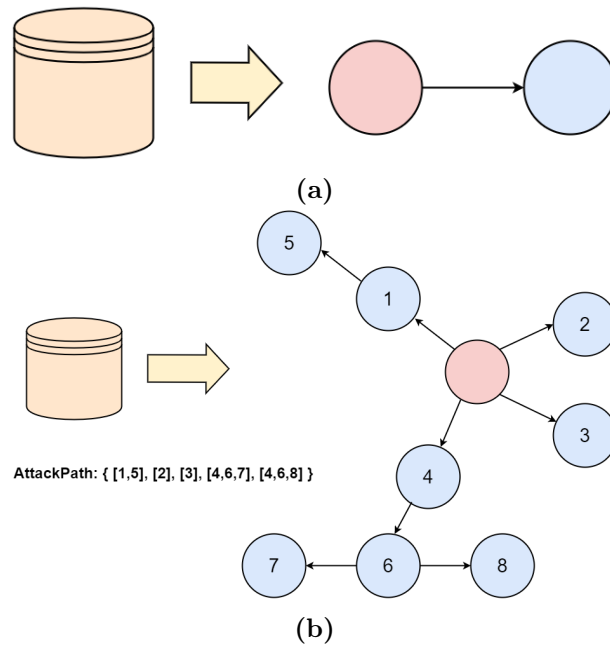


Figure 6.2: Figure (a) shows the single step of the knowledge graph. Instead, figure (b) represents the graph for the multi-stage attack path. In red is the root node.

Finally, all the graphs are combined to obtain a single graph identifying all the attack paths. CVSS 3.0 is used to calculate the severity of each attack path. The higher the severity value, the more severe the impact that the attack can cause. To calculate the severity score for each attack path, the following strategy has been used:

- If an attack has a single step, then the severity of the attack will be equal to the CVSS score multiplied by 10.
- If an attack has multiple steps, then the severity of the attack will be calculated using the maximum values of the mandatory CVSS score metrics associated with that attack. Finally, the value obtained will be multiplied by 10, obtaining the severity.

As a reminder, the CVSS produces a score of 0 to 10. Consider an attack path with two steps having the same values for all metrics except for confidentiality and integrity. Suppose the first step has a value of 6.5 for confidentiality and 5.1 for integrity. The second step has a value of 6.5 for confidentiality and 0 for integrity. The severity will be calculated using the CVSS score by taking the highest values of confidentiality (6.5) and integrity (5.1). The other metrics will receive values that are equal to the two steps. Once obtained, the new CVSS score will be multiplied by 10, obtaining the severity of the attack path. The value 10 is used

to normalize the CVSS score and obtain a numerical representation of the attack risk.

An example of a multi-stage attack is shown in Figure 6.3. Based on the procedure explained before, Auto-CyberKC shows the attack step considering the AAD.

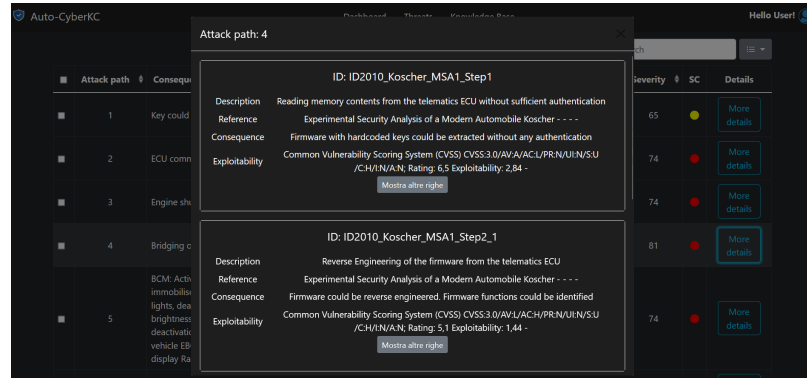


Figure 6.3: Auto-CyberKC Knowledge Base for attack path with the ID four that contains a multi-stage attack.

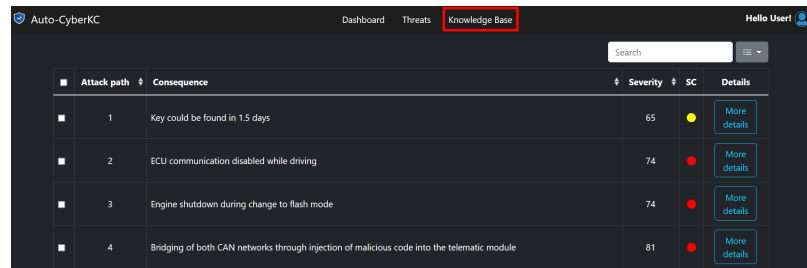


Figure 6.4: Auto-CyberKC Knowledge Base.

In addition to this, the proposed web application can also be connected with SIEM to help cybersecurity analysts understand the attack (described in the following section). Finally, Auto-CyberKC consists of three modules:

- **Dashboard:** provides overall information to the V-SOC analysis. In particular, the dashboard shows the *Open Threat* with the number of attacks identified by IBM QRadar that have not yet been resolved. The *Total Risk Management* (TRM) represents the threat currently open with the highest level of risk (RS_M) (see Chapter 8).
- **Threats:** shows the offenses generated by the SIEM in tabular format.
- **Knowledge Base:** shows information about the attack path. Each attack path is associated with a unique numerical identifier, an attack description, the severity, and the attack step (Figure 6.4). Figure 6.3 shows the

multi-stage attacks obtained from the AAD. The Knowledge Base interface is populated based on the mechanisms described at the beginning of this section.

6.2 Response techniques combining V-SOC

Since the SOC architecture is usually used for IT purposes, there is a need to integrate the response phase with methodologies applied specifically for the automotive context. For this reason, Auto-CyberKC is integrated with the V-SOC (discussed in Section 5.2) to obtain the offenses from IBM QRadar and then, using the taxonomy proposed by Sommer et al. [7], propose remediation to cyber security analysts in order to respond to the identified attack.

In Auto-CyberKC the offenses are shown in the *Dashboard* using REST Application Program Interface (API) that IBM QRadar offers. Figure 6.5 shows the connection between the proposed V-SOC and the Auto-CyberKC regarding the response phase.

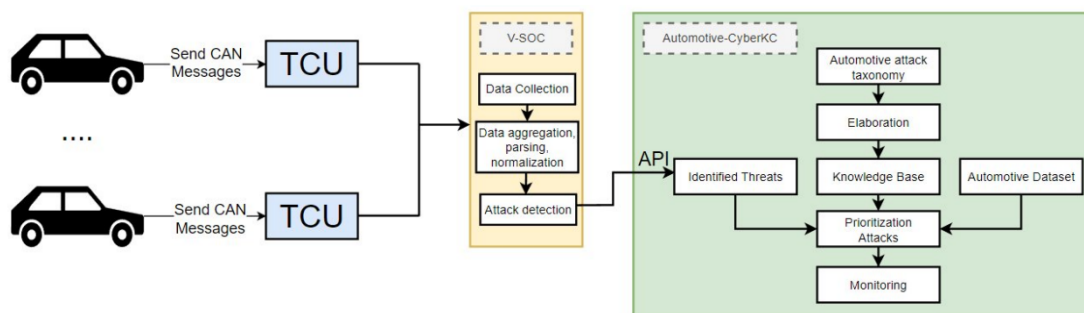


Figure 6.5: Communication between V-SOC and Auto-CyberKC.

To validate the proposed approach, the DoS and Fuzzy attacks are considered (as discussed in Section 7.1). The DoS attack was mapped into IBM QRadar as *Flood*. Instead, the Fuzzy was mapped as *Bad Content*. Flooding and Bad Content are mapped using the Common Attack Pattern Enumeration and Classification (CAPEC) [95] on the Auto-CyberKC. The CAPEC provides a comprehensive dictionary of known attack patterns used by adversaries. This can help the V-SOC analyst to understand how adversaries exploit weaknesses in applications and components. The CAPEC maps the Flood with ID 125¹, while the ID is 28 for bad content².

Since the role of the V-SOC is to monitor a large fleet of vehicles, five vehicles that received DoS and Fuzzy attacks are simulated. Each vehicle sends a payload to the IBM QRadar containing information to identify the attack (Figure 5.9

¹<https://capec.mitre.org/data/definitions/125.html>

²<https://capec.mitre.org/data/definitions/28.html>

Threat: Bad Content
Vehicle Model: Ateca
Automobile Manufacturer: SEAT

Search

Attack path	Consequence	RS	RI	Details
2	38 new vulnerabilities Private RSA keys and self-signed certificates used in approximately 35,000 online devices. Several hard-coded password hashes that could be used to deduce the original passwords. Multiple Backdoors (Authorized Key Files, hardcoded telnet credentials, hardcoded admin logins, backdoor daemons)	728	●	More details
1	Head unit reboots.	648	●	More details
6	Vehicle diagnostics tool can be compromised in such a way that it can infect multiple vehicles to which the tool is connected	592	●	More details
5	Car speed was decreased, car made noises, lights were blinking, odometer was changed	568	●	More details
4	Messages can be sent to the CAN-Bus by using the Bluetooth-OBD-Dongle	520	●	More details
3	Wrong outputs can be forced temporarily.	320	●	More details

Figure 6.6: Auto-CyberKC filtered attack path for the vehicle model Ateca and Bad Content.

and Figure 5.11). If the SIEM identifies this information as an attack, it raises an offense. The generated offense contains a UUID used to identify a vehicle uniquely, the attack category (Flood or Bad Content), and the event name (the attack description). Auto-CyberKC uses the IBM QRadar API to get the UUID, the attack category, and the event name to populate the dashboard and the threats. The V-SOC analyst can obtain the attack path filtered by the attack category for a specific threat. For example, Figure 6.6 shows the filtered information for the vehicle model Ateca.

Open threats: 5

TRM: 728

Attack path KB: 372
Attack KB: 621

ID	Description	Car	Category	Last updated time	RS _M	Status
8	Vehicle: DoS with high priority	Fiat Panda 2015	Flood	2023-02-27 21:59:47	656	OPEN
7	Fuzzing attack: ID not in reference set	SEAT Ateca 2022	Bad Content	2023-02-27 21:59:07	728	OPEN
6	Vehicle: DoS with high priority	Skoda Kamiq 2022	Flood	2023-02-27 21:58:10	656	OPEN
5	Vehicle: DoS with high priority	Jeep Renegade 2022	Flood	2023-02-27 21:55:24	656	OPEN
4	Fuzzing attack: ID not in reference set	Toyota Yaris 2021	Bad Content	2023-02-27 21:30:37	728	OPEN

Figure 6.7: Auto-CyberKC Dashboard.

Instead, Figure 6.7 shows the attacks identified by IBM QRadar. The highest

risk is associated with the category *Bad Content*. The *attack path* and the *attack* refer to the data contained in the knowledge base. When a threat is detected through data analysis in a SOC, it is necessary to associate it with a degree of risk to determine the priority to implement remedial actions. Several attack paths in the AAD were carried out on different vehicles. The role of the Risk Score metric is to associate the risk of a specific threat with a specific vehicle. As reported by Van Schaik et al. [96], risk perception is influenced by factors that can be direct or indirect based on the experience. The numerical risk could influence the direct experience of the analyst because, using a numerical representation, the specialist directly impacts the attack's severity. So, using a numerical risk score can support the V-SOC analyst in making decisions about the impact of the threat that occurred on the vehicle.

This metric is obtained by adding weight to the severity of an attack path and multiplying the result by 10. The weight indicates the feasibility of an attack path and depends on the vehicle's manufacturer, model, and year. The AAD reports the automotive manufacturer (such as Ford and Toyota) for each attack. With this information, it is possible to create the risk score in this manner:

- If the automobile manufacturer, vehicle model, and production year match the vehicle on which the threat was identified, a weight of 1 is returned.
- If the automobile manufacturer and vehicle model match the vehicle, a weight of 0.95 is returned.
- If only the automobile manufacturer matches the vehicle name, a weight of 0.90 is returned.
- If none of the above conditions are true, a weight of 0.80 is returned.

The risk score has been formalized on the Knowledge Base (Figure 6.4 - column 'SC') using the circle stereotype and the following range: if the risk is less than equal to 33, the circle is grey; if the risk is less than equal to 67, the circle is yellow; otherwise if is greater than or equal to 100 is red.

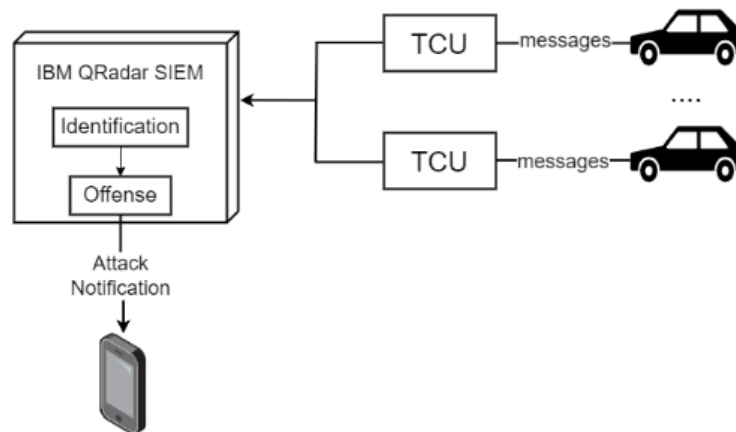


Figure 6.8: Overview of proposed model to respond to an attack using a mobile application.

In addition to providing strategies for analysts to understand the attack, it is necessary to inform the driver. For this reason, this thesis proposes another response technique that aims to advise the driver regarding the attack [60]. A prototype of an Android Mobile application has been created that obtains the offenses from QRadar and provides information regarding the attack. The overview of the proposed model is shown in Figure 6.8 in which the TCUs send data to IBM QRadar SIEM and if detect the received message as an attack, the mobile application shows the information about it. This application consists of two tabs: *Realtime data* and *Attacks*.

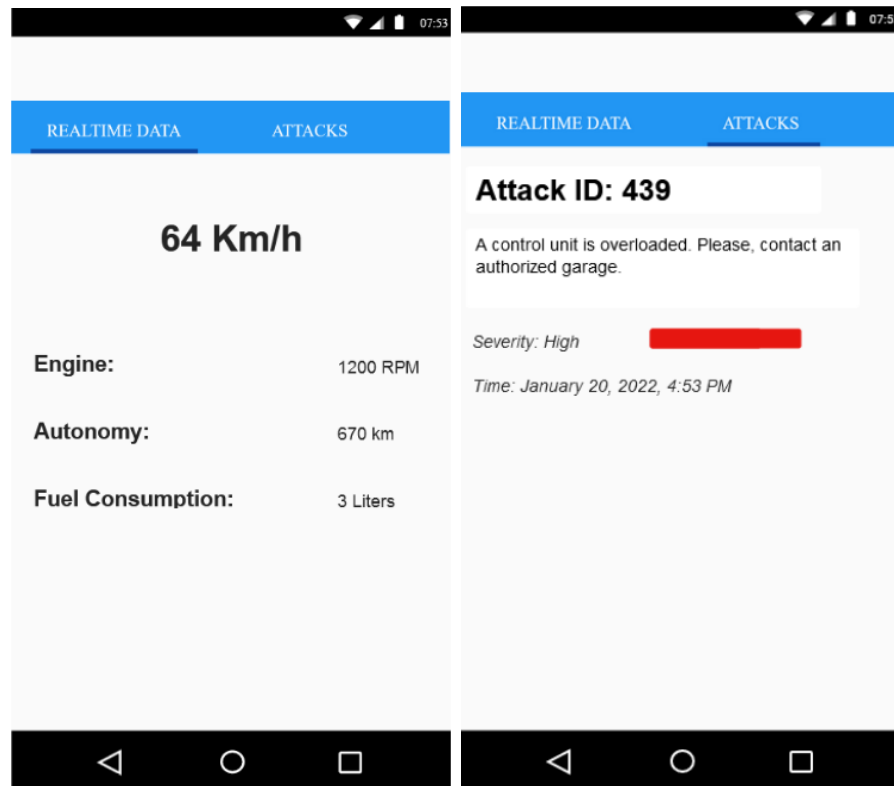


Figure 6.9: Mobile app: on the left the real-time data tab, on the right the attacks tab.

Figure 6.9 shows the app’s User Interface. The *real-time tab* describes information (speed vehicle, engine autonomy, and fuel consumption) about the vehicle in real-time obtained from the back-end. The server will communicate with the client (App Mobile) as soon as it receives the data from the vehicle. The field in real-time data tab can be described as follows:

- **Speed Vehicle:** the current speed of the vehicle expressed in Km/h.
- **Engine:** the engine speed expressed in RPM.
- **Autonomy:** the autonomy of vehicle in Km.
- **Fuel Consumption:** the fuel consumption expressed in liters.

The tab *attacks* (Figure 6.9 on the right) shows both real-time and old attacks. Based on the information reported in this tab, the driver can call an authorized garage to solve the problem. In this tab, we can see different fields. The Attack ID corresponds to the offense’s ID generated by the SIEM so that the driver can call the authorized garage and communicate the ID of the attack. After this, a brief description of the attack will follow. The Severity field indicates the attack severity. This value is calculated in the range between 0 and 10. From 0 to 3 means

a low severity level; the bar will be represented in green. From 4 to 6, there is a medium level of threat of severity that will be shown with a yellow bar. From 7 to 10, a high level of severity threat occurs, and the bar is shown in red. These parameters are taken from the SIEM in the *severity* field (for example, Figure 5.10 shows an example of a severity field related to the DoS attack). Finally, the last field indicates the time and date the attack was received.

6.3 Response techniques combining IDS algorithms

As described in Section 3.3, another problem in the automotive field is correlated to the proposition of new attacks. For example, the attacks in the taxonomy of Sommer et al. are too old. For this reason, obtaining recent vulnerabilities using open-source databases is necessary. For example, the NVD is a good candidate for taking the latest automotive vulnerabilities. For this reason, this thesis proposed an approach using the taxonomy of Sommer et al. to obtain information about the attack identified by the IDS in correlation with the NVD. A Natural Language Processing (NLP) model has been used to retrieve a similar description obtained from the taxonomy to get additional information about new automotive vulnerabilities. To be more specific, the taxonomy was used to obtain information about the attack identified by the IDS. In addition, an NLP pre-trained model has been used to obtain a similar description from the taxonomy to get additional information about new automotive vulnerabilities.

6.3.1 Background to NPL

NLP combines computational linguistics using machine learning, deep learning, or statistical approaches that enable an algorithm to understand and process human language. NLP can measure the semantic similarity between two sentences to determine the degree of similarity or meaning-relatedness between two given sentences. The transformer networks (for example, BERT and RoBERTa) achieve good results in various tasks like semantic textual similarity (STS) [97].

In particular, Reimers and Gurevych [97] presented Sentence-BERT (SBERT), which uses a siamese and triplet network structure to derive semantically sentence embeddings that can be used to calculate the cosine-similarity to determine if two sentences are similar. In other words, these embeddings capture the semantic and contextual information of the sentences, enabling meaningful comparisons between them.

A Python framework called SentenceTransformers³ has been used to perform this task based on the initial work of [97]. The framework is based on PyTorch and Transformers, which offers a pre-trained model that can also be used for semantic

³<https://www.sbert.net/>

textual analysis. A pre-trained model called "multi-qa-MiniLM-L6-cos-v1"⁴ was used. This model was trained using a concatenation of multiple datasets obtaining 215M question and answer pairs.

6.3.2 Methodology

Using the scraping process, 86 automotive vulnerabilities were obtained from NVD. This scraping process consists of using specific keys such as *automotive*, *car(s)*, *infotainment*, and so on. This is important to avoid attacks that are not correlated to this field. In addition, the CVE ID, description, references, CVSS score, CWE ID, and CWE description are also saved for each vulnerability. In particular:

- **CVE ID:** is the identifier of the vulnerability.
- **CVE Description:** is the description about the vulnerability.
- **References:** are the links that provide a PoC about the attacks.
- **CVSS (v2, v3):** is the severity score about a vulnerability.
- **CWE ID:** is a unique identifier about the Common Weakness Exposure (CWE) assigned to the CVE.
- **CWE Description:** is the description about the CWE.

These are important for the response phase because they provide a PoC about the attack. In other words, they show how the attacks have been conducted and, in general, the research(es) that found the vulnerability usually provide information on how to exploit and fix it. On the other hand, the CWE ID and the CWE description are essential because they can support the developer in fixing the vulnerability and understanding the attack consequences. In other words, the information obtained from the NVD can be useful for the automotive industries for different reasons:

1. The reference field contains the link(s) about the attack and, in some cases, how to solve the vulnerabilities or weaknesses.
2. The CWE description is a standardized way of categorizing and describing common software security weaknesses or vulnerabilities. For the automotive industry, the CWE description can help in the risk assessment phase to prioritize their efforts and focus on addressing the most critical security issues first. This prioritization can be conducted in conjunction with the CVSS metric to determine the attack feasibility rating (as reported by the

⁴<https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1>

ISO/SAE 21434). The CWE descriptions often include recommendations and best practices for mitigating vulnerabilities. So, since the ISO/SAE 21434 defines general considerations about the risk assessment methods, the CWE description can be a real criterion for supporting automotive companies in designing safe software and, during the attack phase, in proposing remediations.

As described in the previous section, a pre-trained sentence transformer model automatically obtains similar sentences from the taxonomy and the CVE descriptions from the NVD. Then, the top three similar sentences are recommended to the automotive industry, including the aforementioned NVD fields.

```

"query": "Vehicle theft by using RollJam device to spoof the key fob", Description presents into the taxonomy
"sentence": [
  {
    "Top-three similar sentences"
    "CVE": "CVE-2022-37418",
    "Description": "The Remote Keyless Entry (RKE) receiving unit on certain Nissan, Kia, and Hyundai vehicles through 2017 allows remote attackers to perform unlock operations and force a resynchronization after capturing two consecutive valid key fob signals over the radio, aka a RollBack attack. The attacker retains the ability to unlock indefinitely.",
    "References": "https://hackaday.com/2022/08/17/rollback-breaks-into-your-car/,https://medium.com/codex/rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-df5f99ba9490,https://www.blackhat.com/us-22/briefings/schedule/#rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185,https://www.pcmag.com/news/is-your-car-key-fob-vulnerable-to-this-simple-replay-attack,https://www.youtube.com/playlist?list=PLYody84oQL1gxiuRm13xRXXtQL9c05t",
    "CVSSv3": "5.2",
    "CWE": "CWE-294",
    "CWE Description": "Capture-replay attacks are common and can be difficult to defeat without cryptography. They are a subset of network injection attacks that rely on observing previously-sent valid commands, then changing them slightly if necessary and resending the same commands to the server. "
  },
  {
    "CVE": "CVE-2022-36945",
    "Description": "The Remote Keyless Entry (RKE) receiving unit on certain Mazda vehicles through 2020 allows remote attackers to perform unlock operations and force a resynchronization after capturing three consecutive valid key fob signals over the radio, aka a RollBack attack. The attacker retains the ability to unlock indefinitely.",
    "References": "https://hackaday.com/2022/08/17/rollback-breaks-into-your-car/,https://medium.com/codex/rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-df5f99ba9490,https://www.blackhat.com/us-22/briefings/schedule/#rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185,https://www.pcmag.com/news/is-your-car-key-fob-vulnerable-to-this-simple-replay-attack,https://www.youtube.com/playlist?list=PLYody84oQL1gxiuRm13xRXXtQL9c05t",
    "CVSSv3": "5.2",
    "CWE": "CWE-294",
    "CWE Description": "Capture-replay attacks are common and can be difficult to defeat without cryptography. They are a subset of network injection attacks that rely on observing previously-sent valid commands, then changing them slightly if necessary and resending the same commands to the server. "
  },
  {
    "CVE": "CVE-2020-15912",
    "Description": "** DISPUTED ** Tesla Model 3 vehicles allow attackers to open a door by leveraging access to a legitimate key card, and then using NFC Relay. NOTE: the vendor has developed PinDrive to mitigate this issue.",
    "References": "https://cnn.com/2020/03/09/22:00:00 2020 Speakers,https://twitter.com/Kevin2600/status/1218892338182836224,https://www.carhackingvillage.com/speaker-bios#tm3nrr,https://www.youtube.com/watch?v=VYKsfgox-bs,https://www.youtube.com/watch?v=kQNg-Ywv3S4,https://www.youtube.com/watch?v=nn-3AbtEkl",
    "CVSSv3": "5.0"
  }
]

```

Figure 6.10: The top three sentences that are similar to the description obtained from the taxonomy. CWE and CWE descriptions have been omitted for space reasons from the last sentence. Similar words can be seen underlined in yellow and green.

Figure 6.10 shows a first approach to show the results regarding the NLP process [65]. In the figure, some words have been underlined to show the similarity between the query (the description obtained from the taxonomy) and the description obtained from the NVD. In this case, the suggestions regard DoS and Spoofing attacks previously identified by IDS. The query refers to an attack using the RollJam (a crafted hardware component) that consists of jamming the transmission between the key and the car to obtain the code to open the car door. RollBack is another attack that, unlike RollJam, is time-agnostic.

The obtained references can support the developer in understanding these attacks and what strategies can be applied to increase the vehicle's security or fix the vulnerability. For example, considering the CVE-2022-36945 and the Black-

hat's reference link⁵ provides several mitigations and descriptions about not only for the RollBack attack but also RollJam.

The CVSS, instead, can help the developer to prioritize the attacks based on the score. The CWE and the CWE description add an additional detail to resolve the identified attack. For example, the CWE description *"Capture-replay attacks are common and can be difficult to defeat without cryptography."* suggests adding cryptography solutions to avoid this attack.

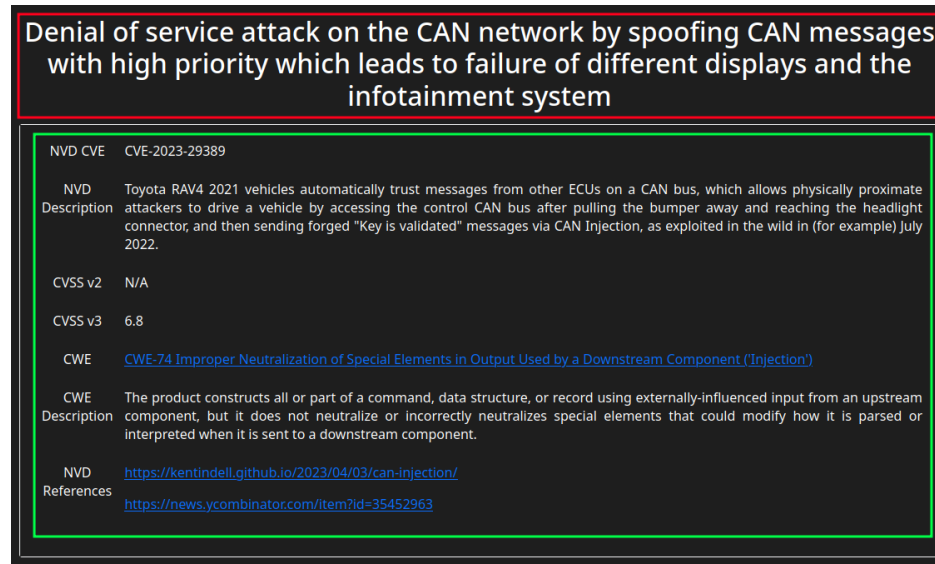


Figure 6.11: A graphical improvement of Figure 6.10. The figure shows an example of a suggestion regarding the DoS attack.

In a second approach [59], those suggestions are modified and integrated into the Auto-CyberKC web application. Figure 6.11 shows the graphical improvement in which the red rectangle describes the attack description obtained from the taxonomy and the green rectangle shows the several fields obtained from the NVD and explained before.

6.4 Comparison with the state-of-the-art

The advanced response prevention in the state-of-the-art (outlined in Section 3.4 and Table 4.1) consists of general methodologies that do not directly address the automotive industry. For instance, in [7], the authors introduced a taxonomy for the Auto-CyberKC, which, as explained in this section, requires substantial

⁵<https://www.blackhat.com/us-22/briefings/schedule/#rollback---a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185>

effort to implement. In [27], the researcher presented a cyber defense center without demonstrating experimental results, but instead proposed architectural design guidelines. In [48], the authors suggested a methodology compliant with the ISO/SAE 21434 to identify attack paths. However, this approach is overly complex for response and prevention purposes. Lastly, [49] defines a risk assessment methodology by providing general guidelines.

In comparison to the current state of the art, the approaches outlined in this section could aid developers or security analysts in understanding attacks and automatically proposing remedies. This promising direction has the potential to significantly reduce the effort required to comprehend received attacks and enhance cybersecurity in the automotive industry.

Chapter 7

Prevention techniques in ANDURIL

As mentioned in Chapter 4.3, prevention techniques aim to identify critical assets and then use methods to protect them from the design phase. An example is the use of the ISO/SAE 21434, which provides a comprehensive framework for managing cybersecurity risks from the vehicle’s concept phase through production, operation, and decommissioning. As mentioned in Section 3.3, a problem with the ISO/SAE 21434 is that it does not address practice guidelines, making it difficult to understand. Several works proposed methodologies or frameworks to facilitate the automotive system designers to follow ISO/SAE 21434 standard guidelines by systematically identifying, assessing, protecting, and managing the cybersecurity risks across the automotive system life cycle [98, 99, 100, 101, 49].

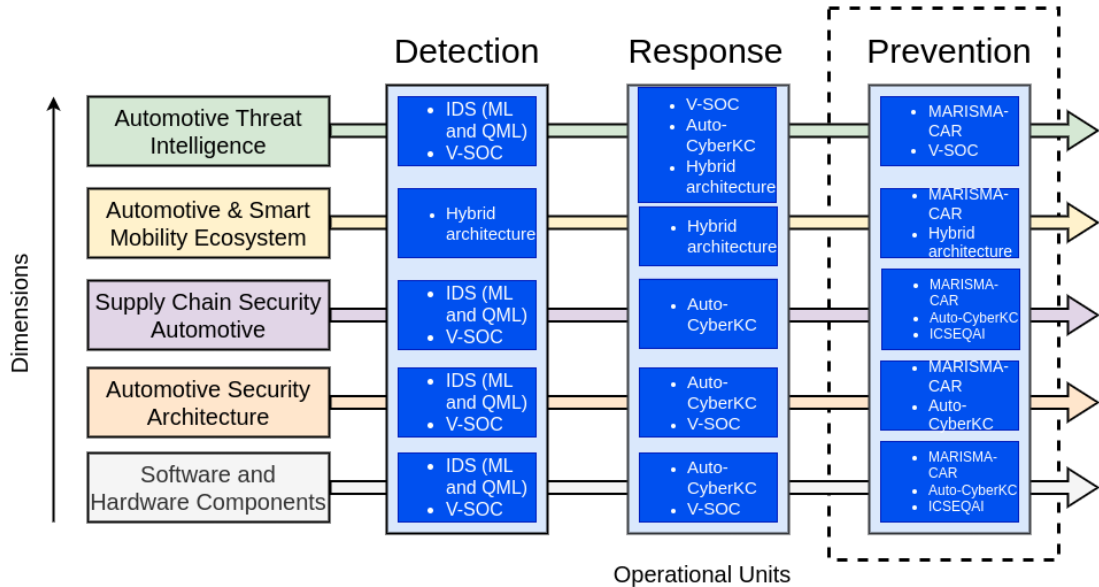


Figure 7.1: Proposed techniques regarding the prevention phase considering the *dimensions* (vertical) and the *operational units* (horizontal).

Generally, these works do not provide an automatic or semi-automatic risk assessment phase and do not use reports or guidelines to detect assets or threats. For those reasons, considering the RQ2.6 "Can Prevention techniques interact across multiple dimensions to mitigate attacks on vehicles?" this thesis proposes two approaches to answer this research question: the first regards the combination of ISO/SAE 21434 and the taxonomy of Sommer et al. [7] to develop a web application that may help the automotive industry to understand attacks, and the second approach by defining a pattern called MARISMA-CAR [67] to manage and control the risk of threats in modern vehicles and future Smart Cars by combining the reports proposed by ENISA and Upstream and the ISO/SAE 21434. In addition, it is also important to teach academia and the industry to make the software secure by exploiting standards like the ISO/SAE 21434. For this reason, a vision model is also presented as a prevention approach by adopting new solutions based on Quantum Artificial Intelligence algorithms and processes that could support Cyber Security Education in the industry. This is also because only broad standards are available, and their implementation is challenging due to the absence of necessary expertise and knowledge.

Figure 7.1 shows the proposed methodologies regarding the prevention mapped based on the domains and the operational units. The figure shows that MARISMA-CAR belongs to all five dimensions since this approach aims to propose a pattern to perform a risk assessment process using those dimensions as domains (more details will be discussed in the following section). Auto-CyberKC belongs to the domains *Supply Chain Security Automotive* and *Software and Hardware Components* since, as shown in Section 6.1, the web application provides the module *Knowledge Base* to shown information about the attack path using the AAD (Figure 6.3 and Figure 6.4). In this manner, a cybersecurity analyst or a developer can use this module to understand specific attacks to protect software and hardware components.

7.1 MARISMA-CAR

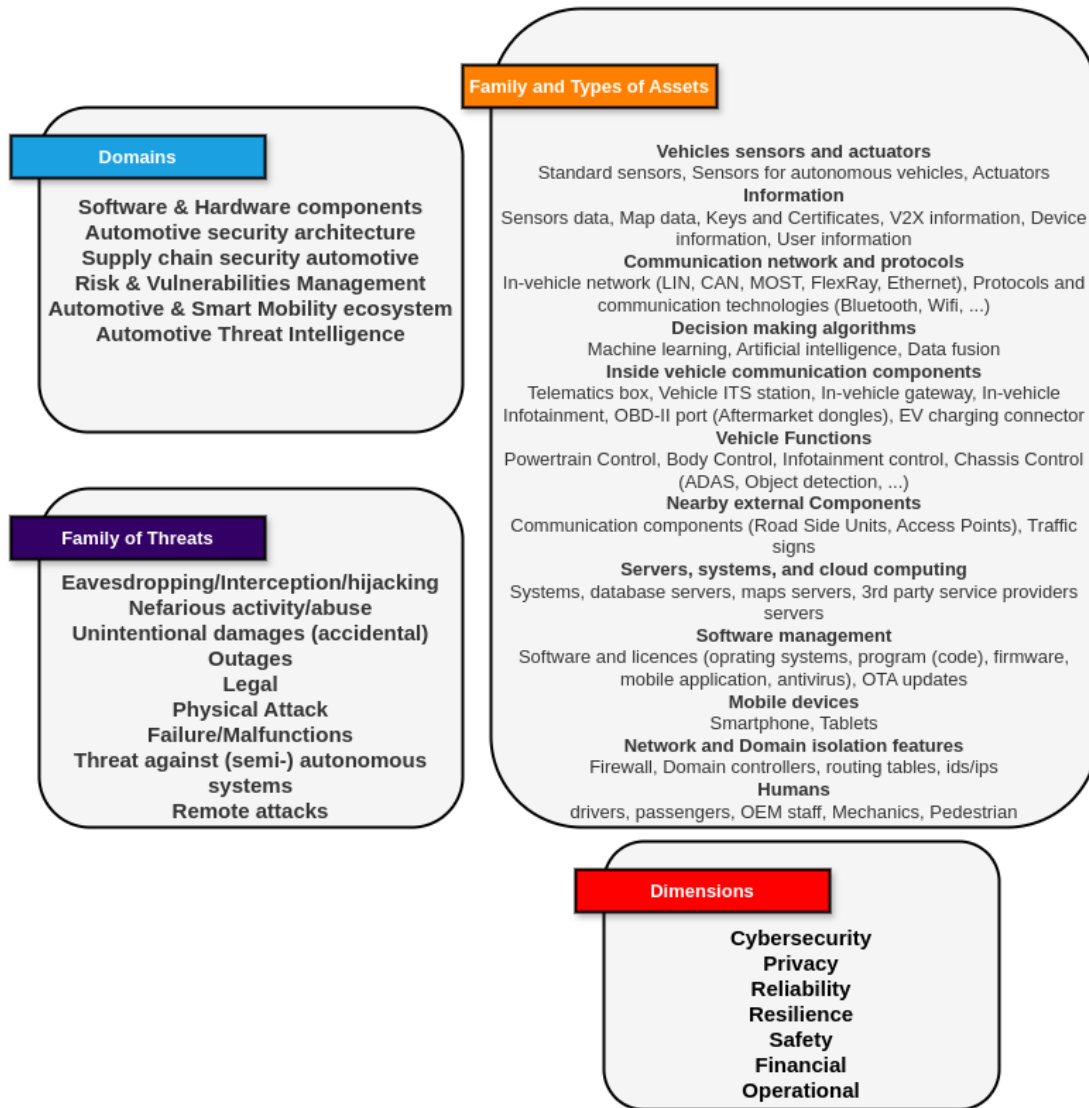


Figure 7.2: Components of MARISMA-CAR pattern.

Introducing a risk assessment methodology inside the automotive and Smart Mobility ecosystem can help actors involved in component manufacturing, developers, and security analysts identify and mitigate assets and threats. For this reason, new techniques that can automatically verify and validate the enhanced risk of cyber attacks in a way that can be managed and controlled are needed.

Rosado et al. [102] proposed a risk analysis methodology called MARISMA (Methodology for the Analysis of Risks on Information Systems, using Meta-

Pattern and Adaptability) and eMARISMA, a technological environment in the cloud, to support risk assessment for CPSs. Using the MARISMA framework idea and combining two reports from ENISA [103], [104], and the ISO/SAE 21434, it is possible to manage automotive and Smart Cars attack risk. The defined patterns can help the automotive industry set up security controls to identify and protect automotive assets and respond to threats. In particular, this last phase can be made using security-by-design approaches (preventing the emergence of vulnerabilities by adopting secure software and hardware development techniques) and real-time threat analysis techniques.

MARISMA framework is a Risk Assessment Methodology (RAM) that can be adapted to many use cases, from IT environments to CPSs. To support different types of cases, MARISMA uses structures known as "patterns" to minimize the effort required in the risk assessment process. A technological component called "eMARISMA" automatically supports the end users in the risk assessment phase. The key elements of MARISMA-CPS to build a pattern are summarized in the following:

- **Domains, objectives and controls:** are security domains that classify security controls according to the defined control objective.
- **Types of assets:** defined in the IoT context using the ENISA agency report.
- **Dimensions:** a set of security dimensions published by the NIST.
- **Family of threats and types of threats:** defines the different families and types of threats that can affect CPSs considering the ENISA IoT report.

Using this approach, Figure 7.2 shows an overview of the MARISMA-CAR pattern [67]. In particular, this is a preliminary work in which a pattern was created in order to be integrated with the eMARISMA tool. On the other hand, this pattern could generally be used for the prevention phase without eMARISMA.

7.1.1 Automotive and Smart Car Standards for MARISMA-CAR

Since the impact of attacks on Smart Cars has far-reaching consequences in terms of safety, in 2017 [103] and 2019 [104], ENISA released two technical documents called "*Cyber Security and Resilience of smart cars*" [103] and "*ENISA good practices for security of Smart Cars*" [104]. The difference between these two documents differs from the defined assets and the security measures. In particular, in [103], ENISA excluded the Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) connection and autonomous vehicles. Since [104] can be applied for all the vehicles and connections, the assets between the [103] and [104] were merged, and

the security measures of [104] were considered to build the family of asset *Vehicle Function*.

These two documents propose guidance on securing vehicles and information exchanged between them, infrastructures, and CPS from cyber threats. They highlight good security practices and propose recommendations to operators, manufacturers, and decision-makers.

Finally, the ISO/SAE 21434 has been considered to define specific domains and dimensions in MARISMA-CAR patterns.

Finally, another report that needs to be mentioned is the one made by Upstream¹, an organization that deals with automotive cybersecurity. Every year, they publish a report that focuses on the automotive cyber threat landscape, taking into consideration autonomous and (semi-) autonomous vehicles. The reports provide insight into creating security measures to increase cybersecurity in automotive and Smart Mobility according to the ISO/SAE 21434 [15] and other standards/regulations such as UNECE WP.29 R155 [105]. The Upstream report of 2024 [18] was used to define new dimensions required by MARISMA.

7.1.2 Cybersecurity Engineering Process for Automotive and Smart Cars

Modern vehicles and the future autonomous cars are safety-critical systems that need security methodology or techniques to avoid threats that could lead to the safety and security of passengers.

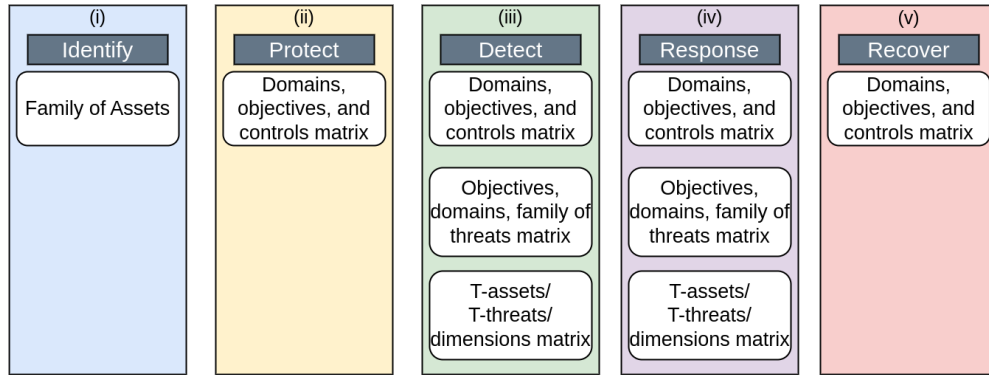


Figure 7.3: Overview of MARISMA-CAR pattern mapping according to the NIST Cybersecurity Framework.

The NIST Cybersecurity Framework [106] provides a risk-based iterative approach for cybersecurity that consists of five phases: **(i) Identify**, **(ii) Protect**, **(iii) Detect**, **(iv) Response**, **(v) Recover** [106, 5] as described also in Chapter 4. MARISMA-CAR takes into consideration [107] by extending the principle of

¹<https://upstream.auto/>

risk assessment for automotive and Smart Cars ecosystems that are in line with the ISO/SAE 21434 [15] and the NIST Cybersecurity Framework [106]. Therefore, the MARISMA-CAR pattern can be mapped with the NIST framework (Figure 7.3): (i) Identify: Family of Assets; (ii) Protect: Domains, objectives, and controls matrix; (iii-iv) Detect - Response: Domains, objectives, and controls matrix - Domains, objectives, and family of threats matrix - T-assets/T-threats/dimensions matrix; (v) Recover: Domains, objectives, and controls matrix. The following components are explained in detail below.

7.1.3 Domain, objective and controls

Since the ENISA report [104] does not define security domains like in MARISMA, the ISO/SAE 21434 [15] and the Upstream report [18] were used as guidelines to define them. The domains and their definition are:

- **Software & Hardware components:** includes security measures that refer to internal or external hardware components like Electronic Control Units (ECUs), Road Side Units (RSUs), and the software used by them.
- **Automotive Security Architecture:** includes security measures regarding the communication between the in-vehicle components.
- **Supply chain security automotive:** includes security measures regarding customer-suppliers' interaction. ISO/SAE 21434 defines an example within a Tier-1 can be a supplier for an OEM in the development, and in another contractual relationship, the Tier-1 can be a customer of a Tier-2 for a component. Between this interaction, the actors shall include a cybersecurity interface agreement.
- **Automotive & Smart Mobility Ecosystem:** includes security measures regarding the Vehicle-to-Everything (V2X), such as communication between Application Programming Interface (API), Cloud, and automotive (for example, ECUs and TCU).
- **Automotive Threat Intelligence:** includes security measures regarding the real-time detection, response, and investigation of threats for modern vehicles and Smart Mobility, such as Smart Cars.

Objectives	DOMAINS					
	D1	D2	D3	D4	D5	D6
Security by design	PS-01,02,03,04	PS-01,02,03,04				
Privacy by design			PS-05		PS-06,07	
Asset management	PS-08,09,10			PS-09	PS-08,09	
Risk and threat management				PS-11,12,14,16,17		PS-13,15
Relationships with suppliers			OP-01,02			
Training and awareness			OP-03,04,05,06			
Security management			OP-10	OP-08,09		OP-07
Incident management				OP-11,14		OP-12,15
Detection	TM-04	TM-01	TM-02		OP-13	TM-05
Protection of networks and protocols	TM-10	TM-06,07,13,14	TM-14		TM-03	
Software security	TM-16,18,19,20		TM-17	TM-21	TM-06,08,09,11,12,14	
Cloud security			TM-24	TM-26	TM-15,22,23	
Cryptography	TM-31,32	TM-28,29			TM-25,26,27	
Access control		TM-33			TM-28,29,30	
Self-protection and Cyber Resilience	TM-38,39,40	TM-41			TM-33,34,36	TM-35
(Semi-) autonomous systems self protection and cyber resilience	TM-43,44,45,47				TM-37	
Continuity of operations	TM-48		TM-50	TM-49	TM-42,46	

DOMAINS D1: Software & Hardware components; D2: Automotive security architecture; D3: Supply chain security automotive; D4: Risk & Vulnerabilities Management; D5: Automotive & Smart Mobility ecosystem; D6: Automotive Threat Intelligence. **CONTROLS** PS: policies; OP: organizational practices; TM: technical practices

Table 7.1: Domains, objective, and controls of MARISMA-CAR pattern.

These domains are also used for ANDURIL. According to the ENISA agency report [104], each control can be defined into three categories: (i) *policies* ensure that appropriate cybersecurity level of policies and procedures are applied; (ii) *organizational practices* ensure security in the Smart Cars; (iii) *technical practices* a set of technical security measures that should be implemented to protect both Smart Cars and back-end systems. The domains, objectives, and the mapping with the controls are shown in Table 7.1. Each cell indicates the control associated with each objective for each domain. In particular, the association of each control is assigned based on the authors' experience. For example, the objective *Security by design* was assigned to a series of domains that are D1 *Software & Hardware components* and D2 *Automotive Security Architecture*. Each control ensures that the software, hardware, and architecture shall be protected by adopting security by design strategies like DevSecOps and Secure Development Lifecycle, and security aspects at the very beginning of the concept phase.

7.1.4 Type of assets

The ENISA agency reports [103, 104] indicate a set of assets that are typical for autonomous and (semi-) autonomous cars. The assets presented in [103] are been merged with the list of assets provided in [104]. In particular, the family **Vehicle functions** are taken from [104] and contain the type of assets of [103] because they are related to general in-vehicle functionalities. The rest of the other families and types of assets are taken from [104]:

- **Vehicles sensors and actuators:** Standard sensors, Sensors for autonomous vehicles, Actuators.
- **Information:** Sensors data, Map data, Keys and Certificates, V2X information, Device information, User information.

- **Inside vehicle communication components:** Telematics box, Vehicle ITS station, In-vehicle gateway, In-vehicle Infotainment, Onboard Diagnostic (OBD-II) port (Aftermarket dongles), Electronic Vehicle (EV) charging connector.
- **Vehicle Functions:** Powertrain Control, Body Control, Infotainment control, Chassis Control (ADAS, Object detection, and so on).
- **Nearby external Components:** communication components (Road Side Units, Access Points), Traffic signs.
- **Servers, systems, and cloud computing:** systems, database servers, maps servers, 3rd party service providers servers.
- **Software management:** software and licenses (operating systems, program (code), firmware, mobile application, antivirus), OTA updates.
- **Mobile devices:** smartphone, and Tablets.
- **Network and Domain isolation feature:** firewall, Domain controllers, routing tables, Intrusion Detection System/Intrusion Prevention System (IDS/IPS).
- **Humans:** drivers, passengers, OEM staff, mechanics, pedestrians.

The families and types of assets are important to understand how to map the control to the domains and objects. For example, the family of asset information is related to the information exchange between the vehicles or infrastructures, the sensors, and so on. Since the data exchange can contain personal information about the passengers or critical information for the autonomous car, it is necessary to first understand the impacted type of asset(s) and then assign the security controls to the respective domains.

7.1.5 Dimensions

To define the security dimensions of MARISMA-CAR, the NIST framework was used [108]. Instead, the description of each security dimension was taken from MARISMA [107]. They defined the concept of trustworthiness, which includes security, privacy, safety, reliability, and resilience that CPSs can adopt. In addition, the financial and operational aspects were considered using the ISO/SAE 21434 [15]. The details of the dimensions are:

- **Security:** to protect, detect, prevent, and recover a system from attacks by ensuring its availability, integrity, and confidentiality [107].

- **Privacy:** to secure and protect individuals from potential risks arising from the use of and access to their personal information within systems or from the manipulation of physical environments [107].
- **Safety:** the absence of catastrophic consequences for the user(s) and the absence of unacceptable risks of physical injury or damage that could cause death, injury, occupational diseases, damage or loss of equipment or property, or damage to the environment [107].
- **Safety:** the absence of catastrophic consequences for the user(s) and the absence of unacceptable risks of physical injury or damage that could cause death, injury, occupational diseases, damage or loss of equipment or property, or damage to the environment [107].
- **Reliability:** the ability of a system or component to function under stated conditions for a specified period of time [107].
- **Resilience:** the ability to adapt to changing conditions and to withstand and recover quickly from deliberate attacks, accidents, or natural hazards or incidents [107].
- **Financial:** to protect an asset from attacks that could cause a potential loss of financial business for a stakeholder.
- **Operational:** to protect an asset (for example, the information exchanged and so on) from attacks that could cause potential damage to a vehicle or other assets.

7.1.6 Family of threats and types of assets

The different families and types of threats that affect the assets are defined using the ENISA. In [104], they developed a threat taxonomy focused on Smart Cars, giving a description of the threat and identifying the assets that may be affected by each threat. The considered family and type of threats are:

- **Eavesdropping/Interception/hijacking:** Communication protocol hijacking, Data replay, Man-in-the-middle attack / Session hijacking.
- **Nefarious activity/abuse:** Denial of service, Malware, Manipulation of hardware and software, Manipulation of information, OEM Targeted attacks, Unauthorised activities, Identify theft, Abuse of authorizations.
- **Unintentional damages (accidental):** Unintentional change of data or car components configuration, Information leakage, Using information and/or devices from an unreliable source, Erroneous use or configuration of car components.

- **Outages:** Loss of GNSS signal, Car depleted battery, Network outage.
- **Legal:** Failure to meet contractual/requirements, Violation of rules and regulations/Breach of legislation/Abuse of personal data.
- **Physical Attack:** Sabotage, Vandalism, Theft, Side channel attacks, Fault injection attacks.
- **Failure/Malfunctions:** Failure or malfunction of a sensor/actuator, Software vulnerabilities exploitation, Failure or disruption of services.
- **Threat against (semi-) autonomous systems:** Threats targeting autonomous sensors, Threats against AI and ML.
- **Remote attacks:** short-range (man-in-the-middle) long-range (API-based attack).

It is important to mention that the family **remote attack** and the type of threats are taken from the Upstream report [18] since the ENISA does not consider this family. For instance, considering that ECUs communicate with each other, an attacker could inject malicious messages using the CAN bus since the protocol does not implement encryption mechanisms. This attack can be mapped as *Nefarious activity/abuse* with *Manipulation of Information*.

7.1.7 Objective-domain-threats matrix

OBJECTIVES	FAMILY OF THREATS								
	FT-1	FT-2	FT-3	FT-4	FT-5	FT-6	FT-7	FT-8	FT-9
Security by design		D3,D5			D3,D5	D1,D2			D1,D2
Privacy by design		D4							
Asset management	D1,D4,D5	D4,D6	D4,D6	D4,D6	D4,D6	D1,D4,D5	D1,D4,D5		D1,D4,D5
Risk and threat management	D4,D6	D3	D3			D4,D6	D4,D6	D4,D6	D4,D6
Relationships with suppliers		D3	D3		D3	D3	D3		
Training and awareness		D3							
Security management	D3,D4	D3,D6	D3	D3	D3,D4,D6	D3,D4	D3	D3,D4	D3,D4,D6
Incident management	D4,D6	D4,D6		D4,D5,D6		D4		D4,D5,D6	
Detection	D1,D2,D3,D5,D6	D3,D5	D3,D5	D3,D5		D1,D3,D5,D6	D1,D2,D3,D5,D6		D3,D5,D6
Protection of networks and protocols	D1,D2,D5	D1,D2,D3,D5				D1,D2,D3,D5	D1,D3,D2,D5		D3
Software security	D1,D4,D5	D1,D3,D4,D5	D1,D4,D5			D1,D5			
Cloud security	D4,D5	D3,D4,D5		D3,D5	D3		D4		
Cryptography	D1,D2,D5	D1,D2,D5	D1,D2,D5		D5	D1,D5	D1,D2,D5		
Access control	D2,D5,D6	D2,D5,D6	D2,D5			D2,D5	D2,D5		
Self-protection and Cyber Resilience	D1,D2,D5	D1,D2,D5					D1,D2,D5		
(Semi-) autonomous systems	D1,D5	D1,D5		D1,D5		D1		D1,D5	
self protection and cyber resilience									
Continuity of operations	D3	D1,D3	D1,D3,D4	D3	D3	D3	D1,D3,D4	D3	

FAMILY OF THREATS FT-1: Eavesdropping/Interception/Hijacking; FT-2: Nefarious activity/Abuse; FT-3: Unintentional damages (accidental); FT-4: Outages; FT-5: Legal; FT-6: Physical attack; FT-7: Failure/Malfunctions; FT-8: Threat against (semi-) autonomous systems; FT-9: Remote attack

Table 7.3: Matrix objective-domain-family of threats of MARISMA-CAR pattern.

Another matrix (objectives, domains, and threats) for MARISMA-CAR has been defined and shown in Table 7.3. The ENISA [104] does not define a specific mapping but reports a description of the family of threats and the objectives in Annex C [104]. In particular, the report defines a set of security controls grouped

by security objective and identifies a threat group for each security measure. The information reported in Annex C can be helpful in understanding which type of security measure can be used to protect one or more assets from a family of threats. Based on this information, the domains are mapped based on the family of threats and the security controls that shall be considered in order to mitigate the threats. For example, we want to prevent physical and remote attacks. We can adopt security controls that belong to the security by design, considering the domains D1, D2, and D5 (Table 7.3). In this way, we can ward off physical and remote attacks by implementing techniques and methodologies that consider the use of cybersecurity from the design phase of the components. Among these, for example, we can have the design of secure software or hardware. Finally, depending on the domain involved for that objective, the threat can be different because a threat can attack an objective but not in all of its domains [107].

7.1.8 T-assets/T-threats/dimensions matrix

The last matrix that needs to be defined is the matrix of type of assets, type of threats, and dimensions, shown in Table 7.4. Since the ENISA report does not provide a specific association between the threat, the affected assets, and the security controls within the security domains, the authors' knowledge of safety and risk assessment was used. In particular, the tables from Annex B and C were used to extract details on the relationship between the threat, the impacted asset, and the security controls from the table in Annex C. The resulting matrix establishes the existing relationship among the types of threats and dimensions for each type of asset that is most likely to be attacked. Table 7.4 shows the type of asset and the type of threats that can attack the given type of assets.

Family of Threats Type of Threats	FAMILY OF ASSETS											
	FA-1	FA-2	FA-3	FA-4	FA-5	FA-6	FA-7	FA-8	FA-9	FA-10	FA-11	FA-12
Malicious activity/Abuse												
Denial of Service	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	C R1 R2 S	S
Malware	C S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	C P S R2	S
Manipulation of hardware and software	C S R1 R2	C P R1 R2 S F O	C P S R1 R2 O	C S R1 R2 O	C P S R1 R2 O	S R1 R2 O	C P S R1 R2 F O	C P S R1 R2 F O	C R1 F	C P R1 R2	C R1 S	S
Manipulation of Information	C S R2 O	C R2 O	C S R2 O	C S R2 O	C S R2 O	S R2 O	C S R2 O	C S R2 O	C S R2 O	C R2 O	C R2	S
OEM Targeted attacks	P F	P F	P F	P F	P F	P F	P F	P F	P F	P F	P F	P F
Unauthorised activities	C F	C F	C S	C S	C S F	C S	C S	C S	F S	C S	C S	S F P
Identity theft	C S O	C S O	C S O	C S	C S O	C S	C S O	C P S O	C S P F	C P	C P	C P S
Abuse of authorizations	C S	C P F	C P S	C S O	C P S	S O	C P S O	C P S O	P F	C P	C P	C P
Protection of networks and protocols	C S	C P F	C P S	C S O	C P S	S O	C P S O	C P S O	P F	C P	C P	C P
Threats against (semi-) autonomous systems												
Threats targeting autonomous sensors	C S O	C S O				C S O						
Threats against AI and ML	C S R2	C S O		C S R2		C S O						
Sabotage	C S	C S	C S	C S	C S	C S	C S	C S	C S	C S	C S	C S
Vandalism	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F
Theft	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F	S F
Side-channel attacks	C P R1 R2	C R1 R2 O	C R1 R2 O		C P R1 R2	C P R1 R2			C P F			
Fault injection	C R1 R2 O	C R1 R2 O	C R1 R2 O			C R1 R2 O						
Failures / Malfunctions												
Failure or malfunction of a sensor/actuator	R1 R2 O	R1 R2 O	R1 R2 O	R1 R2 O		R1 R2 O			R1 R2 O			
Software vulnerabilities	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O
exploitation												
Failure or Disruption of service		C S R1 R2 O		C S R1 R2 O		C S R1 R2 O	C S R1 R2 O	C S R1 R2 O	C R1 R2 O		C S R1 R2 O	
Eavesdropping/Interception/Hijacking												
Communication protocol hijacking	C	C	C	C	C	C	C	C	C	C	C	C
Data replay	C	C	C	C	C	C	C	C	C	C	C	C
Man-in-the-middle attack / Session hijacking	C P S	C P S	C P S	C P S	C P S	C P S	C P S	C P S	C P S	C P S	C P S	C P S
Unintentional Damages (accidental)												
Unintentional change of data or car components	S O	S O	S O	S O	S O	S O	S O	S O	S F O	S O	S O	S O
configuration		P										P
Information leakage												
Using information and/or devices from an unreliable source	S R2	S R2	S R2	S R2	S R2	S R2	S R2	S R2	P F S R2	R2	S R2	S R2
Erroneous use or configuration of car components	S O	S O		S O	S O	S O				S O		S O
Outages												
Loss of GNSS signal	C			C		C	C	C		C		C
Car depleted battery	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O	C S O
Network outage												
Legal												
Failure to meet contractual requirements	P S F	P S F	P S F	P S F	P S F	P S F	P S F	P S F	P S F	P S F	P S F	P S F
Violation of rules and regulations/Breach of legislation / Abuse of personal data	P	P	P	P	P	P	P	P	P	P	P	P
Remote Attack												
Short Range	C S	C P F O	C P F O	C P S	C P F O	C P F O	C P F O	C P F O	C P F O	C P R1 R2	C P R1 R2	
Long Range		C P F O	C P F O	C O	C P F O	C P F O	C P F O	C P F O	C P F O			

DIMENSIONS C: Cybersecurity; P: Privacy; R1: Reliability; R2: Resilience; S: Safety; F: Financial; O: Operational.
Family of Assets FA-1: Vehicles sensors and actuators; FA-2: Information; FA-3: Communication network and protocols; FA-4: Decision making algorithms; FA-5: Inside vehicle communication components; FA-6: Vehicle functions;
FA-7: Nearly external components; FA-8: Servers, Systems, and Cloud Computing; FA-9: Software management; FA-10: Mobile devices; FA-11: Network and Domain isolation features; FA-12: Humans

Table 7.4: Matrix of T-Assets/T-Threats/Dimensions for MARISMA-CAR pattern.

7.2 ICSEQAI: Industrial Cyber Security Education with Quantum Artificial Intelligence

The impact of cybersecurity is global, requiring immediate attention to protect, conserve, and maintain the integrity of any data. The need for cybersecurity is of the utmost importance in industry and academia. All stakeholders should have substantial knowledge about cybersecurity and how to implement it. As described in Section 3.3, the use of published generic standards and guidelines does not describe the technologies or solutions that can be used. For this reason, this thesis proposes a vision model based on Quantum Artificial Intelligence (QAI) that generates secure software development (SSD) rules to educate and train developers and testers during different phases of the Software Development Life Cycle (SDLC). The proposed approach [66], called Industrial Cyber Security Education with Quantum Artificial Intelligence (ICSEQAI), has a high level of abstraction of a learning framework to support developers and testers during the

SDLC (Figure 7.4).

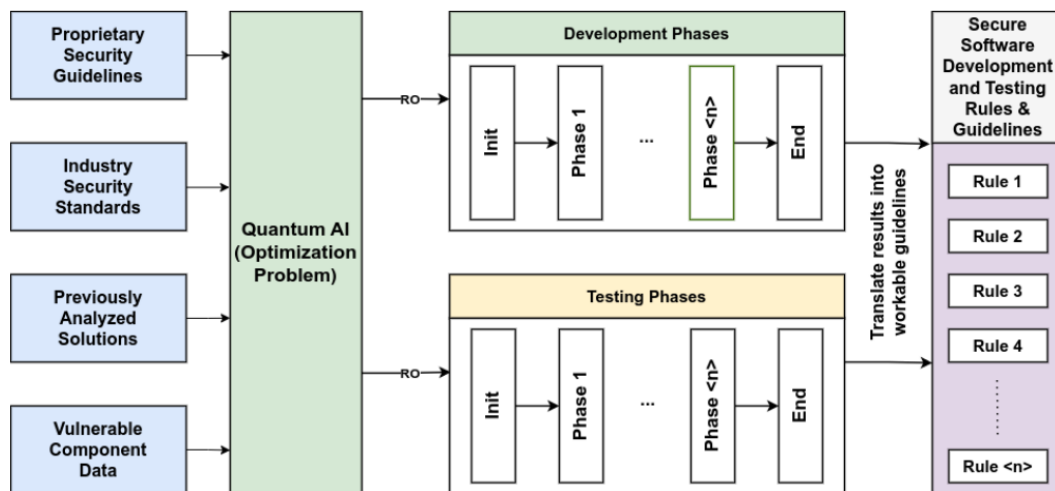


Figure 7.4: ICSEQAI Vision Model to support Developers and Testers during SDLC.

As per the ICSEQAI vision model, artifacts, including but not limited to Company Proprietary Security Guidelines, Industry Security Standards, previously analyzed solutions or reports, and vulnerable component data, can be processed and repurposed to support decision-making by the developers and testers during the SDLC. These artifacts can be translated into specific rules necessary for secure software development. The rules can be mapped using industry standards, such as automotive industry standards or global application standards like OWASP Application Security Verification Standard², to generate a robust model for decision-making. QAI can be trained for such a constrained model to provide an optimal solution, which can be used to update existing security guidelines or standards and educate the developers and testers regarding CS during SDLC.

To understand the model, a secure software development process to fix a vulnerable automobile component has been presented. Since ISO/SAE 21434 does not provide concrete design ideas, it makes it difficult to assimilate directly into the SDLC. Therefore, the guide can be split into multiple rules for SDLC support. Vulnerable ECU, building blocks of a car's internal network, can provide invaluable information about specific attacks and details about the vulnerabilities. Historical security analyses, solutions, and fixes for previous ECU attacks and vulnerabilities are equally important as they contain precious information about specific drawbacks and implemented fixes. A constrained optimization problem can be formulated from the dataset of similar rules, standards, data, and constraints based on ISO/SAE 21434, OWASP, and SDLC, which can be trained

²<https://owasp.org/www-project-application-security-verification-standard/>

on QAI algorithms to perform Rule Optimization (RO in Figure 7.4), generating optimal secure development rules for vulnerability fixes. For example, ICSQAI would suggest the best K strategies among N different actions suited to a specific developmental phase, where K is the constraint (rules to fix vulnerability). These rules can be translated to update company security guidelines, propose updated standards, and educate and train Developers and Testers to adjust to newer security requirements and standards during SDLC.

The proposed ICSQAI model can be efficiently used to continuously train and educate the company resources, like architects, developers, and testers, on CS skills. The model can educate the developers by displaying automated messages providing details about secure coding rules applicable to a specific module during coding/unit testing. At the same time, the testers would receive onscreen comprehensive testing guidance for newly incorporated fixes. Finally, different QAI algorithms can be trained in different contexts to achieve the proposed goal. Quantum classification algorithms can be used to help the system classify vulnerable components. In contrast, optimization algorithms like constrained quadratic models (CQM) or discrete quadratic models (DQM) can generate the rules for training and education. Concurrently, reinforcement learning using variational quantum circuits can also help adapting new development guidelines to update the existing ones.

7.3 Comparison with the state-of-the-art

The state-of-the-art prevention techniques are based on encryption, authentication mechanisms, and risk assessment methodologies. However, as mentioned in Section 3.4, authentication and encryption techniques have several disadvantages that attackers can exploit to send malicious messages to the CAN bus. On the other hand, standard or risk assessment methodologies do not offer specific prevention techniques. In comparison to current approaches, the two methods proposed in this section aim to enhance the prevention phase, starting with the design of the components. The MARISMA-CAR pattern is a risk assessment methodology that can enhance vehicle security by using the suggested controls. This pattern not only aids in the risk assessment phase but also provides security measures based on the threats and assets using the controls. This approach helps automotive organizations improve the design component phase. On the other hand, ICSEQAI is a vision model that supports developers and testers during the SDLC.

Chapter 8

ANDURIL Case Study

This chapter presents an example to validate the model proposed in Chapter 4, considering the three operational units. For each dimension, one or more approaches are taken to create an automotive cybersecurity operational model. To be more specific, Chapters 5, 6, and 7 explain the approaches considering only one dimension. By combining one or more dimensions, an automotive cybersecurity model can be created that considers the three operational units (detection, response, and prevention). For example, if an organization only wanted to identify attacks, it could choose the techniques proposed in the *Automotive Security Architecture* dimension to provide *detection*, *response*, and *prevention* solutions. Taking this into account, the entire cycle is proposed in this chapter. Figure 8.1 shows a graphical representation of the one described above.

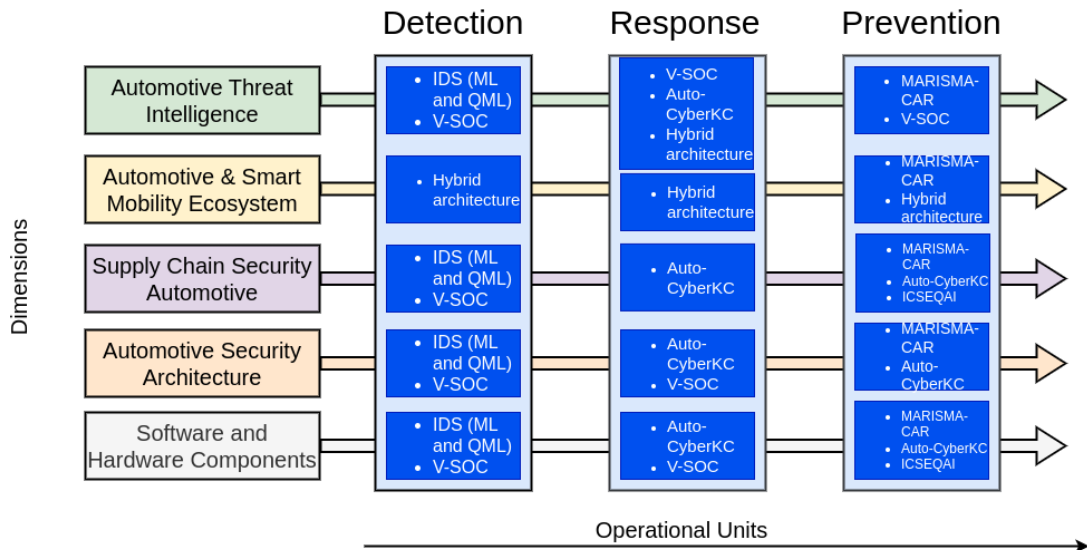


Figure 8.1: ANDURIL considering the horizontal dimension (Operational Units) by taking techniques and methodologies from the vertical dimension (Dimensions).

Regarding the dimension *Automotive Security Architecture*, MaREA (as IDS, see Section 5.4.1) and the V-SOC (see Section 5.2) are used for the detection phase. The decision to use this model is because it reaches good metrics compared to the others. MaREA was deployed into a Raspberry Pi 4 and then tested with an Arduino Elegoo Uno R3 using the same strategy described in Section 5.4.2 regarding the technology transfer. If MaREA identifies an attack, a message is created that contains information about a hypothetical vehicle and sent to the V-SOC. In this example, the Raspberry Pi 4 has the role of IDS and TCU. When an attack is identified, it is necessary to respond to it. For the response and the prevention phases, Auto-CyberKC is chosen because it is important to investigate the threat (with the V-SOC) and take decisions using the suggestions provided by this solution.

In the following section, the application model is explained.

8.1 Pachamama Methodology

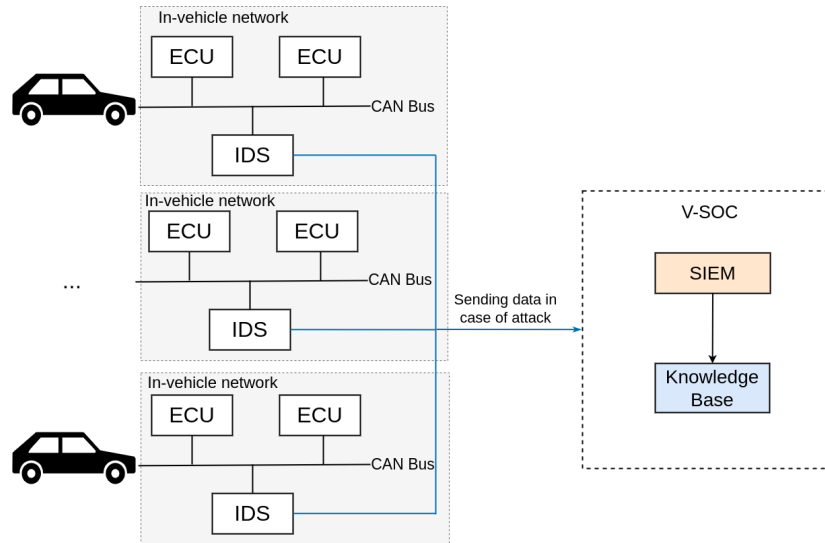


Figure 8.2: An overview of the Pachamama methodology phase.

The proposed architecture, reported in Figure 8.2, uses an IDS to identify CAN attacks, deploying a Machine Learning model into an embedded component to simulate a real-time identification. A SIEM receives a specific message from the IDS if it detects an attack. This message contains information used by the SIEM to generate offenses. Finally, an application extracts these offenses from the SIEM to generate a "*Knowledge Base*" that can be used to help the developers or the cybersecurity experts in the automotive industry understand the attack and make decisions and remediation.

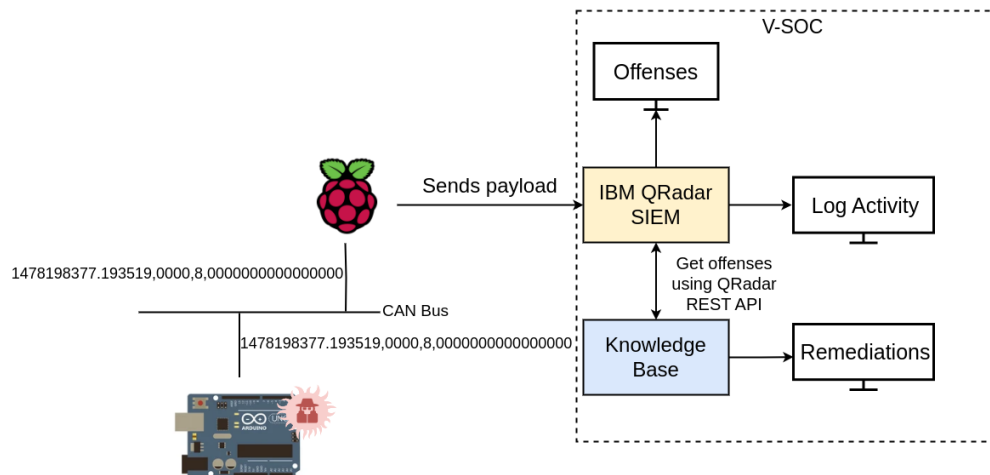


Figure 8.3: Example of DoS attack processed according to the Pachamama methodology.

The DoS attack was considered as an example (Figure 8.3). In the first step, an attacker identifies vulnerabilities in a vehicle to obtain access to the network. After this, the adversarial can inject malicious CAN data frames to disrupt the vehicle's behavior. According to Section 2.3.2, the DoS attack consists of injecting high-priority CAN data frames (encoding the *identifier* with **zeros**), in which an ECU can stop to communicate with the others.

Event Name	Vehicle ID (custom)	CanID (custom)	CanID IDS (custom)	DLC (custom)	DATA (custom)	DATAIDS (custom)
CAN Messages from Vehicle IDS Message	53cd3082a05	0	000000000000	8.0	0000000000000000	0000000000000000
CAN Messages from Vehicle IDS Message	53cd3082a05	0	000000000000	8.0	0000000000000000	0000000000000000
CAN Messages from Vehicle IDS Message	53cd3082a05	0	000000000000	8.0	0000000000000000	0000000000000000
CAN Messages from Vehicle IDS Message	53cd3082a05	0	000000000000	8.0	0000000000000000	0000000000000000

Figure 8.4: Log Activity in IBM QRadar after the SIEM receives messages from the IDS component.

Since the CAN protocol uses a broadcast mechanism to send messages on the bus, the IDS receives them to be analyzed. As explained in Section 5.4.2, the malicious ECU was simulated using an Arduino Elegoo Uno R3 and the IDS with a Raspberry Pi 4 Model B (Figure 8.5). MaREA analyzes the received malicious CAN data frame and creates a custom message for the SIEM that consists of several files:

- **VehicleID:** used in the SIEM to uniquely identify a vehicle and in the web application (Auto-CyberKC) to obtain information about the vehicle's manufacturer, model, and year.
- **EventID:** it maps an event related to the IDS in the SIEM.
- **CANID:** the CAN ID obtained from the bus.

- **DLC**: the data length code.
- **Data**: the payload encoded by a specific ECU and obtained from the bus.
- **CANID IDS**: the message identifier converted in the binary form.
- **DATA IDS**: the payload converted in decimal form.

For each vehicle, we can set a unique identifier (*VehicleID*) that, in this case, is *53cd3082a05*. The **EventID** is used to map the *Event Name* in QRadar. The value *send payload* was chosen, corresponding to a generic flow sent on the CAN bus. The **CANID**, **DLC**, and **DATA** fields were encoded according to the DoS attack (Figure 8.3). Finally, the **CANID IDS** and **DATA IDS** encode the MaREA's prediction. For this example, the "CANID IDS" contains eleven zeros, while the "DATA IDS" stores the DATA value in decimal form.

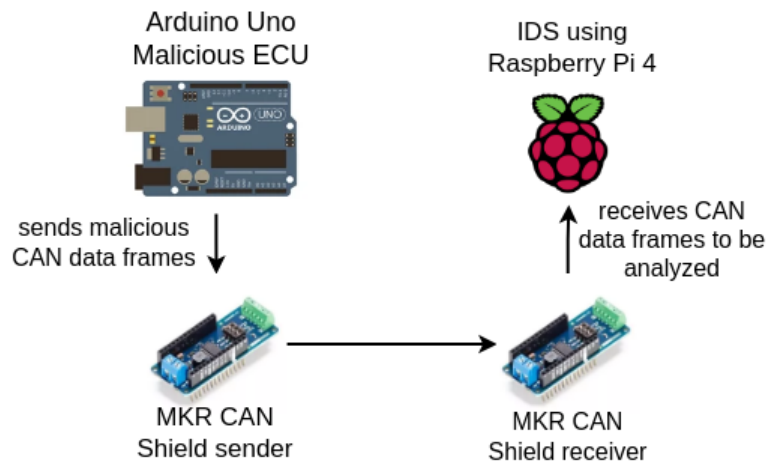


Figure 8.5: Simulation process in which the Arduino Uno (a malicious ECU) sends malicious CAN data frame, and MaREA (deployed on a Raspberry Pi 4) identifies if there are attacks.

Description	Offense Type	Offense Source
Vehicle: DoS attack high priority	Vehicle ID (custom)	53cd3082a05

Figure 8.6: Generated DoS offense.

Then, this message is sent to QRadar using the User Data Protocol (UDP) and port 514, which corresponds to the Syslog protocol to receive generic data. When the SIEM receives the message from the IDS component, it analyzes the DoS rule (see Section 5.2.1 Figure 5.7) and raises an offense (Figure 8.6).

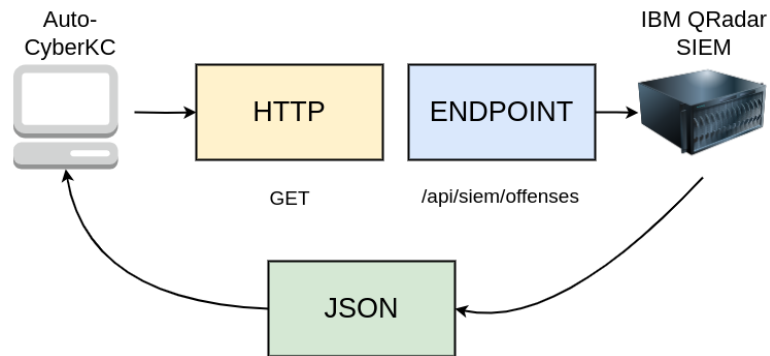


Figure 8.7: Interaction between the Auto-CyberKC and IBM QRadar SIEM using REST API.

In addition, Figure 8.4 shows the *Log Activity* after the SIEM receives messages from the IDS. In the **Knowledge Base**, the Auto-CyberKC obtains the offenses from QRadar using the REST API (Figure 8.7) and uses the AAD to filter the information based on the attack using the *Attack Class* field. For example, in this case, the attack class is DoS that can be mapped as *Denial-of-Service* or *Denial of Service* to be filtered into the AAD (according to the methodology explained in Section 6.2 and Section 6.3).

Year	2002 -
Description	Disrupting direct neighbor in WLAN network with several attack sources
Reference	Denial of service attacks at the MAC layer in wireless ad hoc networks Gupta - - -
Attack Class	Denial of Service UN-T29: Network design introduces vulnerabilities CAPEC-125: Flooding AAD-22: Distributed Denial of Service - -
Attack Base	Wireless Attack -
Attack Type	Simulation -
Violated Security Property	Availability -
Affected Asset	Reliability -
Vulnerability	CWE-664: Improper Control of a Resource Through its Lifetime CWE-770 Capture effect and unfairness at the media access
Interface	WLAN -
Consequence	The throughput is reduced for most nodes. At a high data rate from the attacker node, the throughput goes towards zero.
Attack Path	Single-Stage Attack 3 - - Two nodes can attack a server together even if they are not neighbors of the server node.
Requirements	Required Setup Simulation of the system Simulation with 12x12 network of nodes
Restrictions	Restriction 1: Physical Restriction Range - Restriction 2: Connection Restriction Bandwidth Limited Bandwidth

(a)

Disrupting direct neighbor in WLAN network with several attack sources	
CVE ID	CVE-2020-10281
CVE Description	This vulnerability applies to the Micro Air Vehicle Link (MAVLink) protocol and allows a remote attacker to gain access to sensitive information provided it has access to the communication medium. MAVLink is a header-based protocol that does not perform encryption to improve transfer (and reception speed) and efficiency by design. The increasing popularity of the protocol (used across different autopilots) has led to its use in wired and wireless mediums through insecure communication channels exposing sensitive information to a remote attacker with ability to intercept network traffic.
CVSS v2	5
CVSS v3	7.5
CWE	CWE-319 Cleartext Transmission of Sensitive Information
CWE Description	The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.
References	https://docs.google.com/document/d/1XtbD0ORNkhZ8eKrsbSIZNlyg9sFRXMXbsR2mp37Kblg/edit

(b)

Figure 8.8: The figure above (a) shows a portion of the categories taken from the AAD and suggested. The figure below shows one of the top three similar sentences obtained with the Sentence Transformer. In the red rectangle, the *Description* extracted from the AAD. In the yellow rectangle, the information obtained from the NVD.

Then, Figure 8.8 shows the remediation proposed by Auto-CyberKC using the AAD (Figure 8.8a) and the Sentence Transformer (Figure 8.8b). In more detail, Figure 8.8b shows the information obtained from the NVD about the DoS attack. The Sentence Transformer model probably recommended this **CVE Description** because the words *network*, *attack*, and *attacker* are in common with the **Description**.

The obtained references can be used to support SOC analysts during the attack phase or cybersecurity experts and developers during the design or fixing phase. For example, the reference shown in Figure 8.8b provides solutions to identify and resolve the vulnerability. In addition, the CWE's description suggests that an unauthorized actor can obtain critical information. Since the CAN protocol does not implement encryption techniques, the description can help the cybersecurity analyst find techniques or strategies to solve this problem.

Chapter 9

Conclusion, Limitation, and Future Work

This Chapter concludes this thesis by addressing the potential limitations of the proposed approaches. Additionally, it identifies and discusses avenues for future research and development.

9.1 Conclusion

The increasing integration of automatic and autonomous controls in modern vehicles, combined with the inherent vulnerabilities of automotive protocols and microcontrollers to cyber-attacks, has driven continuous research in automotive cybersecurity. In this context, this thesis aims to contribute to these efforts by proposing solutions to strengthen the security of internal vehicle networks. To respond to the research questions extracted from the first goal, Chapter 2 and Chapter 3 set up the background and threat models in the automotive context and propose an analysis of the current state-of-the-art solutions for secure communications applied to intra-vehicular networks and Electronic Control Units connected to the CAN network. This analysis spans the entire life cycle of a modern vehicle, starting from the design to the disposal phase, showing that many solutions consider only a single cybersecurity phase (for example, detection) using standard techniques like IDSs. Additionally, the results of the first goal highlight the necessity of new methodologies to enhance security in response and prevention and to improve the understanding of traditional automotive standards (like ISO/SAE 21434).

Based on the research questions addressed in the first goal, an **automotive cybersecurity model** named ANDURIL was developed to answer the research questions of the second goal. This model comprises a vertical dimension (*dimensions*) and a horizontal dimension (*operational units*). Various techniques and methodologies were proposed and evaluated for each dimension, where each so-

lution belongs to one or more operational units, encompassing key cybersecurity functions: detection, response, and prevention. The methodologies related to these functions are described in Chapter 5 (detection), Chapter 6 (response), and Chapter 7 (prevention).

Chapter 5 proposes several methodologies to identify CAN attacks. The first novel approach, known as Vehicle-SOC, is designed to support the detection, response, and prevention phase within the automotive context. Additionally, the CAN messages are sent directly to the V-SOC without altering the protocol's structure, unlike several approaches suggested in the literature. Further, various Intrusion Detection Systems (IDS) utilizing both Quantum Machine Learning and traditional Machine Learning algorithms have been proposed, which first address binary classification and then extend to multi-class classification. A cybersecurity educational framework was proposed to educate and train industry resources to raise automotive risk awareness. The next step adopted a technology transfer approach to test an IDS in real-time using a testbed using popular microcontrollers and handheld processors. Finally, a hybrid classical architecture integrating Quantum Machine Learning algorithms and a SIEM is proposed to boost Smart City security.

Chapter 6 combines the proposed V-SOC with Auto-CyberKC to provide a response phase using automotive standards and taxonomy to help cybersecurity analysts understand automotive attacks. Additionally, as it is crucial to inform the driver about any active threat, a prototype of an Android application has been proposed to assist the driver in managing the attack.

Chapter 7 proposes a novel preliminary pattern called MARISMA-CAR to manage the risk during the design phase. Finally, a vision model called Industrial Cyber Security Education with Quantum Artificial Intelligence (ICSEQAI) has been proposed as a high level of abstraction of a learning framework to support developers and testers during the Software Development Life Cycle (SDLC). The final contribution of this thesis is a case study to validate the automotive cybersecurity model considering vertical and horizontal dimensions, which is provided in Chapter 8.

9.2 Limitation

The first limitation of this thesis concerns how the experiments were conducted. Specifically, they were performed *in vitro* rather than *in vivo*. Given the computational and energy constraints of vehicle components, the choice of ML algorithm is crucial. In particular, the experiments conducted with these algorithms were performed using server hardware rather than real-world automotive components. However, the diversity of hardware components across different vehicles presents a significant challenge in developing a standardized hardware platform that can be universally adopted. To address this challenge, preliminary experimentation

was achieved using a Raspberry Pi 4, which aimed to overcome this limitation. By leveraging its computational capabilities and extensive support for various communication protocols, this approach is a potential stepping stone toward developing adaptable and scalable hardware solutions to bridge the gap between different vehicle architectures.

A second limitation concerns the attacks considered. Specifically, ML algorithms used in this thesis were trained with DoS (or flooding), spoofing (or malfunction), and fuzzing attacks. These particular attack types were selected because they are among the most commonly studied in cybersecurity research, allowing for a standardized evaluation of artificial intelligence-based detection methods. However, a notable constraint of the datasets used in this thesis is the lack of differentiation between various attack types, potentially limiting the algorithms' ability to distinguish between distinct threat patterns. To overcome this limitation, a synthetic dataset was generated to introduce additional attack scenarios, specifically incorporating fabrication and malfunction threats. However, since these are synthetic messages, they may not fully capture the complexity and variability of real-world cyber threats.

The third and final limitation identified pertains to the performance analysis of the V-SOC4AS, particularly in the context of scalability and real-world deployment. In practical scenarios, the number of vehicles transmitting data to a central server could be extremely large, potentially leading to network congestion, increased latency, and data processing bottlenecks. This raises concerns regarding the system's ability to handle high-throughput data streams while maintaining reliability and efficiency. To address this challenge, future research should conduct a comprehensive feasibility study to explore and evaluate scalable architectures, such as distributed data processing, or cloud-based load balancing mechanisms. These approaches could help mitigate message loss, optimize data flow, and enhance overall system performance.

9.3 Future Work

Future research will build on the limitations and findings of this thesis. One of the directions is to evaluate machine learning (ML) algorithms within embedded components, such as SMT (Surface-Mount Technology), to assess the trade-offs between cost and performance. Additionally, using the Raspberry Pi 4 in vehicles could be explored further to determine its feasibility in the automotive domain, particularly its ability to withstand extreme temperatures and handle large amounts of data.

Another potential enhancement involves expanding the SA-Hacking dataset by improving dataset generation techniques, incorporating real-world attack simulations, and validating synthetic data against actual vehicular network threats. These improvements aim to enhance the robustness and reliability of machine

learning-based intrusion detection systems. The expanded dataset could then be used to train an ML model, which would be tested on a vehicle in a controlled and secure environment.

A third direction for future research focuses on self-driving vehicles and the potential attacks they may encounter. Given the reliance of these vehicles on sensors and Advanced Driver Assistance Systems (ADAS), it is crucial to evaluate the resilience of machine learning (ML) and deep learning (DL) algorithms against adversarial learning attacks. Additionally, it is essential to adapt the algorithms proposed in this thesis to the emerging bus technologies, such as CAN-FD and automotive Ethernet, which are expected to replace the traditional CAN protocol.

Future research directions will focus on enhancing the response phase by exploring additional techniques to mitigate attacks without compromising vehicle safety. Resetting the ECU in a vehicle in motion poses significant risks due to its critical role in vehicle functionality. Therefore, it is crucial to investigate alternative strategies that can effectively neutralize threats without disrupting vehicle operation. Applying inferential statistics could help determine whether the observed differences in machine learning algorithms are statistically significant. However, this approach presents a challenge due to the limited availability of datasets in the automotive domain, which may not be sufficient for robust statistical analysis. Future research should focus on generating more comprehensive datasets and exploring additional attack scenarios to enhance the applicability of these methodologies.

Bibliography

- [1] LIN Consortium, “LIN Specification Package.” https://www.cs-group.de/wp-content/uploads/2016/11/LIN_Specification_Package_2.2A.pdf. Accessed: 2024-08-31.
- [2] I. A. Grzempa, *MOST: the automotive multimedia network*. Franzis Verlag, 2012.
- [3] S. Lee, B.-S. Cho, Y.-J. Choi, and K.-R. Baek, “Implementation of most/-can network protocol,” in *2011 International Conference on Electrical and Control Engineering*, pp. 5974–5977, IEEE, 2011.
- [4] FlexRay Consortium, “Flexray protocol specification.”
- [5] M. T. Baldassarre, V. Santa Barletta, D. Caivano, D. Raguseo, M. Scalera, *et al.*, “Teaching cyber security: The hack-space integrated model,” in *ITASEC*, 2019.
- [6] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, “Lstm-based intrusion detection system for in-vehicle can bus communications,” *IEEE Access*, vol. 8, pp. 185489–185502, 2020.
- [7] F. Sommer, J. Dürrewang, and R. Kriesten, “Survey and classification of automotive security attacks,” *Information*, vol. 10, no. 4, 2019.
- [8] C. Miller and C. Valasek, “Car hacking: for poories,” *Tech. rep., IOActive Report, Tech. Rep.*, 2015.
- [9] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.
- [10] Bosch, “CAN specification version 2.0.” <http://esd.cs.ucr.edu/webres/can20.pdf>, 1991. Accessed: 2024-06-28.
- [11] E. Aliwa, O. Rana, C. Perera, and P. Burnap, “Cyberattacks and counter-measures for in-vehicle networks,” *ACM computing surveys (CSUR)*, vol. 54, no. 1, pp. 1–37, 2021.

- [12] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, “Survey on security threats and protection mechanisms in embedded automotive networks,” in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 1–12, IEEE, 2013.
- [13] Y. Huo, W. Tu, Z. Sheng, and V. C. Leung, “A survey of in-vehicle communications: Requirements, solutions and opportunities in iot,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 132–137, IEEE, 2015.
- [14] P. Vasile, B. Groza, and S. Murvay, “Performance analysis of broadcast authentication protocols on can-fd and flexray,” in *Proceedings of the WESS’15: Workshop on Embedded Systems Security*, pp. 1–8, 2015.
- [15] I. I. S. Organisation, “ISO/SAE DIS 21434 Road vehicles—cybersecurity engineering,” 2021.
- [16] UNECE, “UNECE WP.29.” <https://unece.org/wp29-introduction>, 2024. Accessed: 2024-08-19.
- [17] T. Le, I. Elsayed-Aly, W. Jin, S. Ryu, G. Verrier, T. Al Rahat, B. B. Park, and Y. Tian, “Evaluating the dedicated short-range communication for connected vehicles against network security attacks.,” in *VEHITS*, pp. 37–44, 2020.
- [18] Upstream, “2024 Global Automotive Cybersecurity Report,” tech. rep., Upstream, 2024.
- [19] K.-T. Cho and K. G. Shin, “Fingerprinting electronic control units for vehicle intrusion detection,” in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 911–927, 2016.
- [20] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, “Cybersecurity challenges in vehicular communications,” *Vehicular Communications*, vol. 23, p. 100214, 2020.
- [21] J. Liu, S. Zhang, W. Sun, and Y. Shi, “In-vehicle network attacks and countermeasures: Challenges and future directions,” *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.
- [22] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, “Automobile driver fingerprinting,” *Proceedings on Privacy Enhancing Technologies*, 2016.
- [23] T. Hoppe and J. Dittman, “Sniffing/replay attacks on can buses: A simulated attack on the electric window lift classified using an adapted cert

- taxonomy,” in *Proceedings of the 2nd workshop on embedded systems security (WESS)*, pp. 1–6, 2007.
- [24] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, “A stealth, selective, link-layer denial-of-service attack against automotive networks,” in *Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14*, pp. 185–206, Springer, 2017.
- [25] S. Kamkar, “Drive it like you hacked it: New attacks and tools to wirelessly steal cars,” 2015. Presentation at DEFCON, 23.
- [26] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, “Practical dos attacks on embedded networks in commercial vehicles,” in *Information Systems Security: 12th International Conference, ICISS 2016, Jaipur, India, December 16-20, 2016, Proceedings 12*, pp. 23–42, Springer, 2016.
- [27] F. Langer, “Establishing an automotive cyber defense center,” 2019.
- [28] P. Meyer, T. Hackel, F. Langer, L. Stahlbock, J. Decker, S. A. Eckhardt, F. Korf, T. C. Schmidt, and F. Schüppel, “A security infrastructure for vehicular information using sdn, intrusion detection, and a defense center in the cloud,” in *2020 IEEE Vehicular Networking Conference (VNC)*, pp. 1–2, IEEE, 2020.
- [29] Upstream, “Vehicle Security Operation Center (vSOC).” <https://upstream.auto/solutions/vehicle-security-operations-center/>. Accessed: 2024-07-15.
- [30] H. Ye, L. Liang, G. Ye Li, J. Kim, L. Lu, and M. Wu, “Machine learning for vehicular networks: Recent advances and application examples,” *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018.
- [31] A. P. Singh and M. Singh, “Analysis of host-based and network-based intrusion detection system,” *International Journal of Computer Network and Information Security*, vol. 6, pp. 41–47, 07 2014.
- [32] E. Seo, H. M. Song, and H. K. Kim, “Gids: Gan based intrusion detection system for in-vehicle network,” in *2018 16th annual conference on privacy, security and trust (PST)*, pp. 1–6, IEEE, 2018.
- [33] M. Sami, M. Ibarra, A. C. Esparza, S. Al-Jufout, M. Aliasgari, and M. Mozumdar, “Rapid, multi-vehicle and feed-forward neural network based intrusion detection system for controller area network bus,” in *2020 IEEE Green Energy and Smart Systems Conference (IGESSC)*, pp. 1–6, 2020.

- [34] H. Lee, S. H. Jeong, and H. K. Kim, “Otds: A novel intrusion detection system for in-vehicle network by using remote frame,” in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 57–5709, 2017.
- [35] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, “Canet: An unsupervised intrusion detection system for high dimensional can bus data,” *IEEE Access*, vol. 8, pp. 58194–58205, 2020.
- [36] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero, “Cannolo: An anomaly detection system based on lstm autoencoders for controller area network,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1913–1924, 2021.
- [37] M. Zago, S. Longari, A. Tricarico, M. Carminati, M. G. Pérez, G. M. Pérez, and S. Zanero, “Recan–dataset for reverse engineering of controller area networks,” *Data in brief*, vol. 29, p. 105149, 2020.
- [38] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, “Intrusion detection system for automotive controller area network (can) bus system: a review,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–17, 2019.
- [39] D. Tian, Y. Li, Y. Wang, X. Duan, C. Wang, W. Wang, R. Hui, and P. Guo, “An intrusion detection system based on machine learning for can-bus,” in *Industrial Networks and Intelligent Systems* (Y. Chen and T. Q. Duong, eds.), (Cham), pp. 285–294, Springer International Publishing, 2018.
- [40] A. Alfardus and D. B. Rawat, “Intrusion detection system for can bus in-vehicle network based on machine learning algorithms,” in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0944–0949, 2021.
- [41] M. S. Salek, P. K. Biswas, J. Pollard, J. Hales, Z. Shen, V. Dixit, M. Chowdhury, S. M. Khan, and Y. Wang, “A novel hybrid quantum-classical framework for an in-vehicle controller area network intrusion detection,” *IEEE Access*, 2023.
- [42] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, “Mauth-can: Masquerade-attack-proof authentication for in-vehicle networks,” *IEEE transactions on vehicular technology*, vol. 69, no. 2, pp. 2204–2218, 2019.
- [43] G. Costantino, I. Matteucci, and D. Morales, “Earnest: A challenge-based intrusion prevention system for can messages,” in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (IS-SREW)*, pp. 243–248, IEEE, 2020.

- [44] D. K. Nilsson, U. E. Larson, and E. Jonsson, “Efficient in-vehicle delayed data authentication based on compound message authentication codes,” in *2008 IEEE 68th Vehicular Technology Conference*, pp. 1–5, 2008.
- [45] P. Mundhenk, S. Steinhorst, M. Lukasiewicz, S. A. Fahmy, and S. Chakraborty, “Lightweight authentication for secure automotive networks,” in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 285–288, IEEE, 2015.
- [46] VDA QMC Working Group 13 / Automotive SIG, “Automotive spice process assessment / reference model.” https://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf, 2015.
- [47] J. Dobaj, G. Macher, D. Ekert, A. Riel, and R. Messnarz, “Towards a security-driven automotive development lifecycle,” *Journal of Software: Evolution and Process*, p. e2407, 2021.
- [48] C. Plappert, D. Zelle, H. Gadacz, R. Rieke, D. Scheuermann, and C. Krauß, “Attack surface assessment for cybersecurity engineering in the automotive domain,” in *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 266–275, 2021.
- [49] J.-P. Monteuis, A. Boudguiga, J. Zhang, H. Labiod, A. Serval, and P. Urien, “Sara: Security automotive risk analysis method,” in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, CPSS ’18, (New York, NY, USA), p. 3–14, Association for Computing Machinery, 2018.
- [50] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner, “Sahara: a security-aware hazard and risk analysis method,” in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 621–624, IEEE, 2015.
- [51] NIST, “National vulnerability dataset.” <https://nvd.nist.gov/>, 2023. Accessed: 2023-07-09.
- [52] FIRST, “Common vulnerability scoring system.” <https://www.first.org/cvss/>, 2023. Accessed: 2024-07-09.
- [53] V. S. Barletta, D. Caivano, M. D. Vincentiis, A. Ragone, M. Scalera, and M. Á. S. Martín, “V-soc4as: A vehicle-soc for improving automotive security,” *Algorithms*, vol. 16, no. 2, p. 112, 2023.
- [54] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Magrì, and A. Piccinno, “Quantum optimization for iot security detection,” in *International Symposium on Ambient Intelligence*, pp. 187–196, Springer, 2022.

- [55] D. Caivano, M. De Vincentiis, F. Nitti, and A. Pal, “Quantum optimization for fast can bus intrusion detection,” in *Proceedings of the 1st international workshop on quantum programming for software engineering*, pp. 15–18, 2022.
- [56] M. De Vincentiis, A. Pal, A. Ragone, and M. Scalera, “A multi-class intrusion detection system for cyber security education in automotive industry,” in *IS-EUD Workshops*, 2023.
- [57] V. S. Barletta, D. Caivano, C. Catalano, M. De Vincentiis, and A. Pal, “Machine learning for automotive security in technology transfer,” in *World Conference on Information Systems and Technologies*, pp. 341–350, Springer, 2023.
- [58] D. Caivano, C. Catalano, M. De Vincentiis, A. Lako, and A. Pagano, “Marea: Multi-class random forest for automotive intrusion detection,” in *International Conference on Product-Focused Software Process Improvement*, pp. 23–34, Springer, 2023.
- [59] V. S. Barletta, D. Caivano, C. Catalano, and M. De Vincentiis, “Quantum-based automotive threat intelligence and countermeasures,” in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 548–554, 2024.
- [60] V. Santa Barletta, M. De Vincentiis, P. Buono, A. Pagano, and D. Caivano, “Detecting attacks on in-vehicle networks through a mobile app,” in *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 148–153, IEEE, 2023.
- [61] V. Santa Barletta, D. Caivano, M. De Vincentiis, A. Pal, and F. Volpe, “Automotive knowledge base for supporting vehicle-soc analysts,” in *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRINE)*, pp. 960–965, IEEE, 2023.
- [62] V. Santa Barletta, D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, “Hybrid quantum architecture for smart city security,” *Journal of Systems and Software*, vol. 217, p. 112161, 2024.
- [63] V. S. Barletta, C. Catalano, M. De Vincentiis, A. Pal, and M. Scalera, “Artificial intelligence for automotive security: How to support developers in automotive solutions,” in *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRINE)*, pp. 954–959, 2023.

- [64] M. De Vincentiis, F. Cassano, A. Pagano, and A. Piccinno, “Qai4ase: Quantum artificial intelligence for automotive software engineering,” in *Proceedings of the 1st International Workshop on Quantum Programming for Software Engineering*, pp. 19–21, 2022.
- [65] D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, “Extending developer support: Quantum artificial intelligence for automotive security,” QP4SE 2023, (New York, NY, USA), p. 7–12, Association for Computing Machinery, 2023.
- [66] M. T. Baldassarre, M. De Vincentiis, A. Pal, and M. Scalera, “Quantum artificial intelligence for cyber security education in software engineering,” in *IS-EUD Workshops*, 2023.
- [67] V. S. Barletta, D. Caivano, M. De Vincentiis, L. E. Sánchez, A. Santos-Olmo, and M. A. Serrano, “Marisma-car: A preliminary approach to risk assessment in automotive security,” in *IEEE MASS 2024 Workshop on Enhancing Security and Privacy in IoT-Enabled Applications*, IEEE, 2024.
- [68] N. I. of Standards and T. (NIST), “Framework for Improving Critical Infrastructure Cybersecurity.” <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf>, 2018. Accessed: 2024-08-19.
- [69] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Vehicular Communications*, vol. 21, p. 100198, 2020.
- [70] M. L. Han, B. I. Kwak, and H. K. Kim, “Anomaly intrusion detection method for vehicular networks based on survival analysis,” *Vehicular communications*, vol. 14, pp. 52–63, 2018.
- [71] A. Venturi, D. Stabili, F. Pollicino, E. Bianchi, and M. Marchetti, “Comparison of machine learning-based anomaly detectors for controller area network,” in *2022 IEEE 21st International Symposium on Network Computing and Applications (NCA)*, vol. 21, pp. 81–88, IEEE, 2022.
- [72] M. Marchetti and D. Stabili, “Read: Reverse engineering of automotive data frames,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.
- [73] A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, “Can-match: a fully automated tool for can bus reverse engineering based on frame matching,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12358–12373, 2021.
- [74] Vector Informatik GmbH, “DBC file format documentation,” 2007.

- [75] “Socketcan - Controller Area network.” <https://docs.kernel.org/networking/can.html>. Accessed: 2024-09-03.
- [76] N. Moustafa, “A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets,” *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.
- [77] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, “Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment,” *Sensors*, vol. 23, no. 13, p. 5941, 2023.
- [78] A. Madani, S. Rezayi, and H. Gharaee, “Log management comprehensive architecture in security operation center (soc),” in *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, pp. 284–289, IEEE, 2011.
- [79] G. Dupont, A. Lekidis, J. Den Hartog, and S. Etalle, “Automotive controller area network (can) bus intrusion dataset v2,” *4TU. Centre for Research Data*, 2019.
- [80] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, “Training a binary classifier with the quantum adiabatic algorithm,” *arXiv preprint arXiv:0811.0416*, 2008.
- [81] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, “Qboost: Large scale classifier training with adiabatic quantum optimization,” in *Asian Conference on Machine Learning*, pp. 333–348, PMLR, 2012.
- [82] M. Sami, *Intrusion Detection in Controller Area Network Bus Using Deep Neural Network*. California State University, Long Beach, 2020.
- [83] R. Gundu and M. Maleki, “Securing can bus in connected and autonomous vehicles using supervised machine learning approaches,” in *2022 IEEE International Conference on Electro Information Technology (eIT)*, pp. 042–046, IEEE, 2022.
- [84] S. C. Kalkan and O. K. Sahingoz, “In-vehicle intrusion detection system on controller area network with machine learning models,” in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, 2020.
- [85] R. Petersen, D. Santos, M. C. Smith, K. A. Wetzel, and G. Witte, “Workforce framework for cybersecurity (NICE framework),” *NIST Special Publication 800-801*, 2020.
- [86] E. M. Rogers, S. Takegami, and J. Yin, “Lessons learned about technology transfer,” *Technovation*, vol. 21, no. 4, pp. 253–261, 2001.

- [87] Microchip, “MCP2515 Stand-Alone CAN Controller with SPI Interface,” tech. rep., Microchip Technology Inc, Chandler, Arizona, USA, 2019.
- [88] “Superflow.” <https://www.ibm.com/docs/en/qsip/7.4?topic=monitoring-superflows>, 2023. Accessed: 2024-09-03.
- [89] G. Wassermann and Z. Su, “Static detection of cross-site scripting vulnerabilities,” in *Proceedings of the 30th international conference on Software engineering*, pp. 171–180, 2008.
- [90] “DVWA damn vulnerable web application.” <https://github.com/digininja/DVWA>, 2017. Accessed: 2024-08-27.
- [91] W. G. Halfond, J. Viegas, A. Orso, *et al.*, “A classification of sql-injection attacks and countermeasures,” in *Proceedings of the IEEE international symposium on secure software engineering*, vol. 1, pp. 13–15, IEEE, 2006.
- [92] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.
- [93] F. Sommer and J. Dürrwang, “IEEM-HsKA / AAD: Automotive Attack Database (AAD).” <https://github.com/IEEM-HsKA/AAD>, 2019. Accessed: 2024-08-22.
- [94] Microsoft, “The STRIDE Threat Model.” [https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)](https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)). Accessed on 2023-05-17.
- [95] MITRE, “CAPEC common attack pattern enumeration and classification.” <https://capec.mitre.org/>. Accessed on 2024-08-23.
- [96] P. Van Schaik, D. Jeske, J. Onibokun, L. Coventry, J. Jansen, and P. Kusev, “Risk perceptions of cyber-security and precautionary behaviour,” *Computers in Human Behavior*, vol. 75, pp. 547–559, 2017.
- [97] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2019.
- [98] F. Siddiqui, R. Khan, S. Y. Tasdemir, H. Hui, B. Sonigara, S. Sezer, and K. McLaughlin, “Cybersecurity engineering: bridging the security gaps in advanced automotive systems and iso/sae 21434,” in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pp. 1–6, IEEE, 2023.

- [99] G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, “Threat and risk assessment methodologies in the automotive domain,” *Procedia computer science*, vol. 83, pp. 1288–1294, 2016.
- [100] S. Chlup, K. Christl, C. Schmittner, A. M. Shaaban, S. Schauer, and M. Latzenhofer, “Threatget: towards automated attack tree analysis for automotive cybersecurity,” *Information*, vol. 14, no. 1, p. 14, 2022.
- [101] Z. Ma and C. Schmittner, “Threat modeling for automotive security analysis,” *Advanced Science and Technology Letters*, vol. 139, pp. 333–339, 2016.
- [102] D. G. Rosado, A. Santos-Olmo, L. E. Sánchez, M. A. Serrano, C. Blanco, H. Mouratidis, and E. Fernández-Medina, “Managing cybersecurity risks of cyber-physical systems: The marisma-cps pattern,” *Computers in Industry*, vol. 142, p. 103715, 2022.
- [103] ENISA, “Cyber Security and Resilience of smart cars,” tech. rep., 2017.
- [104] ENISA, “Good practices for security of Smart Cars,” tech. rep., 2019.
- [105] G. W. UNECE, “Un regulation no. 155—cyber security and cyber security management system,” tech. rep., Technical Report. United Nations, 2021.
- [106] National Institute of Standards and Technology (NIST), “Framework for improving critical infrastructure cybersecurity.” <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf>. Accessed: 2024-08-29.
- [107] D. G. Rosado, A. Santos-Olmo, L. E. Sánchez, M. A. Serrano, C. Blanco, H. Mouratidis, and E. Fernández-Medina, “Managing cybersecurity risks of cyber-physical systems: The marisma-cps pattern,” *Computers in Industry*, vol. 142, p. 103715, 2022.
- [108] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, “Framework for cyber-physical systems: Volume 2, working group reports,” tech. rep., NIST, 2017.
- [109] Tencent Security Keen Lab, “Experimental security assessment of mercedes-benz cars.” <https://keenlab.tencent.com/en/2021/05/12/Tencent-Security-Keen-Lab-Experimental-Security-Assessment-on-Mercedes-Benz-Cars/>. Accessed: 2021-05-12.
- [110] Tencent Security Keen Lab, “Experimental security assessment on lexus cars.” <https://keenlab.tencent.com/en/2020/03/30/Tencent-Keen-Security-Lab-Experimental-Security-Assessment-on-Lexus-Cars/>. Accessed: 2020-03-30.

- [111] Tencent Security Keen Lab, “Exploiting wi-fi stack on tesla model s.” <https://keenlab.tencent.com/en/2020/01/02/exploiting-wifi-stack-on-tesla-model-s/>. Accessed: 2020-01-02.
- [112] S. C. HPL, “Introduction to the controller area network (can),” *Application Report SLOA101*, pp. 1–17, 2002.
- [113] J. Wynn, J. Whitmore, G. Upton, L. Spriggs, D. McKinnon, R. McInnes, R. Graubart, and L. Clausen, “Threat assessment and remediation analysis (tara),” *MITRE Corporation: Bedford, MA, USA*, 2014.