Comparing Different Vehicle Architectures Based On Attack Path Analysis

Emilija Kastratović
1052407

Bachelor Thesis
VS-2019-B18

Examined by
Prof. Dr. rer. nat. Frank Kargl

Supervised by
M.Sc. Michael Wolf

Institute of Distributed Systems
Faculty of Engineering, Computer Science and Psychology
Ulm University

May 9, 2023

I hereby declare that this thesis titled:

Comparing Different Vehicle Architectures Based On Attack Path Analysis

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source.
I further declare that all my academic work was written in line with the principles of proper academic research according to the official "Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis" (University Statute for the Safeguarding of Proper Academic Practice).

Ulm, May 9, 2023

Emilija Kastratović, student number 1052407

ABSTRACT

The increased use of technology in modern vehicles has made cybersecurity a crucial part of the development process of modern vehicles, making it more and more critical for the safety and security of the vehicle and the privacy and personal data of drivers and passengers.

The internal vehicular network of such E/E systems plays a vital role in the overall cybersecurity of a modern vehicle. For example, attackers might exploit potential attack paths in the network to gain unauthorized access to a vehicle's systems or networks.

However, most security testing is done at later stages of development, when it is more complex and costly to address vulnerabilities. Additionally, due to the nature of security testing, it is carried out manually by experts with a high level of expertise and experience with other cybersecurity tools and techniques. These factors result in a stagnant security testing process that cannot maintain pace with the increasing complexity of modern vehicles.

This thesis focuses on comparing various vehicular network architectures in terms of which offers more security based on their attack path feasibility. A comprehensive evaluation of multiple architectures is conducted by discussing different criteria to evaluate the security of vehicular networks in an automated manner. The results offer a guiding beacon for future development and evaluation of vehicular network designs.

CONTENTS

1 INTRODUCTION

1.1 RESEARCH FIELD: CYBERSECURITY OF VEHICULAR NETWORKS

Modern vehicles are becoming increasingly reliant on technology, with a wide range of systems and components being connected to the internet and each other. This includes everything from infotainment systems and navigation to advanced driver assistance systems and autonomous driving features. ISO 26262 describes these so-called "Electronic/Electrical (E/E) systems" as systems that consist of electrical and electronic elements and components. Examples include Electronic Control Unit (ECU)s, sensors, actuators, connections, and communication systems like Controller Area Network (CAN), Ethernet, and Bluetooth [5]. As these technologies become more and more important, strong cybersecurity measures are also becoming increasingly important. Cybercriminals are constantly finding new ways to exploit vulnerabilities in these systems which can have serious consequences for users, which includes their safety, privacy, finances, and operational viability [4]. Ensuring the safety and security of connected vehicles is crucial to the success of the emerging world of connected and autonomous transportation.

The internal vehicular network architecture plays a crucial role in the overall cybersecurity of a modern vehicle as it determines how different vehicle systems and components are connected and communicate. Attack paths play an important role in vehicle networks and security because they define the specific routes that a malicious actor might use to attack a vehicle's systems or networks. A well-designed internal vehicular network architecture can help minimize cyberattack risk. With the increasing number of systems and components that are connected to the internet and each other, the complexity of the internal vehicular network architecture is also increased. This can make it more challenging to design and implement effective cybersecurity measures as more potential points of vulnerability arise that need to be addressed. Therefore, companies developing connected and autonomous vehicles need to prioritize cybersecurity in designing their internal vehicular network architecture, which can help to ensure the safety and security of the vehicle, as well as protect the privacy and personal data of drivers and passengers.

Security testing is, therefore, a crucial part of the development process. A Threat Analysis and Risk Assessment (TARA), for example, is performed early during the development process to identify and prioritize potential risks. This information can be used to guide the design and implementation of controls or countermeasures to reduce or mitigate these risks. Once the system is developed, e.g. a penetration test is carried out to validate the effectiveness of these controls and to identify any remaining vulnerabilities. The results of the pentest can then be incorporated back into the TARA process to update the risk assessment and prioritize future risk mitigation efforts. For example, companies can implement appropriate security measures by understanding the attack paths that might be used to gain unauthorized access to a vehicle's systems. These include encryp-

tion, authentication protocols, and firewall systems, etc. to protect against cyber threats.
In this way, the combination of a pentest and TARA provides a complete and iterative approach to security assessment.
However, security testing is often carried out in the late stages of development, which can lead to the discovery of vulnerabilities at a time when it is more difficult and costly to address. Additionally, they are considered to be a skill-based activity that is still carried out manually. It requires a high level of expertise and experience with other cybersecurity tools and techniques. The increased complexity of modern vehicles and the arduous nature of state-of-the-art security testing methods make it more unfeasible for companies to conduct them as is done now.
Thus, the need for a more efficient approach to improve overall security testing is evident. By automizing the process of evaluating automotive network architectures and their attack paths, potential vulnerabilities can be assessed early on in the development process resulting in overall more efficient security testing, improving the flexibility, costs and accelerateing it.

In this thesis we will explore a Criteria for Evaluation that can be used to evaluate the security of vehicular network architectures and how it can be used to automate the process of security testing.
This criteria is based on the experience of security experts and the results of a Survey conducted by the author of this thesis. This survey will feature a training set of architectures which the security experts will evaluate and give feedback on the architectures' security. Their ratings and feedback will be used as calibration for the criteria.

In order to automize the evaluation of vehicular network architectures, we will use a graph-based approach to model the network architecture and its attack paths and develop a Tool that can be used to evaluate the security of vehicular network architectures based on the criteria in an automated manner. The Proof Of Concept contains another set of architectures that will be used to test the criteria and the tool. In the end, we will evaluate the results of the tool and the criteria and discuss the limitations and potential improvements.

1.2 THESIS QUESTIONS

The questions this thesis will answer include:

- How can different E/E architectures be rated based on attack paths?
- How secure is the given vehicular network architecture?
- What architectural approach makes a network safer than others?
  - How do small changes in network positioning affect the network security overall?
  - How do simple and more branched out networks compare in terms of security?

2 STATE OF THE ART

There are various approaches to assessing the security of vehicular networks. Cybersecurity standards and frameworks give guidance and best practices for designing, implementing, and testing the cybersecurity of automotive systems and networks. The following standards are mentioned in virtually every piece of literature; thus, they are the most important ones to consider. Moreover, they offer a basis for the thesis itself as well as the development of a tool to assess the security of vehicular networks, as their further use is intended to be used in conjunction with these standards and frameworks to ensure compliance with them:

ISO 26262 is an international standard for ensuring the functional safety of E/E systems in vehicles. It provides a framework for the development and management of safety-related systems throughout the lifecycle of a vehicle, from design to operation. The standard is divided into several parts, each addressing a specific aspect of functional safety. It aims to help organizations manage the functional safety of their systems, ensure they meet a defined level of safety, and demonstrate that the systems are safe for the intended use. ISO 26262 helps organizations identify and mitigate potential hazards, reducing risks to an acceptable level. It provides a systematic approach to functional safety, ensuring the safety of vehicles and their passengers.

ISO 21434 is an international standard for the execution of functional testing and specifies engineering requirements for cybersecurity risk management regarding the lifecycle of E/E architecture systems in road vehicles, including their components and interfaces [4]. ISO 21434 outlines the security requirements, goals, and measures to consider throughout the entire lifecycle of a vehicle, including design, development, production, and operation. It aims to assist organizations in managing the security of their vehicles and ensuring they meet a defined level of security by providing a systematic approach to identifying and evaluating security-related risks and implementing measures to reduce them to an acceptable level. The standard is intended to be used in conjunction with ISO 26262, which focuses on functional safety, to provide a comprehensive approach to ensuring the safety and security of vehicles. It also explains how to integrate security into the functional safety process, helping organizations manage security risks similarly to how they manage functional safety risks.

SAE J3061 is a guide for cybersecurity best practices for the automotive industry and was created based on existing methods. The guide provides a comprehensive set of best practices for securing automotive systems and vehicles from cyber threats and covers various aspects of cybersecurity, including threat modeling, risk management, security testing, incident response, and security management. They are intended to be flexible, pragmatic, and adaptable in their further application to the vehicle industry and other cyber-physical vehicle systems. It provides a framework for organizations to incorporate cybersecurity into the lifecycle of vehicle systems, information on standard tools and methods used in designing and verifying systems, basic principles for cybersecurity, and a foundation for further standards development. SAE J3061 is intended to be used in conjunction with other standards and guidelines, such as those mentioned above [8].

AUTOSAR is a standard for the development of software for ECUs in the automotive industry [1]. The goal of AUTOSAR is to create and establish an open and standardized software architecture for automotive ECUs that is scalable to different vehicle and platform variants, transferable of software, considering availability and safety requirements, collaboration between various partners, sustainable use of natural resources, and maintainable during the product lifecycle. This improves the efficiency of development, enables the integration, exchange, reuse, and transfer of functions within a vehicle network, and helps manage the growing complexity of technology and economics in automotive software development.

TARA is an essential process for ensuring the cybersecurity of systems and networks, especially in the automotive industry. In the context of automotive systems, a TARA can be used to identify and evaluate potential threats and vulnerabilities to the electronic and electrical (E/E) architecture of vehicles. This includes identifying assets such as connected systems and networks and evaluating the likelihood and potential impact of threats such as cyberattacks, hacking, and software vulnerabilities as well as determine appropriate countermeasures to mitigate these risks. In addition, it can help prioritize these threats and vulnerabilities based on their potential impact on safety, financial, operational, and privacy aspects.
An attack path analysis can be an important component of TARA, helping to identify and evaluate the potential pathways that an attacker might use to gain unauthorized access to the vehicle's system. Based on such analysis, appropriate changes can be made to the vehicle's network architecture as a countermeasure. The TARA process can be used to ensure compliance with industry standards, such as ISO 26262 or ISO 21434, and can be integrated with other frameworks, like HEAVENS or EVITA. Regularly reviewing and updating the TARA process is crucial to keep up with the evolving threats and vulnerabilities in the automotive industry, which is constantly evolving with the integration of new technologies such as connected cars, autonomous driving, and V2X communication [7].

Important frameworks include HEAVENS, and EVITA. HEAVENS performs risk assessments of general IT systems and models explicitly built for automotive systems and uses threat and impact levels to calculate risks [3]. The primary objective of the framework is to identify security requirements and vulnerabilities in automotive systems and to provide countermeasures to minimize the risks associated with these vulnerabilities. It uses the Microsoft STRIDE model for threat modeling and aligns its impact level estimation parameters with established industry standards such as ISO 26262. It is a great candidate as a framework for automotive risk assessments over traditional IT risk assessment models.
The EVITA framework, which essentially performs the same things as HEAVENS, but also considers the potential of attacks to impact the privacy of vehicle passengers, financial losses, and the operational capabilities of the vehicles sys-

tems and functions [6].

Many of these approaches aim to standardize the process of assessing the security of vehicular networks. However, most of them are based on manual penetration testing and manual vulnerability assessment today. This is because penetration testing is an experienced-based and explorative skill that is difficult to automate. Further research aims to improve or couple existing approaches like performing a TARA, as well as automate and accelerate the process of security testing.

F. Sommer et al. introduce the concept of Model-Based Security Testing using an EFSM model [10]. The Automotive Security Model section describes an E/E architecture, security measures, and further development artifacts to protect vehicles against attacks. The model is based on the EFSM, with nodes representing attacker privileges and transitions representing a vulnerability. The Proof of Concept section of the paper demonstrates how the model can be used to identify different attack paths, analyze the model for attack paths, and execute the attack paths on a real vehicle. The incremental approach allows for the redefinition of attack paths, making it useful at different stages during development. The paper concludes that this approach can be helpful in identifying attack paths and potential vulnerabilities in automotive systems, thus helping to improve the security of vehicles.

F. Sommer et al. further expand on the same model-based approach by using a database of successful vehicular penetration tests [9]. The attack path generation process involves using an attack database, which describes vulnerabilities and exploits that can be used, including attack taxonomy and classification, attack steps, requirements, restrictions, components, and interfaces. The database can also be used to find new attack paths by permuting existing attack steps. Additionally, the process of creating the database can be done iteratively over several penetration tests and can be transferred to test scripts. However, there is a risk that the attack path generated may not be transferable, which can be circumvented by permuting previous attacks. The authors also suggest that further testing activities, such as black-box testing, should be carried out. Despite its limitations, this approach can be used as a useful tool to automate the process of penetration testing and improve the efficiency of security testing in the automotive industry.

J. Dürrwang et al. further describe the concept of attacker privileges mentioned in the papers above [2]. Several types of privileges that an attacker may seek to gain access to a vehicle's communication system and components are defined, such as "Read/Write," "Execute," "Read," "Write," and "Full Control.". They note that channels that are not protected by security measures can be immediately accessed by an attacker, but interpretation is needed in other cases. It also mentioned that the attacker needs to reach one of these privileges to access further attached communication systems and components. The authors applied their privilege model to real-world automotive security attacks to demonstrate its practical use, in which an attacker is connecting to the vehicle via the OBD port, which is connected to the central gateway via CAN, and then uses the central gateway to gain access to the internal vehicle network. They also showed the automatic generation of attack trees using a model checker in a custom software tool and an application of their privileges in security testing by describing attack paths. In future work, the authors plan to formalize the security testing approach to allow for early testing during development and to evaluate the TARA and security testing approach in a case study.

In contrast, D. Zelle et al. introduce a concrete approach, "ThreatSurf" [11], which presents an algorithm for automated generation and rating of attack paths in the automotive industry, using various attack building blocks and assessing attack feasibility. The attack feasibility assessment can be used in a TARA to assess an entire attack path of a threat scenario. It also discusses different methods for weighing attack paths, such as Sum, Average, Maximum, and Hybrid-weighted Sum. The paper describes four different types of threat agents - Thief and Owner, Terrorist, Organized Crime and Mechanic, Hacktivist, and Foreign Government - and their motivations, capabilities, and window of opportunity for performing an attack. The paper provides an example of how the proposed attack feasibility rating concept can be applied to threat scenarios derived from the use cases and also compares it with other rating approaches such as attack-potential-based approaches, CVSS based approaches, and attack vector-based approaches. The proposed attack feasibility rating concept is based on the attack-potential approach due to the complex nature of attacks against electric vehicles.
Other approaches include the CVSS and attack vectors. They conclude that attack-potential-based approaches have high flexibility but high complexity, CVSS-based approaches are easier to handle but have lower flexibility, and attack vector-based approaches are simpler but less suited for automotive applications. The tool and algorthmic approach fits the needs of the thesis much better than the model-based approach, as the feasibility rating is represented as a single variable and it is easier to compare multiple architectures.

While those papers mentioned above are a great foundation for this thesis, they do not provide a complete solution for the problem of automating the process of security testing in the automotive industry. The most important missing piece is the automatic evaluation of the possible attack paths and thus the automatic evaluation of the vehicle's architecture in general. While all of the approaches include a tool and an automaitc attack path search, they do not further consider the evaluation of the attack paths and architecture. This thesis aims to provide further steps to fill this gap by providing a tool as well as discuss different criteria to evaluate and even compare multiple architectures.

3 ARCHITECTURES

Similarly to a common network, a vehicle network is a system of electronic components and devices within a vehicle that communicate with each other to perform different functions, such as engine performance, safety, entertainment, and navigation.

What started as a simple network of a few electronic control units (ECUs) has evolved into a complex network of ECUs, and while the evolution of these systems has resulted in improved vehicle efficiency and features, it has also created new cybersecurity concerns.

There are several types of components in a vehicle network, and in order to understand these and their security features, we will first elaborate on those and define certain terms in this chapter that will be used throughout this thesis. Throughout this thesis, we will use the term *architecture* to describe a vehicular network architecture of ECUs and their connections.

3.1 FEASIBILITY

First, we want to explain and define the term *feasibility* in the context of this thesis. Normally, the term feasibility is used to describe the possibility of something to be done or achieved, or the likelihood of something to happen. This applies to the term in this thesis as well, as we will use it to describe the likelihood of an attack to be successful. This means, a higher *attack feasibility* means that an attack is more likely to be successful, while a lower *attack feasibility* means that an attack is less likely to be successful.

However, due to further factors used in this thesis, this thesis will also refer to the term of *feasibility rating* which refers to the security feasibility of a component, i.e. the likelihood of an attack is unsuccessful. Architectures and their components will receive a security feasibility, which a higher rating means that the component is more secure, while a lower rating means that the component is less secure. The rating or score an architecture receives is similar to a benchmark that is used to compare the security of different architectures. The higher the score, the better the rating and the more secure the architecture is.

3.2 COMPONENTS

While architectures in the automotive industry are complex, often featuring up to 150 ECUs in a single model, this thesis will focus on more manageable examples, each consisting of 20 ECUs Each architecture was represented as a graph and was modeled using these components:

- **ECUs**: The different ECUs in the architecture. They are the nodes of the graph.
- **Entry points**: The entry points to the architecture. The only possible entries are the external interfaces that an ECU might have.

- **Targets**: ECUs that are considered targets for an attacker. They are targets because they contain sensitive data or because they are critical for the vehicle's functionality.
- **Bus systems**: ECUs are connected to each other using bus systems. The bus systems are the edges in the graph. The possible bus systems are *CAN, CAN FD, LIN, FlexRay, MOST* and *Ethernet*.
- **Interfaces**: The interfaces are the connections between the ECUs and the external world. They are the connections between the ECUs and the bus systems. The possible interfaces are *Bluetooth*, *WiFi*, and *GNSS*.
- **Attack Feasibility**

Using the scale mentioned in 4.1 for each component, a higher rating means that the component is more secure, i.e. might have more security mechanisms in place, while a lower rating means that the component lacks such mechanisms. This also indicated that the higher the overall rating one architecture receives, the more secure it is. This subsection will further explain some the aforementioned components as well as their attack feasibility based on their attributes.

- **ECU**: An ECU is an embedded system in an automotive system that cotrnols various systems in the vehicle. ECUs are typically connected to each other using a bus system, enabling communication between the each other. Over the years, vehicles have become more and more complex, and thus the number of ECUs has increased. Since each ECU has an individual task, whose criticality varies, the attack feasibility of an ECU is based on the criticality of the task it performs. For example, the Engine Control Unit is responsible for the engine, and thus is a critical ECU, whereas the SEAT ECU, that are responsible for the seats in a vehicle, are not as critical. For simplicity, the ECUs and their feasbility rating in this thesis are divided into three as seen in 3.2.
- **CGW**: In the context of automotive network architectures, the CGW acts as a key communication node. It's a device that interconnects various ECUs or different networks within the vehicle. This can include networks operating on different protocols such as those mentioned below. The CGW is responsible for routing and sometimes filtering messages between these different networks. It also plays a crucial role in managing security, as it can help to prevent unauthorized access to the vehicle's internal networks. With the rise of connected and autonomous vehicles, the CGW has become more complex and sophisticated, capable of handling higher data throughput and managing more advanced security measures. It's an important component for the operation and security of modern vehicles.
- **CAN**: CAN a communication protocol utilized in modern vehicles and industrial applications to enable devices and microcontrollers to communicate with one another. It was developed by Bosch in the 1980s and has since become a widespread standard for in-vehicle communication. It is a dependable, durable, and cost-effective communication protocol that has become an indispensable component of modern vehicle electronics. Since the CAN bus is the most common bus system used in a vehicle, it will be set as a baseline for security in this thesis.

- **CAN FD**: It is an extension of CAN bus. It provides higher data transfer rates, more efficient data communication, and increased bandwidth compared to the original CAN bus. It is backward compatible with the original CAN bus, which means it can function on the same network as older CAN devices. In terms of security it will be rated a bit higher than CAN. Modern implementations of both protocols incorporate security features, such as message authentication and encryption.
- **FlexRay**: It is a bus system that is used in the automotive industry used to facilitate high-speed real-time communication. It was developed by a group of automotive companies and aims to meet the growing need for real-time communication in complex automotive systems. FlexRay is utilized to link ECUs requiring high bandwidth, such as advanced driver-assistance systems and safety-critical systems. It ensures deterministic data transfer, delivering data at predetermined intervals with a guaranteed maximum latency. This is crucial for safety-critical systems that rely on fast and dependable data transfer. FlexRay is designed for use in safety-critical systems and provides deterministic data transfer, making it highly reliable and resistant to interference. It also supports encryption and authentication, providing additional security features.
- **LIN**: LIN is a communication protocol used to connect and communicate with low-speed peripherals in modern vehicles and other industrial applications. It is a lower-cost alternative to the CAN bus communication protocol, developed by a group of automotive companies and generally considered to be less secure than other protocols, as it doesn't support encryption or authentication LIN is weaker bus system than CAN and CAN FD, in terms of technology but also use, which will contribute to the rating.
- **MOST**: It is a communication protocol that enables high-bandwidth multimedia data transfer between devices in modern vehicles. It was developed by a consortium of automotive and multimedia compaGNSnies and offers high-speed, dependable, and cost-effective data transfer. It is a highly dependable communication protocol that has gained increase popularity in the automotive industry and provides secure communication between devices through encryption and authentication
- **Ethernet**: Ethernet is a wired networking technology that enables the transfer of data between computers and devices within a local area network (LAN) It delivers fast, secure, and reliable data transfer, which makes it ideal for a variety of applications such as internet connectivity, file sharing, and multimedia streaming. Ethernet is commonly used in modern vehicles for connecting various devices, including infotainment systems, sensors, and advanced driver-assistance systems. It is a crucial part of modern automotive systems, providing dependable and swift data transfer for various applications, however, it is also expensive and complex to implement. While it is vulnerable to physical attacks, such as cable tapping, it is generally considered to be more secure than wireless technologies.
- **Bluetooth**: Bluetooth is a wireless technology used to exchange data between electronic devices over short distances. In modern vehicles, Blue-

tooth is used for hands-free phone calls, music streaming, and other features that require wireless connectivity. It has become an essential component in modern automotive systems, providing a convenient and safe way for drivers to interact with their vehicles. Though Bluetooth uses encryption to secure the communication between devices, vulnerabilities have been discovered in the past that allow attackers to intercept and decrypt Bluetooth traffic.

- **WiFi**: WiFi is a wireless communication technology that enables electronic devices to connect to the internet and exchange data wirelessly over a local area network (LAN) It uses radio waves to transmit data between devices, typically using the 2.4GHz or 5GHz frequency bands. In modern vehicles, WiFi is used for a variety of applications, such as infotainment systems, navigation, and telematics. It enables passengers to access the internet, stream video content, and browse the web, enhancing the in-car entertainment experience. WiFi also provides a means for vehicles to communicate with other devices, such as smartphones or smart home devices, to access remote services or home automation features. Though WiFi has improved in terms of security over the years, it is still vulnerable to various attacks, including weak encryption, insecure network configurations, and attacks on the wireless network itself.

- **GNSS**: GNSS is a satellite-based navigation technology that can determine precise positioning and timing information anywhere on Earth. It comprises various satellite navigation systems, such as GPS, GLONASS, Galileo, BeiDou, and other regional systems. GNSS uses a constellation of satellites orbiting the Earth to provide accurate location and timing data to receivers on the ground. These receivers analyze signals from the satellites to determine their position and velocity, making it possible to navigate precisely over long distances. GNSS is used in vehicles primarily for navigation purposes. It is generally considered to be the most secure technology since it is difficult to intercept and manipulate signals from GNSS satellites, and modern receivers incorporate security features to mitigate the risk of attacks.

Ranking these technologies in terms of security is a complex task, as each technology works differently, has different use cases and different security characteristics and vulnerabilities. However, based on experts' experience, these values were given each component:

Technology	Feasibility Rating
CAN	0.5
CAN FD	0.6
FlexRay	0.9
LIN	0.1
MOST	0.3
Ethernet	0.8
Bluetooth	0.4
WiFi	0.4
GNSS	0.8

Table 3.1: Ranking of automotive communication technologies based on security

ECU	Feasibility Rating
Target	0.9
LIN connected ECUs	0.1
Other	0.5

Table 3.2: Ranking of ECUs based on security

4 TOOL

Evaluation of multiple architectures by hand is a tedious and time-consuming task, so finding a way to automate this process is essential. A tool was developed to accelerate the evaluation of a vehicle network architecture based on their attack path feasibility by automatically generating the most feasible attack paths for the different complex architectures, as well as evalaute them based on the criteria.

The following sections will describe the tool's implementation.

4.1 CONFIGURATION FILES STRUCTURE

First, the architecture, ECUs and buses need to be defined in a configuration file. The configuration files for the simulation are stored in JSON format, which is a lightweight data interchange format that is easy to read and write, and easy for machines to parse and generate.

There are three main configuration files used in the simulation: buses.json, ecus.json, and graph.json. Each file has a specific structure and contains different attributes.

BUSES.JSON

The tool takes as input three sets of configuration files, each of them represented in the JSON format: the system architecture, the ECUs in the system, and the buses connecting the ECUs.

This file defines the different communication buses that are used in the simulation. It contains an array of objects, where each object represents a bus and has the following attributes:

- **type**: The type of the bus: *CAN, CAN FD, FlexRay, Ethernet, MOST, LIN*.
- **feasibility rating**: A value between 0 and 1 indicating the bus' attack feasibility (*see 3.1*).

ECUS.JSON

This file defines the electronic control units (ECUs) that are used in the simulation.

It contains an array of objects, where each object represents an ECU and has the following attributes:

- **name**: The name of the ECU.
- **type**: The type of the ECU (e.g., entry, target, interface).
- **feasibility rating**: A value between 0 and 1 indicating the ECU's attack feasibility (*see 3.2*).

GRAPH.JSON

This file defines the topology of the system being simulated, i.e. the architecture. It contains an array of objects, where each object represents a communication link which contains the ECUs and interfaces and has the following attributes:

- **name**: The name of the link.
- **type**: The type of the link (*CAN, CAN FD, FlexRay, Ethernet, MOST, LIN Bluetooth, WiFi, Ethernet, GNSS*).
- **ECUs**: An array of the names of the ECUs that are connected by this link.

Note that external interfaces (e.g., Bluetooth, WiFi, GNSS) are also considered as ECUs as well as buses in the simulation. Treating them as such simplifies the implementation of the tool using the following libraries. The feasibility of the interface can be set in either the buses.json or ecus.json file, however it is recommended to set it in the buses.json file. Whichever file is used, note that the feasibility must be set to 0 in the other file, to avoid double counting.

4.2 TOOL IMPLEMENTATION

The tool is implemented in Python 3.10 and newer. Python was chosen as the implementation language because it is a script-based language, allowing for quick prototyping and development, and is popular in the security domain. Additionally, the language offers a vast variety of libraries that can be used to implement the tool. The most important and useful library in the tool is *networkx* for graph manipulation and generation.

`main()` calls the following functions, which are integral to the tool's functionality:

`parse_files(architecture, ecu, bus)`: it takes in three parameters: `architecture, ecu, bus`. They are the above described configuration files and loads the contents of these files. Specifically, the function loads the contents of the architecture file, ECU file and bus file, and extracts the architecture name, architecture, ECUs configuration, and buses configuration, respectively. The function returns these four objects as a tuple.

`generate_graph(architecture, ecus_config, buses_config)`: This function takes three arguments as input: `architecture`, `ecus_config`, and `buses_config`. The function processes this input to create a directed graph (`nx.DiGraph()`) that represents the system's architecture and computes the feasibility between various nodes (ECUs). It returns the generated graph, a list of entry points, and a list of target ECUs.

Note: This is why it was easier to treat external interfaces as ECUs as well as buses. Since we are looking for the shortest path for an interface type, not the ECU using it, treating all interfaces as a bus and ECU conntected to the internal ECUs using such interface facilitates the shortest path generation.

The function returns the generated graph, along with the entry ECUs and target

Algorithm 4.1: Generate Graph

```
procedure GENERATE_GRAPH(architecture, ecus_config, buses_config)
    entry_points ← EmptyList
    target_ecus ← EmptyList
    G ← NX.DiGraph
    for all bus ∈ architecture do
        bus_type, bus_feasibility, bus_ecus ← ExtractInfo(bus)
        for all ecu ∈ bus_ecus do
            ecu_type, ecu_feasibility ← ExtractInfo(ecu, ecus_config)
            if ecu_type is entry or both then
                Append(entry_points, ecu)
            end if
            if ecu_type is target or both then
                Append(target_ecus, ecu)
            end if
            for all target_ecu ∈ bus_ecus do
                if target_ecu is interface then
                    G.addEdge(ecu, target_ecu, weight = bus_feasibility)
                else
                    target_ecu_feasibility ← ExtractInfo(target_ecu, ecus_config)
                    G.addEdge(ecu, target_ecu, weight = bus_feasibility + target_ecu_feasibility)
                end if
            end for
        end for
    end for
    entry_points ← MakeUnique(entry_points)
    target_ecus ← Set(target_ecus)
    return G, entry_points, target_ecus
end procedure
```

ECUs lists.

`find_attack_path`: This function is designed to find the feasibility and the shortest path from each entry ECU to each target ECU in a given directed graph (`nx.DiGraph`).

The resulting table can be used to analyze the most feasible attack paths between various entry points and target ECUs in the system's architecture.

Finally, `apply_criteria` applies the Criteria for Evaluation to the table. Each architecture receives a feasibility score and the result can be interpreted as a benchmark for the architecture's security.
Since the evaluation of the feasibility of the architecture is represented by a single number, it is easy to rank and compare the results of different architectures,

Algorithm 4.2: Find Attack Path

```
procedure FindAttackPath(G, entry_points, target_ecus_names)
    table ← EmptyDict
    table["feasibility"] ← EmptyDict
    table["shortest_path"] ← EmptyDict
    table["hops"] ← EmptyDict
    for all entry ∈ entry_points do
        table["feasibility"][entry] ← EmptyDict
        table["shortest_path"][entry] ← EmptyDict
        table["hops"][entry] ← EmptyDict
        for all target ∈ target_ecus_names do
            table["feasibility"][entry][target] ← Bellman-Ford(G, entry, target)
            table["shortest_path"][entry][target] ← nx.shortest_path(G, entry, target)
            if length(table["shortest_path"][entry][target]) ≥ 2 then
                table["hops"][entry][target] ← length(table["shortest_path"][entry][target]) − 2
            else
                table["hops"][entry][target] ← 0
            end if
        end for
    end for
    return table
end procedure
```

which the function is also designed to do.

It is easy to extend the tool to support other criteria.

Various *I/O functions* are responsible for printing the results, saving them to an external file, or quickly saving the graph as an image file and other *helper functions* are used to simplify the code.

Below is the implementation of the Bellman-Ford algorithm used in `Find Attack Path`.

The Bellman-Ford algorithm is a graph search algorithm that computes the shortest paths from a single source vertex to all other vertices in a weighted graph. It is very similar to the Dijkstra algorithm, but it can work with negative edge weights.

T Both algorithms are possible solutions to the our proble, but we chose the Bellman-Ford algorithm because it was mentioned in [11].

Algorithm 4.3: Bellman-Ford Algorithm

```
procedure BellmanFord(Graph, source)
    for all vertex ∈ Graph.vertices do
        distance[vertex] ← ∞
        predecessor[vertex] ← null
    end for
    distance[source] ← 0
    for i ← 1 to |Graph.vertices| − 1 do
        for all edge ∈ Graph.edges do
            if distance[edge.source] + edge.weight < distance[edge.target] then
                distance[edge.target] ← distance[edge.source] + edge.weight
                predecessor[edge.target] ← edge.source
            end if
        end for
    end for
    for all edge ∈ Graph.edges do
        if distance[edge.source] + edge.weight < distance[edge.target] then
            return "Graph contains a negative-weight cycle"
        end if
    end for
    return distance, predecessor
end procedure
```

5 CRITERIA PREPARATION

One of the most important aspects of this thesis is the criteria used to evaluate the different architectures.

In order to find and calibrate a good criteria, a survey was conducted in which ten architectures were evaluated by security experts in the field of automotive security. Their rating and feedback on this "training set" was used to calibrate the criteria used on the main architectures in this thesis. Doing a survey with such a set rather than the architectures themselves was done to avoid biasing the results, as well as have a well evaluated criteria. In addition, the training set is much larger than the main set, thus the results are more accurate.

5.1 TRAINING SET ARCHITECTURES

Ten architectures were used as a *training set* to determine and calibrate the criteria. The following section will describe the test architectures and estimate the impact the architectural choices have on the security of the system.

5.2 ARCHITECTURE 1

Architecture 1 represents a scaled down version of an architecture used in a real vehicle. It offers six entry points with three being targets, one of which is the *CGW*, and eight targets in total. Due to its many entry points, the attack surface is enlarged and the attack paths are more spread out, and we expect this architecture to not perform as well in terms of security as other architectures in this set.

5.3 ARCHITECTURE 2

Architecture 2 is essentially the same as Architecture 1, but without a CGW, to see how the architecture would perform without it. We expect this architecture to perform similarly or slightly worse to Architecture 1 and Architecture 5.

5.4 ARCHITECTURE 3

Architecture 3 was designed to group all the entry interfaces to see how a centralized entry would affect the architecture. Isolating the possible entry locations reduces the attack surface, prolongs the attack path to each target, and increases the number of ECUs used for the path, thus making the attack more difficult as the attacker would have to compromise more ECUs. It is similar to Architecture 10, in a sense that the entry points are centralized, however, in this case, the entry points are not isolated to a single ECU. We expect this architecture to be one of the most secure architectures out of this set due to its secluded entry points.

5.5 architecture 4

Architecture 4 offers entry points from all domain controllers, however, notice how every domain controller has only one type of interface. Having an entry point from each domain controller enlarges the attack surface, as the attacker would only need to compromise one ECU in each domain controller to gain quick access to the whole domain. It is essentially the same as Architecture 9, however, including the CGW. We expect this architecture to be one of the least secure architectures out of this set due to its many, spread out entry points.

5.6 architecture 5

Architecture 5 uses only Ethernet as the bus system, to see how the architecture would perform without any other bus system. Ethernet was chosen because it receives the securest attack feasibility rating of any bus system. We expect this architecture to perform similarly to Architecture 1 and Architecture 2.

5.7 architecture 6

Architecture 6 puts all the ECUs onto their own bus system. This increases complexity in communication between the ECUs but it also offers great isolation between the ECUs, however, it also reduces the hops per attack path and number of ECUs used for the path. This architecture does not make sense in a real world scenario becuase of its complexity but it is interesting to see how it performs and is not expected to be favorable to the security experts.

5.8 architecture 7

Architecture 7 has a similar idea to Architecture 6, but instead of putting all the ECUs on their own bus system, it puts all the ECUs that used the same bus system on the same bus system. Now, the feasibility of the bus rating is much more significant to the overall architecture rating and can indicate whether the used bus systems are secure enough. It will be interesting to see how this architecture performs and how it compares to Architecture 6.

5.9 architecture 8

Architecture 8 only has one entry point, namely the CGW. This architecture eliminates other entry points meaning the attack paths all start at the same point, thus reducing the attack surface. Additionally, the feasibility rating of the interfaces is more significant than before and the attack paths might become much more linear, thus the feasibility rating of the components. We expect this architecture to also do well in terms of security because of the singular entry point, however, the entry point is also very central which can reduce the security of the architecture.

5.10 ARCHITECTURE 9

Architecture 9 is essentially the same as Architecture 4, but without a central gateway. Again, the entry points are more dispersed resulting in an enlarged attack surface and the significance of the CGW can be deduced more from such a comparison and we do not expect this architecture to do well in terms of security and perform worse than Architecture 4.

5.11 ARCHITECTURE 10

Architecture 10 is the final architecture of the test set. It only includes one of each interface but more dispersed than that of Architecture 8. Note that the entries are also grouped together which shares the same idea ofArchitecture 3. We expect this architecture to be one of the most secure architectures out of this set due to its few, secluded entry points.

5.12 SURVEY

Security experts at Mercedes-Benz Tech Innovation were given ten architectures as shown in Architectures. The experts were asked to rank the architectures and give reasoning behind their ranking, as well as add any comments they had. Since there are ten architectures, the ranking went from 1 to 10, where 1 is the most secure and 10 is the least secure. To get to a result in the end, each rank was given a score, where the first place got 1 points, the second place got 2 points, and so on. In the end, each ranking or score each architecture got was summed up, and the architecture with the lowest score was the most secure.

Of course there are many ways how a security architect approaches an evaluation. Each architect comes from a different background with different experience and thus will see some factors as more or less important than another. Even though architects tend to agree in general about what makes an architecutre secure and what does not, opinions differ in the details. This result thus represents the architects' mean opinion on the security of the architectures.

The experts agreed that the number of hops between the entry and target is a criteria in evaluating the security of attack paths. The more hops there are, the more challenging it becomes for attackers to navigate from the entry point to the target, however not every hop is the same.

Separation and isolation, both of domains and ECUs, is a crucial consideration, as it enables compartmentalization and application of security measures to each domain or ECU individually. The Powertrain domain, which controls the engine, transmission, and other critical systems, is of particular concern, and its position in the network must be carefully considered. It was thus somewhat expected that Architecture 3 received the highest score, as it has all of the external intefaces placed in the same domain. Architecture 10 was also expected to receive a high score, as it is similar to Architecture 3 in terms of the placement of the external interfaces, but the Telematic domain controller being a target was a

concern and we actually expected it to rank better than the survey results did in the end.
Architecture 6 was also not expected to receive a good score, as the the maximum number of hops between the entry and target is 2 on average, but the experts agreed that isolation played a huge role in the evaluation.

However, excessive isolation would make the system too complex. Architectures Architecture 6 and Architecture 7 also provide high levels of isolation, but at the cost of increased overall complexity. Communication between ECUs must also be considered, as it is an essential part of the vehicle. Too many isolated ECUs communicating with each other can result in significant overhead. Media change, such as from CAN to CANFD or FlexRay or Ethernet, also introduces additional hurdles. That is why it is very surprising to see that Architecture 6 received such a good score in the survey and in addition for it to be in such stark contrast to Architecture 7 which received one of the worst ratings.

Other factors that need to be evaluated include the presence of a CGW and the number of external interfaces, as more interfaces mean more attack vectors and the need for more security mechanisms. Architecture 4 and Architecture 9 have most spread out interfaces, and thus they were expected to receive a low score, whereas Architectures Architecture 8 was expected to receive a high score, as the CGW is the only entry vector into the network. However, a CGW with no external interfaces is overall likely more secure than an architecture with no CGW at all, and that is why Architecture 2 received a higher score than Architecture 1. Of course, the type and the security of the external interface plays a role as well, because every interface has different security properties.

While the attack feasibility of each component accounts for the security mechanisms implemented in the vehicle, such as firewalls or IDCs, these mechanisms and the component's feasibility have not been explicitly stated in the architectures under consideration, since every component can vary in their implementation. However, these feasibility assessments are represented through the feasibility of each ECU, bus, and interface as explained in Section Configuration Files Structure. The experts agreed that the presence of security mechanisms is a critical factor in evaluating the security of the architectures. Such security measures impede attackers from progressing quickly through the attack path. For example, intelligent domain controllers that differentiate between domains are more effective than those that only forward messages. As a result, the final results varied between experts, as expected.

To summarize, architectures Architecture 3, and Architecture 8 received the best feedback, closely followed by architectures Architecture 6 and Architecture 10 and Architecture 2, whereas architectures Architecture 4, Architecture 7, and Architecture 9 received a the worst feedback, as they have been ranked last place at least once.

5.13 RESULTS - TRAINING SET ARCHITECTURES

The result is as follows:

Table 5.1: Rank and Architecture

Rank	Architecture	Score
1	Architecture 3	10
2	Architecture 8	11
3	Architecture 6	15
4	Architecture 10	16
5	Architecture 2	17
6	Architecture 1	25
7	Architecture 5	27
8	Architecture 7	28
9	Architecture 9	29
10	Architecture 4	30

6 CRITERIA FOR EVALUATION

Just as there are many ways to evaluate an architecture, there are many ways to a general criteria for this thesis. Similar to [11], we decided to approach the criteria as a mathematical equation that can be applied to every architecture equally. This equation takes into account the architecture feasibility, attack path feasibility, presence and influence of a CGW, amount of hops, isolation of ECUs and amount of interfaces/entry points.

Finding the most fitting equation was done in a manual, brute-force like attempt. The following chapter will describe the equation, the factors that are taken into account, and the reasoning behind them.

6.1 EQUATION FACTORS

We define a set of architectures A, where each architecture a has a set of attack paths P:

$$A := Architectures \qquad a \in A \tag{6.1}$$

$$P := AttackPaths_a \qquad p \in P \tag{6.2}$$

Next, we define the factors that are taken into account for the equation.

Since the Bellman-Ford algorithm finds and takes the lowest weighted attack path, i.e. the least secure one, a high score for this path means the most feasibile path receives a high security rating. Multiplying the feasibility of each attack path with the amount of hops it has, we also take into account the amount of hops in each attack path. The more hops there are, the better. This is because the more hops there are, the more ECUs there are to compromise, and the more ECUs there are to compromise, the more likely it is that the attacker will be hindered to reach the target ECU.
We calculate the architecture feasibility $feasibility_a$ by multiplying the feasibility of each attack path $feasibility_p$ with the amount of hops $hops_p$ in each attack path:

$$feasibility_a := \sum_p feasibility_p * hops_p * cgw_a \tag{6.3}$$

The CGW also plays a role in the security of an architecture. As mentioned before, the benefit of a CGW is dependent on whether it is present, an entry point, and or a target. A CGW is beneficial if it is present, but not an entry point or a target. The CGW factor is 1 by default, and is then decreased by 0.15 if the CGW is not present, by 0.1 if the CGW is an entry point, and by 0.05 if the CGW

is a target. This reflects the opinion of the security experts.
To include the influence of a CGW, we define the CGW factor cgw_a as follows:

$$cgw_a := \begin{cases} 1, & \text{base case} \\ cgw_a - 0.15, & \text{if } \notin a \\ cgw_a - 0.1, & \text{if external interface} \in a \\ cgw_a - 0.05, & \text{if target} \in a \end{cases} \tag{6.4}$$

In this thesis we define *hops* in an attack path as the amount of $ECUs - 1$ in the path from entry point to target. The interface itself does not count as a hop, only the entry point, i.e. if the entry point is also the target, the amount of hops is 0. To later norm the amount of hops, we set the hops factor $hops_a$ as follows:

$$hops_a := \sum_p hops_p \tag{6.5}$$

Isolation of an architecture is defined as the amount of ECUs per bus, which we call the separation factor $separation_a$:

$$isolation_a := \text{average amount of ECUs per bus in } a \tag{6.6}$$

And lastly, we define the amount of entry points as the interfaces factor $interfaces_a$:

$$entrypoints_a := \text{total amount of entry points in } a \tag{6.7}$$

6.2 EQUATION

Iterating over each of the ten architectures, their factors were given a weight and were then arranged in different mathematical operations. These weights were iterated from factors between 0 and 100, resulting in millions of combinations for every attempted equation. Each combination of the ten architectures using the same weights was then represented as a ranked table. In the end, the weights of the tables which had the smallest eucledian distance to the survey table 7.1 were taken into consideration. There were hundreds of feasible weight combinations, and in the end the smallest possible weights were chosen.

Finding an equation that fit the criteria was a step-by-step progress. In genreal, we asked ourselves the questions:
Which of the factors is it favorable to have more of? and *Which of the factors is it unfavorable to have more of?*

Let's consider any one architecture:
It is clear that a high architecture feasibility score and a high number of hops in each attack path is the most desirable. To get the architecture feasibility, each attack path feasibility is multiplied by the amount of hops it has 6.3. We know

that the more difficult the path, the more secure the architecture and the more hops there are in a path, the more difficult it is to reach the target ECU.
This is then muplitplied by the CGW factor 6.4.
To get a reasonable rating score, the product of the feasibility and CGW factor are multiplied by 100.

Next, we consider the factors that are unfavorable to have more of.
The isolation of an architecture, i.e. the amount of ECUs per bus, is unfavorable to have more of. This is because the more ECUs there are per bus, the more ECUs there are to compromise on one bus. Taking a look back at Architecture 6, which received a good rating, we can deduce that the more isolated the ECUs are, the better.
The amount of interfaces is also unfavorable to have more of. In this case, we only take into account the amount of entry points, and not the amount of total interfaces. Doing so, we are able to differentiate between one ECU with many interfaces and many ECUs with one interface since more entry points are also likely more spread out. Combining entry points and the amount of hops, we can deduce whether the entry points are secluded or spread out. High hops most likely means that the entry points are all grouped together, and low hops means that the entry points are spread out.

In the end, we receive the following equation:

$$rating_a := \frac{100 * feasibility_a * cgw_a}{hops_a * w_1 + isolation_a^{w_2} + interfaces_a * w_3} \tag{6.8}$$

To figure out the weights w_1, w_2, and w_3, we iterated over the factors from 0 to 100 for each architecture. Each permutation was then ranked, and the weights of the permutations that had the smallest eucledian distance to the survey table 7.1 were taken into consideration. In the end, the permutation with the smallest weights was chosen, however, the other options were possible as well, as the resulting scores did not differ much in terms of how close an architecture is ranked. w_1 was chosen to be 1, w_2 was chosen to be 4, and w_3 was chosen to be 32.

Notice how w_2 is an exponent, and not a multiplication. It did start out as one, but during our testing we found that the best results were achieved with the weight being 99, indicating that the isolation factors plays a more important role than the other factors and initially believed, and therefore decided to make it an exponent. We experimented with omitting w_1 completely, as the weight is only 1, however, results were not as good as with w_1 included. For example, the score discrepancy between Architecture 10 and the other architectures was much higher without w_1 included, Architecture 8 would be ranked higher than Architecture 3 and Architecture 6 which did not represent the survey results. We concluded that including the total hops was necessary to get the desired results. Weighing the entry points factor more, gave similar results, however, Architecture 6 was ranked much lower in this case compared to the desired results. Weighing it less, however, gave completely different and undesired results, with Architecture 4 being ranked as one of the most secure architectures instead of one of the least secure, and Architecture 8 being evaluated as one of the least secure ones. Looking at 7.1, this is exactly the opposite of what we want.

The resulting table had an eucledian distance of 6 to the survey table 7.1, which was the best result out of all the permutations, and looked as follows:

Table 6.1: Rank and Architecture

Rank	Architecture	Rating
1	10	151.48
2	3	60.58
3	8	59.85
4	6	59.8
5	2	56.52
6	5	44.69
7	1	43.46
8	4	42.21
9	9	31.38
10	7	13.02

As we can see, the table is very similar to the survey table 7.1, with the eucledian distance being 6. The architectures that received the best rating, which were Architecture 10, Architecture 3, Architecture 8, and Architecture 6 also being the ones that were ranked the highest, and the architectures that received the worst rating also being the ones that were ranked the lowest. Differences are that Architecture 10 jumped from rank 4 to rank 1, Architecture 1 and Architecture 5 switched places, and so did Architecture 9 and Architecture 7.

We were able to replicate the order of architectures 3, 8, 6. Looking at their feasibility score, it is evident that their rating was very close to each other, just like the security experts' opinions. Concentrated entry points were an advantage of Architecture 3 and 8, however the CGW being the entry point in Architecture 8 is what made it rank below Architecture 3. In contrast, Architecture 6's strong isolaiton is what made it rank high. This distribution let's us consider this rating a success.

Architecture 10 ranked first in our table, which was not the case in the survey table 7.1. This is due to the entry points being isolated, the CGW only being a target and not an entry point, and the amount of hops being high, as no interface shares an ECU with another interface. The priorities of the security experts are in fact reflected in this architecture, and strictly sticking to the criteria, Architecture 10 is the best architecture by far. The opinion was not shared by the security experts, ranking it fourth, but this opinion was also surprising to us.

Architecture 2 is correctly ranked above Architecture 1 and Architecture 5.. Though it is missing the CGW, the CGW in 1 is an entry point, which is more unfavorable, as it reduces the amount of total hops. 5 uses Ethernet only, which has a higher security rating than CAN, thus placing above 1.
The result is interesting because the architectures are the same with one key difference each, thus we are able to exactly see the priorities of the security experts as reflected with the criteria. 5 and 1 ranked close to each other, thus we

consider this result a success.

Unfortunately, we were not able to replicate the low ranking of 4. As our criteria weighs the isolation factor the most out of all the factors, it is understandable that 7 is placed last, even though it is placed third in the survey. Isolation in this architecture was intentionally removed, with every ECUs having to share the bus with every ECU using the same bus technology. It is surprising to see that Architecture 4 ranked much better than 9 and even 7. 9 and 4 are the same, except for the presence of the CGW in 4. However, this does reflect the opinion of some security experts that having a CGW is better than having none, if it is not an entry point.

To summarize, we were able to choose a criteria and according weights in such a way that the resulting table was very similar to the survey table 7.1. The criteria was constructed so that the priorities of the security experts were reflected, and the weights were chosen so that the resulting table was as close to the survey table as possible. Though Architecture 10 and 4 were the only major exceptions, we find that the results, and thus the criteria, are nevertheless valid.

7 PROOF OF CONCEPT

To prove that the criteria is properly calibrated and gives desired results, another set of architectures is evaluated.
These architectures were designed with the survey feedback in mind. In the following, they are referred to as *Proof Of Concept* architectures.
The training set architectures sometimes different immensely from each other to be able to better crystallize the criteria, whereas these now are kept relatively similar to each other to ensure the criteria is not biased towards a specific architecture.
Similarly to the test set, this chapter will describe the architectures, estimate the desired results, and evaluate the architectures using the tool. The following architectures all use the same building blocks with the same values as described in 3.

7.1 ARCHITECTURE A

Architecture A features four entry points, and six interfaces - the highest amount of all the architectures. However, the strong point here is that they are are relatively separated from the rest of the components. In addition, it features a CGW that is not an entry point.
These factors indicate a high amount of hops for some targets. In terms of isolation of ECUs, most ECUs share their bus with at least two or more other ECUs, resulting in a low isolation compared to the other architectures.
Based on the criteria, this architecture is expected to do the best out of the three architectures.

7.2 ARCHITECTURE B

Architecture B features three entry points and five interfaces. Though the number is lower that that of Architecture A, the entry points are more spread out, meaning that the maximum amount of hops is lower. The CGW, however, is not an entry point and the isolation of ECUs is better compared to 11.
This architecture, though offering better isolation, is expected to perform worse than 11 due to the lower amount of hops.

7.3 ARCHITECTURE C

Architecture C has only three entry points and three interfaces - however one of them being the CGW and in addition, the entry points are distributed across the architecture. This results in a lower amount of hops for the targets, putting this architecture at a disadvantage compared to the other two. Isolation of ECUs here is similar or somewhat better to Architecture B.

Due to the low amount of hops and the CGW being an entry point, this architecture is expected to perform similar to Architecture B, but still the worst out of the three.

7.4 RESULTS - PROOF OF CONCEPT ARCHITECTURES

The results are as follows:

Table 7.1: Rank and Architecture

Rank	Architecture	Rating
1	Architecture A	152.97
2	Architecture B	129.2
3	Architecture C	124.66

Taking the criteria into account, the results reflect the expectations. Architecture A, with the highest amount of hops, due to the secluded entry points, is ranked first. Architecture B, with a lower amount of hops, but better isolation, is ranked second. Architecture C, with the lowest amount of hops and the CGW being an entry point, is ranked third.

Architecture A was expected to perform the best due to the high amount of hops and the secluded entry points. It was predicted that Architecture B and Architecture C would perofrm similarly and score lower than Architecture A, but the difference between Architecture B and Architecture C is smaller than initially thought. However, we expected Architecture B to perform better than Architecture C, because the CGW being an entry point in Architecture C is the deciding factor for the lower score.

Overall, we are satisfied with the results, as our predictions matched the actual results. In conclusion, this Proof of Concept confirms that the criteria is properly calibrated and gives the desired results.

8 CONCLUSION

In this thesis we have explored the security of automotive network architectures.

We have done this by first researching the current state of the art in automotive network security, finding that the current state of the art is lacking in the area of security evaluation. To fill this gap, we have developed a new method for evaluating the security of automotive network architectures. We conducted a survey to find out what factors are important for security architects when evaluating an architecture, and used these factors to develop a criteria that can be used to evaluate the security of an architecture. A tool helped us to automate the evaluation of architectures, by automatically generating model-based architectures, finding their most feasibile attack path and calculating their security feasibility by applying the criteria.

Key findings showed that the most important factors for security architects are the attack path feasibility, hops between entry points and targets, isolation of ECU on a bus, and the presence and influence of a Central Gateway (CGW), as well as the amount and distribution of entry points actross a network architecture.

The results and findings of this thesis can be used to aid security architects in their evaluation and conception of future automotive network architectures by also providing a tool that automoizes the process of graph generation, attack path finding, evaluation, and comparison of architectures.

Despite providing valuable insights into the security of automotive network architectures, this thesis is not without its limitations.

The criteria is based on the opinions of a small pool of security experts. More security experts should be consulted to further validate the criteria.
In addition, the architectures in the training set as well as the training set itself used for the survey was small. A larger training set would have provided more data to work with, and would have allowed for a more accurate criteria. More accurate and architectures actually used in the automotive industry would have also been beneficial, as out architectures were kept small and simple. This was done to keep the conception and survey short and simple, but it would have been interesting to see how the criteria would have performed on more complex architectures.
Another limitation was the way the criteria was developed and calibrated. Using machine learning to build a neural network that would have learned the criteria would have been more accurate and would have allowed for more accurate criteria.

Based on the findings and limitations, future work could include the aforementioned suggestions for improvement. In addition, the work of a security architect for this process could be further automated by having different data formats for the input files, such as XML, PDF, or image files, and generation of a report on the security of the architecture.

Despite these limitations, our research has shown to be successful as the criteria, represented as a mathematical formula, was able to closely replicate the evaluaiton of the security experts.

Overall, this thesis has provided valuable understanding on how can different E/E architectures be rated based on their attack path feasibility, what architectural approach makes a network safer than others and how secure a vehicular network architecture is by using a criteria calibrated by security experts.

BIBLIOGRAPHY

[1] AUTOSAR Development Partnership. *AUTOSAR 4.2.2 Specification*. Munich, Germany: AUTOSAR Development Partnership, 2019. URL: https://www.autosar.org/standards/standard-downloads/.

[2] Jürgen Dürrwang, Florian Sommer, and Reiner Kriesten. "Automation in Automotive Security by Using Attacker Privileges". In: (2021). URL: https://hss-opus.ub.ruhr-uni-bochum.de/opus4/frontdoor/index/index/year/2021/docId/8357.

[3] European Union's Horizon 2020 research and innovation programme. *HEAVENS: HEAling Vulnerabilities to ENhance Software Security and Safety*. Brussels, Belgium: European Union's Horizon 2020 research and innovation programme, 2019. URL: https://cordis.europa.eu/project/id/866041.

[4] International Organization for Standardization. *Road vehicles – Cybersecurity engineering – Guideline and general aspects*. Geneva, Switzerland: International Organization for Standardization, 2020. URL: https://www.iso.org/standard/73547.html.

[5] International Organization for Standardization. *Road vehicles – Functional safety*. Geneva, Switzerland: International Organization for Standardization, 2018. URL: https://www.iso.org/standard/64539.html.

[6] National Institute of Standards and Technology. *Evaluating the Vulnerability of Information Technology Assets (EVITA)*. Gaithersburg, Maryland, USA: National Institute of Standards and Technology, 2003. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-4/final.

[7] National Institute of Standards and Technology. *Threat Analysis Risk Assessment (TARA)*. Gaithersburg, Maryland, USA: National Institute of Standards and Technology, 2011. URL: https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final.

[8] SAE International. *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (SAE J3061)*. Warrendale, Pennsylvania, USA: SAE International, 2018. URL: https://www.sae.org/standards/content/j3061_201801/.

[9] Florian Sommer and Reiner Kriesten. "Attack Path Generation Based on Attack and Penetration Testing Knowledge". In: (2022).

[10] Florian Sommer, Reiner Kriesten, and Frank Kargl. "Model-Based Security Testing of Vehicle Networks". In: *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2021, pp. 685–691. DOI: 10.1109/CSCI54926.2021.00179.

[11] Daniel Zelle et al. "ThreatSurf: A method for automated Threat Surface assessment in automotive cybersecurity engineering". In: *Microprocessors and Microsystems* 90 (2022), p. 104461. ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2022.104461. URL: https://www.sciencedirect.com/science/article/pii/S0141933122000321.

ACRONYMS

AUTOSAR AUTomotive Open System ARchitecture. 4

CAN Controller Area Network. 1, 5, 8, 9, 11–13

CAN FD Controller Area Network Flexible Data Rate. 8, 9, 11–13

CGW Central Gateway. 8, 17–20, 22–29

CVSS Common Vulnerability Scoring System. 6

E/E Electronic/Electrical. 1, 3, 5

ECU Electronic Control Unit. 1, 4, 7–9, 11–14, 17–20, 22–27, 29

EFSM Extended Finite State Machine. 5

EVITA E-safety vehicle intrusion protected applications. 4

FlexRay FlexRay bus. 8, 9, 11–13

GNSS Global Navigation Satellite System. 8, 10, 11, 13

HEAVENS HEAling Vulnerabilities to Enhance Software, Security, and Safety. 4

LIN Local Interconnect Network. 8, 9, 11, 13

MOST Media Oriented Systems Transport. 8, 9, 11, 13

OBD On-board diagnostics. 5

STRIDE Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. 4

TARA Threat Analysis and Risk Assessment. 1, 2, 4–6

Figure 1: Architecture 1

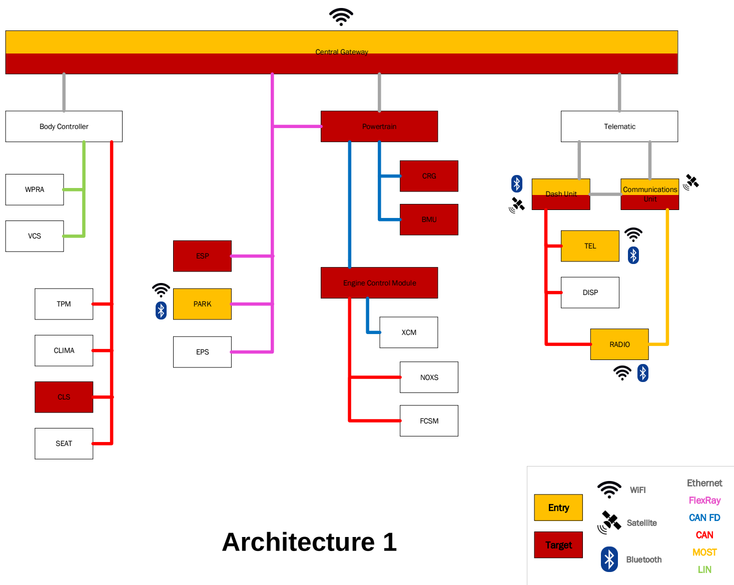

Figure 2: Architecture 2

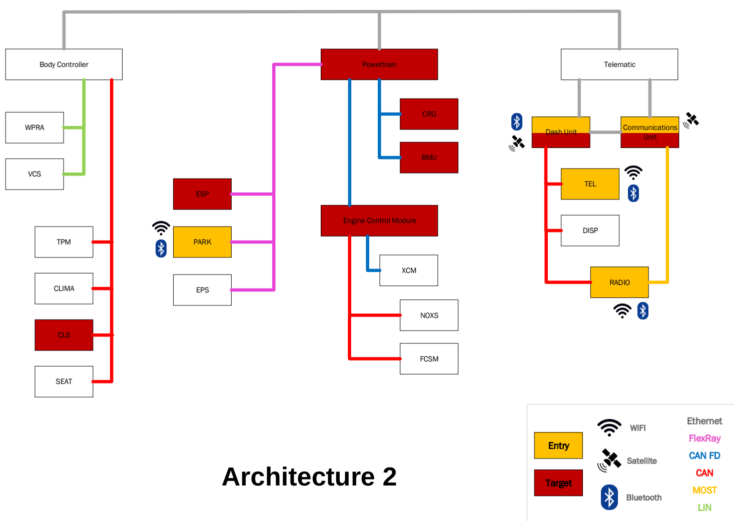

Figure 3: Architecture 3

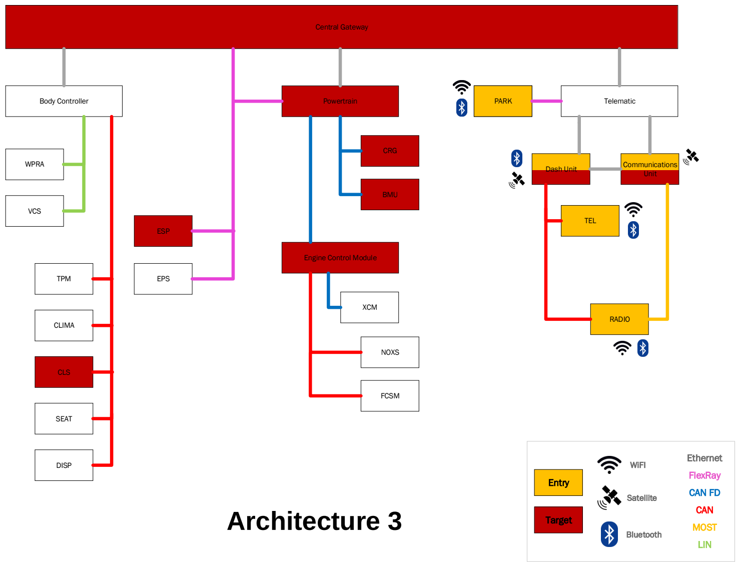

Figure 4: Architecture 4

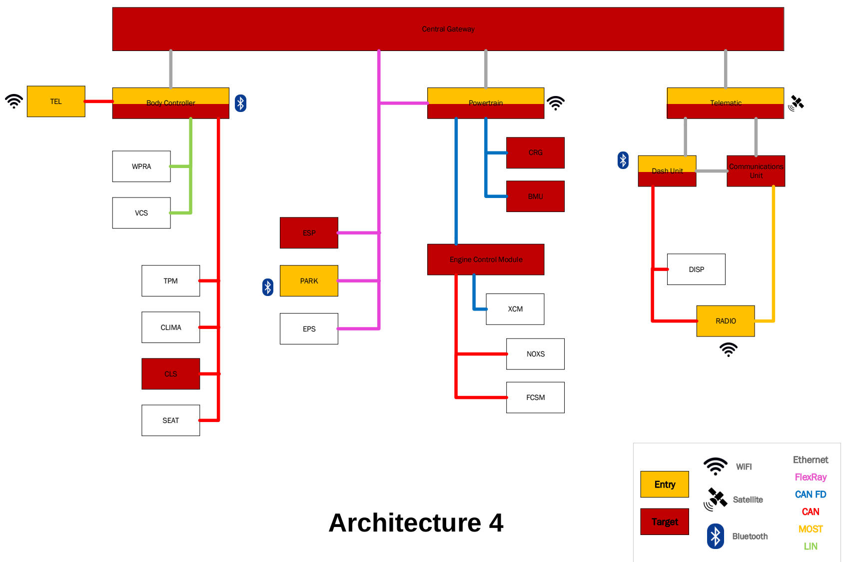

Figure 5: Architecture 5

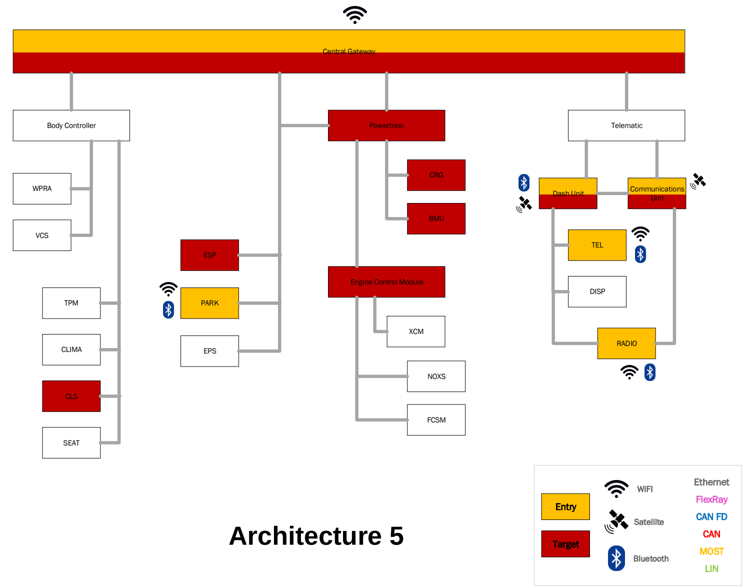

Figure 6: Architecture 6

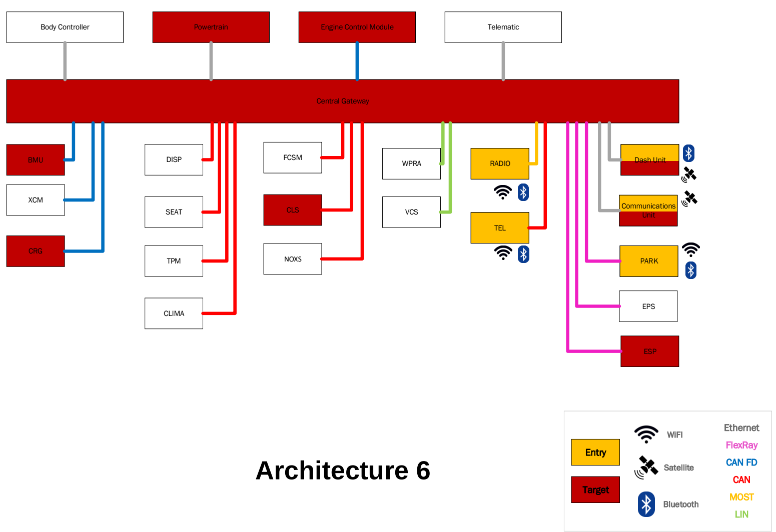

Figure 7: Architecture 7

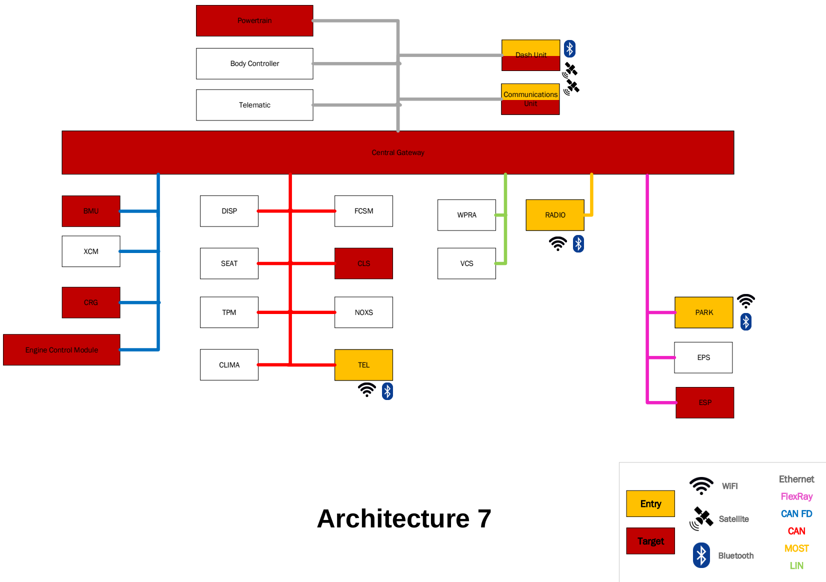

Figure 8: Architecture 8

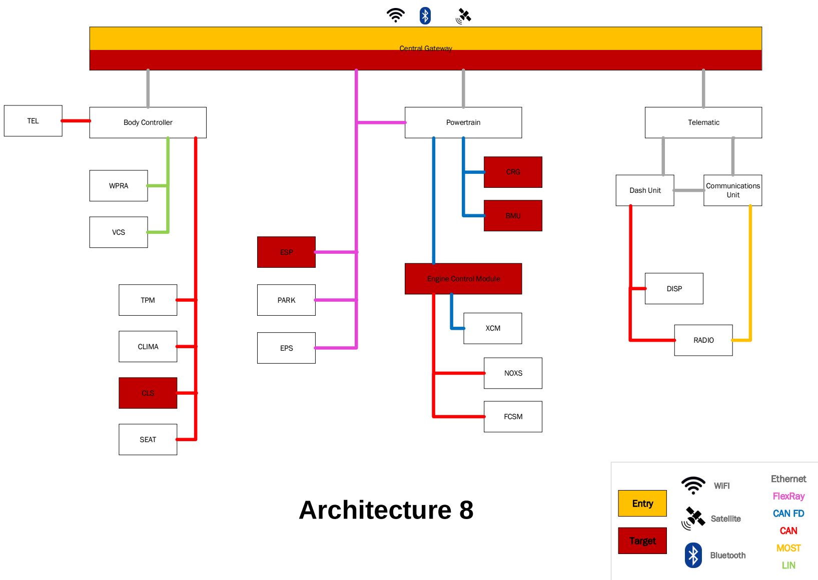

Figure 9: Architecture 9

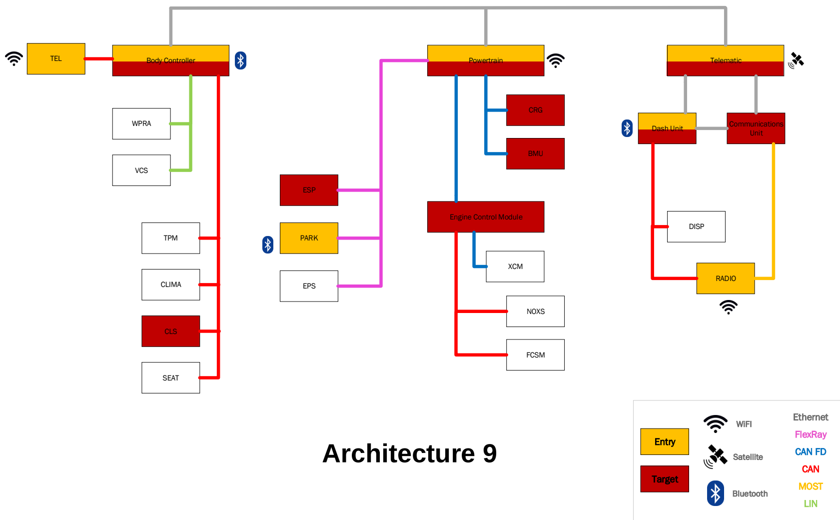

Figure 10: Architecture 10

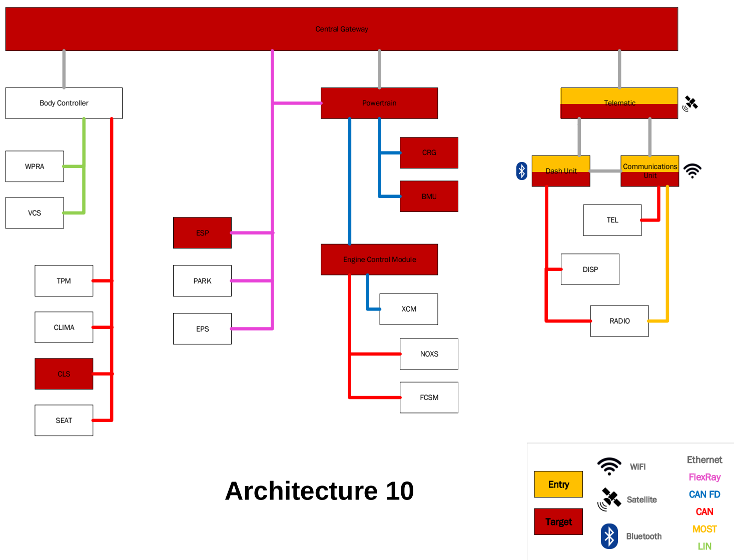

Figure 11: Architecture A

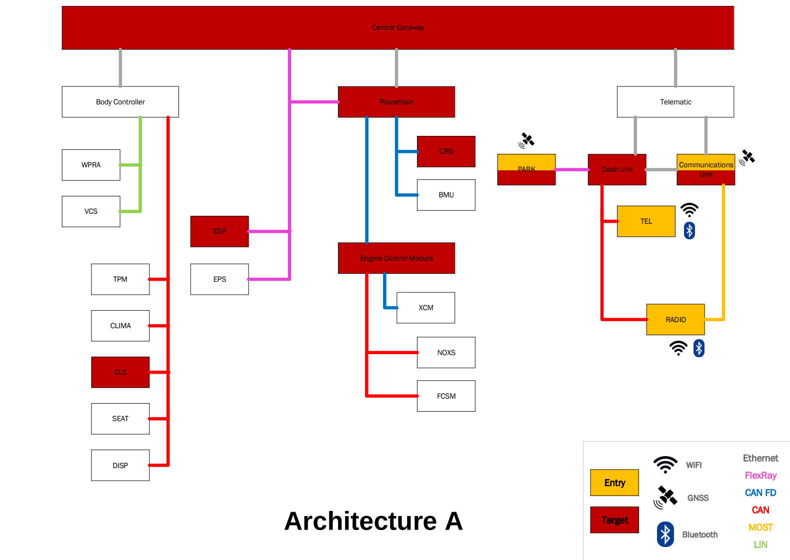

Figure 12: Architecture B

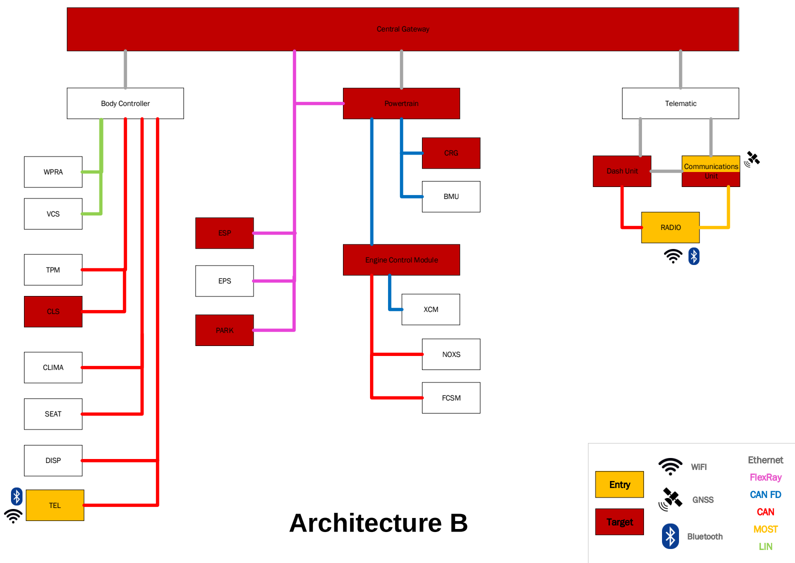

Figure 13: Architecture C