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

April 17, 2023

Issued: April 17, 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, April 17, 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. Cybersecurity plays an increased role in the safety and security of the vehicle and the privacy and personal data of drivers and passengers.

Most security testing, such as pentesting, is done at later stages of development, a point in time at which it is more complex and costly to address vulnerabilities. Additionally, due to the nature of security testing, it is carried out manually by experts. These factors result in a stagnant security testing process that cannot maintain pace with the increasing complexity of modern vehicles.

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

In this thesis, we propose a solution by presenting a tool that can automate the evaluation of vehicular network security based on attack path feasibility. This tool can automatically generate attack paths in a given vehicular network, find the most feasible ones and evaluate the network's overall security. The tool will be used to evaluate multiple different vehicular networks and compare their security in terms of attack path feasibility.

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 such as 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 important, the need for strong cybersecurity measures becomes increasingly important. Hackers and cybercriminals are constantly finding new ways to exploit vulnerabilities in these systems, which can have serious consequences. This includes their safety, privacy, finances, and operational viability. 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 because 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 help companies understand the specific routes or methods that a malicious actor might use to attack a vehicle's systems or networks. For example, companies can implement appropriate security measures to prevent these attacks by understanding the attack paths that might be used to gain unauthorized access to a vehicle's systems. These include encryption, authentication protocols, and firewall systems to protect against cyber threats. A well-designed internal vehicular network architecture can help minimize cyberattack risk. As the number of systems and components that are connected to the internet and each other increases, so too does the complexity of the internal vehicular network architecture. This can make it more challenging to design and implement effective cybersecurity measures, as more potential points of vulnerability 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,

Methods for security testing, like penetration testing, are 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, pentesting is 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. A Threat Analysis and Risk Assessment (TARA) is a crucial element for security assessment. Companies perform a TARA during the development process to identify and prioritize potential risks and to implement controls or

countermeasures to reduce or mitigate those risks to an acceptable level. The increased complexity of modern vehicles and the arduous nature of the state-of-the-art security testing methods make it more unfeasible for companies to conduct security assessment testing as is done now. Thus, a need for an automated tool that can help resolve this issue and couple to the TARA process is apparent.

TODO: Was ich in der arbeit mache und herausgefunden habe

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:

The thesis aims to help organizations in the automotive industry to assess the security of their vehicular networks and to provide a tool to assist in this process, thus it is important to understand the existing approaches and frameworks that are used in the industry today and adhere to them. For the reasons mentioned above, the thesis can be used in conustion with the following standards and frameworks:

ISO 26262 is an international standard for functional safety of E/E systems in vehicles that provides a framework for the development of safety-related systems, and lays out the safety requirements, safety goals, and safety measures to be considered during the whole lifecycle of a vehicle, including the design, development, production, and operation phases. ISO 26262 is divided into several parts, each of which addresses a specific aspect of functional safety. ISO 26262 is designed to help organizations manage the functional safety of their systems and ensure that the systems meet a defined level of safety and helps them to identify and mitigate potential hazards, and to demonstrate that the systems are safe for their intended use. This standard helps to ensure the safety of the vehicles by providing a systematic approach to identify and evaluate the risks associated with the safety-related systems and to implement appropriate measures to eliminate or reduce those risks to an acceptable level [5].

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, re-use, and transfer of functions within a vehicle network, and helps manage the growing complexity of technology and economics in automotive software development.

The following frameworks also, are mentioned frequently in literature and offer a base for the development of a tool to assess the security of vehicular networks:

TARA is an important process for ensuring the cybersecurity of systems and networks, especially in the automotive industry. In the context of automotive systems, 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 cyber-attacks, hacking, and software vulnerabilities. TARA can help prioritize these threats and vulnerabilities based on their potential impact on the safety, financial, operational, privacy, and aspects. Then, it can be used to determine appropriate controls or countermeasures to mitigate these risks, such as implementing security protocols, software updates, and network segmentation. TARA process can be used to ensure compliance with industry standards such as ISO 26262 or ISO 21434. Additionally, TARA can be integrated with other frameworks, such as the HEAVENS security model, which also focuses on identifying security requirements in the context of the automotive E/E architecture systems. 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. By identifying and mitigating potential risks through the TARA process, automotive systems can be made more secure and reliable, protecting the assets, and the safety of the passengers, and the data they hold [7].

An important framework is HEAling Vulnerabilities to Enhance Software, Security, and Safety (HEAVENS), which performs risk assessments of general IT systems and models explicitly built for automotive systems. The HEAVENS framework 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.

Another framework is the E-safety vehicle intrusion protected applications (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 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 in their paper "Model-Based Security Testing of Vehicle Networks" [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 useful 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 in their paper "Attack Path Generation Based on Attack and Penetration Testing Knowledge." 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. [9].

J. Dürrwang et al. further describe the concept of attacker privileges mentioned in the papers above in "Automation in Automotive Security by Using Attacker Privileges" [2]. It defines several types of privileges that an attacker may seek to gain access to a vehicle's communication system and components, 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 On-board 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.

These approaches are all closely related to each other and provide a strong foundation for this thesis, however the focus is on the EFSM model, the transitions between nodes and the attacker privileges. Those factors focus on the actual security mechanisms of the components, which is already induced with the feasibility rating of each component.

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 calculating 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 a criteria to evaluate and even compare multiple architectures.

3 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.

3.1 CONFIGURATION FILES STRUCTURE

TODO: decide on the numbers

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 feasibility rating is between 0 and 1, where 0 is most feasible, i.e. the attack is most likely to succeed, and 1 is least feasible, i.e. the attack is least likely to succeed.

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.2 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, loads the contents of these files and returns them as a tuple. 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 vehicle 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. 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 two keys, *distance* and *shortest_path*, 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.

In the end, a funciton which applies the criteria to the table is called, which returns the feasibility rating of the architecture.

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.

4 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.

While architectures used in the automotive industry are complex, with sometimes over 80 ECUs in one model, the architectures used in this thesis were sized down to a more manageable size, with ca. 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.

4.0.1 COMPONENTS

Using the scale mentioned in 3.1 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 score 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**: ECU. An ECU is an embedded system

  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, are not as critical.

- **CAN**: CAN. It is a serial bus standard for connecting electronic control units in automotive applications. It is the most common bus system used in the automotive industry. The security of the CAN bus for this thesis and the tool later on is set in the middle of all the other bus systems and will receive a 0.5 out of 1.0
- **CAN FD**: Controller Area Network Flexible Data Rate (CAN FD). It is an extension of the CAN standard that allows for higher data rates. In terms of security it will be reated higher than CAN and will receive a 0.6 out of 1.0
- **LIN**: Local Interconnect Network (LIN). It is a weaker bus system than CAN and CAN FD and will receive a 0.1 out of 1.0.
- **FlexRay**: FlexRay bus (FlexRay). It is a bus system that is used in the automotive industry for real-time communication. Due to its use for the *Powertrain*, and its advantages over CAN, it will receive a 0.9 out of 1.0.
- **MOST**: Media Oriented Systems Transport (MOST). It is a bus system that is used in the automotive industry for media. It receives a lower rating than CAN, CAN FD and FlexRay, as it is not used for critical systems, and will receive a 0.3 out of 1.0.
- **Ethernet**: Ethernet will receive a rating of 0.8 out of 1.0, as it is used for critical systems, but is not as secure as the other bus systems.
- **Bluetooth**: It receives a rating of 0.4 out of 1.0.
- **WiFi**: It receives a rating of 0.6 out of 1.0, as it is more secure than Bluetooth, but not as secure as Ethernet.
- **GNSS**: It receives a rating of 0.5 out of 1.0.

4.1 TRAINING SET ARCHITECTURES

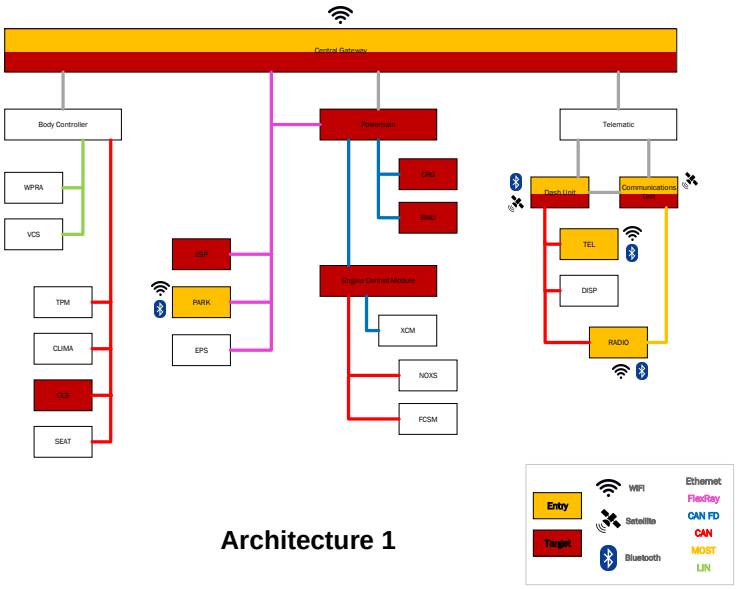
Ten architectures were used as a "training set" to determine and calibrate the criteria used to evaluate the remaining architectures, which are the focus of the comparison. This was done to avoid biasing the results, as well as have a well evaluated criteria. In addition, the training set is much larger than the main set, thus the results are more accurate.

The following section will describe the test architectures and estimate the expected behavior of the architectures.

ARCHITECTURE 1

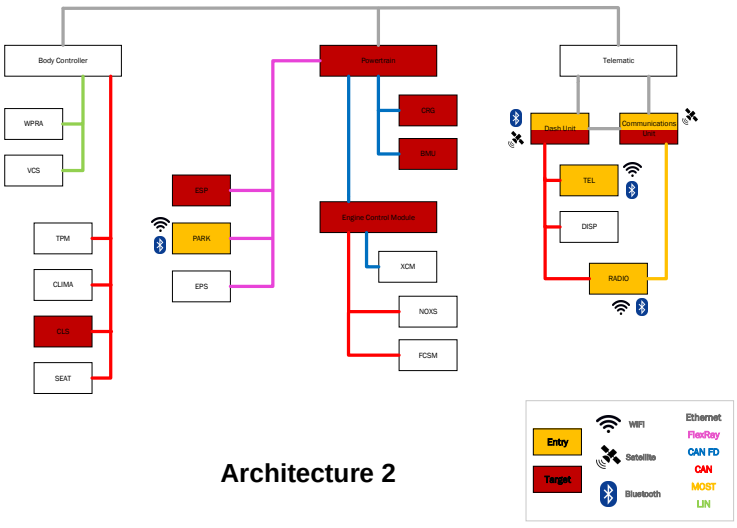
Architecure 1 represent the most realistic architecture, as it is modeled after an actual architecture used in the automotive industry. It was important to include this architecture, since it could offer valuable insight into the real world. The first architecture offers six entry points with three being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

Figure 4.1: Architecture 1



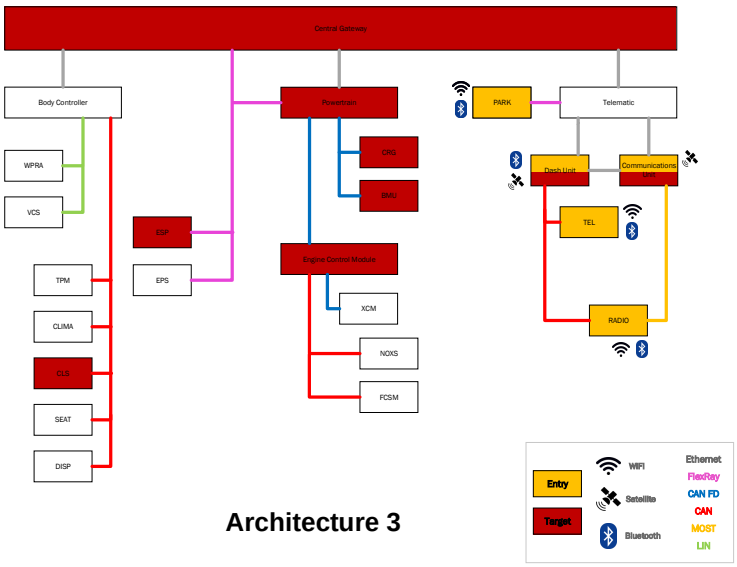
ARCHITECTURE 2

Figure 4.2: Architecture 2



Architecure 2 is essentially the same as architecture 1, but without a central gateway, so we could see how the architecture would perform without it.

Figure 4.3: Architecture 3



ARCHITECTURE 3

The idea of architecture 3 was 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.

ARCHITECTURE 4

Architecture 4 offers entry points from all domain controllers, however notice how every domain controller only has one type of interface. In addition, ECUs that rely on more than one interface would now need to use that interface from another ECU. For example, *TEL* has to use the Bluetooth interface of the *Body Controller* instead of its own Bluetooth interface. Having an entry point from each domain controller enlargens 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.

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 recieves the securest attack feasibility rating. In addition, there has been a trend in the automotive industry to increasingly use Ethernet as the bus system and maybe even some day replace most of the traditional CAN bus system.

Figure 4.4: Architecture 4

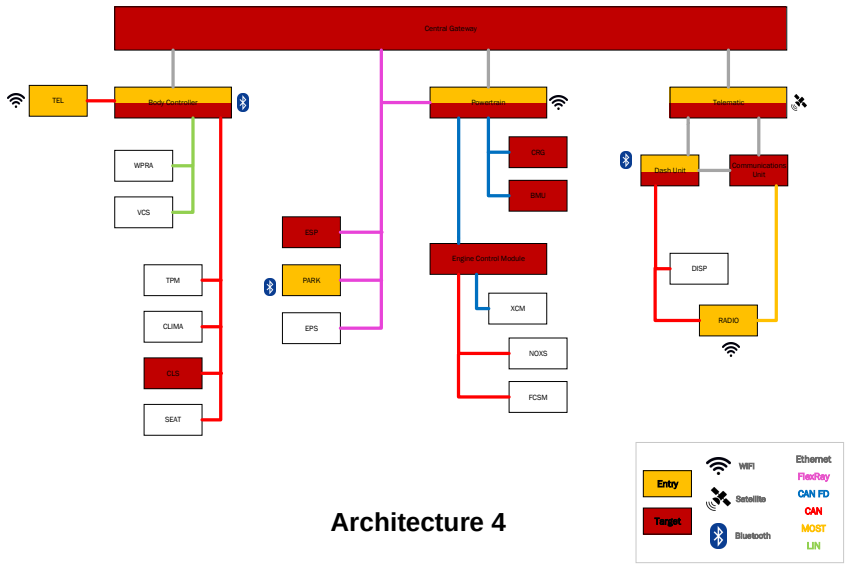
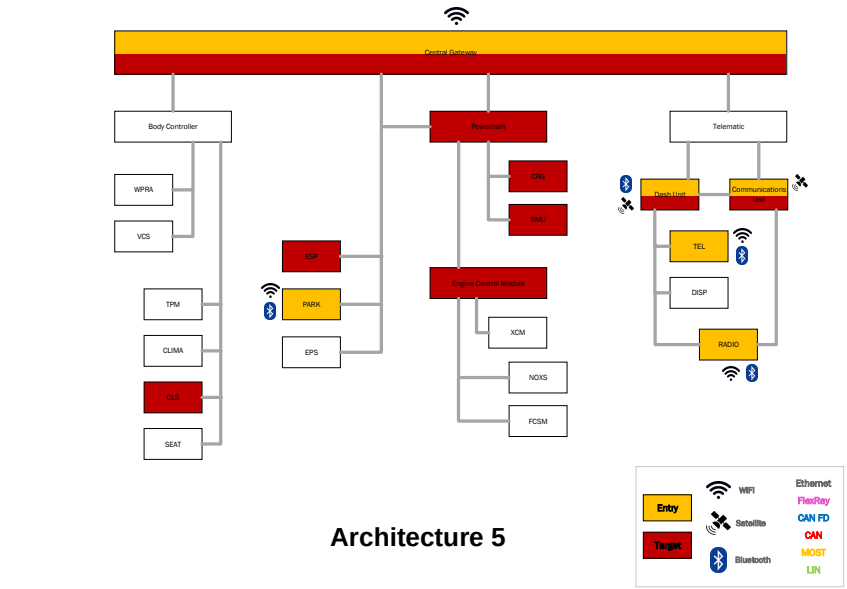


Figure 4.5: Architecture 5



ARCHITECTURE 6

Architecture 6 puts all the ECUs onto an own bus system. This is interesting, as the possible ECUs on a bus are limited to just one, thus many of the attack paths and number of ECUs used for the path are trimmed down to a few.

Figure 4.6: Architecture 6

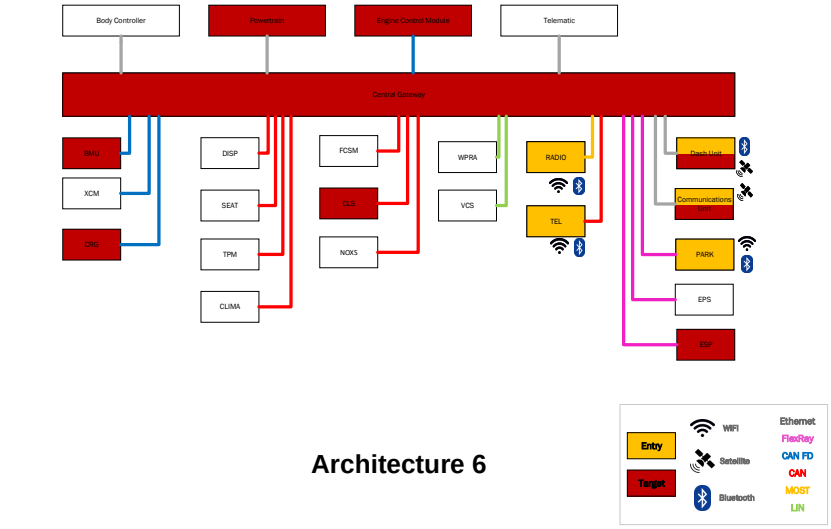
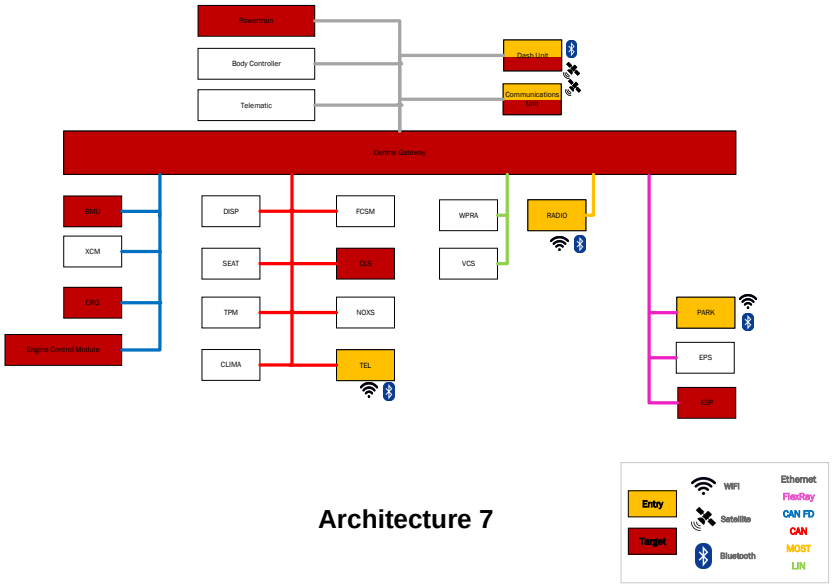


Figure 4.7: Architecture 7

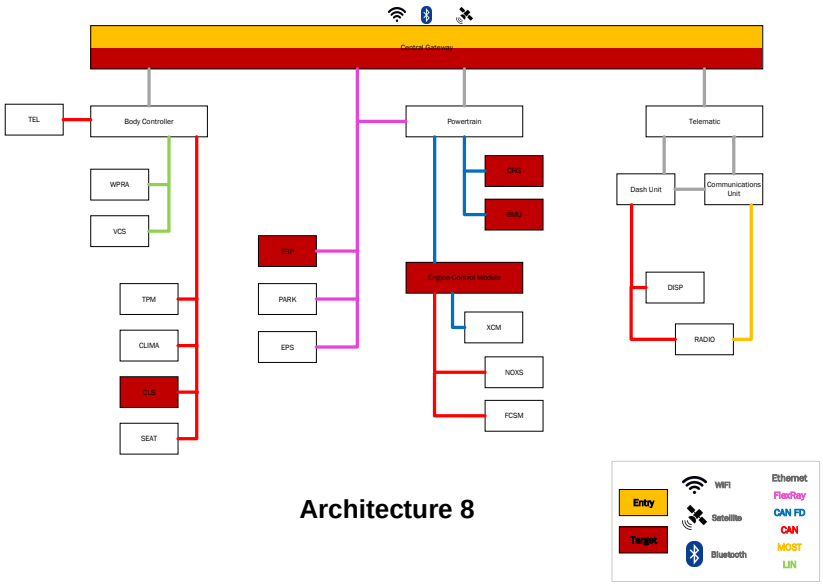


ARCHITECTURE 7

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

Figure 4.8: 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 themselves becomes a greater factor.

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

ARCHITECTURE 10

The final architecture of the test set only includes one of each interfaces but more dispersed than that of architecture 8. Note that the entries are also grouped together which shares the same idea of architecture 3.

Figure 4.9: Architecture 9

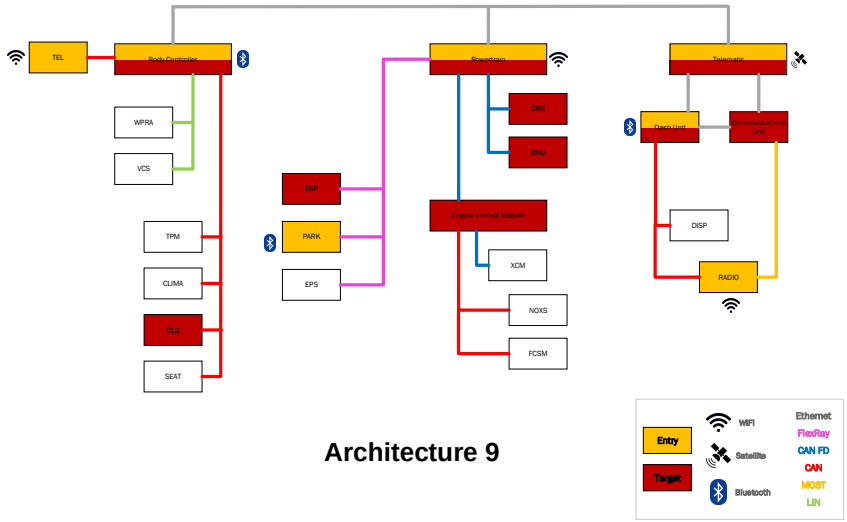
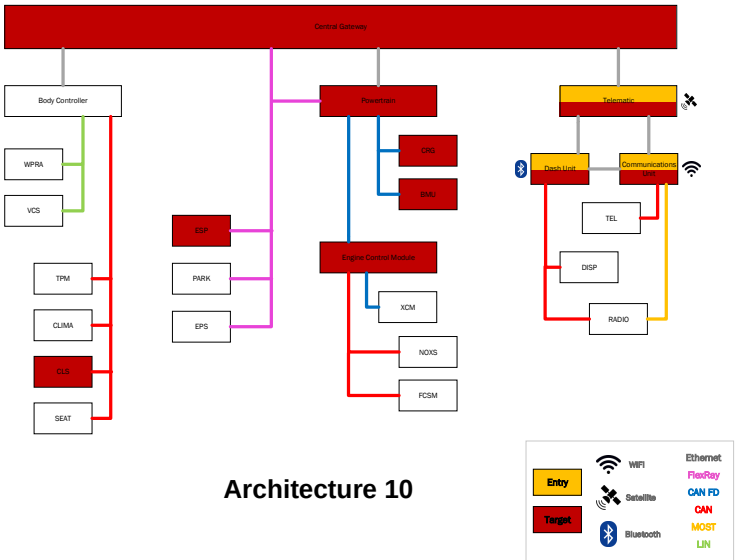


Figure 4.10: Architecture 10



5 DECIDING ON THE CRITERIA

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 SURVEY

Security experts at Mercedes-Benz Tech Innovation were given ten architectures as shown in 4. 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.

The result is as follows:

Table 5.1: Rank and Architecture

Rank	Architecture
1	3
2	6
3	8
4	2
5	10
6	7
7	9
8	1
9	4
10	5

Looking at the reason behind the ranking, the experts gave the following feedback:

They all agreed that one of the most important, if not the most important, criteria is the amount of hops there is between the entry and target. The fewer hops there are, the easier it is for an attacker to get to the target.
Another crucial factor is isolation - isolation of domains as well as the ECUs themselves, and especially an isolated Powertrain domain. It gives the opportunity to compartmentalize the system, and security mechanisms can be applied

to each domain or ECU individually. The Powertain domain is the "heart" of the vehicle because it controls the engine, transmission, and other important systems, thus it is crucial to isolate it. However, it is also not feasibile to have too much isolation, as it would make the system too complex. For example, Architectures 4.1 and 4.1 are very isolated, but they are also very complex and unfeasible. Communication between the ECUs must also be considered, as it is a crucial part of the vehicle itself. Too many ECUs communicating with each other that way can lead to a lot of overhead.
Other factors include whether a *Central Gateway* is present, which is always beneficial, as well as the amount of external interfaces. More external interfaces means more attack vectors, and thus more security mechanisms must be implemented.

One thing that was not considered in this survey was the security mechanisms implemented in the vehicle, such as firewalls or IDCs. However, it is of course important to consider them when evaluating the architectures. In this thesis, they are represented through the feasibility of each ECU, bus and interface as mentioned in 3.1.

5.2 CRITERIA

Having the result from the survey, the criteria used to evaluate the architectures was calibrated. The amount of hops between the entry and target being the most important criteria, each attack path feasibility is divided by the amount of hops it took from entry to target. This way, the more hops there are, the less feasible the attack path is.

6 COMPARISON AND EVALUATION

This chapter will compare the three main architectures based on the criteria described in 5.

7

CONCLUSION

Of course, there is room for improvement. For exapmle, the security mechanisms implemented in the vehicle, such as firewalls or IDCs, were not considered in great detail, but just as the feasibility of each component. A more fine granular analysis of each component would be beneficial. This way, you would have to consider more variables and thus get a more accurate result. However, considering them in this thesis would have made it too complex and too long, and thus it was not done.

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, 12

CAN FD Controller Area Network Flexible Data Rate. 12

CVSS Common Vulnerability Scoring System. 6

E/E Electronic/Electrical. 1, 3

ECU Electronic Control Unit. 1, 4, 11

EFSM Extended Finite State Machine. 5

EVITA E-safety vehicle intrusion protected applications. 5

FlexRay FlexRay bus. 12

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

LIN Local Interconnect Network. 12

MOST Media Oriented Systems Transport. 12

OBD On-board diagnostics. 6

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. . 1, 5, 8