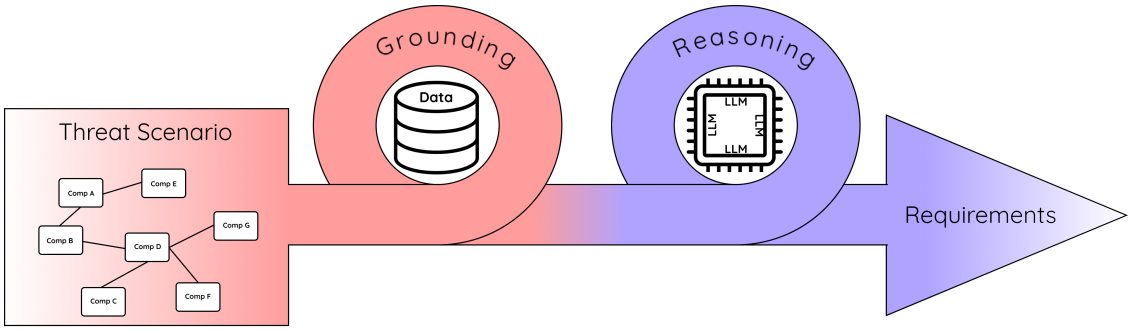
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Cybersecurity requirements identification using LLMs

A design science study

Master's thesis in Computer science and engineering

Filip Linde
Oscar Sanner

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Cybersecurity requirements identification using LLMs

A design science study

Filip Linde
Oscar Sanner

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Cybersecurity requirements identification using LLMs
A design science study

© Filip Linde 2024.
© Oscar Sanner 2024.

Supervisor: Farnaz Fotrousi, Department of Computer Science and Engineering
Examiner: Jennifer Horkoff, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Cybersecurity requirements identification using LLMs
A design science study
Filip Linde
Oscar Sanner
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Context: Threat analysis and risk assessment (TARA) is a widely used approach for conducting cybersecurity analysis in the automotive industry. The process is initiated early in the development process and continuously iterated.
Problems: Automotive systems continue to rely more on software. Additionally, the National Vulnerability Database (NVD) show that more vulnerabilities are found each year. As a result, much time has to be spent continuously ensuring that systems have updated TARA analysis.
Method: We designed a Large Language Model (LLM) based artifact to help security engineers by automatically identifying attack paths and security requirements. The artifact achieved this via a combination of prompt engineering and grounding in both the Common Vulnerabilities and Exposures (CVE) database, and the Automotive Information Sharing and Analysis Center (Automotive-ISAC) Automotive Threat Matrix (ATM).
Result: The artifact could define security requirements which met the expected standards of practitioners and were correct based on the attacks they were generated to mitigate. However, challenges were identified in the generation of attacks paths, where the generated output was less consistent in how well it met expectations. Experts perceived it to be able to generate appropriate requirements for an initial TARA analysis, however future work is needed to determine how more complex paths and requirements could be identified automatically.

Keywords: requirements engineering, threat analysis and risk assessment, large language models, automotive industry, cybersecurity, attack elicitation, RAG, prompt engineering

Acknowledgements

We would like to thank Farnaz Fotrousi for her guidance as a supervisor in this research project.

We'd also like to thank the members of Systemites research and development team, for their help and hospitality shown throughout this project.

Oscar Sanner, Filip Linde
Gothenburg, June 2024

Contents

List of Figures

List of Tables

Listings

1
Introduction

A modern vehicle is heavily reliant on sophisticated software systems that manage everything from basic engine functions to advanced driver-assistance systems. From a security perspective, each of these subsystems has interfaces connected to other subsystems or external networks which need to be analyzed and properly defended. As modern vehicles transform into more autonomous software-driven machines, they become prime targets for cyber-threats, ranging from data theft to remote vehicle control. Securing such a system is not only complicated due to the vast range of attack vectors, but also due to the constant change of the system which has to be reassessed.

Addressing these cybersecurity challenges requires a robust systematic engineering approach, central to which is the Threat Analysis and Risk Assessment (TARA) process. TARA is an industry standard for cybersecurity in vehicles specified in ISO/SAE 21434 [18]. TARA is an example of how to systematically identify, evaluate, and prioritize risks associated with cyber-threats of automotive systems. It serves as a foundational element in the elicitation of security requirements and ensures compliance with regulations.

However, the application of TARA is a resource-heavy task. One of the primary issues is the amount of manual labor related to ensuring continuous compliance with the evolving software system. This requires continuous adaptation of TARA processes, which can be both resource-intensive and time-consuming.

One key activity in the TARA analysis is the reading of external data-sources, such as vulnerability databases, regulations and documentation. A result of this type of structured textual input to individual TARA activities, opens up for the potential for automation in the process. In this context, the rise of Large Language Models (LLMs) presents a opportunity for the field of cybersecurity, especially in gathering and presenting information relevant to specific threat scenarios. LLMs, such as GPT (Generative Pre-trained Transformer), have demonstrated great capabilities in understanding and generating human-like text [10, 3, 39], potentially making them useful in cybersecurity applications. Their ability to quickly process vast amounts of unstructured data and generate insights can be leveraged to automate routine tasks associated with TARA, such as the identification of new threats, generation of threat models, and updating of risk assessments.

Moreover, by employing techniques such as Retrieval Augmented Generation (RAG), LLMs could potentially facilitate the continuous compliance efforts by providing up-to-date information on regulatory changes and emerging threats. By integrating LLM technologies into cybersecurity practices, organizations can enhance their ability to respond dynamically to threats, reduce the workload on human analysts, and improve the overall efficacy of their cybersecurity processes. This thesis aims to explore the potential of LLM technologies in eliciting cybersecurity requirements, with a specific focus on automating and optimizing the TARA process within the automotive industry. Through a comprehensive examination of current challenges and the capabilities of LLMs, this study will seek to demonstrate how these advanced technologies can complement established cybersecurity practices.

This study is in collaboration with Systemite. Systemite provides a platform, Systemweaver, for automotive companies to help them manage their development processes. Systemite is exploring the integration of AI tools into their platform to further enhance customer efficiency. A notable application area identified is security, where Systemweaver supports working with the TARA methodology. Systemite has contributed to this study by providing several experts in automotive security, related documents and introduced the authors to some of their customers who use their platform for conducting security-related tasks.

1.1 Objective

This study was in part guided by the design science research methodology outlined by R.Wieringa [59]. By utilizing the objective template 1.1 provided by Wieringa [59], our research objective may be formulated as:

Improving the TARA process by designing an LLM based program that reduces time spent on certain activities to support engineers deriving cybersecurity related requirements

- Improve <a problem context>
- by <(re)designing an artifact>
- that satisfies <some requirements>
- in order to <help stakeholders achieve some goals>.

Figure 1.1: Design problem template presented by Wieringa [59].

In order to reach the design objective a set of research questions are defined. The first question relates to problem investigation and will aid in justifying the research. The second is about designing and incorporating the solution to the TARA process while the third is about evaluation.

RQ1 : What are the primary challenges encountered by practitioners when eliciting security requirements?
RQ2 : How can an LLM based artifact be designed to assist practitioners in security requirements elicitation?
RQ3 : How do experts evaluate the designed artifact?
RQ3.1 How well does the output produced by the designed artifact meet the expected standards and goals of a TARA project?
RQ3.2 How could the designed artifact influence the existing workflow in TARA?

1.2 Structure of the report

The report will begin by presenting a set of methods and techniques in Chapter 2. These were either employed in, or are otherwise related to the research project. In Chapter 3 the report will present related research which is relevant to the project presented in this report. The research method is then presented in Chapter 4. The Chapter will align the method used in this project with established guidelines and methodologies, as well as present the data collection methods used, and the various phases of the project. Chapter 5 will explain the artifact from a technical and process perspective, as it was at the completion of the project. Findings are presented in Chapter 6. The findings are divided into three separate sections, each focusing on a single research question. Chapter 6.1 will present findings related to challenges identified in the industry. Chapter 6.2 will present findings which motivate the design of the artifact presented in Chapter 5. The last sub-chapter related to findings, Chapter 6.3, will present the final evaluation of the artifact. Finally Chapter 7 will present a discussion of the findings, as well as limitations, ethics, and threats to validity. Chapter 8 will present a brief conclusion and the potential for future work.

2
Background

This chapter will present topics relevant for the study. It begins with an overview to security requirements engineering and later discusses certain documents and technologies used in the development of the artifact.

2.1 Security Requirements Engineering (SRE)

Security requirements are requirements aimed to "Protect against abuse and disaster" [24]. Security requirements are often classified as a category of non-functional requirements by organizations such as IEEE and ISO [24].

Whilst such organizations legitimize security requirements as a type of non-functional requirements, there is less consensus in the literature on which exact frameworks and tools such requirements should be elicited from. In [24], Søren Lausen presents an abstract method of SRE where the central activities are threat mapping and risk analysis. Threat mapping is done to investigate which potential threats exist in the system, whilst risk assessment is performed to assess the potential consequences of a realized threat. If a risk is assessed to be feasible enough to mitigate, a security requirement is specified with the goal to mitigate it. This method of working can be projected on most SRE frameworks presented in the literature as well as the TARA process used by the company in the study.

In the field of SRE, various methodologies and frameworks have been proposed to systematically address security requirements. The work by Haley et al. [14] discusses a framework that integrates security requirements into the early stages of the software development lifecycle, emphasizing the importance of identifying security threats and corresponding mitigation strategies during the requirements engineering phase in the beginning of the project. Similarly, Mead et al. [33] introduces the Security Quality Requirements Engineering (SQUARE) methodology, which provides a structured process for eliciting and prioritizing security requirements through stakeholder collaboration and threat modeling.

These methodologies highlight the significance of a proactive approach to security requirements, where potential threats are anticipated and addressed before they appear as vulnerabilities in the system. Such methods align closely with TARA, which also emphasizes the identification and assessment of threats and risks to guide the development of security requirements.

2.2 Threat Analysis and Risk Assessment (TARA)

TARA is an efficient method to ensure cybersecurity defense mechanism and reduce cost by introducing security concepts early in the development process [29]. It analyzes the threat of systems and determine hierarchical defence and corresponding mitigations. TARA is a broad term, and can thus be applied in different ways [29], however this study will consider the application of TARA to cybersecurity as specified in the ISO/SAE 21434 standard, since Systemite and their customers base their implementation of TARA on these guidelines. This adaptation was explained by a practitioner during an interview in the first iteration of the project. The interviewee explained TARA to be a structured approach to identifying vulnerabilities, evaluating their impact and developing strategies to mitigate the risks identified. The TARA process is illustrated in Figure 2.1.

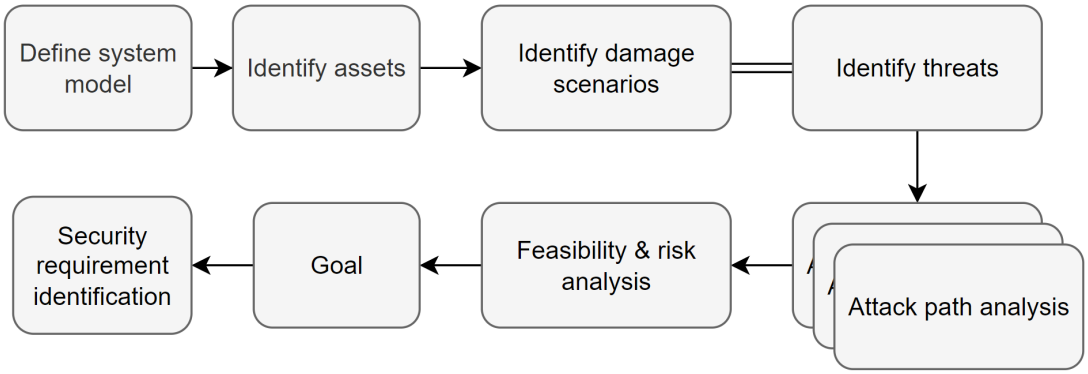


Figure 2.1: The TARA process, as explained by one practitioner in the study.

2.3 Security standards in automotive SRE

With the aim to standardize automotive security, organizations have set out to come up with standards and regulations. In this chapter, two such documents are presented, which have to be adhered to throughout the project.

2.3.1 ISO/SAE 21434:2021

According to its authors, the purpose of ISO/SAE 21434 is to provide cybersecurity guidelines for electrical and electronic (E/E) systems in the automotive industry [18]. It strives to achieve this by discussing and presenting several different tools and methods for working with cybersecurity. While the document discuss several different topics throughout the whole production chain, the ideas that are most relevant to this thesis are the concrete suggestions for performing TARA.

2.3.2 UNECE World Forum for Harmonization of Vehicle Regulations (WP.29) Regulation No. 155

Regulation 155 is a document that contains a list of security specifications that any approved vehicle manufacturer must follow to. Adherence to these specifications has to be demonstrated yearly to an Authentication Authority. To help manufactures comply with the specifications, WP.29 presents a concrete list of security threats and mitigations that the manufacturer must consider throughout the whole production cycle.

2.4 Prompt engineering

Prompt engineering is a relatively new field of research that refers to the practice of designing, refining, and implementing prompts or instructions that guide the output of large language models (LLMs) to help in various tasks [9].

2.4.1 Retrieval Augmented Generation (RAG)

An effective strategy for grounding LLMs in domain-specific knowledge is to employ Retrieval Augmented Generation (RAG). In response to domain-specific questions, the LLM can retrieve information from an external data source to ground itself with more knowledge about the specific context [25]. In this study, RAG is applied by first performing a semantic search into a vulnerability database, and then using the result to ground subsequent chat prompts. RAG addresses one common challenge associated with LLMs, namely hallucination. Hallucination occurs when the LLM gets asked something outside of its training data, potentially leading to the generation of inaccurate or fictive information [4]. Research indicates that RAG improves the accuracy and relevance in the responses generated by LLMs [46]. In comparison to alternative approaches for grounding LLMs in domain knowledge, such as fine-tuning, RAG offers a more cost-effective and straight-forward option [2]. However, the effectiveness of RAG depends on the composition of the external data source [46].

2.4.2 LLM/Prompt chaining

To improve the reliability and performance of LLMs, one of the most applied prompt technique is to break the task into subtasks [44]. In summary, this results in pipelines, where the output from one LLM-call is fed into the next one. This technique has been utilized since the first iteration of the research project, as it was discovered that the LLMs accuracy decreased for more complex prompts.

2.4.3 Chain-of-thought

Introduced in [58], chain-of-thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. In practice, this is achieved by using phrasing such as: *"Begin by considering X, continue by considering Y. Combine the two results to come up with Z"*. The goal of such prompting is to impose a though pattern on the LLM. Experiments has shown that CoT improves the commonsense reasoning of the LLM [58]. This has been relevant for solving some of the reasoning tasks in our study.

2.5 Semantic search

Semantic search is a search technique where the semantic meaning of an item is searched for, as opposed to matching words [8]. One method to this approach, supported by Neo4j, is to embed the query into a vector and query large datasets of high dimensional embeddings of texts [37]. In Neo4j, such embeddings can be used as the subject for an index, allowing fast searches. The purpose of semantic search is to improve the semantic relevance from the search, by capturing the semantic intention of the query. In the context of this research, textual descriptions of vulnerabilities were embedded to aid in the search for relevant data to ground the LLMs.

2.6 Neo4j

Neo4j is a Graph Database Management System (GDBMS)[38]. As a graph database, data is represented as nodes and edges. Nodes represent data points, and edges represent relationships between data points. Nodes and edges may have an arbitrary amount of properties. Unlike SQL databases, data is not categorized into types using constructs such as tables. Instead, data points are categorized using labels. The primary purpose of labels is to tag data, as opposed to mandate that a specific set of properties must exist in labeled data points. Similarly to SQL database, Neo4j supports indexing on specific properties to increase retrieval performance [36]. In the context of this study, Neo4j was used as a database which hosted vulnerabilities and attack techniques, as well as semantic embeddings used during semantic search. This database was queried during execution of the software artifact which was designed as part of this study.

2.7 Public vulnerability databases and matrices

There are several public cybersecurity vulnerability databases, as well as cybersecurity attack and defence matrices available. This section will briefly present and discuss such databases relevant to the thesis.

2.7.1 National Vulnerability Database (NVD)

NVD is a database hosted by the United States Government which collects and catalogues software vulnerabilities[41]. The database hosts several different lists of data. Relevant to this thesis is the Common Vulnerabilities and Exposures (CVE) program, which contains individual vulnerabilities. Other examples of lists are the common weakness enumeration (CWE) containing common vulnerability categories, as well as the Common Product Enumerator (CPE) containing a list of products and product-versions in a highly standardized format. Generally each CVE is assigned both a CPE and CWE.

2.7.2 Automotive Information Sharing and Analysis Center (Automotive-ISAC) Automotive Threat Matrix (ATM)

Automotive-ISAC is a collaboration between several automotive companies with the goal to share information. One of the resources presented by Auto-ISAC is a threat matrix, mapping out common threats to automotive vehicles, many of which are adopted from the larger MITRE attack matrix. ATM contains a list of 72 different attack techniques, each mapped to one or more attack tactic. A tactic in ATM describe a what an attacker would want to achieve, with examples such as "Privilege escalation" or "Affect vehicle function". Techniques describe how an attacker would achieve a given tactic, examples being "Process injection" to achieve "Privilege escalation" or "Modify bus message" to achieve "Affect vehicle function". Additionally, each tactic and technique has a detailed description.

2.8 ChatGPT

ChatGPT, developed by OpenAI, is a publicly accessible LLM which was released in the year 2022. It has several models available to the public, but the most common are GPT-3.5 (free and paid) and GPT-4 (paid). GPT-4 is a significantly larger model than GPT-3.5 in terms of parameters and outperforms its precursor in most benchmarks [42]. However, at the time of this thesis, utilizing GPT-4 costs more than GPT-3, approximately by a factor of 25, and each call takes more time. This makes GPT-3.5 a preferable model for handling easier tasks. Both models was utilized in the developed artifact. GPT-4 was used for the reasoning tasks, while GPT-3.5 was used for more well-defined tasks, such as filtering and summarizing. Additionally, calls to ChatGPT can be varied based on their temperature parameter. The temperature parameter controls randomness of the LLM. A low temperature yields a more deterministic output [45]. The usage of this parameter is later discussed in Chapter 5 - Artifact.

3
Related works

Recent works [1, 16, 23] have focused on the mapping of CVE vulnerabilities to attacks outlined in the MITRE ATT&CK framework. In [1], an attempt was made to employ similarity search but due to difference in vocabulary got poor accuracy. Consequently, a two-step classification was adopted, initially mapping vulnerabilities to common weaknesses. Similarly, this research project will utilize similarity search to fetch relevant CVEs and attacks, however to get better accuracy the search string will be extended with information about the scenario, inputted by the user. Another difference between this research and the ones previously mentioned [1, 16, 23] is the validation of the research.

LLMs ability in risk analysis was recently studied in another study [39], with the goal to elicit requirements through a full Hazard analysis and Risk Assessment (HARA) for a vehicle. [39] share several ideas with this project, however differences in their scope and designed solutions exist. [39] tries to mimic all steps in the HARA process. It was decided to scope the project this way due to the limited time-frame and resources available to the project. The artifact designed in [39] got mixed feedback from practitioners and it was concluded that LLMs with the prompts designed by the authors was incomplete. By not relying on other generated elements as [39] did, and instead generating requirements directly from a system model and knowledge databases this study evades threats like *propagation error*. Furthermore, the feedback from practitioners in the prior study [39] criticized the generated HARA for not including explanations for its decisions.

4
Research method

This research project was conducted as design science by following the methodology presented by Wieringa [59] and the guidelines given by Knauss [22] to conducting DSR (design science research) in the context of a master thesis. The goal of design science is to investigate and design an artifact for a certain context to contribute to a set of specified goals [59]. Wieringa decompose the design iteration into three tasks, namely *problem investigation*, *treatment design* and *treatment validation*, which may be iterated over multiple times. The result of the iterations is then transferred to the real world, applied and evaluated.

For the context of this research, the artifact is defined as an LLM based program which contributed to the TARA process as employed by Systemite. The goal of the program was to aid SRE practitioners by eliciting potential attack paths and requirements which could be used in TARA. To achieve this goal the project was devised in to three major iterations, a high-level overview can be seen in Figure 4.1. Although, the figure contains similar phases in the iterations, their emphasis differed. Knauss [22] argues in guideline 4 that the first iteration should focus more on the problem (RQ1), the second on the solution (RQ2) and the third on evaluation (RQ3). This approach is suitable for this project due to the uncertainty in the first iteration of the project. Because of this uncertainty more time has to be spent studying the problem in this iteration compared to the latter ones.
Several TARA practitioners participated in the research project. One practitioner was an external customer to Systemite, while the remaining participants were employees of Systemite. As they were mostly involved in the evaluation phase, senior practitioners with good domain knowledge were recruited to get relevant feedback. The practitioners involved can be seen in Table 4.1.

4.1 Data collection & analysis methods

Three different methods of data collection were employed throughout the project: Interviews, literature review, and document analysis. This section will describe these in detail, as well as which methods of analysis were used.

4.1.1 Interviews

This section presents the interviews conducted as part of the project.

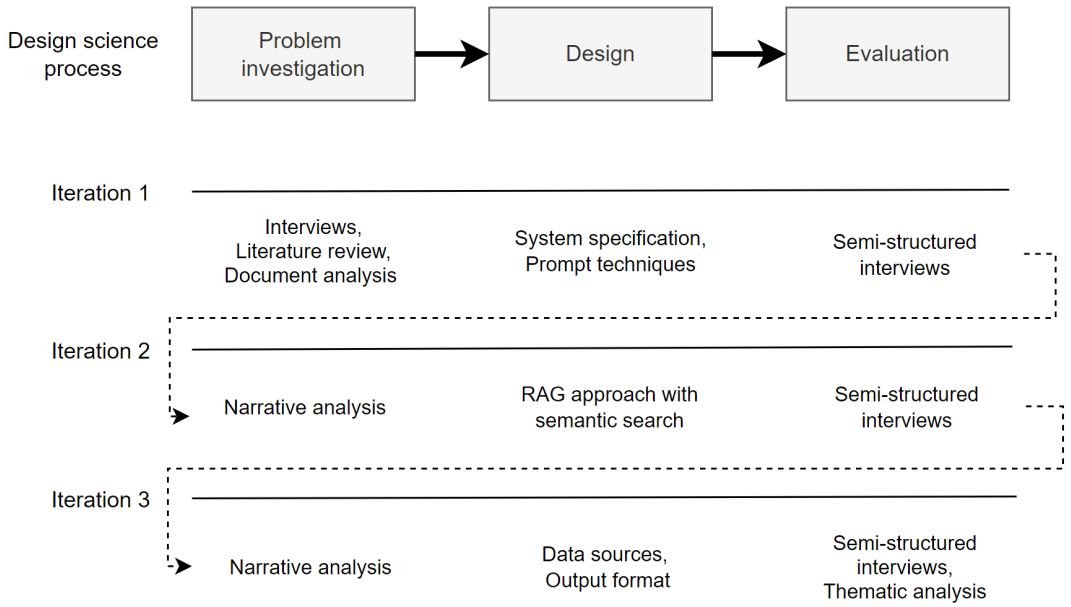


Figure 4.1: Research plan of the activities performed in this study.

As detailed in Figure 4.1, interviews were conducted during various phases of the research project. These interviews were transcribed or recorded. The interview questions varied depending on the context and goal of the interview. In total, three different semi structured interviews were designed, each with a different goal and context. One interview was designed for the initial problem investigation, one interview was designed for intermediate evaluations of iteration 1 and iteration 2. Finally, one interview was designed for the final evaluation of the artifact. The individuals interviewed throughout the project are listed in Table 4.1.

Problem investigation was conducted with all individuals listed in Table 4.1. Intermediate evaluations of the artifact conducted in iteration 1 and 2 were conducted with individual 3 and individual 2. The final evaluation was conducted with individual 3, individual 2, individual 1 and individual 5.

Table 4.1: Individuals who were involved in the research project

ID	Expertise	Background
Individual 1	Cybersecurity	Worked towards an ISO/SAE 21434 certification in 2 large organizations. Led cyber concept development in high profile vehicle project.
Individual 2	Cybersecurity and Machine learning	Experience in applying TARA in automotive industry with several years of research in machine-learning applications in cybersecurity.
Individual 3	Cybersecurity	History being the process manager for cybersecurity processes in SystemWeaver.
Individual 4	Management	Development manager.
Individual 5	Product Developer	Senior product developer with multiple years of applying the TARA process in an automotive company.

All interviews conducted shared the same initial structure. Interviewees were first

explained the purpose of the interview. This was done to provide the interviewees with a context which they could relate the questions to. After this, interviewees were explained that the data collected would be anonymized. Permission to record the meeting was then asked, if such permission was applicable. Finally, if the practitioner was interviewed for the first time, two demographic questions were asked. One to determine the what type of expertise the interviewee had, in relation to the project. One to determine how much experience the interviewee had utilizing this expertise.

To analyse the data, two different methods were employed: *narrative analysis* and *thematic analysis*. Narrative analysis was performed by investigating the transcripts and identifying a set of key narratives focusing on information relevant to the purpose of the interview. This type of analysis was employed in the problem investigation, as well as during intermediate evaluations of iteration 1 and 2.

Thematic analysis was performed during the final iteration. The implementation of the analysis inspired by the steps presented by Cruzes and Dybå [11]. To achieve this, transcripts were divided into paragraphs. From each paragraph, a set of codes were identified. Codes were defined as words or very short short phrases which could could capture the semantic meaning of a paragraph. Codes were then grouped together into minor themes, based on similarity. Finally, minor themes were grouped together to form major themes. Due to the codes and themes being subject to biases by the authors [11], analysis was done by both authors independently and cross-checked.

4.1.2 Literature review

Literature review was conducted with the purpose of further identifying challenges related to automotive TARA. Literature review was conducted through publication libraries such as IEEE Xplore [17] and Google Scholar [12]. To find relevant literature, the below search string was used:

> *Security* ***AND*** *("Threat"* ***AND*** *"Risk Assessment")* ***OR*** *"Security Requirement Engineering"* ***AND*** *("Automotive"* ***OR*** *"Vehicle")*

Literature written before 2012 was discarded, and literature was selected based on the contents of abstracts. Any abstract which did not indicate that the literature would present or investigate challenges of TARA analysis were discarded. Papers written in other languages then English were discarded.

4.1.3 Document analysis

From the discussions in the investigative interviews some documents relevant for performing TARA were retrieved and analyzed:

Table 4.2: Relevant documents mentioned by practitioners

Name	Category	Description
ISO/SAE 21434 [18]	Standard	ISO/SAE 21434 is a standard with the focus on cybersecurity in electronic components of road vehicles. The goal of the standard is to provide a systematic approach to cybersecurity across the automotive industry.
UNECE WP.29, Regulation No. 155 [56]	Standard	WP.29 is a regulatory body under the United Nations Economic Commission for Europe(UNECE). Regulation No. 155 is a regulation issued by WP.29, mandating vehicle manufacturers to work systematically with cybersecurity.
Headlamp TARA	Historical data	The headlamp TARA example is a historical artifact produced by Systemite AB. The artifact includes documentation from all activities performed in a single isolated TARA process, as well as the security requirements elicited from the process.
Auto-ISAC Automotive Threat Matrix [6]	Public vulnerability database	Auto-ISACs threat matrix is a public vulnarability matrix specifically targeting the automotive industry. Many of the vulnerabilities found in the matrix are vulnerabilities found in the MITRE Att&ck database, adapted to the automotive industry.
National Vulnerability Database (NVD) [41]	Public vulnerability database	NVD is a repository of public security vulnerabilities hosted by the U.S. Government. The vulnerabilities found in this repository targets specific products, including products used in the automotive industry.
UNECE WP.29, Regulation No. 155 (Annex 5) [56]	Public vulnerability database	See "UNECE WP.29, Regulation No. 155". Annex 5 contains several public examples of cyber attack methods which an attack could utilize in the context of automotive vehicles.

4.2 Design science phases

This section will explain how the design science phases. They are discussed in relation to their purpose, their associated data collection and analysis methods, and the activities performed in each phase.

4.2.1 Problem investigation

According to Wieringa [59], the goal of problem investigation is to gain knowledge about the problem by identifying, describing and explaining the problem to be treated. This task was mostly emphasised during the first iteration of the project. Data was collected through interviews, literature review and document analysis. Figure 4.1 further describes that problem investigation was in part carried out through narrative analysis of intermediate treatment validation. This analysis mainly focused on the identification of problems with the artifact itself and is described in more detail in Chapter 4.2.3 - Treatment Validation.

A total of 3 initial interviews were conducted with practitioners. The goal of the interviews was to identify challenges associated with current TARA procedures. Interviews followed the common interview structure presented in Chapter 4.1.1. The questions which were asked in the interviews are described in Table A.1. To understand and describe problems, narrative analysis was performed as described in Chapter 4.1.1. Transcripts were analyzed to identify narratives that discussed challenges within the TARA process, and to determine the characteristics of a potential solution.

To understand which constraints and regulations a potential solution must adhere to, several documents were analysed. The full list of documents is listed in Table 4.2. Standards were explored with the purpose of understanding which constraints the process must adhere to, and how these constraints should be incorporated into the process. The headlamp TARA example, which is part of the ISO/SAE 21434 standard was investigated to get a better understanding of what a completed TARA could look like. Finally, public vulnerability databases, used by the practitioners in their work, were investigated to understand how real world threats and mitigation's are presented and used today.

4.2.2 Treatment design

The treatment was defined as a software program which could generate potential automotive cybersecurity attacks and requirements.

The LLM used during the design of the software artifact was ChatGPT 3.5 and 4.0. These models were selected as they provide an easy to use API. The models were selected as they encapsulate current day capabilities of LLMs, and other models were not tested since this was not part of the scope in the thesis. Since ChatGPT relies on a publicly available API, permission was asked before any data was sent to it.

Treatment design was conducted based on the preceding problem investigation. To guide the design, the system was specified with initial requirements elicited from the initial problem investigation. Following the intermediate evaluations of the artifact, additional requirements were specified to address to the issues identified.

To implement the design, three types of activities were carried out: planning sessions, workshops, and internal verification meetings. During the planning sessions, actions and requirements were discussed, and broken down into smaller sub-tasks which could be individually implemented and internally verified. Workshops were then held where sub-tasks were implemented. Finally, an internal verification session was held to verify that each sub-task had been implemented according to its specification.

4.2.3 Treatment validation

According to Wieringa [59] the goal of validation research is to develop a design theory of an artifact in a context which allows for prediction into what would happen if the artifact was transferred to its problem context. In this research project, the selected validation method to be used is expert opinion. Expert opinion is one of the simplest ways to validate an artifact [59], as opposed to other options like applying prototypes to real-world problems to assist stakeholders. It is the chosen validation method because it is time efficient, which allows more time to improve the artifact, and because it promotes a natural way to involve the stakeholders in the project.

Treatment validation was performed iteratively, as illustrated in Figure 4.1. For

each treatment validation phase, interviews were used to collect data. As implied in Chapter 4.1.1, all interviews were initialized in the same way.

To evaluate iteration 1 and iteration 2 a total of 4 interviews were conducted, 2 in each iteration, with individual 1 and individual 2. The participants were shown the input and the output of the artifact by applying it on the headlamp example from the ISO/SAE 21434 [18] standard. Interviewees were then asked the questions listen in Table A.1. The questions were deliberately constructed to be open-ended to encourage a broader discussion.

The data collected from the interviews was then analysed using the narrative analysis method described in Chapter 4.1.1. The goal of the analysis was primarily to identify which potential problems existed in the iteration which was being evaluated. This was achieved by extracting narratives which outlined key issues in the artifact.

During the final evaluation performed in iteration 3, the focus and goal of the evaluation shifted to answering **RQ3**. To evaluate the artifact, 4 interviews were conducted with individual 1, individual 2, individual 3, and individual 5 seen in Table 4.1. For each interview, 60 minutes were scheduled. Before interviews were conducted, the artifact was applied to a TARA analysis scenario. This TARA analysis scenario was different from the one used during the design of the artifact. Each interview began with a brief presentation of the system under analysis, onto which the artifact had been applied. Interviewees were then shown the output of the artifact, which contained a list of 10 identified attacks and corresponding requirements. Finally, questions listed in appendix A.2 were asked to the interviewees.

Questions 1-6 were defined to determine the perceived quality of individual requirements and attack paths, as they relate to the expected standards in the industry, such that **RQ3.1** could be answered. Questions 7-9 were deliberately more open ended, with the objective of collecting data which could be used to answer **RQ3.2**. The answers from the final evaluation were analysed using thematic analysis as specified in Chapter 4.1.1.

5

Artifact

This section of the report will describe the artifact designed in the project. The section will begin by presenting an overview of the artifact, describing inputs, outputs and execution. Following this, the artifact is described in two different levels of abstractions, here referred to as *views*. This style of presentation is chosen to provide a comprehensive view of the software artifact, presenting it both as an individual executable software program and within its intended context. The *Process view* will present how the executable program is implemented into the TARA process. The *Technical view* will present the software artifact, it's individual modules and how they communicate with each other. The section will present the artifact without motivations for design decisions. Motivations are presented in Chapter 6.2.

5.1 Overview

The artifact is implemented as a python application, which is executed via the terminal. Input to the program is provided as a CSV file, which has to be placed in a specific directory for the program to execute. Finally, output is generated into a new CSV which the program places in the same folder as the input.

Input

The input to the program was designed to be an automotive system model. System models are defined during the first step of the TARA analysis, as illustrated by Figure 2.1. System models describe the different components of the TARA system under analysis, and how they are connected with each other. An example of a system model is visualised in Figure 5.1.

The stick figures in the model denotes potential points of entry for an attacker to get into the system.

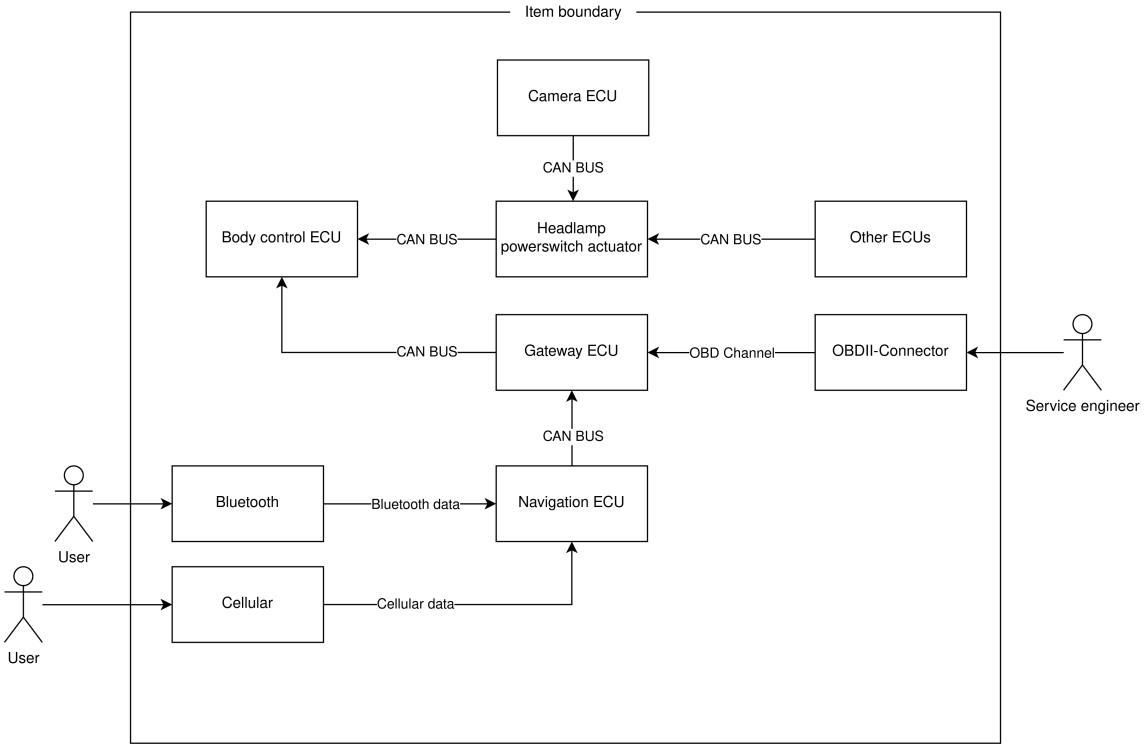


Figure 5.1: The system model which was used as input during the design of the artifact.

To use a system model as input for the program, it needs to be formatted as a CSV file. In this file, each system model edge is represented on a separate line, specifying the source node and the destination node. Listing 5.1 displays the first four lines of such an input file, which correspond to the system model shown in Figure 5.1. The "RootPath" property on each line indicates whether the edge represents a possible entry point into the system.

Listing 5.1: Example of an input to the artifact. This example shows the first three edges of a longer CSV file.

```
head -n 4 input.csv

Source,          Channel,          Destination,     RootPath
Bluetooth,       Bluetooth data,   Navigation ECU,  True
Cellular,        Cellular data,    Navigation ECU,  True
Navigation ECU,  CAN BUS,          Gateway ECU,     False
```

Execution

Execution of the artifact is done by running the `main.py` code using the python interpreter. This is done with the `inputs.csv` file present in the same folder as the `main.py` file.

Output

The output of the program is placed in a csv file. In the implementation of the artifact, the file was denoted as `output.csv`. The file contains attacks and security requirements which the artifact has identified. Each line in the output file `output.csv` can be mapped to an edge in the input file `input.csv`. The header of an output file can be seen in Listing 5.2. A full example of a single output from the artifact, corresponding to a single edge in an input file, can be seen in Figure A.1.

Listing 5.2: Example of an output from the artifact. This example only shows the header of the output CSV file.

```
head -n 1 output.csv

Source, Destination, Channel, Attack, Mitigation
```

5.2 Process view

The software artifact is intended for use within the standard TARA process performed by engineers, as detailed in Chapter 2.2 and illustrated in Figure 2.1. To integrate this artifact, engineers may modify the normal TARA process as described below:

1. The engineer performs all of the TARA activities preceding the attack path analysis.
2. The engineer converts or exports the system model as an input CSV file, as described in Chapter 5.1.
3. The CSV file is placed in the root folder of the software artifact, and the program is executed.
4. Once the execution has finished, the engineer has access to generated attacks and requirements, and may use these in the continuation of the TARA process.

The normal TARA process is illustrated in Figure 2.1. Figure 5.2 shows the same TARA process, updated with activities automated by the program. The activities illustrated in gray boxes are performed manually, while the activities illustrated in green boxes are performed automatically by the program.

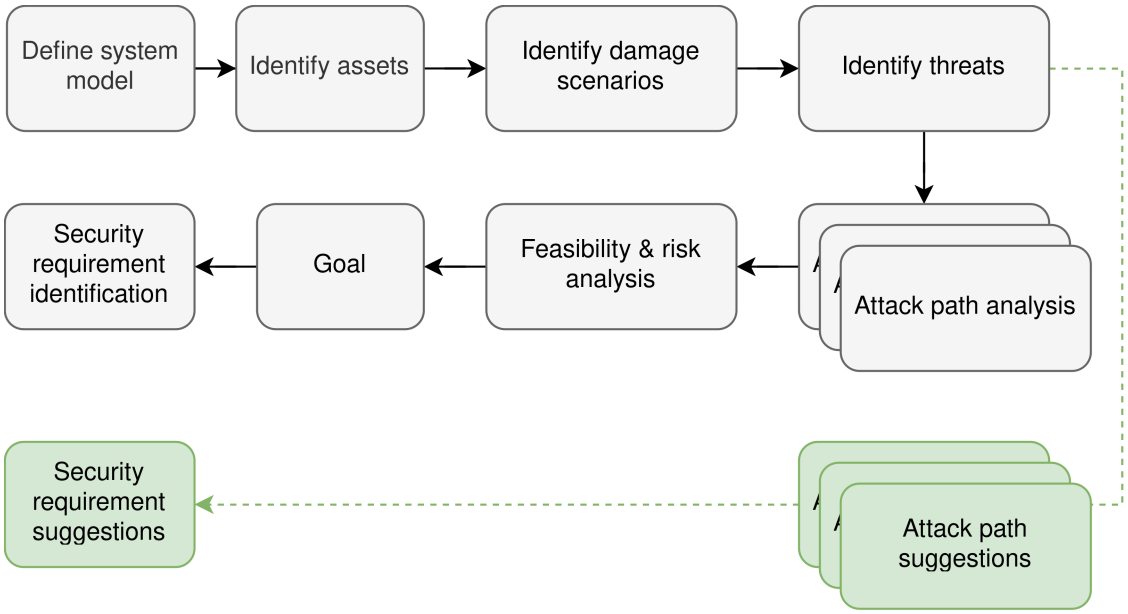


Figure 5.2: An overview of the updated process, with the software artifact implemented. Gray boxes represent the regular TARA activities performed manually. Green boxes represent automated activities.

5.3 Technical view

The technical implementation of the software artifact, in this chapter referred to as the program, was done in python. In summary, the goal of the application is to elicit security requirements by coming up with potential cyberattacks on various paths in the system under analysis. This is achieved by iterating over the edges in the system model provided as input. For each edge, a RAG approach is applied where the program finds relevant CVE's from the NVD database, as well as grounds itself in the Automotive-ISACs ATM matrix. An excerpt of the tools used to achieve this can be seen in Table 5.1.

Table 5.1: Tools used in the program.

Tool	Version	Type	Purpose
Python	3.10.x	Language	Main programming language used.
Neo4j	5.19.0	Database	Store data accessible by the program during execution.
ChatGPT	3.5-turbo	LLM	Used in some activities performed by the program.
ChatGPT	4.0	LLM	Used in some activities performed by the program
text-embedding-ada-002	1.0	Semantic Embedding model	Used to embed text into semantic encodings to perform similarity search

The specific LLM used during the execution of the program depends on the task which the LLM has to perform. Generally, tasks which require a higher degree of

reasoning use ChatGPT 4.0, and tasks used to summarize or filter data use ChatGPT 3.5-turbo. This is further detailed in Chapter 5.3.2. ChatGPT 4.0 is not used for all LLM tasks due to the longer round trip times a single call to ChatGPT 4.0 takes, compared to ChatGPT 3.5-turbo.

This section will now explain detailed technical aspects of the program. In the next chapter, Chapter 5.3.1, pre-processing and other prerequisites for the program are explained. In Chapter 5.3.2, the execution of the program is explained in detail.

5.3.1 Preprocessing

For the program to execute, it requires a connection to a Neo4j database. The database must be filled with the correct data from the NVD and Automotive ISAC databases. Additionally, all CVE descriptions has to be encoded into semantic arrays. An overview of the data which was present in the Neo4j database as the project was finalized, can be seen in Table 5.2. Finally, an index has to be in place in order for semantic search to be performed efficiently. Methods of importing and indexing data is explained in detail below.

Table 5.2: An overview of the data present in the database, used to ground the program during execution.

Neo4j Label	Count	Properties
CVE	241252	[name, description, embedding]
CVE-Reference	523909	[name, url]
ATM-Technique	72	[title, description]
ATM-Tactic	13	[title]

CVE import

CVEs are imported into the Neo4j database with the properties displayed in Table 5.3. The CVEs imported into the database during the design of the artifact were all CVEs released between the year 2002 and 2023. The CVEs were imported using the json datafeed hosted by NVD [40].

Table 5.3: The properties of CVE labeled nodes in the Neo4j database used in the implementation of the designed software artifact. The corresponding CVE json property references the CVE format used in the json data feed hosted by NVD.

Key	Value	Corresponding CVE json property
description	The textual description of the vulnerability.	`cve.description.description_data.value` where `cve.description.description_data.lang == 'en'`
id	The NVD ID assigned to the vulnerability.	`cve.CVE_data_meta.ID`
embedding	The calculated semantic embedding of the vulnerability.	None

Additional to the CVEs, references to the CVEs are also imported from the NVD JSON file. Each reference contain a link to additional information about the CVE

vulnerability, or alternate sources to the CVE vulnerability. Each CVE generally has one or more references, which point to external websites outside of NVD. In the CVE json data feed, each CVE contains an array of references. In the implementation of the designed artifact, references are imported into the Neo4j database as nodes labeled "Reference". Table 5.4 describes the properties of references.

Table 5.4: The properties of "Reference"-labeled Neo4j nodes used in the execution of the software artifact. The corresponding CVE json property references the CVE format used in the json data feed hosted by NVD.

Key	Value	Corresponding CVE json property
name	The name of the reference as specified in the CVE.	`cve.references.reference_data.name`
url	The url of the reference, pointing to an external website.	`cve.references.reference_data.url`

Reference nodes and CVE nodes are linked to each other by using Neo4j relationships. Figure 5.3 shows a screenshot from the Neo4j browser view, showing 2 CVEs and their corresponding references, imported into the database.

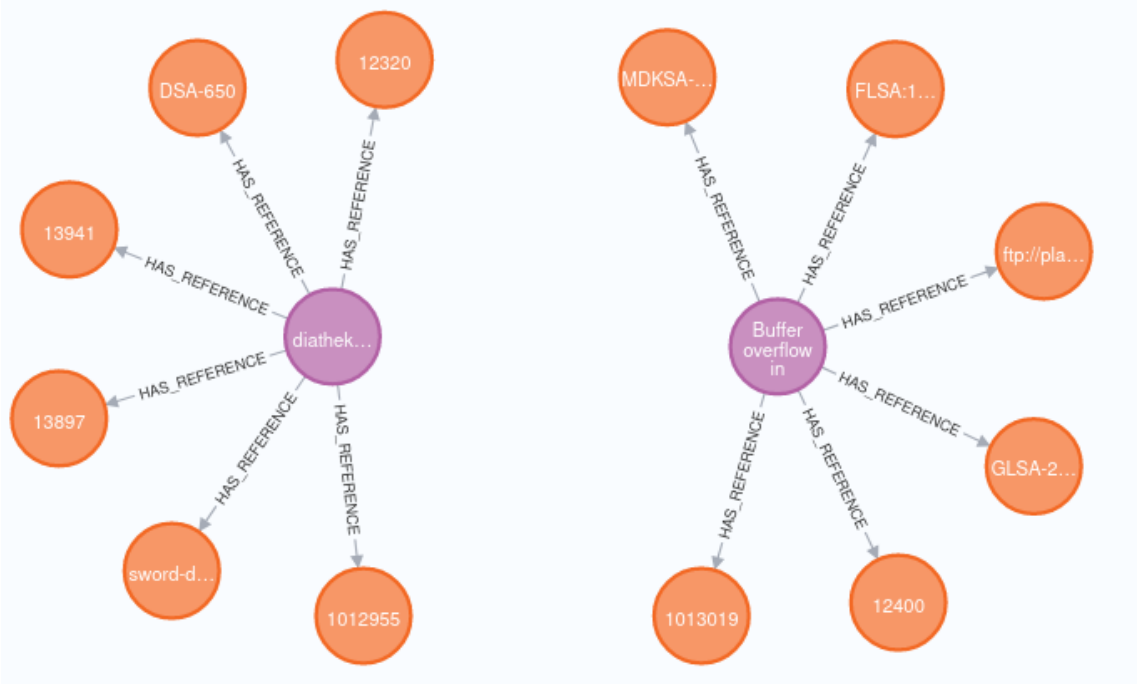


Figure 5.3: A screenshot showing the output from the Neo4j browser illustrating the relationship between CVEs and references. Purple nodes represent vulnerabilities. Orange nodes represent references.

Automotive ISAC import

Data from automotive ISAC is imported into the database. Two labels are defined in the database to represent the datatypes imported from automotive ISACs

attack matrix: ATM-Technique, and ATM-Tactic. The two datatypes have the same properties, listed in Table 5.5. Nodes labeled "ATM-Tactic" represent abstract attack methods in the matrix, and nodes labeled "ATM-Technique" represent concrete techniques which may be applied. Each technique references at least one tactic. Data is imported from ATM via the json resource file provided by automotive ISAC [7].

Table 5.5: The properties of ATM attacks and techniques in the Neo4j database used in the implementaion of the designed software artifact. The corresponding ATM json property references the format used in the downloadable json file of the matrix, hosted by Automotive ISAC.

Key	Value	Corresponding ATM json property
description	The textual description of the tactic or technique.	`atm.description`
title	The title of the tactic or technique.	`atm.title`
type	The type of data. Either tactic or technique.	`atm.type`

Relationships between tactics and techniques are captured in Neo4j as illustrated in Figure 5.4.

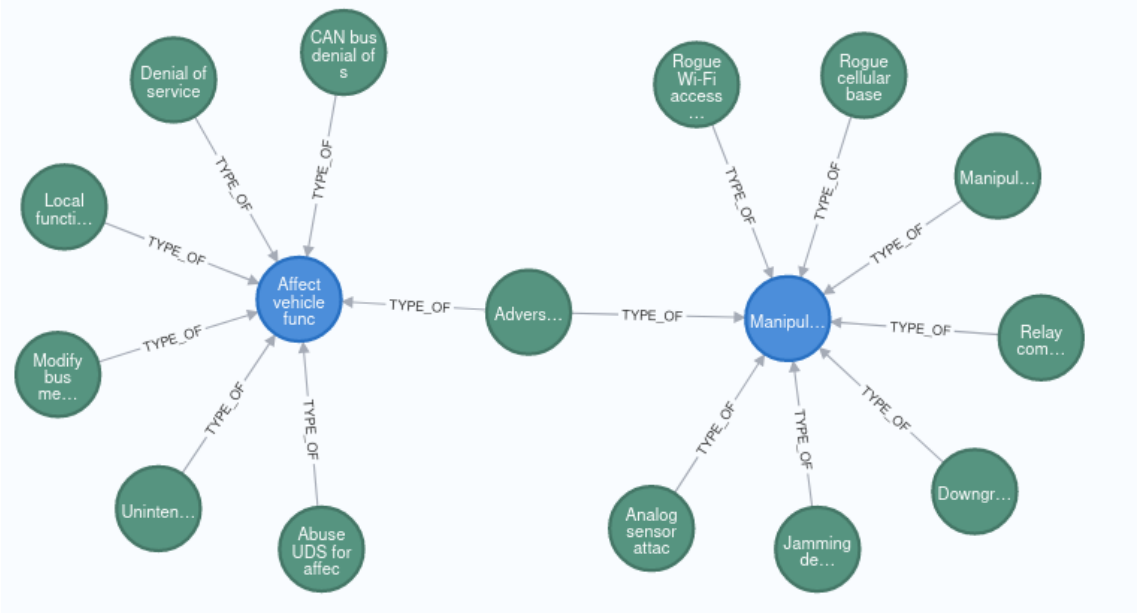


Figure 5.4: A screenshot showing the output from the Neo4j browser illustrating the relationship between ATM-techniques and ATM-tactics. Blue nodes represent tactics, green nodes represent techniques.

Creating CVE embeddings

Semantic search done as part of the execution of the program requires the CVE nodes in Neo4j to contain an array of semantic embeddings. These embeddings are created by iterating through all of the CVE nodes, and embedding the description property. The embedding model used to create embeddings was the `text-embedding-ada-002` provided by OpenAI. Once an embedding had been created for a CVE node, the node was updated by populating the `embedding`-property of the node. Table 5.3 shows the full list of CVE properties in the database.

Creating index

Finally, the software artifact is dependant on a Neo4j index to perform similarity search on CVEs during execution. The properties of an active and correctly configured index are shown in Table 5.6.

Table 5.6: Properties of the required index on CVE nodes in the Neo4j database.

Property	Value
name	"cve_descriptions"
state	"ONLINE"
populationPercent	100
type	"VECTOR"
entityType	"NODE"
labelsOrTypes	["CVE"]
properties	["embedding"]
indexProvider	"vector-1.0"

5.3.2 Execution

This section will present the execution of the program, by explaining the various modules in the program. A detailed program flow chart can be seen in Figure A.2. A highly simplified code snippet describing how the modules are called can be seen in Listing 5.3. In the code of the artifact, modules are represented as single files. Each module takes in a set of parameters, performs one or more calls to the LLM, and processes the output which is finally returned. The modules are described in the order that they are executed by the program. In addition, LLM prompts used in the modules are described, and the model and model temperatures are given. Prompts which rely on a higher level of reasoning use ChatGPT 4, and prompts with simpler tasks rely on ChatGPT 3.5. As mentioned in Chapter 2.8, ChatGPT has a better general performance, but it is notably slower, and thus impractical to use for all prompts in the program. Higher temperatures were used when the LLM had to perform a higher degree of reasoning, to encourage creativity, such as when coming up with requirements and attacks. Lower temperatures were used when the LLM was asked to perform closed end tasks such as filtering.

Listing 5.3: A simplified version of the programs main function, implemented in python. The version shows how and when different modules are called, and how the output from the modules are used.

```
1  def simplified_program_flow(input_cve_path):
2      input_as_array = read_input_from_csv(input_cve_path)
3      for input_edge in input_as_array:
4          potential_attacks_list = AutoIsacAttackModule.
               get_auto_isac_attacks(input_edge)
5          for potential_attack in potential_attacks_list:
6              relevant_cves = CveModule.get_relevant_cves(
                   potential_attack, input_edge)
7              attacks = AttackModule.get_attacks(
                   relevant_cves, input_edge,
                   potential_attack)
8              requirement = RequirementModule.
                   get_requirements(attacks)
9              store_attack_and_requirement_for_edge(
                   input_edge, attacks, requirement)
10     write_stored_attacks_and_requirements_to_csv_file()
```

Automotive ISAC technique module

The first module called by the program is responsible for determining which types of attack techniques a specific edge in the system model could be susceptible to. In Listing 5.3, the call to the module is found on line 4. In Figure A.2, the same call is denoted with the letter A. An example of inputs and outputs to and from the module are shown in Figure 5.5.

Example 5.5: An example of the inputs and outputs in the Automotive ISAC technique module. The output is the description of the Automotive ISAC attack technique which the module identified to be the most fitting to the scenario.

Input: `(source = 'Bluetooth', destination = 'Navigation ECU', channel = 'Bluetooth data', root_path = True)`

Output: "Exploit via radio interface:

The vehicle may be targeted for exploitation through its interface to cellular networks or other radio interfaces.

Baseband Vulnerability Exploitation

A message sent over a radio interface such as cellular, Bluetooth, GPS, NFC, Wi-Fi, TPMS, DAB, or others to the vehicle could exploit a vulnerability. Malicious SMS Message

An SMS message could contain content designed to exploit vulnerabilities in the SMS parser on the receiving device. An SMS message could also contain a link to a web site containing malicious content designed to exploit the device web browser. Vulnerable SIM cards may be remotely exploited and reprogrammed via SMS messages.

This technique was adapted from MITRE ATT&CK."

The module begins by fetching the names of all techniques which are relevant to the edge. If the edge is a potential point of entry into the system, such as the first two examples of edges listed in Listing 5.1, techniques which are types of the "Initial access" tactic are fetched. Otherwise, if the edge is not a potential point of entry, techniques which are types of "Lateral movement" are fetched. The tactics "Lateral movement" and "Initial access" are two of the automotive ISAC techniques present in the database as described in Chapter 5.3.1.

Once the list of techniques have been fetched, a single call to ChatGPT is made. The template specifies that the LLM is an automotive cybersecurity expert. It is then asked to select one of the techniques, based on the edge it has been provided. The prompt is sent to ChatGPT 3.5 with a temperature of 0.3.

Once the technique has been chosen, the full description of the technique is fetched from the Neo4j database and returned.

CVE module

The next module called by the program is responsible for finding and returning a set of CVEs which are relevant to the system components. Apart from the CVEs, the module should also return a summarised version of the CVE in a common format. As detailed in Listing 5.3, the module is called once per combination of input edge and Automotive ISAC attack technique. The module is called on line 6 in the listing, and the same call is denoted with the letter B in Figure A.2. Figure 5.6 shows an example of the inputs and outputs of the module.

To find relevant CVEs, four search strings are first crafted. The channel, source component and destination component are used as base for three of the search strings. These strings are concatenated with an additional search string to better find CVEs relevant to automotive vehicles. An example of a search string made from the "bluetooth" component is shown below:

$$\underbrace{\textit{"Bluetooth}}_{\textit{Component}}\ \underbrace{\textit{vulnerability in an automotive vehicle or car"}}_{\textit{Helper text}}$$

The fourth search string is composed solely from the AutoISAC attack technique identified in the previous step.

To perform a similarity search into the database of CVEs, the search string is first embedded. This embedding is done in the same way as the description strings of the CVEs in the Neo4j database were embedded. The model used to convert the strings to embeddings is the `text-embedding-ada-002` model. As described in Chapter 5.3.1, the model is hosted by OpenAI, and http requests are sent to retrieve the embeddings.

Once the embeddings have been fetched. Similarity searches are made to the Neo4j database, to find CVEs with descriptions which are similar to the search strings. The method used to achieve this is cosine similarity. Listing 5.4 shows how this can be achieved using cypher code. In the artifact, this code is integrated into a python script used by the CVE module.

Listing 5.4: The Cypher call used to perform similarity search in the software artifact. `index_name`, should be the name of the index, as set in Table 5.6. `limit` is the amount of nodes that should be returned. `query_embeddings` is the resulting array of the embedded search string.

```
1 CALL db.index.vector.queryNodes(index_name, limit,
      query_embedding)
2 YIELD node AS similarAttack, score
3 RETURN similarAttack.description AS description, score
```

Once four lists of CVE vulnerabilities have been identified, one for each search string, then CVEs are filtered. First, duplicates are removed. Duplicates can occur if the Neo4j database contains multiple instances of the same CVE, or if a single CVE has been found in more than one of the four searches. The second step in filtering is done by querying ChatGPT. The LLM is provided with the full list of identified CVEs, and it is then asked to remove any CVEs which do not explicitly relate to vulnerabilities that occur in vehicles. In particular, ChatGPT is asked to provide a list of all CVEs which do not mention automotive car components, brands or otherwise specify that the vulnerability was found in an automotive context. The model used to handle this query is ChatGPT 3.5, and the temperature used is 0.2.

When the module has filtered out a single list of relevant CVEs, it proceeds to fetch all references relevant to the CVEs, from the Neo4j database. Information in references are then downloaded using http calls to the url in the reference, and for each CVE, the references and the CVE are summarized into a short summary. The summary is created using a single call to ChatGPT, with the information in the reference, as well as the CVE itself as input. ChatGPT is prompted to write a short summary of 40-60 characters, specifically including the component which is vulnerable, the type of vulnerability and the communication channels used. The main goal of writing summaries was to provide later prompts with a common format for all CVEs. It was implemented to mitigate issues found in the intermediate evaluations, where it was discovered that the program had a tendency to select the same CVE repeatedly for several potential attacks. ChatGPT 3.5 is used to achieve this with a temperature of 0.2.

Finally, each CVE is returned in a tuple, together with it's summary. The output is in the form of an array of tuples, as described by Example 5.6.

Example 5.6: An example of the inputs and outputs in the CVE Module. The inputs are a set of parameters, and the output is an array of tuples. The first item in the tuple is a CVE. The second item in the tuple is a summary of the CVE.

Input:
```
(source = 'Bluetooth', destination = 'Navigation ECU',
       channel = 'Bluetooth data', root_path = True,
       attack_technique = attack_technique)
```

Output:
```
[
('CVE-2021-46145: The keyfob subsystem in Honda Civic 2012
vehicles allows a replay attack for unlocking. This is related
to a non-expiring rolling code and counter resynchronization.',
'The vulnerability in Honda Civic 2012 vehicles involves a replay
attack on the keyfob subsystem. It exploits a non-expiring
rolling code and counter resynchronization, allowing unauthorized
unlocking.'),
...]
```

Attack finding module

The attack finding module is tasked with finding potential attacks to a specific edge and Automotive ISAC attack technique. The inputs and outputs to the module are shown in Example 5.7. In Listing 5.3 a call to the module is made on line 7, and the call is denoted by the letter B in Figure A.2.

Example 5.7: An example of the inputs and outputs in the Attack finding module. The inputs are a set of parameters, and the output is a string describing one or more attacks. The `cve_list` parameter is a list of CVEs and their summaries, identified in previous modules. The `attack_technique` parameter is an attack technique identified in the Automotive ISAC technique module.

Input:
```
(source = 'Bluetooth', destination = 'Navigation ECU',
       channel = 'Bluetooth data', root_path = True)
       cve_list = cve_list, attack_technique = attack_technique
```

Output: See 'Attack', 'Description', 'Execution', and 'CVE References' shown in Figure A.1

The module uses a single call to ChatGPT to identify attacks. All inputs listed in Example 5.7 are used as inputs to the ChatGPT prompt. Previous ChatGPT calls have been done with the primary use of filtering or selecting. Unlike these prompts, this call to ChatGPT relies heavier on the reasoning capabilities of the module. As a result, the attack finding module relies on the reasoning capabilities of the LLM to a greater degree.

The prompt starts with general instructions to the model describing its role as a security requirements engineering expert, and a general explanation of the task which the model has to perform. Following this, the inputs are given to the model. The inputs are the selected technique, the full list of relevant CVEs, as well a description of the edge.

Finally chain of thought instructions are given to the model. The instructions begin by asking the model to group the vulnerabilities into separate groups based on common patterns or themes in the vulnerabilities. This is followed by asking the model to come up with attacks methods for each identified theme in the list CVEs.

An important finding was that the prompt must specify that the model should not assume that specific vulnerabilities are present in the system under analysis. During evaluation, it was observed that ChatGPT often assumed that vulnerabilities listed in the CVEs were directly applicable to the system being analyzed. It frequently made statements suggesting that an attacker could directly exploit a CVE. These statements were typically expressed as follows:

> *"...the attacker then applies the bluetooth vulnerability identified in CVE-2017-9212, to compromise the Navigation ECU"*

The chain of thought technique described in the previous paragraph aids in mitigating this behaviour. However, to further mitigate this, explicit instructions were given to the model to guide it away from making such statements. The model used for this prompt is ChatGPT 4.0, with a temperature of 0.3.

Requirements module

Finally, once an attack has been identified for a specific edge and attack technique, then requirements are identified to mitigate the attacks. Similarly to the attack finding module, the module uses on a single call to ChatGPT. An example of inputs and outputs from the module is shown in Example 5.8. The module is called on line 8 of Listing 5.3, and is denoted by the letter D in Figure A.2.

Example 5.8: An example of the inputs and outputs in the Requirements Module. The input is an attack identified in the previous module. The output is a security requirement, returned as a string.

Input: `(attack = attack)`

Output: See 'Requirement' shown in Figure A.1.

The prompt used to find potential mitigations required less tricks and mitigations then the prompt used to come up with attacks. The prompt begins by the standard role description, specifying that the LLM is a security requirements expert, with great knowledge of the NVD and MITRE databases. The prompt then explains

what a security requirement is, that is, a mitigation which could be implemented into the system. This is followed by the input, which is made up of all of the parameters present in Example 5.8.

6
Findings

This chapter presents findings from the design science study related to the research questions defined in 1.1. The first sub-chapter identifies the problem with the current way of eliciting security requirements (RQ1). The second presents insights learned while developing the artifact (RQ2). Finally the third sub-chapter presents the effectiveness of the artifact by presenting the results from the validation with experts (RQ3).

6.1 Problem investigation (RQ1)

The first research question was answered during the problem investigation phase of the first iteration with a literature study, interviews and document analysis. During these activities the authors learned both about the functions of TARA and its challenges. The identified challenges are presented in the following sub chapters.

6.1.1 Challenges from interviews

During the problem investigation in the first cycle a set of investigative interviews were conducted. The findings of these interviews were:

Information processing
The main challenge, according to some interviewees, is the amount of information that has to be gone through to conduct a proper risk assessment. Eliciting threat scenarios, attacks vectors and mitigation for a technical system are all activities which require the practitioner to read through and analyze substantial information. Relevant information could be found in public security standards like wp.29 [55] and ISO/SAE 21434 [18], internal company documents, vulnerability databases, system documentation or on the internet. To not miss any important details during the information processing, one practitioner said that the TARA was developed in iterations by several people.

Continuous compliance
Continuous compliance was mentioned to be a challenge, as it demands ongoing attention and resources throughout the entire development process. An interviewee said that a TARA needs to be updated frequently because of changes to the system and the rise of new vulnerabilities. When asked how frequently, the interviewee said it depends on the resources available, but said that it is something that should be

done more often then it is today. Beside the need for resources, it also requires effective communication within the organization. Without such communication, there is a risk for gaps in security measures, potentially allowing for exploits to be untreated and thus threatening of compromising the system.

6.1.2 Challenges from literature review

The search string yielded 70 papers in IEEE Explore [17] and 38 700 in google scholar [12]. The papers from google scholar were sorted on relevance and the top 50 were selected. After filtering the 120 papers based on their titles and abstracts, a set of 19 papers, shown in appendix A.3, were selected for a full read. Table 6.1 shows the challenges found from these papers:

Table 6.1: Challenges in cybersecurity requirements engineering.

Challenge	**Source**	**Explanation**
cross-domain expertise	[30, 32, 49]	This challenge was mentioned as a consequence from the growing complexity of software and its interaction with hardware.
Rapid changing environment	[28, 51, 52, 27]	Software implemented functions increases quickly, making it difficult for security requirements to stay compliant with the system
Wireless connections, IoT	[35, 47, 32, 51]	Connections to networks and software functions make the vehicle more vulnerable to cyber attacks.
Different variants of components	[50]	Many components are developed in several different variants and integrated over a sequence of prototype phases.
Redundancy	[50]	In large projects such as development of automotives, several hundred requirements and related documentation exist. These documents normally contains significant amount of redundant information.

6.2 Solution candidate (RQ2)

To mitigate the challenges identified in Chapter 6.1, an artifact was designed. The purpose of the artifact was to elicit requirements and potential attacks in a TARA scenario, which could be provided to practitioners conducting the analysis. The artifact achieved this with a pipeline of calls to ChatGPT, in combination with grounding.

The final answer to RQ2 is the designed artifact outlined in Chapter 5. To motivate the design of the artifact, specified requirements from the initial problem investigation are presented. This is followed by the result of the intermediate evaluations, and an additional set of requirements to improve the problems identified in these evaluations.

Table 6.2: Table listing the initial requirement of the artifact, elicited during the initial problem investigation phase.

ID	Requirement
R1	The system shall support output to CSV files.
R2	The system shall support input from CSV files.
R3	The system shall generate TARA attack paths.
R3.1	The attack paths shall include a description of the attack.
R3.2	The attack paths shall include a list of steps incorporated into the attack.
R3.3	The attack paths shall include motivations based on real world attacks or vulnerabilities.
R4	The system shall generate security requirements.
R4.1	The system shall provide motivations for the suggested requirements it elicits.
R5	Output produced by the program must be traceable to preceding TARA activities.

6.2.1 System requirements

To specify the system, a set of requirements were defined. Requirements of the system were in part elicited from practitioners during the initial problem investigation, and in part from the research questions and objective of the study. These requirements are listed in Table 6.2.

CSV support (R1 & R2)

The specification to produce and receive CSV files was elicited from initial interviews with practitioners from Systemite. During this interview, it was discovered that the main software used in the company, Systemweaver, can convert any intermediate TARA output as well as the output from the full TARA analysis scenario into an structured table format. This format could easily be converted to a CSV file. Additionally, CSV files is have a simple structure which is easy to work with in the selected programming language. The use of CSV as the primary input and output format was thus specified as a requirement.

Attack paths (R3, R3.1, R3.2, R3.3)

The artifact was in part designed to generate attack paths. The format of the attack paths was specified to mirror the format of the output from an attack path analysis. By investigating the headlamp TARA introduced in Table 4.2, attack paths were specified to include:

1. An attack scenario description, as specified in **R3.1**.
2. A numbered sequence of tasks which an attacker would have to perform, in order to carry out the scenario, as specified in **R3.2**.

Additionally, to help in evaluation, and to validate the suggested attacks to practitioners, the system was also specified to include motivations for the attacks.

Requirements generation (R4, R4.1)

One of the goals of the artifact was to enable the generation of requirements. This was design goal was included in the specification of the design as **R4**. Similar to

R3.3, the requirements were specified to include a motivation to validate the requirement as a mitigation to its corresponding attack path suggestion.

Traceability (R5)

During initial interviews, the need to maintain TARA tractability was expressed by one practitioner:

> *"... you can choose to design something which only elicits requirements, however, even if you do so you must maintain traceability. Even if practitioners trust the results from the requirements, legal compliance requires us to trace the outcome of all activities back to previous activities."* - Individual 4

This discovery lead to requirement **R5**.

6.2.2 Intermediate evaluation

Intermediate evaluations highlighted a set of challenges and an yielded an additional list of requirements which were implemented to improve the output of artifact. The primary purpose of intermediate evaluations was to identify issues in the artifact, rather then to elicit new requirements or features of the artifact. Regardless, additional requirements were found and are listed in Figure 6.3.

Table 6.3: Table listing the requirements elicited during the intermediate interviews.

ID	Requirement
LR1	The system shall ground itself in public threat and vulnerability databases.
LR1.1	Grounding shall include NVEs CVE database
LR1.2	Grounding shall include MITREs attack databases.
LR1.3	Grounding shall include automotive ISACs attack matrix.
LR2	The system shall summarize CVEs in a structured format before applying them to a prompt.
LR3	The system shall filter CVEs to discard CVEs not relevant to automotive systems.
LR4	The system shall apply chain of thought prompting to encourage the LLM to find common patterns in vulnerabilities.
LR5	The system shall provide attacks for each edge in the system model.

Grounding (LR1)

During the initial implementation of the artifact, the artifact was designed to rely on prompt engineering. As specified in **R3.3**, a requirement was that the artifact provided references to real world vulnerabilities to validate the attack paths it generated. The initial design to achieve this was to prompt ChatGPT to provide the program with a list of real CVE's which could be relevant to the program. However, during the intermediate evaluations it was found that this approach, which relied soley on the training of ChatGPT, was insufficient. It was determined that the program tried to came up with fictional CVEs which were tailored to be as relevant as possible.

To mitigate this issue, **LR1** was defined. Initially, grounding was based on 4 databases:

1. *NVD database of CVEs*
2. *MITRE Att&ck Enterprise*
3. *MITRE Att&ck Mobile*
4. *MITRE Att&ck ICS*

As detailed in Chapter 5.3, grounding was done by encoding the description of each database entry into an array. Grounding was then achieved by encoding a semantic context into an array, and performing a similarity search to find the most relevant vulnerabilities and attacks in the databases.

However, during the evaluation of iteration 2, it was discovered that many of the databases were too general for the use-case of the artifact. Individual 2 directly suggested that it would be good to use a different data source, instead of MITREs databases. MITREs which were very general and not specifically targeting the automotive field:

> *"You can of course [use Automotive ISACs attack matrix], I mean some of the attacks in this matrix are actually adopted from MITRE attack. It's a good idea to also try to add this ATM to your new project database as well"* - Individual 2

This led to a new low level requirement in the final iteration, **LR1.3**. Finally, AutoISACs attack matrix was introduced to the artifact, as described in Chapter 5.

Generalizing attacks (LR2, LR4)

During the evaluation of the second iteration, it was discovered that the program made assumptions about specific components which may not be present in the system under analysis.

> *"... I don't know what an OnStar system is, but I'll assume it's some type of component present in certain vehicles..."* - Individual 3

> *"... it seems like it does not take the system model into consideration, and instead tries to apply specific vulnerabilities from the CVEs."* - Individual 3

The components mentioned were explicitly named in the referenced vulnerabilities that the program found, which indicates that the program overestimates how applicable the identified vulnerabilities are in a specific context.

To address this, the two requirements **LR2** and **LR4** were specified. Using ChatGPT and specific prompts to process the CVEs, they were summarized into a common format. The format was specified to include the type of components which the vulnerability affected, and the vulnerability itself. By employing these prompts, the goal was to reduce the specificity of the CVE descriptions such a way that they did not describe specific versions. The summarized CVEs were then used for future

attack path elicitation.

Additionally, to further discourage the program from making general assumptions, prompt engineering was employed with chain-of-thought as a central focus. The prompt which was targeted was the final prompt for generating attacks. This prompt was grounded with a list of CVEs which the program had identified as relevant, the source and destination component, their communication channel, as well as an Automotive-ISAC attack technique. Additionally, the prompt contained instructions for the LLM to generate a step-by-step attack targeting the destination component. Figure 6.1 illustrates the goal of the chain of thought prompting used in the program.

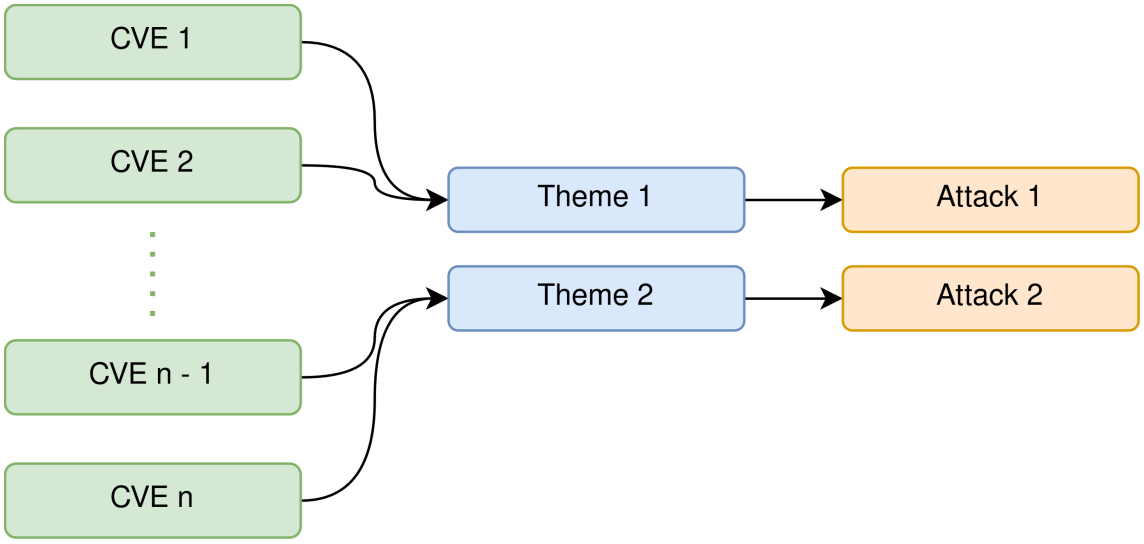


Figure 6.1: Illustration of the chain-of-thought pattern imposed on the ChatGPT model to encourage an improved output.

CVE filtering (LR3)

Intermediate evaluations in the second iteration also identified the need to filter CVEs. When looking at the output of the code, it was discovered that some CVEs were irrelevant to the scenario. Individual 2 mentioned this, and further mentioned a technique which could mitigate the issue.

> *"You are using similarity search, so it is always possible that it doesn't match very well with what you're looking for."*
>
> *"...[To mitigate this, you] can use a reflection agent. So basically an LLM tries to identify vulnerabilities and then another LLM tries to judge the findings and give feedback to the previous LLM."* - Individual 2

To adhere to these concerns, requirement **LR3** was defined. Instead of a conventional reflection agent, mentioned by Individual 2, a simpler filter method was employed. Filtering was conducted by a single call to ChatGPT. The prompt template contained all of the identified CVEs in a list, and the prompt contained instructions to

discard all CVEs which were not related to the automotive industry.

Attack path presentation (LR5)

One final important discovery in the evaluations related to the presentation of the attacks and requirements. **R3.0** specified that the system should generate TARA attack paths. This specification was based on the TARA process. As mentioned in Chapter 2.2, a TARA attack path analysis identified attack paths, based on the preceding TARA activities performed during analysis. Generally, a TARA attack path is designed to show that a predetermined threat can be carried out to compromise the item which is being analysed. This attack may involve compromising several of the listed components of the system model.

During evaluations, a different approach was suggested. Instead of attempting to find a single attack compromising several components in the system to reach a final goal, the program may instead try to identify every possible attack between every *pair* of components in the system model. Individual 2 discussed how security analysis may approach problems by building attack graphs, and suggested that the artifact could mirror this approach.

> *"Basically [security analysts] start from the system architecture and try to analyse different paths in their models and then encapsulate those paths in a graph. And then then then they try to make connections between vulnerabilities in the components of the graph. If I have "vulnerability X" in "component A" and "vulnerability Y" in "component B", then exploiting vulnerability X could be an entry point towards compromising "component B". They then try to come up with different sort of combinations."* - Individual 2

Presenting attacks in this format will produce more attacks, and more requirements, but the attacks vectors will be shorter. To complete TARA according to the aforementioned standards and regulations, several of the attacks would have to be combined to build complete attack vectors. It was decided that the approach should be adopted as a requirement of the artifact, and **LR5** was specified.

6.3 Evaluation (RQ3)

The evaluation of the artifact was conducted through a demo of the program on a vehicle system. The program generated 10 requirements for the system which were shown for 4 experts. An example of one of the requirements being evaluated can be found in Appendix A.1. For each requirement a set of evaluative questions were asked to evaluate its quality and afterwards some final questions regarding the programs usefulness in TARA were asked.

During the evaluation of the artifact, several strengths and weaknesses were identified, as well as potential use cases of the artifact. The chapter will begin by a summary of the findings in the evaluation. It will then present how the data was collected and analyzed and end with a presentation of the findings tied to RQ3.1 and RQ3.2.

RQ3.1: It was discovered that the artifact is capable of producing well formulated security requirements which are in line with expectation of practitioners. The requirements made reasonable assumptions of the system for which they were elicited, and were considered to be unambiguous by all practitioners. However, findings also indicate that the requirements may not be completely correct, as 1 of the 10 requirements was found to not fully mitigate its corresponding attack during evaluation.

Findings further identify challenges in the elicitation of TARA attacks. The reasoning behind how an attack should be performed, according to the program, was in some cases considered to be lacking. In instances where such issues were identified, the main issues were traced to assumptions made by the artifact. Either the artifact assumed that the attacker has an unreasonable amount of power, yielding a highly unfeasible attack, or the artifact assumes that the system has functionality which it can not be assumed to have.

RQ3.2: The evaluation identified that the artifact could be usefull in a TARA process, but that there are limitations to it's use. Incomplete or completely new TARA processes could benefit the most from the artifact, as it is capable of identifying generic attacks and requirements, showing that the security engineers have not missed more essential threats. However, it is limited in that it may not be able to come up with requirements which are complex enough to complete a mature TARA process.

6.3.1 Data collection and analysis

The evaluation data was purely qualitative and was collected during four 1 hour long interviews described in Chapter 4.2.3. These interviews were transcribed and analyzed using thematic analysis. As shown in Figure 6.2, 21 minor themes were identified, which were grouped into 4 major themes.

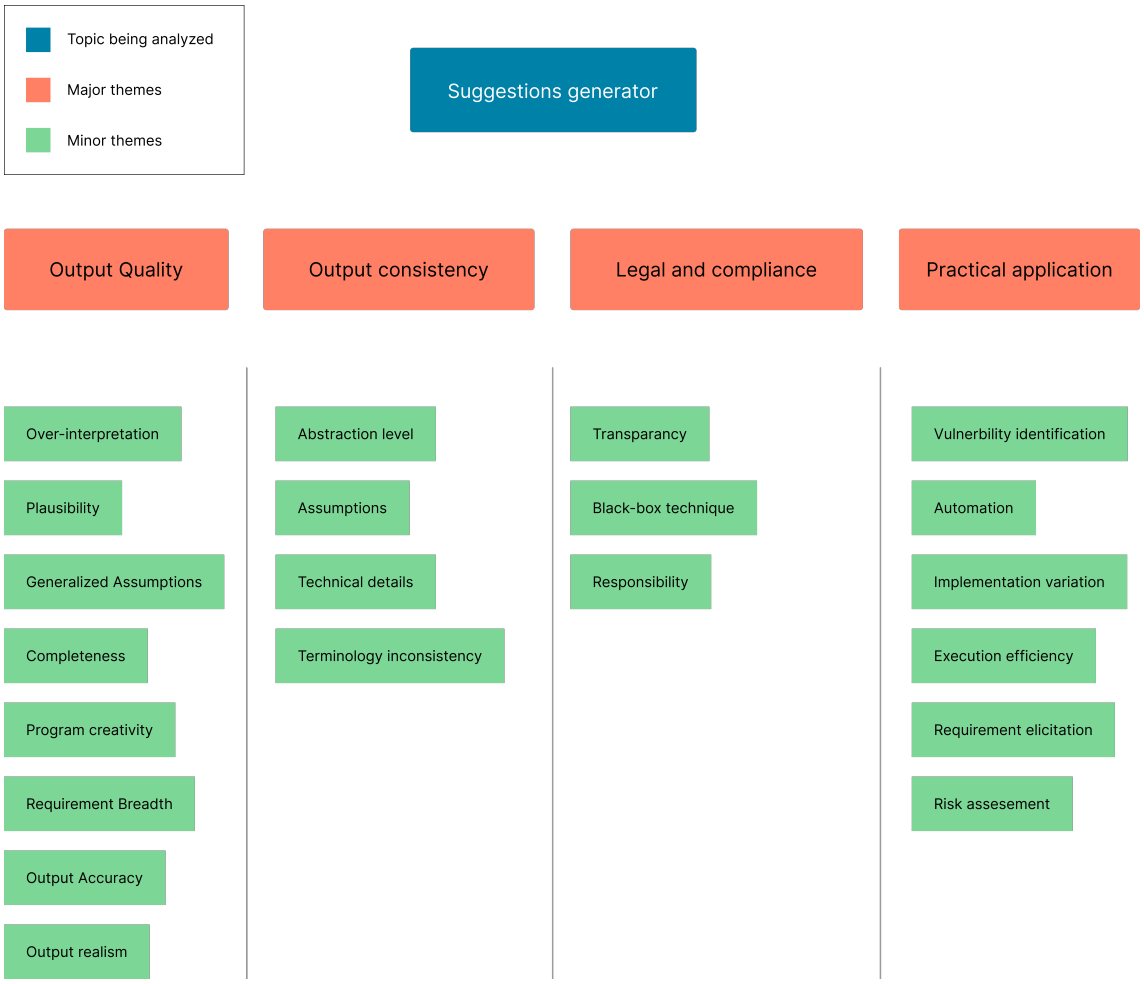


Figure 6.2: Identified themes in the evaluation interviews.

A breakdown of the result from the interviews continues in the following 2 sub-chapters, one explaining the findings related to the output of the program (RQ3.1) and one on findings related to the effects from applying the artifact on the current TARA process (RQ3.2).

6.3.2 Evaluation of output (RQ3.1)

RQ3.1 was defined in order to aid in generating insight to RQ3 and it states that:

> ***RQ3.1:*** *How well does the output produced by the designed artifact meet the expected standards and goals of a TARA project?*

One of the identified themes which almost exclusively related to RQ3.1 was "Output Quality". To some extent, this was by design as many of the questions in the semi structured interview targeted the quality of individual outputs.

Interviews indicated that the output provided by the program perceived to be of high quality. However, the interviews also highlighted a difference in quality between the generated attacks and the generated requirements.

Generated attacks

In general, requirements were perceived to on average maintain a higher standard then attacks, with one interviewee stating that:

> *"I'm positively surprised by the quality of the requirements..."* - Individual 1

> *"...many of the attacks are highly credible as well, however it seems as though the consistency of this quality is a bit wobbly."* - Individual 1

Two primary issues were identified with the attacks. One of these issues was related to the assumptions that the program made when it generated the attacks. A perception shared by 2 of the interviewed individuals, was that the program assumes that an attacker has an unreasonably high level of control of certain components.

> *"In some attacks, all of the steps are present, while in other attacks, the program assumes that the attacker has gained access to more control than one could expect it to have, without properly explaining how such access is gained. Once this assumption has been made, subsequent steps in the attack are both coherent and plausible."* - Individual 1

On a similar topic, one individual mentioned that the assumptions made by the program in some cases relies on specific software or hardware protocols that can not be assumed to exist inside of the program. In one notable instance of this issue, it was remarked that the program sometimes directly states that an attack should use one of the referenced CVEs in the attack. This happened despite the prompt-engineering techniques employed to generalize the attacks, mentioned in Chapter 6.2.2.

> *"... part of an attack step is that [The attacker sends a 'download and execute' or 'write memory by address' command to the Body Control ECU.]. This would be a plausible route for an attacker to take, but it relies on the mentioned protocols being implemented into the components which are being analysed, which could be a far fetched assumption to*

make." - Individual 1

This behaviour of the program is however viewed in another perspective by a different practitioner. In reference to the same attack, Individual 5 mentions that:

> *"This is a really nice catch by the program, I always tell people to not allow "write memory by address commands"* - Individual 5

In summary, this shows that the attack does highlight important risks in an automotive system, however the attack itself may not be relevant if the system does not incorporate the specific functions, as mentioned by individual 1.

Generated requirements

The disparity in quality between attacks and requirements was emphasised by how the practitioners perceived the generated requirements. When asked about the correctness of the requirements according to the definition in question 2, all practitioners found that the requirements were correct to a very high degree, with one practitioner identifying a requirement where which may be improved.

> *"Each requirement that you have presented me with have accurately targeted its corresponding attack."* - Individual 1

> *"... each requirement is the common mitigation which one would apply to it's corresponding attack."* - Individual 2

Individual 5 identified that an issue related to the correctness of one of the requirements was that recent attacks had been constructed to make the mitigation presented in the requirement.

> *"This one [Requirement specifying that rolling codes should be implemented in the key fob] is tricky. It reduces the feasibility of the attack, but more recent attacks have been identified where the attackers interfere with the signals from the key fob in such a way that a code rotation never takes place."* - Individual 5

An indication of this is that the requirements generated by the artifact fail to consider more recent attacks in some cases, which has a negative affect on its correctness.

The interviews showed that the individuals considered the requirements to be unambiguous, based on how they personally have worked with TARA. This was expressed by all 4 practitioners interviewed. However, with 2 out of 4 practitioners, the question on ambiguity in the generated requirements raised a discussion on requirement abstraction levels.

> *"This is the level of abstraction that I have used in the companies where I have conducted TARA analysis"* - Individual 1

> *"I think that these requirements are 'right on the money' in terms of their level of abstraction..."* - Individual 5

Individual 5 however elaborates on the potential perspectives of other practitioners.

> *"... but the engineers implementing the requirements may request them to be expressed in a lower level. The way I see it, requirements identified in TARA should keep this abstraction level, and additional implementation specific requirements should be derived from them."* - Individual 5

Individual 1 highlighted the difference in abstraction levels with an example.

> *"The requirements 'The ECU shall encrypt and verify outgoing and messages to the backend' and 'The ECU shall implement TLS' both treat the same problem. The latter is just specified on a lower level."* - Individual 1

6.3.3 Evaluation of process (RQ3.2)

To further aid the answering of **RQ3**, a second research question, **RQ3.2** was defined. The research question states that:

> ***RQ3.1:*** *How could the designed artifact influence the existing workflow in TARA?*

Legal compliance

When asked whether the program could face challenges to be implemented in the industry, several interviewees mentioned the problem of using generated data. The interviewees were generally skeptical about relying on a machine's security requirements because of the lack of reasoning and justifications made by the program. Which according to standards that were enforced by the company had to be present. However, one subject said that the program's effectiveness depended on its implementation, proposing that it could serve better as a suggestion tool rather than a replacement.

> *"If the artifact would be developed into an full automation tool, as opposed to tool for generating suggestions, then it may be constrained for legal reasons."* - Individual 3

Many of the interviewees aligned with this opinion. One individual used the words "black box solution" to describe the artifact and said that if the purpose of the artifact were to automate some part of the TARA, it might be better to strive for a more white box approach, to make it easier to legally incorporate it into the process.

Another opinion related to legal compliance, which was brought up by Individual 5, was that many companies may reject to implement such a solution, if a publicly available LLM is used.

> *"...[Using ChatGPT] that could be a problem, or I can see that some of the car companies will see that this could be a problem from a security perspective. Because you have to send data to it to get a response."* - Individual 5

This implies artifact used in this research would have to be re-implemented with an LLM which could be hosted by the companies using the artifact.

Practical implementation

When asked about how the program could be implemented in a TARA process in practice, the individuals aligned to a large degree on the possibilities and limitations of the artifact. Interviews showed that the artifact would be a helpful starting point for eliciting basic requirements in a TARA, but that it's potential was limited, and that more mature TARAs may not benefit very much from it.

> *"... this is good for engineers who are working on a relatively new TARA, but the requirements are not complete..."* - Individual 3

> *"...some of them are pretty basic compared to what you would expect to see in a completed TARA"* - Individual 3

> *"From the requirements that you've shown me, this is really really good if your drawing your system to start with. That you get your generic requirements and you can see if you missed something."* - Individual 5

Another interviewee believed the program to be of value even for mature TARA models by assisting with continuous compliance.

> *"One use case could be to deploy the program as a static analyser... to react on changes for the system architecture"* - Individual 1

The interviewee proposed that the program could maintain a changelog of the systems architecture, eliciting attacks and requirements only for recent changes and thus avoiding collisions. Potentially reducing the resources spent on continuous compliance. However, this would require some small modifications to the program as no snapshot of the system is saved nor compared between executions.

7
Discussion

This chapter will present a discussion surrounding the project which has been conducted. The discussion aims to show how the result presented in Chapter 6 could improve the current state of research, as well as present topics of validity threats, limitations and ethics.

7.1 Implications of research

The implications of the project have meanings for different contexts and is thus separated into two chapters, one for researchers and one for practitioners.

7.1.1 Implication for researchers

Numerous attempts to apply LLMs to domain-specific problems exist in literature, with popular domain including medicine [10][60][57] and translation [15][26]. This study further builds on the boundaries of LLM applications by applying them on a new domain through a structured development process involving practitioners from the automotive industry. The findings showed that cybersecurity requirements can successfully be elicited with the help of LLMs when provided with a limited context of the system under analysis.

A similar study [39] attempted to automate the process of eliciting safety requirements with an LLM. That study [39] concluded that the LLM was insufficient on its own, due to weaknesses in its reasoning, limiting its role to making suggestions for HARA practitioners. This study aligns with those findings, as the output of the LLM program sometimes made unrealistic or incorrect assumptions and therefore should be verified by an practitioner. This issue is arguably not a problem with this specific study but rather with LLMs in general. For example [54] discusses the mixed result of applying LLM chatbots in medicine. The study [54] shows that LLM chatbots often successfully help patients, however occasionally generating information dangerous for the patient. It seems that LLMs, as of today, are not mature enough to be fully autonomous.

Several implications can also be drawn from the design choices of the artifact. It was indicated by the practitioners in iteration 3 that the program produced better output compared to previous iterations. For instance, the artifact initially struggled with generating coherent attack-steps, but ultimately managed in the final iteration

after applying prompt techniques such as CoT. Implying that this prompt technique improves the accuracy of LLMs. Which aligns with previous research on the subject [58].

The use of RAG demonstrates two things: First is that RAG can be employed to mitigate hallucinations by the LLM and improve the accuracy of domain-specific tasks, as was shown in [25]. This became evident when comparing the quality of the generated requirements from iteration one and two. Where the requirements from the former iteration suffered from both hallucinations and irrelevant details not related to the scenario. Second is that public data sources such as the wp.29 threat list [55], CVE database [41] and MITRE ATT&CK database [34] provides sufficient data for an LLM to derive accurate cybersecurity requirements, encouraging future researchers to utilize said data sources in their LLM applications.

7.1.2 Implication for practitioners

The evaluation of the artifact revealed that LLMs are capable of generating well-formulated and unambiguous security requirements. This indicates that LLMs can be a valuable tool in eliciting cybersecurity requirements for systems, thereby reducing time spent on this activity in the TARA. However, the findings also highlighted that the requirements can not be guaranteed to be correct, suggesting that the output has to be validated before use.

Moreover, the integration of LLMs into TARA presents some challenges, particularly regarding legality and completeness. Findings indicate that a complete suit of requirements is not feasible with the artifact that we have presented. This indicate that some time could be saved when performing initial TARA analysis, however in more mature scenarios, humans engineers have to be involved to the same degree as before. Findings also indicate that an automated process may have legal complications, due to regulations imposed on the process. This indicate that the solution presented in this paper is sufficient to save some time from the practitioners in early stages, but may be limited in helping engineer in later stages. We can not show that a solution such as this can reduce the time and effort spent by engineers with continuous compliance.

7.2 Threats to Validity

This section presents discussions related to potential validity threats, as well as how the authors have taken actions to mitigate these threats. It does so by separating external and internal validity into two separate sub-chapters.

7.2.1 Internal Validity

When working with probabilistic tools such as ChatGPT, one clear threat to validity is that of over-fitting the solution to the specific problem instance on which the solution was built. This means that if a solution is built incorporating LLMs such

as ChatGPT, the risk is that it shows very good results for the specific problem that it was trained on, while the general performance is poor on problem instances which have not been presented to the designed solution.

In the context of this report, the problem instances are partially completed TARA scenarios. To mitigate this potential threat, the solution designed in this project was evaluated on one TARA scenario for the first two iterations, and evaluated on a second TARA scenario for the final iteration. The first TARA can be seen in 5.1 and features a headlamp component. The final evaluation of the artifact, which used a new TARA scenario, indicated that the program could perform well on unseen data. However, as will be discussed in Chapter 8.1.2, future work is warranted to analyse the generalizability of the designed solution. Such analysis would benefit from more TARA scenarios.

7.2.2 External validity

One potential threat to external validity which was taken into consideration throughout the report, was the sample size of the individuals who participated in the evaluations. To evaluate the final version of the design, a total of 4 individuals were interviewed, presented in Chapter 4.1. Similarly to the aforementioned internal threat, smaller sample sizes may affect generalizability in the findings.

However, as mentioned in Chapter 4, the evaluation was designed with this in mind. The individuals selected for the evaluation were of varying backgrounds, all with experience in the field of automotive SRE.

7.3 Limitations and Delimitations

This section will discuss the limitations and delimitations which were deliberately scoped or imposed on the project.

7.3.1 LLMs in use

One delimitation made in the project was the specific LLMs to use. In particular, this project limited itself to using ChatGPT. To show which LLM is *best* suited for this specific context, a study would have to be designed and performed comparing the performance of various LLMs within our artifact.

7.3.2 Model tuning

The artifact presented in this report relies solely on public general LLMs. The only techniques employed to improve the output from these LLMs were prompt engineering techniques. Other techniques which may have been employed are model training and fine-tuning, however this was not done in this project.

7.3.3 Project scope

One delimitation introduced in Chapter 6.2, is related to the project scope. It was decided not to design an artifact to generate any other topics than TARA attack scenarios and security requirements. This delimitation was made to allow the project to focus more on the chosen TARA activities. As mentioned in Chapter 3, at least one attempt has been made to build a software pipeline using LLMs to automate a full HARA analysis [39].

7.3.4 Data

The project was limited to two different TARA scenarios when designing and testing the software artifact. This was a practical limitation. Due to the sensitive nature of security data, the process of getting access to more data was beyond the scope of this project.

7.4 Ethics

The project presented in this report have several ethical aspects which should be considered. One ethical perspective is the responsibility of the users of the artifact. Similarly to data found in public vulnerability databases and attack matrices, there exists a risk that the information gained by the executing the artifact can be used for malicious purposes.

Another ethical perspective is that of the carbon footprint produced by LLMs. Studies have been carried out, investigating the energy expenditure of LLMs, and found that both training and deployment of these models have an significant impact on the environment [19].

8
Conclusion

In this study we identify a set of requirements, and the design of a software program which can be used to elicit attack path analysis and requirements.

By employing the program om a real world TARA scenario, we showed that general LLMs can be employed in a program to produce well formulated and relevant suggestions for attack path analysis within automotive TARA. We also found that the designed program could potentially be used in the TARA process as a good starting point to quickly elicit a set of attack paths and requirements which could be iterated on.

However, we also found challenges and limitations of a potential implementation of such program. Some of the attacks which the program elicits are based on assumptions which are not reasonable in the scenario being analysed. Furthermore, the suite of attacks was not complete for the scenario which was being analysed. As a result, practitioners who were interviewed identified that an appropriate use case of the program was restrained to relatively, or completely new TARAs, to identify generic requirements and to show engineers if they had missed anything.

8.1 Future work

As mentioned in Chapter 7.3, the project was affected by several limitations and delimitations which may be considered for future work. Additionally, some of the insights discovered during the evaluation of the artifact are further ground for potential future work.

8.1.1 Automation

To design an artifact to perform TARA automation, instead of requirements generation, was considered early on in the project. As presented in Chapter 6.3, such an approach may face some legal issues if implemented into a tool as of the writing of this report, due to rules and regulations imposed on cybersecurity within the Automotive sector. Additionally, this study as well as [39], highlights some difficulties in implementing LLM based tools to perform complex requirements analysis.

However, as the technology of LLMs improve, artifacts relying on such technologies may improve, and future attempts to automate the TARA process, built on

newer technologies, may show more promising results.

8.1.2 Analysis

To better understand the applications of LLMs in the requirement engineering field, future studies are warranted. The generalizability of any findings in this report are limited to automotive TARA analysis. However, TARA is a broadly applied methodology, and understanding the applications of LLMs is an ongoing initiative across multiple industries. Thus, we propose a field study with a wider range of practitioners across multiple industries to understand how well such a solution could be generalized.

8.1.3 Project continuation

The authors would also like suggest a few potential improvements to the current artifact. One such improvement is what Individual Individual 1 referred to as "attack trees". As presented in Chapter 6.3, attacks could be presented in trees, as opposed to sequential lists such as those made by the program. Building attack trees would mean that the program found and identified common attack steps between several attacks. This identification could then yield a tree, where common attack steps branch out into multiple possible attack steps. Individual 1 mentioned this way of viewing attacks to be more beneficial and said its frequently employed by practitioners in the industry.

A potential method to improve the accuracy of the LLM is to compliment the prompt-engineering techniques applied in this project with model fine-tuning. As mentioned in Chapter 7.3, such fine-tuning was not possible within the scope of this project. However, other research made within the field of LLMs show that fine-tuning of LLMs can significantly improve domain specific knowledge of an LLM [57].

Bibliography

[1] Constantin Adam et al. *Attack Techniques and Threat Identification for Vulnerabilities*. 2022. arXiv: 2206.11171 [cs.CR].

[2] Balaguer et al. "RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture". In: *arXiv.org* (2024). DOI: https://doi.org/10.48550/arXiv.2401.08406.

[3] Y. Liu et al. "Summary of ChatGPT-Related Research and Perspective Towards the Future of Large Language Models". In: *arXiv (Cornell University)* (2023). DOI: https://doi.org/10.1016/j.metrad.2023.100017.

[4] Zhang et al. "Siren's Song in the AI Ocean: A survey on hallucination in large Language models". In: *arXiv.org* (2023). DOI: https://doi.org/10.48550/arXiv.2309.01219.

[5] Abdullah Al Mamun, Md. Abdullah Al Mamun, and Abdullatif Shikfa. "Challenges and Mitigation of Cyber Threat in Automated Vehicle: An Integrated Approach". In: 2018. DOI: 10.23919/EETA.2018.8493171.

[6] *Auto-ISAC*. [Online; accessed 02-May-2024]. URL: https://automotiveisac.com/.

[7] *Automotive Threat Matrix*. [Online; accessed 26-April-2024]. URL: https://atm.automotiveisac.com/matrix.

[8] Hannah Bast, Björn Buchhold, and Elmar Haussmann. "Semantic Search on Text and Knowledge Bases". In: *Foundations and Trends® in Information Retrieval* 10.2-3 (2016), pp. 119–271. ISSN: 1554-0669. DOI: 10.1561/1500000032. URL: http://dx.doi.org/10.1561/1500000032.

[9] Bertalan. *Prompt engineering as an important emerging skill for medical professionals: Tutorial*. DOI: doi:10.2196/50638. URL: https://www.jmir.org/2023/1/e50638.

[10] Siyuan Chen et al. *LLM-empowered Chatbots for Psychiatrist and Patient Simulation: Application and Evaluation*. 2023. arXiv: 2305.13614 [cs.CL].

[11] Tore Dybå Daniela Cruzes. "Recommended Steps for Thematic Synthesis in Software Engineering". In: *Empirical Software Engineering and Measurement (ESEM)* (2011). DOI: https://doi.org/10.1109/ESEM.2011.36.

[12] *Google Scholar*. [Online; accessed 08-May-2024]. URL: https://scholar.google.com/.

[13] Jose Gumiel et al. "A Holistic Approach on Automotive Cybersecurity for Suppliers". In: 2023.

[14] Charles Haley et al. "A framework for security requirements engineering". In: (May 2006). DOI: 10.1145/1137627.1137634.

[15] Zhiwei He et al. “Exploring Human-Like Translation Strategy with Large Language Models”. In: *Transactions of the Association for Computational Linguistics* 12 (Mar. 2024), pp. 229–246. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00642. URL: https://doi.org/10.1162/tacl%5C_a%5C_00642.

[16] Erik Hemberg et al. *Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting*. 2021. arXiv: 2010.00533 [cs.CR].

[17] *IEEE Xplore*. [Online; accessed 07-May-2024]. URL: https://ieeexplore.ieee.org/Xplore/home.jsp.

[18] *ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering*. Standard. Geneva, CH: Society of Automotive Engineers, Aug. 2021.

[19] Peng Jiang et al. “Preventing the Immense Increase in the Life-Cycle Energy and Carbon Footprints of LLM-Powered Intelligent Chatbots”. In: (2024). DOI: https://doi.org/10.1016/j.eng.2024.04.002.

[20] Matthias Kern et al. “Model-based Attack Tree Generation for Cybersecurity Risk-Assessments in Automotive”. In: 2021. DOI: 10.1109/ISSE51541.2021.9582462.

[21] Marzana Khatun, Michael Glaß, and Rolf Jung. “An Approach of Scenario-Based Threat Analysis and Risk Assessment Over-the-Air updates for an Autonomous Vehicle”. In: *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*. 2021. DOI: 10.1109/ICARA51699.2021.9376542.

[22] Eric Knauss. *Constructive Master’s Thesis Work in Industry: Guidelines for Applying Design Science Research*. 2021. arXiv: 2012.04966 [cs.SE].

[23] Aditya Kuppa, Lamine Aouad, and Nhien-An Le-Khac. “Linking CVE’s to MITRE ATTCK Techniques”. In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ARES ’21. Vienna, Austria: Association for Computing Machinery, 2021. ISBN: 9781450390514. DOI: 10.1145/3465481.3465758. URL: https://doi.org/10.1145/3465481.3465758.

[24] Søren Lauesen. “Software Requirements-Styles and Techniques”. In: (Jan. 2002).

[25] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL].

[26] Chaoqun Liu et al. *Is Translation All You Need? A Study on Solving Multilingual Tasks with Large Language Models*. 2024. arXiv: 2403.10258 [cs.CL].

[27] Feng Luo et al. “A Framework for Cybersecurity Requirements Management in the Automotive Domain”. In: *Sensors* 23.10 (2023). DOI: 10.3390/s23104979.

[28] Feng Luo et al. “Threat Analysis and Risk Assessment for Connected Vehicles: A Survey”. In: *Security and Communication Networks* (Sept. 2021), pp. 1–19. DOI: 10.1155/2021/1263820.

[29] Feng Luo et al. “Threat analysis and risk assessment for connected vehicles: A survey”. In: *Security and Communication Networks* 2021 (Sept. 2021), pp. 1–19. DOI: 10.1155/2021/1263820.

[30] Georg Macher et al. “A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context”. In: Sept. 2016, pp. 130–141. DOI: 10.1007/978-3-319-45477-1_11.

[31] Georg Macher et al. “ISO/SAE DIS 21434 Automotive Cybersecurity Standard - In a Nutshell”. In: 2020, pp. 123–135. DOI: 10.1007/978-3-030-55583-2_9.

[32] Georg Macher et al. “Threat and Risk Assessment Methodologies in the Automotive Domain”. In: *Procedia Computer Science* 83 (2016), pp. 1288–1294. DOI: https://doi.org/10.1016/j.procs.2016.04.268. URL: https://www.sciencedirect.com/science/article/pii/S1877050916303015.

[33] Nancy R. Mead and Ted Stehney. “Security quality requirements engineering (SQUARE) methodology”. In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005). DOI: https://doi.org/10.1145/1082983.1083214.

[34] *MITRE ATTCK Knowledge base*. [Online; accessed 06-May-2024]. URL: https://attack.mitre.org.

[35] Dietmar P. F. Möller and Roland E. Haas. “Automotive Cybersecurity”. In: *Guide to Automotive Connectivity and Cybersecurity: Trends, Technologies, Innovations and Applications*. Springer International Publishing, 2019, pp. 265–377. DOI: 10.1007/978-3-319-73512-2_6.

[36] *neo4j search performance*. [Online; accessed 10-May-2024]. URL: https://neo4j.com/docs/cypher-manual/current/indexes/search-performance-indexes/overview.

[37] *neo4j semantic indexes*. [Online; accessed 10-May-2024]. URL: https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/overview.

[38] *neo4j website*. [Online; accessed 10-May-2024]. URL: https://neo4j.com/docs/getting-started.

[39] Ali Nouri et al. *Engineering Safety Requirements for Autonomous Driving with Large Language Models*. 2024. arXiv: 2403.16289 [cs.AI].

[40] *NVD Data Feeds*. [Online; accessed 03-May-2024]. URL: https://nvd.nist.gov/vuln/data-feeds#JSON_FEED.

[41] *NVD vulnerability database*. [Online; accessed 13-May-2024]. URL: https://nvd.nist.gov/vuln.

[42] Flavio Petruzzellis, Alberto Testolin, and Alessandro Sperduti. *Benchmarking GPT-4 on algorithmic problems: A systematic evaluation of prompting strategies*. Feb. 2024. URL: https://arxiv.org/abs/2402.17396.

[43] Christian Plappert et al. “Attack Surface Assessment for Cybersecurity Engineering in the Automotive Domain”. In: 2021. DOI: 10.1109/PDP52278.2021.00050.

[44] *Prompt Engineering Guide*. [Online; accessed 21-May-2024]. URL: https://www.promptingguide.ai/techniques/prompt_chaining.

[45] *Prompt Engineering Guide - LLM Settings*. [Online; accessed 18-May-2024]. URL: https://www.promptingguide.ai/introduction/settings.

[46] Ori Ram et al. “In-context retrieval-augmented language models”. In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 1316–1331. DOI: 10.1162/tacl_a_00605.

[47] Mera Nizam-Edden Saulaiman et al. “Overview of Attack Graph Generation For Automotive Systems”. In: 2022. DOI: 10.1109/ICCC202255925.2022.9922866.

[48] Mera Nizam-Edden Saulaiman et al. “Use Cases of Attack Graph in Threat Analysis And Risk Assessment for The Automotive Domain”. In: 2022. DOI: 10.1109/CogMob55547.2022.10118297.

[49] Christoph Schmittner, Bernhard Schrammel, and Sandra König. “Asset Driven ISO/SAE 21434 Compliant Automotive Cybersecurity Analysis with ThreatGet”. In: 2021. DOI: 10.1007/978-3-030-85521-5_36.

[50] Christoph Schmittner et al. “Using SAE J3061 for Automotive Security Requirement Engineering”. In: vol. 9923. Sept. 2016, pp. 157–170. ISBN: 978-3-319-45479-5. DOI: 10.1007/978-3-319-45480-1_13.

[51] Rakesh Shrestha Shiho Kim. “Automotive Cyber Security”. In: *Automotive Cyber Security: Introduction, Challenges, and Standardization*. Springer Singapore, 2020. DOI: https://doi.org/10.1007/978-981-15-8053-6.

[52] Fahad Siddiqui et al. “Cybersecurity Engineering: Bridging the Security Gaps in Advanced Automotive Systems and ISO/SAE 21434”. In: 2023. DOI: 10.1109/VTC2023-Spring57618.2023.10200490.

[53] Madhusudan Singh. “Cybersecurity in Automotive Technology”. In: *Information Security of Intelligent Vehicles Communication: Overview, Perspectives, Challenges, and Possible Solutions*. Springer Singapore, 2021, pp. 29–50. DOI: 10.1007/978-981-16-2217-5_3.

[54] Arun James Thirunavukarasu et al. *Large language models in medicine*. 2023. DOI: https://doi.org/10.1038/s41591-023-02448-8.

[55] *UNECE WP.29*. [Online; accessed 08-May-2024]. URL: https://unece.org/wp29-introduction.

[56] United Nations Economic Commission for Europe. *Agreement Concerning the Adoption of Harmonized Technical United Nations Regulations for Wheeled Vehicles, Equipment and Parts which can be Fitted and/or be Used on Wheeled Vehicles and the Conditions for Reciprocal Recognition of Approvals Granted on the Basis of these United Nations Regulations*. UN Regulation No. 155 E/ECE/TRANS/505/Rev.3/Add.154. Revision 3, including amendments entering into force on 14 September 2017. United Nations Economic Commission for Europe, 2021. URL: https://unece.org/.

[57] Haochun Wang et al. *HuaTuo: Tuning LLaMA Model with Chinese Medical Knowledge*. 2023. arXiv: 2304.06975 [cs.CL].

[58] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[59] Roel J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[60] Chaoyi Wu et al. *PMC-LLaMA: Towards Building Open-source Language Models for Medicine*. 2023. arXiv: 2304.14454 [cs.CL].

[61] Shiying Zhou et al. “Data Security Risk Assessment Method for Connected and Automated Vehicles”. In: 2022. DOI: 10.1109/ICITE56321.2022.10101389.

A

Appendix 1

Table A.1: Questions in the semi-structured interviews.

Id	Question
Q1	*How realistic do you find the attack scenarios presented in the artifact, particularly concerning assumptions made by the program?*
Q2	*What challenges arise when the artifact assumes specific system configurations that may not align with the actual environment?*
Q3	*In what ways can the artifact be improved to better reflect the real-world systems and configurations you typically encounter in your TARA?*
Q4	*Can you elaborate on the balance between specificity and generalization in the described attack paths? How does this impact the ability to derive actionable security requirements?*
Q5	*Based on your experience, how critical is it for the attack descriptions to provide precise details about vulnerabilities, and what are the consequences of using general terms in these contexts?*
Q6	*How coherent is the output? Does the attack steps exploit the vulnerability explained in the description? Does the requirement mitigate the attack?*

Table A.2: Questions in final evaluation for RQ3.

Id	Question
Q1	How realistic do you deem the attack to be? A realistic attack is physically possible to perform. An unrealistic attack has fictive or otherwise unplausable elements.
Q2	To which degree are the **attacks** based on reasonable assumptions of the automotive system under analysis? The **attack** may mention protocols, subsystems, etc which it assumes the system to have.
Q3	To which degree does the attack steps contain all necessary information required to perform the attack?
Q4	To which degree is the requirement correct? Correctness is evaluated by the following factors: 1. The requirement mitigates the attack. 2. The requirement can be implemented into the system.
Q5	Based on the expressed abstraction level of the requirement, to which degree is the requirement unambiguous? A requirement is considered unambiguous when it has a single interpretation.
Q6	To which degree is the **requirement** based on reasonable assumptions of the automotive system under analysis? The **requirement** may mention protocols, subsystems, etc which it assumes the system to have.
Q7	What advantages could you see in implementing the program in the TARA process?
Q8	What challenges could exist in implementing it?
Q9	How could the program be improved?

Figure A.1: Example of scenario evaluated during final evaluation

Source: KeyFob, **Dest**: KeyLessEntry, **Channel**: Keyfob channel, **Attack**: Keyfob Replay Attack

Description:

An attacker can gain unauthorized access to a vehicle by exploiting the vulnerabilities in the keyless entry system. This attack involves capturing the radio frequency (RF) signals sent by the keyfob to the keyless entry system and replaying these signals to unlock the vehicle. The attacker can perform this attack without needing to physically modify the vehicle's ECUs or wiring harnesses.
Execution:

1. The attacker needs to be physically close to the vehicle when the owner uses the keyfob to unlock the vehicle. This is necessary to capture the RF signals sent by the keyfob to the keyless entry system.
2. The attacker uses a radio frequency (RF) signal capture device to capture the RF signals sent by the keyfob to the keyless entry system. The device can be a simple software-defined radio (SDR) or a more sophisticated RF signal capture device.
3. Once the RF signals are captured, the attacker can replay these signals to unlock the vehicle. The replay can be done using the same device used to capture the RF signals or a different device capable of transmitting RF signals.
4. After unlocking the vehicle, the attacker can gain physical access to the vehicle's interior and potentially start the vehicle if the keyless start system is also vulnerable to replay attacks.

CVE References:

- 'The keyfob subsystem in Honda Civic 2012 vehicles allows a replay attack for unlocking. This is related to a non-expiring rolling code and counter resynchronization.'
- 'The remote keyless system on Honda HR-V 2017 vehicles sends the same RF signal for each door-open request, which might allow a replay attack.'
- 'The Remote Keyless Entry (RKE) receiving unit on certain Mazda vehicles through 2020 allows remote attackers to perform unlock operations and force a resynchronization after capturing three consecutive valid key-fob signals over the radio, aka a RollBack attack. The attacker retains the ability to unlock indefinitely.'
- 'The remote keyfob system on Nissan Sylphy Classic 2021 sends the same RF signal for each door-open request, which allows for a replay attack. NOTE: the vendor's position is that this cannot be reproduced with genuine Nissan parts: for example, the combination of keyfob and door handle shown in the exploit demonstration does not match any technology that Nissan provides to customers.'

Requirement:

Security requirement: The keyless entry system must implement rolling code mechanisms for keyfob authentication to prevent replay attacks.
Motivation: By using rolling code mechanisms, the keyfob and the keyless entry system can synchronize and generate a new code for each communication session. This prevents attackers from capturing and replaying RF signals to gain unauthorized access to the vehicle.

Table A.3: Papers selected for the literature review.

Paper	Type	Title
[28]	J	Threat Analysis and Risk Assessment for Connected Vehicles: A Survey
[30]	C	A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context
[32]	J	Threat and Risk Assessment Methodologies in the Automotive Domain
[50]	C	Using SAE J3061 for Automotive Security Requirement Engineering
[49]	C	Asset Driven ISO/SAE 21434 Compliant Automotive Cybersecurity Analysis with ThreatGet
[47]	C	Overview of Attack Graph Generation For Automotive Systems
[51]	B	Introduction to Automotive Cybersecurity
[35]	B	Guide to Automotive Connectivity and Cybersecurity
[48]	C	Use Cases of Attack Graph in Threat Analysis And Risk Assessment for The Automotive Domain
[43]	C	Attack Surface Assessment for Cybersecurity Engineering in the Automotive Domain
[21]	C	An Approach of Scenario-Based Threat Analysis and Risk Assessment Over-the-Air updates for an Autonomous Vehicle
[61]	C	Data Security Risk Assessment Method for Connected and Automated Vehicles
[52]	C	Cybersecurity Engineering: Bridging the Security Gaps in Advanced Automotive Systems and ISO/SAE 21434
[20]	C	Model-based Attack Tree Generation for Cybersecurity Risk-Assessments in Automotive
[5]	C	Challenges and Mitigation of Cyber Threat in Automated Vehicle: An Integrated Approach
[53]	B	Cybersecurity in Automotive Technology
[31]	C	ISO/SAE DIS 21434 Automotive Cybersecurity Standard - In a Nutshell
[27]	J	A Framework for Cybersecurity Requirements Management in the Automotive Domain
[13]	C	A Holistic Approach on Automotive Cybersecurity for Suppliers

Explanation of types:
J = Journal
C = Conference paper
B = Book

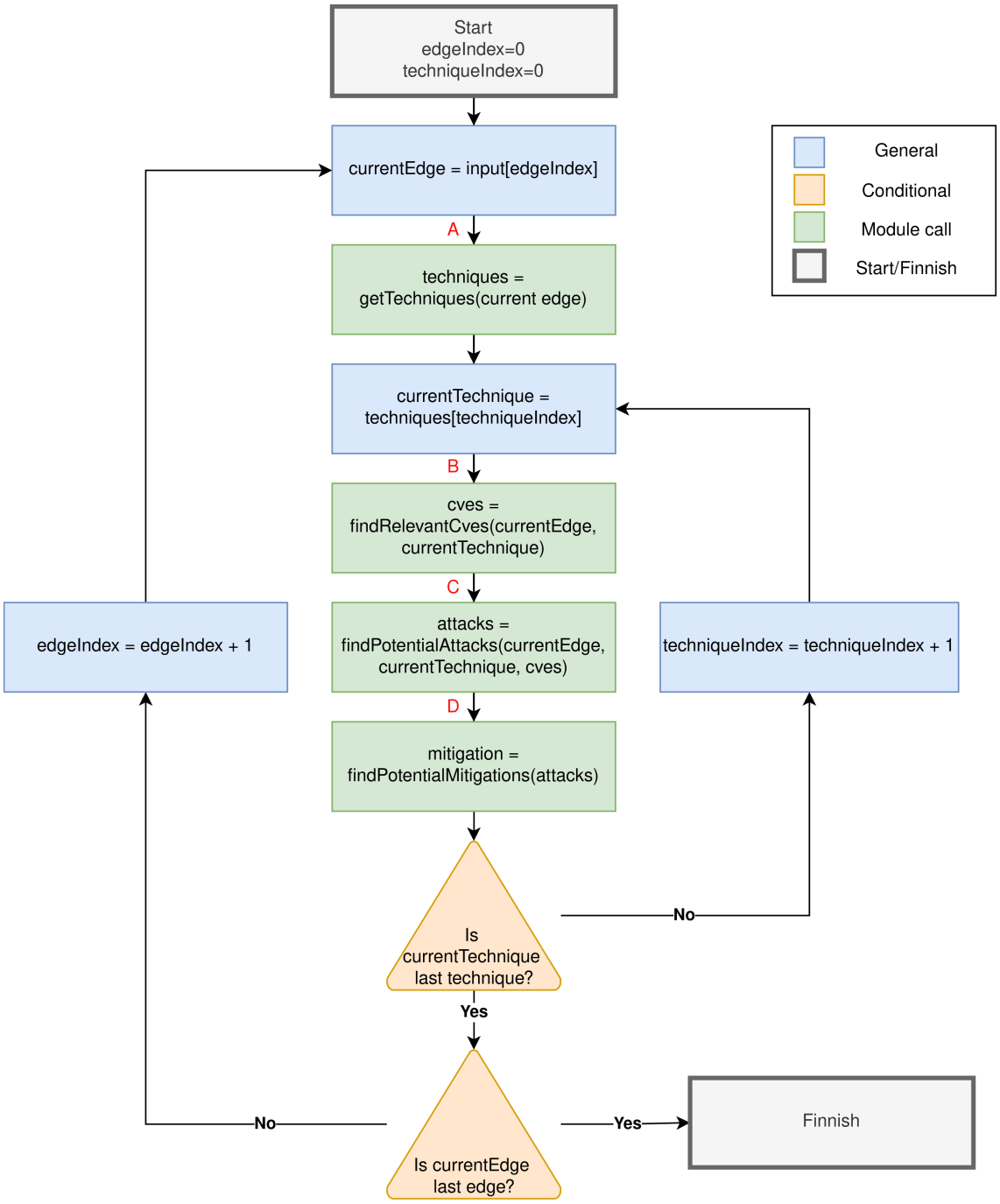


Figure A.2: A flow chart describing the execution flow of the program. Each module call is made to a separate module, where module refers to the four modules described in Chapter 5.3.2. Letters A - D denote the calls made to the modules.