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 6, 2023

Issued: May 6, 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 6, 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. TODO: Was ich in der arbeit mache und herausgefunden habe The purpose of this thesis is to

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 electrical and electronic 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].

AUTomotive Open System ARchitecture (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 HEAling Vulnerabilities to Enhance Software, Security, and Safety (HEAVENS), and E-safety vehicle intrusion protected applications (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 HEAV-

ENS, but also considers the potential of attacks to impact the privacy of vehicle passengers, financial losses, and the operational capabilities of the vehicles systems 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 Extended Finite State Machine (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 Onboard diagnostics (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, Common Vulnerability Scoring System (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

The following chapter will show and explain the different architectures used in this thesis. The architectures were based on existing architectures used in the automotive industry as well as experimental ones based on certain attributes.

3.1 FEASIBILITY

First, we want to explain and define the term *feasibility* in the context of this thesis. Normally, a high attack feasibility in a vehicular architecture means a less secure architecture and that an attacker has less difficulty attacking the vehicle. However, due to the nature of the mathematics used in this thesis, we will conveniently define feasibility as the opposite of that, or to an extend apply it to another functionality. Imagine the term feasibility, or feasibility rating, to be the security rating of an architecture. In this thesis, the higher the feasibility score an architecture receives, the more secure it is.

3.2 COMPONENTS

While architectures used in the automotive industry are complex, with sometimes up to 150 ECUs in one model, the architectures used in this thesis were sized down to a more manageable size, with 20 ECUs for each architecture. Each architecture 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, CANFD, 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**

  Each component of the architecture (ECU, bus system, interface) has its own rating for its 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. 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. Thus, each ECU will have an own attack feasibility. For example, the Engine Control Unit (ECU) 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. Over the years, vehicles have become more and more complex, and thus the number of ECUs has increased.

- **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 base line for security in this thesis.

- **Controller Area Network Flexible Data Rate (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 bus (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.

- **Local Interconnect Network (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.

- **Media Oriented Systems Transport (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 companies 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, Bluetooth 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 uses encryption to secure the communication between devices, but 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 it has improved in terms of security over the years, is vulnerable to various attacks, including weak encryption, insecure network configurations, and attacks on the wireless network itself.

- **Global Navigation Satellite System (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
CANFD	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

Note: the higher the rating, the more secure the technology is.

4 TOOL

In order to evaluate the different complex architecture, a tool was developed. The tool's purpose is to help security architects quickly evaluate the given vehicular network architecture based on their Attack Path feasibility.
Furthermore, the tool is used in this thesis to automatically generate 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 easyto 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.

Note that the higher the feasibility rating is, the lower the likelihood that the attack will succeed.

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, CANFD, FlexRay, Ethernet, MOST, LIN*.
- **feasibility**: 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**: 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, CANFD, 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. Python is also used in the security domain, as well as the proprietary tool used by MBTI. 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 generates a directed graph based on the input architecture, ECU configuration, and buses configuration dictionaries. It then returns the generated graph along with two lists of entry and target ECUs. The function first creates empty lists for entry ECUs and target ECUs, and a directed graph `G` representing the vehicular network architecture using the *networkx* library. It then creates dictionaries for ECUs configuration and buses configuration using the *name* and *type* attributes respectively. The function then iterates through each bus in the architecture and extracts its type and feasibility from buses configuration. It also extracts the list of ECUs on the bus and for each ECU, it extracts its type and feasibility from ECUs config. Based on the ECU type, it adds the ECU to either

the entry ECUs or target ECUs list. Finally, for each ECU on the bus, it iterates through all other ECUs on the same bus and checks their types. If the target ECU has the *interface* type and at least one of the ECUs in the current bus is an entry ECU, then an edge is added to the graph with a weight equal to the bus feasibility. Otherwise, an edge is added between the two ECUs with a weight equal to the sum of the bus feasibility and the target ECU's feasibility. *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 ECUs lists.

`find_attack_path`: To evaluate the graph, the function takes the graph G, the dictionary representing the entry ECUs, and a list of strings containing the target ECUs. It then finds the distance and shortest path from each entry ECU to each target ECU in the graph using the *Bellman-Ford algorithm* and returns a dictionary table containing the distance and shortest path for each combination of entry and target ECU.
The table dictionary has three keys, *distance*, *shortest_path*, and *hops* each of which has a value of another dictionary. The *distance* dictionary maps each entry ECU to another dictionary that maps each target ECU to its distance from the entry ECU. The *shortest_path* dictionary maps each entry ECU to another dictionary that maps each target ECU to its shortest path from the entry ECU. *Hops* shows the amount of hops between the entry and target ECUs.

In the end, a funciton which applies the criteria (6) to the table is called, which returns the feasibility rating of the architecture.
The evaluation of the architecture is done in a single function and it is easy to extend the tool to support other criteria. Since the evaluation of the attack feasibility of the architecture is represented by a single number, it is easy to rank and compare the results of different architectures.

Various *I/O functions* take care of 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.

5 SURVEY AND CRITERIA CALIBRATION

One of the most important asepcts 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 used to evaluate the remaining architectures, which are the focus of the comparison. This 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.

The following section will describe the test architectures and estimate the impact the architectural choices have on the security of the system.

ARCHITECTURE 1

Architecture 1 represents the most realistic architecture, as it is modeled after an actual architecture used in the automotive industry. It offers six entry points with three being targets, one of which is the *Central Gateway*, and eight targets in total. It was important to include this architecture, since it could offer valuable insight into the real world.

ARCHITECTURE 2

Architecture 2 is essentially the same as Architecture 1, but without a central gateway, to see how the architecture would perform without it.

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.

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 *Central Gateway*.

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.

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.

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.

ARCHITECTURE 8

Architecture 8 only has one entry point, namely the *Central Gateway*. 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.

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 Central Gateway can be deduced more from such a comparison.

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 of Architecture 3.

5.2 SURVEY

Security experts at Mercedes-Benz Tech Innovation were given ten architectures as shown in 3. 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 1 got 1 points, 2 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, it's in the details where opinions will digress. 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 critical criterion 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 3 in terms of the placement of the external interfaces, but the Telematic domain controller being a target was a concern. Architecture 6 was also not expected to receive a high 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 6 and 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.

Other factors that need to be evaluated include the presence of a Central Gateway 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 8 was expected to receive a high score, as the Central Gateway is the only entry vector into the network. However, a Central Gateway with no external interfaces is overall likely more secure than an architecture with no Central Gateway 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 4.1. 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.

In general, Architectures 3, 6, 8, and 10 received a positive feedback, whereas Architectures 4, 7, and 9 received a negative feedback.

5.3 RESULTS - TRAINING SET ARCHITECTURES

The result is as follows:

Table 5.1: Rank and Architecture

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

6 CRITERIA

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 Central Gateway, 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.

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 Central Gateway also plays a role in the security of an architecture. As mentioned before, the benefit of a Central Gateway is dependent on whether it is present, an entry point, and or a target. A Central Gateway is beneficial if it is present, but not an entry point or a target. The Central Gateway factor is 1 by default, and is then decreased by 0.15 if the Central Gateway is not present, by 0.1 if the Central Gateway is an entry point, and by 0.05 if the Central Gateway is a target. This reflects the opinion of the Security Architects.
To include the influence of a Central Gateway, we define the Central Gateway factor cgw_a as follows:

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

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 $interfaces_a$:

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

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 general, 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 Central Gateway factor 6.4.

To get a reasonable rating score, the product of the feasibility and Central Gateway 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:

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

7 COMPARISON AND EVALUATION

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 Centra Gateway 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 Central Gateway, however, is not an entry point and the isolation of ECUs is better compared to A. This architecture, though offering better isolation, is expected to perform worse than A 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 Central Gateway 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 B. Due to the low amount of hops and the Central Gateway being an entry point, this architecture is expected to perform similar to 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. Architectre B, with a lower amount of hops, but better isolation, is ranked second. Architecture C, with the lowest amount of hops and the Central Gateway being an entry point, is ranked third.

As shown by the score, Architecture B and Architecture C are relatively close to each other, but the Central Gateway being an entry point in Architecture C is the deciding factor for the lower score.

Overall, this Proof of Concept confirms that the criteria is properly calibrated and gives the desired results.

8

CONCLUSION

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, 6, 8, 9

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

CVSS Common Vulnerability Scoring System. 6

E/E Electronic/Electrical. 1, 3

ECU Electronic Control Unit. 1, 4, 8

EFSM Extended Finite State Machine. 5

EVITA E-safety vehicle intrusion protected applications. 4

FlexRay FlexRay bus. 8

GNSS Global Navigation Satellite System. 10

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

LIN Local Interconnect Network. 8

MOST Media Oriented Systems Transport. 9

OBD On-board diagnostics. 5

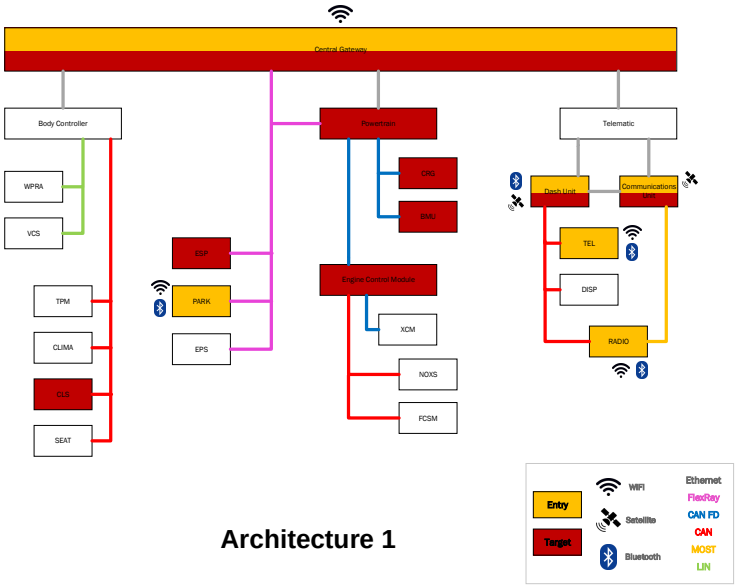
STRIDE Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. 4

TARA Threat Analysis and Risk Assessment. 1, 4

GLOSSARY

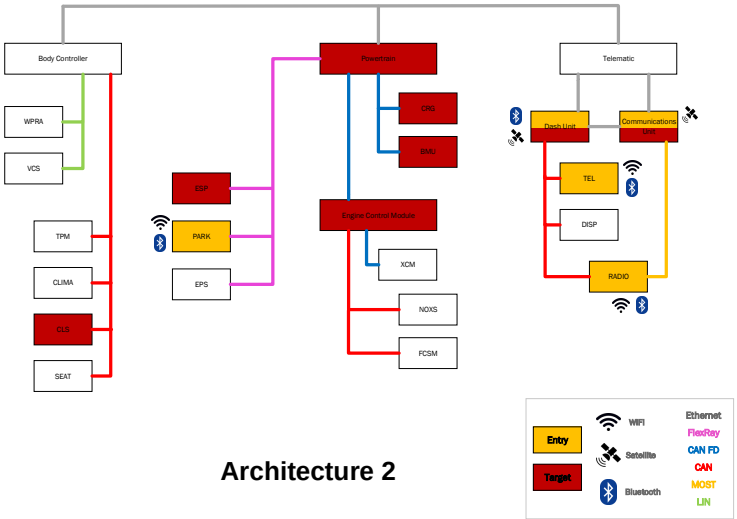
Attack Path As defined in [4], an attack path is a set of deliberate actions that an attacker takes to realize a threat scenario, a potential cause of compromise of cybersecurity properties of one or more assets in order to realize a damage scenario.. 11

Figure 1: Architecture 1



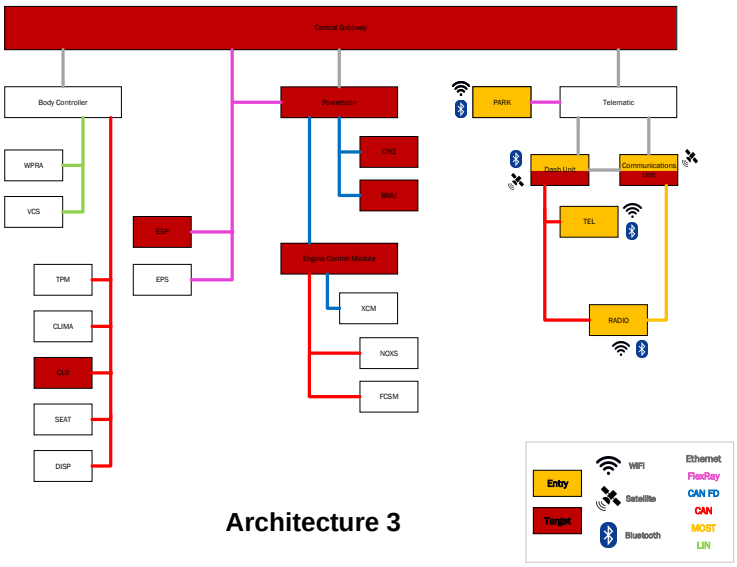
Architecture 1

Figure 2: Architecture 2



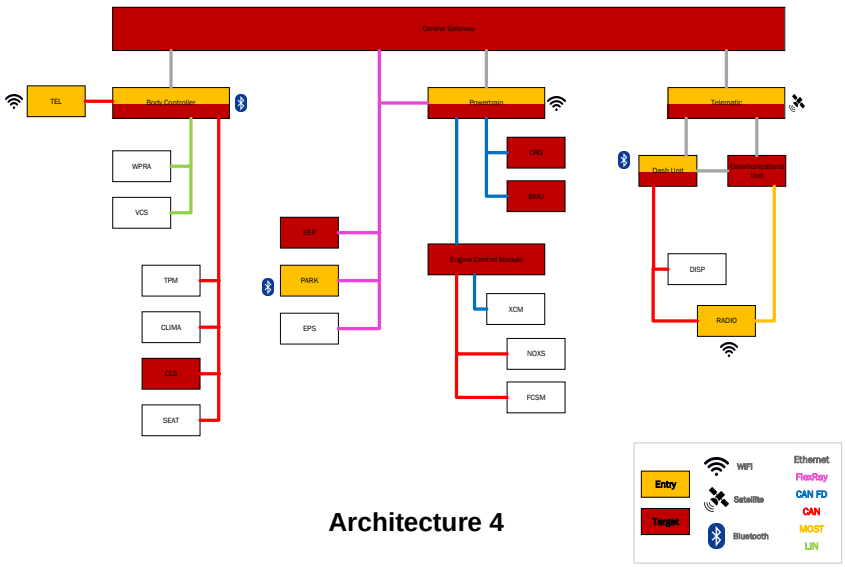
Architecture 2

Figure 3: Architecture 3



Architecture 3

Figure 4: Architecture 4



Architecture 4

Figure 5: Architecture 5

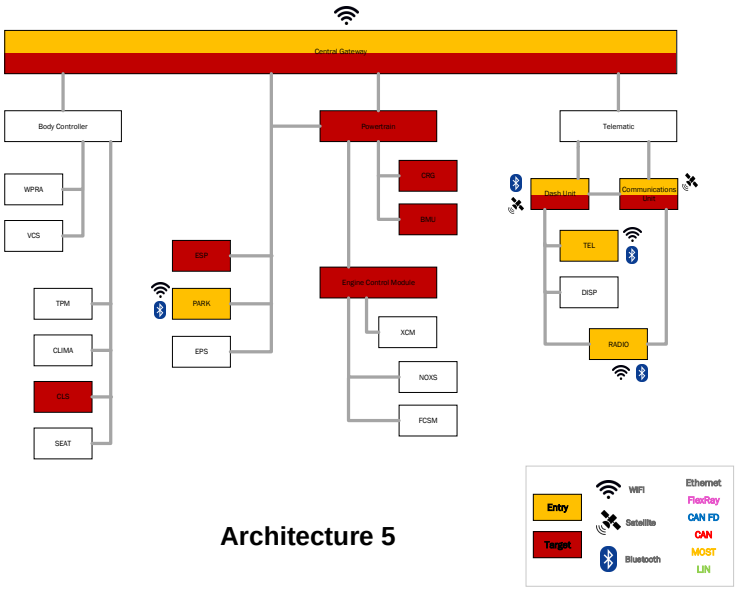


Figure 6: Architecture 6

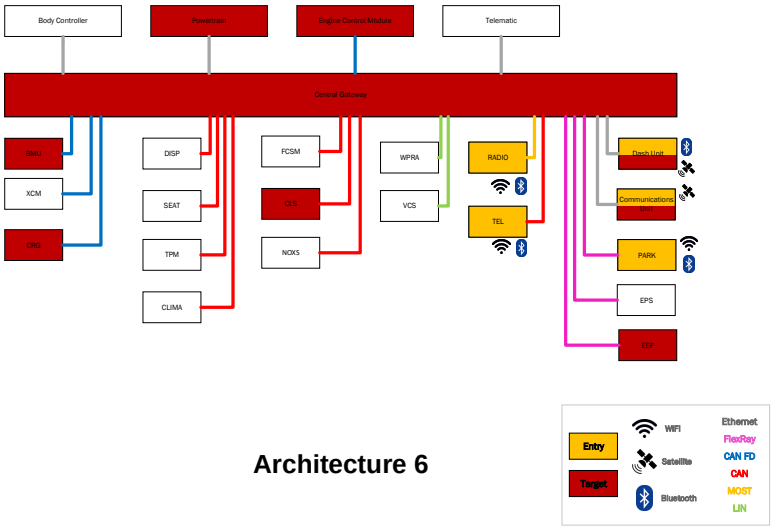
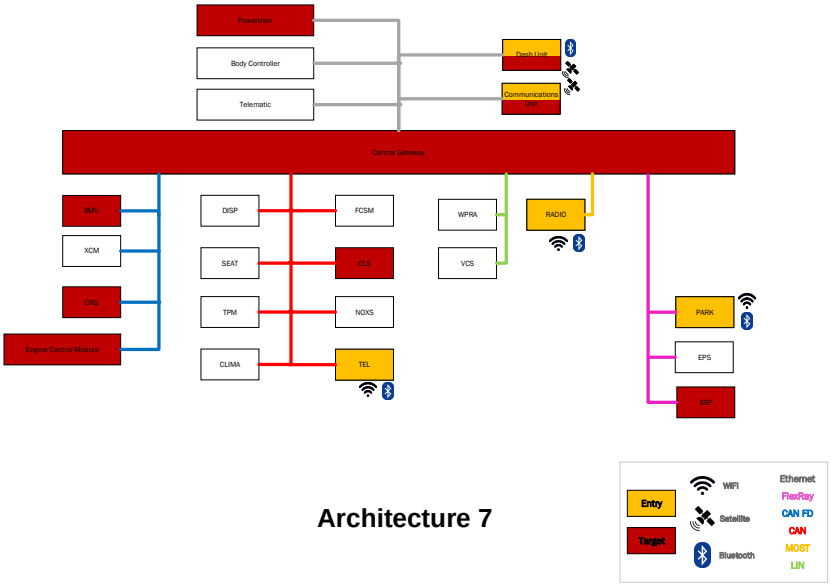
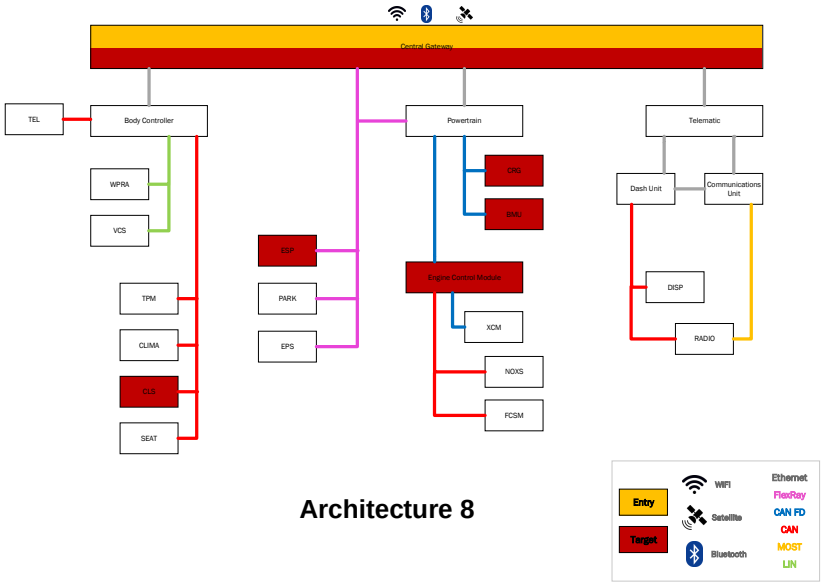


Figure 7: Architecture 7



Architecture 7

Figure 8: Architecture 8



Architecture 8

Figure 9: Architecture 9

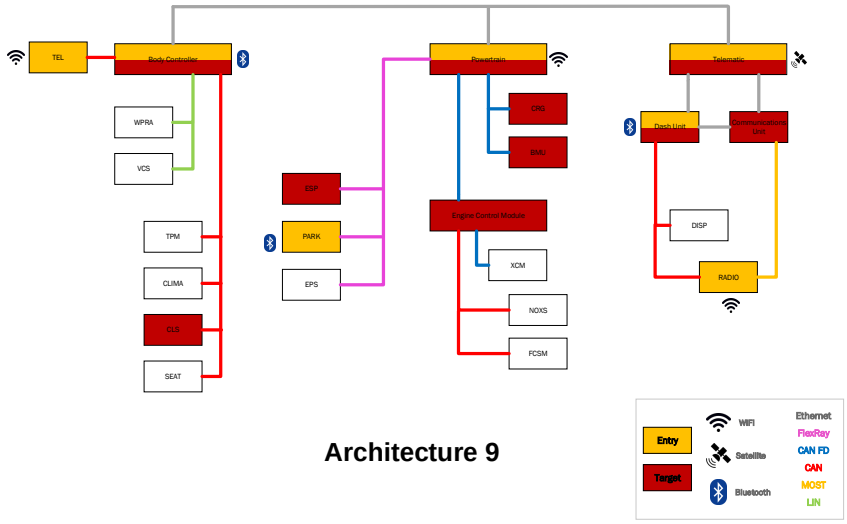


Figure 10: Architecture 10

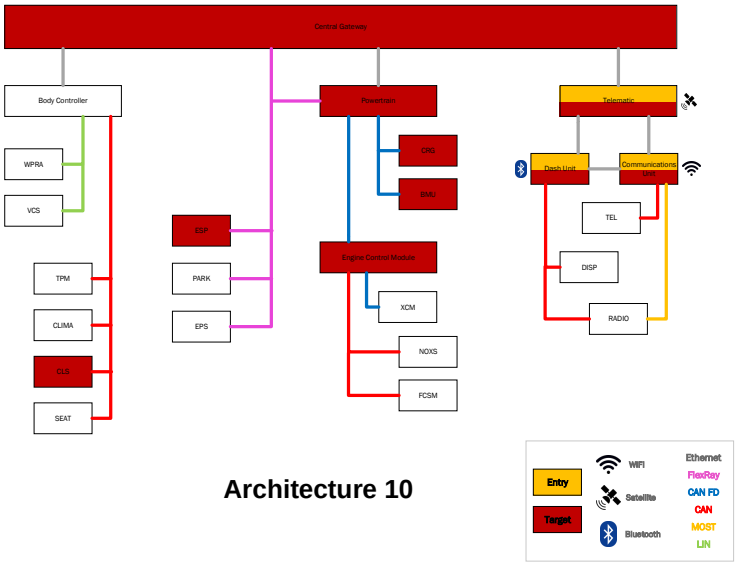


Figure 11: Architecture A

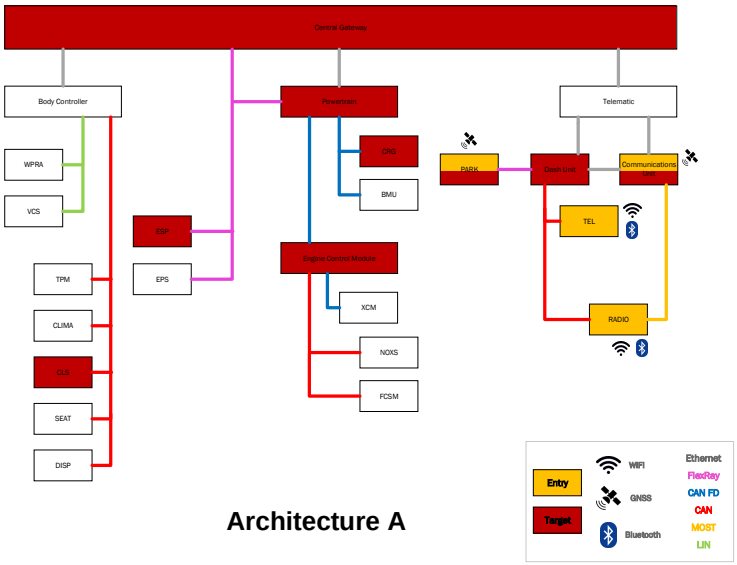


Figure 12: Architecture B

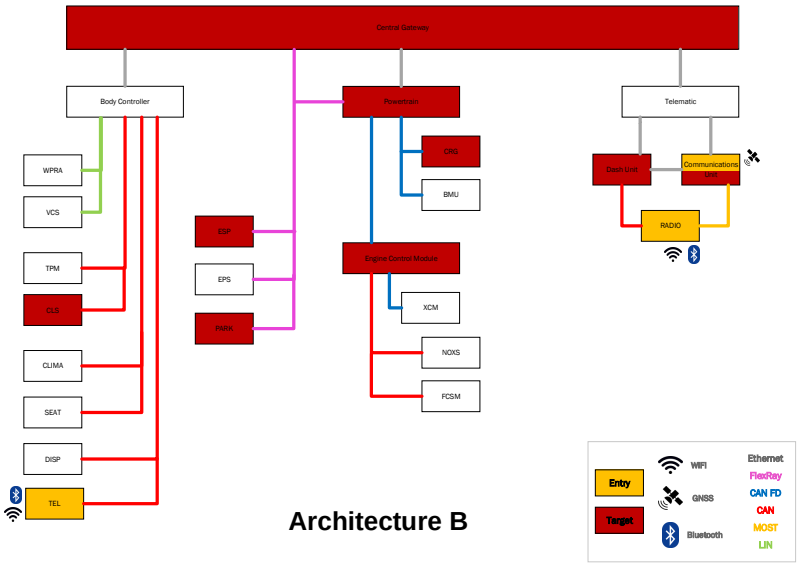
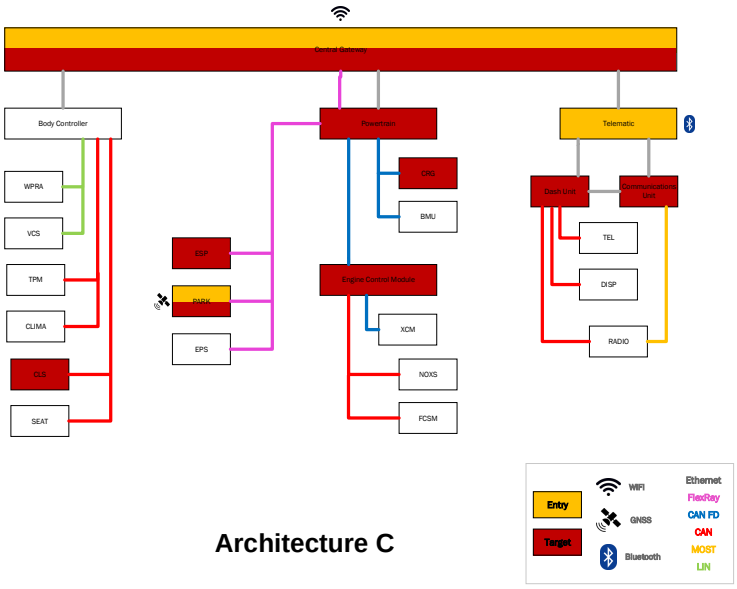


Figure 13: Architecture C



Architecture C