DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2019

Validating vehicleLang for Domain-specific Threat Modelling of In-vehicle Networks

NEDO SKOBALJ

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Validating vehicleLang for Domain-specific Threat Modelling of In-vehicle Networks

NEDO SKOBALJ

Master in Computer Science
Date: January 30, 2019
Supervisor: Pontus Johnson
Examiner: Elena Troubitsyna
School of Electrical Engineering and Computer Science
Host company: Scania CV AB
Swedish title: Validering av vehicleLang för domänspecifik hotmodellering av fordons interna nätverk

Abstract

Due to technological developments, vehicles have gone from mechanically controlled machines to software controlled, connected machines. Increased reliance on software to operate and wireless connections to external services have exposed vehicles to a new threat, cyber attacks. Because of this threat, development of new tools has been required to improve and ease the process of securing vehicles against attacks. One such tool is vehicleLang, a probabilistic threat modelling and attack simulation language specifically designed for the automotive domain. However, vehicleLang has been developed based on information of in-vehicle networks and cyber attacks on vehicles found in scientific literature. As such, little validation has been done in terms of its real-world applicability as a threat modelling language.

The aim of this study was to validate vehicleLang by modelling an in-vehicle network from Swedish vehicle manufacturer Scania and examining the results with domain experts. Two models of the in-vehicle network were created, the first covering roughly half of the network and the second covering the entire network. Semi-structured interviews were held with experts where the first model and results from attack simulations on it were presented. This was followed by a Turing test where one expert rated solutions to attack goals by other experts and solutions created using the second vehicleLang model.

Results from the interviews and Turing test show that vehicleLang cannot accurately model in-vehicle networks, and results from attack simulations on its models are not comparable to that of domain experts. This was mainly due to vehicleLang both being unable to model certain important aspects of networks, and models lacking information which impacts networks' security. This work summarises experts' comments and presents a list of suggested changes to improve vehicleLang.

Sammanfattning

På grund av den tekniska utvecklingen som har skett har fordon utvecklats från mekaniskt styrda maskiner till mjukvarustyrda, uppkopplade maskiner. Ett ökat beroende på mjukvara för att uppfylla sina funktioner, och trådlösa uppkopplingar till externa tjänster har utsatt fordon för ett nytt hot, cyberattacker. Som en konsekvens av detta hot har utveckling av nya verktyg krävts för att förbättra och underlätta processen av att säkra fordon mot cyberattacker. Ett sådant verktyg är vehicleLang, ett probabilistiskt hotmodellering-och attacksimuleringsspråk utvecklat specifikt för fordonsindustrin. Dock har vehicleLang utvecklats baserat på information om fordons interna nätverk och cyberattacker på fordon som funnits i vetenskaplig litteratur. Därav har lite validering gjorts för att styrka dess verkliga tillämplighet som ett hotmodelleringsspråk.

Syftet med denna studie var att validera vehicleLang genom att modellera ett internt nätverk från den svenska fordonstillverkaren Scania, och undersöka resultaten med hjälp av domänexperter. Två modeller av det interna nätverket skapades där den första täckte ungefär hälften av nätverket, och den andra täckte hela nätverket. Semistrukturerade intervjuer hölls med experter där den första modellen och resultat från dess attacksimulationer presenterades. Detta följdes upp med ett Turing-test där en expert bedömde lösningar till attackmål av andra experter och lösningar som skapades med hjälp av den andra modellen.

Resultaten från intervjuerna och Turingtestet visar att vehicleLang inte kan modellera fordons interna nätverk med en tillräcklig nivå av noggrannhet. Dessutom är inte resultat från attacksimulationer på dess modeller jämförbara med domänexperters analyser. Detta berodde främst på att vehicleLang både saknar möjligheten att modellera viktiga aspekter av fordons nätverk, och att modeller saknar information som påverkar nätverkens säkerhet. Detta arbete sammanfattar experternas kommentarer och presenterar en lista över förslag av förändringar som kan förbättra vehicleLang.

Contents

Chapter 1

Introduction

This chapter serves as an introduction to the area of vehicular threat modelling and how it relates to the report. The aim is to give the reader context to the report's problem statement and its relevance in the field. Moreover, the limitations, objective and outline of the report are presented.

1.1 Industry Demand

Developments in both the automotive industry and the IT industry have slowly turned vehicles from mostly mechanically controlled entities to machines controlled by software. Modern cars are controlled using tens of millions lines of code [1]. This developmental direction is showing few signs of stopping as the automotive industry is pushing for autonomous driving and more network connected vehicles in the future. The reason behind this development has been to provide customers with more features, increased efficiency and improved vehicle safety.

Consequently, this increased reliance on software to operate vehicles has exposed them to possible cyber attacks. As newer, and future, cars are connected to the Internet and other Wide Area Networks, hackers may not even require physical access to vehicles to perform cyber attacks. This was demonstrated in the famous case of hackers remotely taking control of an unaltered 2014 Jeep Cherokee [2]. One can easily imagine scenarios that lead to physical and societal damage if attempts are not made to thwart cyber attacks on vehicles. Therefore, it is in the automotive industry's and society's best interest to prevent such attacks.

Other companies from a wide range of industries also have an interest in protecting themselves from cyber security threats. From this interest, a de-

mand has arisen for techniques and tools which facilitate security auditing of systems. One area of research which has seen rapid expansion is threat modelling. Threat modelling is the process of modelling a system, such as an IT infrastructure, and identifying potential threats and vulnerabilities in the system while assuming the perspective of a potential attacker. The Royal Institute of Technology (KTH) has conducted research in this area and from it, the company foreseeti was founded in 2014 [3]. securiCAD® is a modelling tool by foreseeti which enables user-friendly threat modelling of IT infrastructures by providing a drag-and-drop interface, and the ability to run attack simulations on models.

As a result of the automotive industry's increasing desire to prevent cyber attacks and foreseeti's tooling and knowledge around threat modelling, a joint research project between foreseeti, F-secure, KTH, Scania and Volvo Cars has been initiated. The project is named THREAT MOVE and has the overarching goal of developing an open-source domain-specific threat modelling language for the automotive industry [4]. A previous master's thesis project completed the development of the first version of this language, named vehicleLang [5]. However, vehicleLang is mostly a theoretical construct at the moment and has not been used to model an actual in-vehicle network, which means that its real-world application as a threat modelling language has not been established. This report will be a continuation of the previous work with the aim of validating vehicleLang by modelling the in-vehicle network of a Scania truck.

1.2 Objective

The desired outcome of this report is to validate vehicleLang to ensure that it is applicable as a domain-specific threat modelling language used for security auditing/analysis of in-vehicle networks.

1.3 Problem Statement

The aim of the report is to answer the problem statement below:

Can vehicleLang be used to accurately model the in-vehicle network of a truck such that the model, combined with attack simulations, can be used for security auditing of the network?

This problem statement can be broken down into two sub-questions:

1. *Can vehicleLang be used to accurately model the in-vehicle network of a truck?*

2. *Can models of in-vehicle networks created using vehicleLang, combined with attack simulations, be used for security auditing?*

1.4 Delimitations

The aim of the report is to validate vehicleLang, therefore no other threat modelling language applicable to the automotive domain will be considered.

The study is limited to the in-vehicle network of a Scania truck. This means that parts of the network such as the infotainment system and vehicle-to-everything (V2X) communication are excluded from the modelling process.

The probabilistic aspect of vehicleLang, namely *time to compromise* values (explained in Section 2.2.3), will not be considered in the validation as this is not fully implemented at the time of writing.

1.5 Outline

The rest of the report is outlined as follows. Chapter 2 presents the theory and knowledge necessary to follow the rest of the rapport. This includes an overview of the technologies commonly found in vehicles, the theory behind vehicleLang and previously published works related to the field. Chapter 3 describes and discusses the methodology used in the study. Chapter 4 presents the results of the study, followed by Chapter 5 which analyses and discusses the results and relates them to the problem statement. Chapter 6 summarises the conclusions of the report.

Chapter 2

Background

The aim of this chapter is to give the reader an overview of the theoretical background of the report. This is to explain the context of the report and give the reader the necessary knowledge to interpret the rest of it.

2.1 In-vehicle Networks

To better understand vehicleLang and the modelling process in this report, the reader needs to have a basic understanding of in-vehicle networks. This section will present a brief overview of the components, protocols and typical architecture commonly found in vehicles. As there are too many aspects of in-vehicle networks to cover in this report, only the parts relevant to the vehicleLang models will be presented.

2.1.1 Electronic Control Units

The computerised components of in-vehicle networks that execute code are known as Electronic Control Units (ECUs). These ECUs are microprocessor-based and are found in every modern vehicle. The number of ECUs in a vehicle depends on the year it was manufactured and if the vehicle is a low-, mid- or high-end model. Newer and higher-end vehicles tend to have more ECUs since they provide more features. On average, a modern vehicle has around 100 ECUs in its internal network [1].

An ECU is generally assigned one function or a subset of a vehicle's functions to execute. For example, one ECU may be associated with the braking system of a vehicle where it provides Anti-Lock Braking (ABS) functionality. Another ECU may control the amount of fuel being injected into the engine

[6]. ECUs can receive data necessary for performing their functions from sensors in the vehicle. Following the ABS example, the ECU may receive data from a sensor measuring the speed at which the wheels are rotating. ECUs also communicate with other ECUs as many functions in vehicles work together or rely on each other [7]. ECUs and the software they execute, commonly referred to as firmware, are either created by the vehicle manufacturer or purchased from third party suppliers. Naturally, exactly how the ECUs are divided, what sensors they receive data from and if they are designed in-house or not varies depending on the manufacturer.

There is a certain type of ECU known as a Gateway ECU. This type differs from the ones previously discussed in that they function as a bridge/gateway connecting several vehicle networks together, similar to a router in a WAN. To perform this function, Gateway ECUs generally have more computational power available since they provide more complex functions, such as a firewall or an Intrusion Detection and Prevention System (IDPS) [6].

2.1.2 Bus Systems

ECUs need to communicate with each other to perform their functions correctly and there are different technologies available to facilitate this communication. The most common and widespread is the Controller Area Network (CAN) which was introduced in 1991 [8]. CAN is a standardised (ISO 11898) bus protocol meaning that ECUs connected on a network will broadcast their messages to all other ECUs. It is the receiving ECUs' responsibility to determine if a received message should be processed or discarded, which is done using predefined headers in the CAN-protocol. This type of communication allows for simple adding of new ECUs to a network and, theoretically, no limit to the number of ECUs that can be connected. However, in practice the load a high number of ECUs places on a CAN-bus will lead to problems such as increased latency between messages. This is partly because CAN assigns a priority to each message which suppresses lower priority messages from being sent [8].

Other bus technologies with different bit rates or transmission modes are available. Some examples are LIN-buses, MOST-buses and FlexRay. These can work as alternatives to CAN in some cases by providing functionality more suited for specific use cases, sometimes at a lower price [8].

2.1.3 On-board Diagnostics

Vehicles' ECUs have the ability to log data in memory related to errors or faults while they operate which can later be used to diagnose problems. ECUs also have the ability to provide data such as values measured while operating. These functions make up vehicles' on-board diagnostics (OBD) systems and can be used to both repair vehicles and make sure they comply with legislation. Most countries have legislation which requires manufacturers to provide this functionality to the public via standards. The standards will specify a universal port to access the network and a diagnostics communication protocol. Some examples are OBD-II in the US and EOBD in Europe [8].

One diagnostics communication protocol which is commonly implemented in ECUs is Unified Diagnostic Services (UDS), specified in the ISO 14229 standard [9]. The word "unified" indicates that it is an international standard that is not tied to any specific manufacturer/company. UDS works on the application layer and provides functions such as reprogramming of and reading/writing data to ECUs, as well as the previously mentioned common diagnostic services [9].

2.1.4 Architecture

Because of the described attributes of bus systems and the fact that there are different technologies available, the typical network architecture consists of several buses connected to a central gateway. Gateways may also be used to create sub-networks within the network [10, 11, 6]. Figure 2.1 is a simple example of this type of architecture. Note that Figure 2.1 is not the architecture modelled in this report.

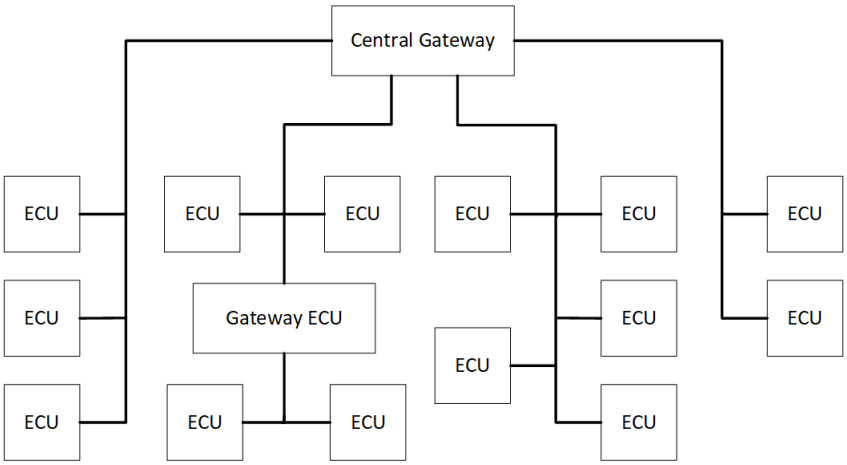

Figure 2.1: *Typical network architecture found in modern vehicles*

The central gateway is responsible for routing messages between the different buses. With this approach, ECUs can communicate with other ECUs connected on a different bus even if the other bus uses a different protocol. The load of the network is also distributed across the different buses compared with having all communication occur via a central bus system. Another benefit of this approach is that it creates a separation of concern in the network [6].

2.2 Threat Modelling

As the report aims to validate vehicleLang it is necessary for the reader to have an understanding of probabilistic threat modelling, vehicleLang and the meta-language used to create it. First, a description of common characteristics of probabilistic threat modelling and attack simulation languages is given. Then, an overview of the Meta Attack Language, its formalism and syntax are presented followed by an overview of vehicleLang.

2.2.1 Probabilistic Threat Modelling and Attack Simulations

As previously mentioned, threat modelling is the process of modelling a system and identifying potential threats and vulnerabilities in the system, while assuming the perspective of a potential attacker. The results of the threat modelling process give feedback to system designers about potential threats which allows them to assess and mitigate them. There are different methods that facilitate threat modelling, some require manual analysis of models by security

experts while others utilises tools which automate the analysis. The category vehicleLang belongs to is the latter. Different threat modelling methods applicable to the automotive domain are discussed in Section 2.3.1.

vehicleLang is categorised as a probabilistic threat modelling language facilitating its analysis of models by performing attack simulations on them [5]. The results of the simulations will produce one or several sequences of steps an attacker could take to compromise some resource/asset in the system, known as an attack path. Generally, attack paths by themselves do not provide sufficient security assessment of systems. The probabilistic aspect of these types of languages is introduced as a way to quantify the threat level of attacks and systems' security [5, 12, 13]. Exactly how the probabilistic aspect is implemented varies between languages. Some examples will be given in Section 2.3.2 where other probabilistic threat modelling languages are discussed.

vehicleLang and its meta-language introduce probability by assigning a probability distribution to every attack step [5, 14]. These distributions represent the probability that an attacker successfully performs the attack step as a function of time invested. The more time an attacker spends trying to perform an attack, the higher the probability of success. The attack simulations will use these distributions to generate an estimation of the time it will take to perform an attack step, known as the *local time to compromise* ($\texttt{TTC}_{\texttt{loc}}$). The sum of all $\texttt{TTC}_{\texttt{loc}}$ in an attack path gives the *global time to compromise* ($\texttt{TTC}_{\texttt{glob}}$), which can be used to compare different attack paths and systems as previously mentioned.

One of the main goals of probabilistic threat modelling languages is to have security expertise and analysis built into the language itself. This is to enable users without security expertise to model large-scale systems and receive accurate threat assessments of them. There are two main reasons behind this goal. The first being that cyber security experts are expensive. The second being that it is difficult to perform manual analysis of large systems especially when taking into account how changes to the system will impact its security [14, 12].

2.2.2 The Meta Attack Language

The Meta Attack Language (MAL) is a probabilistic threat modelling meta-language that can be used to specify and generate domain-specific threat modelling languages [14]. These languages can then be used to model systems in the specific domain and perform attack simulations on models. Using this approach, different domain-specific languages can be created from the same

meta-language. This allows the process of creating a new domain-specific language to mainly focus on gathering information and specifying the attack logic, assets and their required information instead of software development.

The underlining basis for the MAL is a mathematical formalism which will now be described. This description is adapted from the one found in the original paper presenting the MAL [14]. For a more detailed explanation of the formalism behind the MAL, refer to the original paper.

To enable threat modelling there must be a way to define and refer to entities in a model. These entities are called *objects* or *assets* in the MAL, the rest of this report will refer to them as assets. Depending on the domain that is being considered, assets can represent many different things. Examples from the automotive domain could be `steeringECU` or `drivetrainCAN`. All assets are partitioned into a set of *classes*, for example:

$$\texttt{steeringECU} \in \texttt{ECU}$$

and

$$\texttt{drivetrainCAN} \in \texttt{VehicleNetwork}$$

Every class has a set of attack steps associated with it which every asset inherits. These are denoted as `Asset.AttackStep`, for example `steeringECU.shutdown` refers to the `shutdown` attack step on the asset `steeringECU`.

Assets by themselves will not enable fruitful modelling, there must also be a formalism for connecting assets with each other. This is done with a *link* relationship defined as a binary tuple of assets.

$$\texttt{connected} = (\texttt{steeringECU}, \texttt{drivetrainCAN})$$

This indicates that there is a `connected` link between the `steeringECU` and `drivetrainCAN` assets. A link relationship only represents a link between two assets. Therefore, links are partitioned into *associations* such that two links creating two pairs of assets of the same classes are part of the same association. For example, the two links $(\texttt{steeringECU}, \texttt{drivetrainCAN})$ and $(\texttt{heaterECU}, \texttt{climateControlCAN})$ are part of the same association since they are links between the same classes.

The classes in associations play different *roles* which is the way the opposite end of an association is referred to, denoted as `Asset.Role` or `Class.Role`. For example:

$$\texttt{steeringECU.vehicleNetworks} = \{\texttt{drivetrainCAN}\}$$

`vehicleNetworks` is the role of `drivetrainCAN` with respect to `steeringECU`. Note the use of set notation as multiple assets can play the same role in associations, such as one ECU being connected to multiple CAN-buses.

To enable navigation over associations, attack steps are connected to each other using directed edges. The starting point of the edge refers to an attack step on a class, and the end point of the edge refers to an attack step of the associated class. For example, an edge `e` could be described as:

$$\texttt{e} = (\texttt{ECU.access}, \texttt{ECU.vehicleNetworks.access})$$

This describes that if an attacker is able to obtain full access on an `ECU`, they can then attempt to gain access on a `VehicleNetwork` asset it is connected to. Note that the two `access` attack steps are different despite having the same name.

A probability distribution is defined with every attack step which is used to indicate the difficulty for an attacker to perform a certain step. This is defined as $\phi(\texttt{AttackStep}) = \texttt{P}(\texttt{TTC}_{\texttt{loc}}(\texttt{AttackStep}) = \texttt{t})$. Attack steps may also be of two different types, `AND` or `OR`. An attack step of type `AND` denotes that *all* parent steps, steps which lead to this attack step, must be compromised before the attacker can attempt this step. Conversely, `OR` types only require that *one* parent step is compromised.

Lastly, classes can have *defences* defined on them which can have one of two states, `TRUE` or `FALSE`, denoting if they are enabled or disabled. Defences are described as parents of `AND` attack steps that prevent steps from being performed if they are `TRUE`. For example, if

$$\texttt{gatewayECU.firewall} = \texttt{TRUE}$$

and there is some edge `e` such that

$$\texttt{e} = (\texttt{gatewayECU.firewall}, \texttt{gatewayECU.forwarding})$$

then the firewall on the ECU is preventing the attacker from attempting the `forwarding` attack step.

The described formalism can be implemented in several different programming languages, for example, Java or Python. These implementations are then used for the execution of attack simulations and calculation of $\texttt{TTC}_{\texttt{glob}}$ values. However, the creators of the formalism knew that requiring users to implement

it using available programming languages was unsustainable in terms of user-friendliness. Therefore, they created the MAL which allows users to specify the formalism with a simpler specification. This specification is then taken as input to a MAL-compiler which generates the equivalent Java code.

What follows is a brief overview of the MAL syntax. Again, refer to the original work [14] for a more in-depth explanation. The code snippet below is an example of an asset definition with some attack steps:

```
asset ECU {
   | connect
     -> access
   & uploadFirmware
     -> shutdown
   # firmwareValidation
     -> uploadFirmware
}
```

This example defines an asset called `ECU` with two attack steps, `connect` and `uploadFirmware`, and one defence `firmwareValidation`. The arrow (`->`) indicates the attack steps which compromise of the current attack step will lead to. The symbol `|` defines the attack step as an `OR` step, the symbol `&` defines an `AND` step and the symbol `#` defines a defence.

Similar to many object-oriented languages, MAL supports inheritance of assets:

```
asset GatewayECU extends ECU {
   | access
     -> networks.eavesdrop
}
```

The asset `GatewayECU` will inherit all the attack steps defined in `ECU`, as well as having its own attack step `access`. The step `networks.eavesdrop` makes use of associations and indicates that compromise of `access` will lead to the attack step `eavesdrop` on all networks connected to this `GatewayECU`.

An example of how to define associations in MAL is shown on the next page:

```
associations {
   GatewayECU [trafficGatewayECU] * <- Connection
   -> * [networks] VehicleNetworks
}
```

This code snippet defines a `Connection` association between the assets `GatewayECU` and `VehicleNetwork`. The roles of the assets in this association are defined inside the brackets (`[]`), and the stars (`*`) indicate the multiplicities of the asset similar to UML class diagrams. In this case they indicate a many-to-many association.

2.2.3 vehicleLang

vehicleLang is a probabilistic threat modelling language created specifically for the automotive domain using the MAL [5, 15]. As such, vehicleLang contains definitions of assets, associations, attack steps etc. found in the automotive domain.

The design process of vehicleLang began with an extensive domain survey of assets, possible attacks, defence mechanism etc. in the automotive domain. Using the results of the domain survey, vehicleLang was implemented using the MAL specification. Another MAL-defined language, coreLang, was used as the basis for vehicleLang as it contains the core assets needed to model an IT infrastructure. This enabled the designer of vehicleLang to reuse some of coreLang's assets through inheritance, thus adapting them to the automotive domain. Figure 2.2 below shows an overview of the assets and associations present in vehicleLang, and the extended coreLang assets.

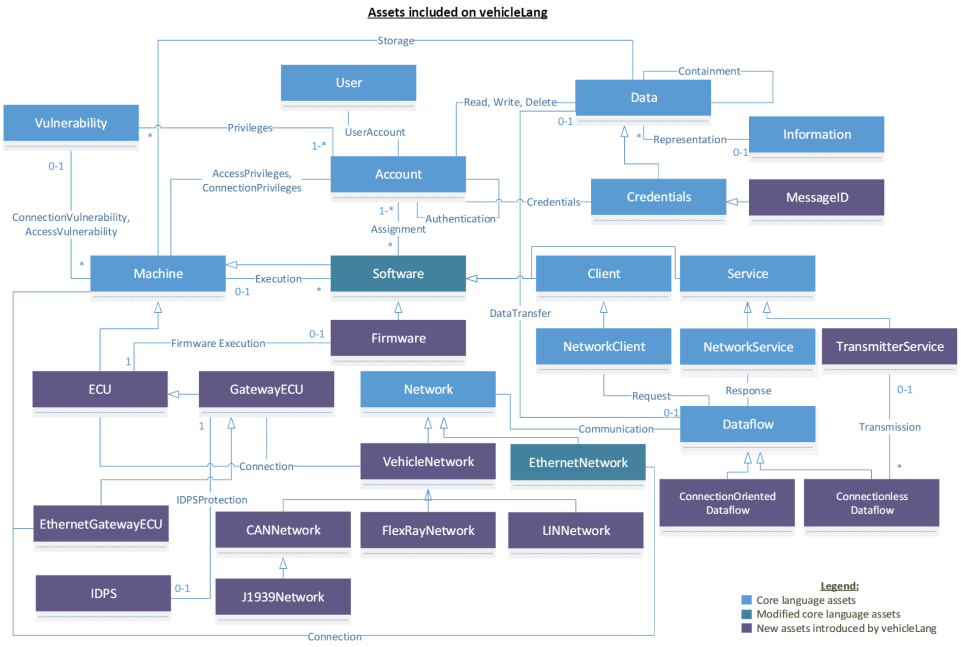

Figure 2.2: *Assets and associations implemented in vehicleLang. Figure provided by Sotirios Katsikeas, creator of vehicleLang.*

Two different types of tests were used to evaluate vehicleLang. Firstly, unit tests were used to verify the correctness of the implementation. The implementation was also cross-checked with another developer who worked with a different MAL project. The second type were integration tests which aimed to test the connection between assets and attack steps. The test cases for the integration tests were based on the attacks found during the domain survey. This was to ensure that the results produced by vehicleLang were feasible and corresponded with the literature.

An interview with a domain expert from the automotive industry was conducted to validate vehicleLang. During the interview, the language and some test cases were presented to the expert at which point the expert gave the designer feedback. This was followed by a brainstorming session were all parties participated in the implementation of real-world test cases, the results of which were evaluated by the domain expert.

The development process of vehicleLang excluded the implementation of correct probability distributions and `TTC` values, which was left for future works. According to the creator of the language, the expected time to perform the attacks found in the literature has not been well researched and was

therefore outside the scope of the project. As a consequence, this study will not take the probabilistic aspect of vehicleLang into account when validating the language.

2.3 Related Work

This section aims to give an overview of previous works published in the domains related to the report. As the report relates to several separate domains each will be presented separately.

2.3.1 Vehicular Threat Modelling

As the report evaluates a threat modelling language specific to the automotive domain, other works concerning the application of threat modelling tools and frameworks to this domain are relevant.

Several papers have been published with the goal of creating a threat modelling framework specifically designed for the automotive industry. Papers most relevant to this report are ones presenting vehicleLang and its functions [5, 15], which were discussed in Section 2.2.

Other works have taken a different approach and tried to adapt existing threat modelling tools to the automotive domain. Ma & Schmittner [16] customised the Microsoft Threat Modeling Tool to make it viable for use in the automotive domain using only built-in functionality. The tool enables threat modelling by allowing users to create Data Flow Diagrams (DFD) of a system which the tool then automatically analyses using the STRIDE methodology. They validated their work by applying the tool to an automotive cockpit unit demonstrating that existing tools for threat modelling are a viable option for use in the automotive domain. A similar approach was taken by Karahasanovic, Kleberger & Almgren in [17]. They modelled the interior light application of a vehicle using the Threat Modeling Tool, but instead used an open-source automotive threat modelling template compatible with the tool. The template provided elements to the DFD creation related to components of vehicles, messages sent in vehicles, attack methods which exploit found threats etc. They validated the threats found using the tool by testing them in a simulated vehicle environment, which proved successful.

However, Microsoft's Threat Modeling Tool only performs static analysis of the DFDs and suggests if threats *may* be present and how one could prevent/mitigate them. Unlike vehicleLang it does not take into account countermeasures already in place and how difficult an attack step is to perform. There

are also no papers published, as far as the author is aware, that apply the Threat Modeling Tool to an entire, or majority of a vehicle network.

Published works have not only focused on adapting threat modelling tools, but methods and development processes as well. In the previously mentioned paper by Karahasanovic, Kleberger & Almgren [17], they adapted the Threat Agent Risk Assessment (TARA) method to the automotive industry with the help of domain experts. The new method was applied to a vehicle and the results were validated with domain experts. Schmittner, Ma, Schoitsch & Gruber [18] applied two known safety and security analysis methods, Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) and Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS), to a case study of a cyber-physical system in the automotive domain. They found that both methods had the potential to solve safety and security challenges present in cyber-physical systems. van Winsen [19] created a composite threat model by performing a systematic literature review of threat models applied to the automotive industry, such as DFD with STRIDE, the ISO 26262 standard and the composite threat model by the National Highway Traffic Safety Administration. The proposed threat model was evaluated through interviews with a cyber security systems architect at a major European truck manufacturer.

Despite these approaches possible benefits during development, they are designed to be used by cyber security experts for manual analysis of systems. This process is time consuming, costly and makes the analysis highly dependent on the experts' knowledge, experience and opinions.

2.3.2 Threat Modelling Languages

vehicleLang is a probabilistic threat modelling and attack simulation language. Therefore, works covering other languages of the same type are relevant to the report.

The most relevant papers in this area are ones which have laid the foundation for vehicleLang's creation, namely CySeMoL [12], P^{2}CySeMoL [20], pwnPre3d [13], securiCAD® [21] and the Meta Attack Language (MAL) [14]. CySeMoL, The Cyber Security Modeling Language, is one of the first modelling languages able to produce attack paths with an estimated probability of success from models of enterprise-level system architectures. CySeMoL differs from other threat modelling languages in that it uses a model-driven approach that does not require the user to have cyber security expertise. The goal of CySeMoL was for it to perform analyses of models equivalent to that of a domain expert. To validate this, a Turing test was employed where do-

main experts were tasked with rating analyses by other domain experts and CySeMoL. The rating expert was unaware of which analyses were done by CySeMoL. They were also tasked with rating analyses from novices in the field to ensure that they could differentiate reasonable analyses from less reasonable ones. The results showed that CySeMoL's analyses were comparable to that of a domain expert.

P^{2}CySeMoL improved on CySeMoL by adding new assets usable for modelling and changing the probability of successful attacks to a function of the time spent by an attacker. Generally, the more time an attacker spends trying to perform an attack the higher the probability is that the attack is successful. CySeMoL had based its probability on how likely a professional penetration tester would be in successfully performing an attack given one week. pwnPr3d can be seen as the next evolution of P^{2}CySeMoL as it provides much of the same functionality while using a closed meta-modelling architecture. pwnPr3d defines its classes using a layered approach as opposed to a flat one as in P^{2}CySeMoL. This makes pwnPr3d more flexible, making it easier to introduce new assets while reducing complexity. All the previously mentioned research has now been developed into a commercial product known as securiCAD®.

The one common drawback of all these threat modelling and attack simulation languages is that they are hard coded. This makes them both cumbersome to modify and inflexible in terms of adapting them to threat modelling of new domains. The MAL is the proposed solution to these problems as it allows the creation of new domain-specific languages without much software development.

2.3.3 Cyber Attacks on Vehicles

As the report's main field of study is vehicular threat modelling, there is an inherent need for the reader to have an understanding of cyber attacks on vehicles. This gives context as to what vehicular threat modelling aims to prevent and why.

The work of Koscher *et al.* [7] is often cited in the discussion of cyber attacks on vehicles. In their work, they outline their process of hacking two 2009 cars of the same model enabling them to perform several different cyber attacks. These attacks ranged from benign such as displaying an arbitrary message on the radio, to safety critical such as disabling the brakes or stopping the engine. These attacks were tested with the car placed on stands with the wheels removed and while driving on a closed course. They argued that

these types of attacks were not specific to the vehicles tested, but that they are possible as a result of the network architecture common in modern vehicles.

Their work, and others', received criticism for assuming an attacker was able to physically connect to the target vehicle's internal network. Critics pointed out that an attacker with physical access to the vehicle could perform attacks other than cyber attacks, such as cutting wires. Koscher *et al.* [22] responded to this criticism by systematically analysing the external attack surface of vehicles. They demonstrated several remote attacks which gave them complete control over a vehicle's systems. This proved that cyber attacks on modern vehicles do not require physical access to be performed.

However, cyber attacks on vehicles were not in the public eye until the paper commonly known as "The Jeep Hack" was published by Miller & Valasek [2]. In their work, they outline how they were able to gain remote control of an unaltered 2014 Jeep Cherokee using a vulnerability in its internet-connected infotainment system. Their work generated public attention as Fiat Chrysler Automobiles recalled 1.4 million vehicles as a result of it. There are many other examples of published papers which discuss/demonstrate cyber attacks on vehicles, such as [23, 24, 25, 26].

There have also been papers published that focus on cyber attacks on trucks and other heavy vehicles. The work of Burakova, Hass, Millar & Weimerskirch [27] exploits the SAE J1939 standard, commonly used in heavy vehicles, to perform safety-critical attacks on a school bus and a semi-tractor. They argue that since these attacks focus on vulnerabilities in the J1939 protocol they can be implemented on a wide range of heavy vehicles. Similarly, Murvay & Groza [28] demonstrate several cyber attacks that can be performed while still conforming to the J1939 standard. They used industry-standard CAN simulation software to demonstrate and validate their suggested attacks.

Chapter 3

Methods

This chapter presents the methodology used to examine if vehicleLang is applicable for threat modelling and security auditing of in-vehicle networks. The process of modelling Scania's in-vehicle network is presented, followed by a description of the methods used to validate vehicleLang. Motivations for the chosen methods are also presented.

3.1 Modelling Process

This section describes the process of creating two different models of the given in-vehicle network. As both models share a similar process, the general process is presented first followed by descriptions of each model and their processes. Exact low-level details about the models are omitted as the specification of the network is protected by confidentiality.

3.1.1 General Process

Scania provided a high-level diagram of the in-vehicle network used in their vehicles. The network used in actual vehicles may be modified to a small extent depending on the specifications provided by customers. However, for the purpose of this study, the diagram was sufficient as it was an accurate representation of vehicle manufacturers' in-vehicle networks.

The provided diagram was then used as a basis for the vehicleLang models that were created. Specifying vehicleLang models was done in JSON-format following a predetermined syntax. Models in this format can be used as input to a simulation tool that performs attack simulations and generates a visual representation of attack paths. This tool was provided by foreseeti.

During the modelling process, continuous correspondence was kept with the creator of vehicleLang and supervisor at Scania. The documentation of vehicleLang was lacking in some areas and spread out in different documents so correspondence with the creator was ongoing to ensure the language was used correctly. In some cases, the diagram representing the in-vehicle network provided by Scania lacked details necessary for the modelling process. Questions of this nature were directed to the supervisor at Scania and other employees to make sure the vehicleLang model was an accurate representation of the network. The diagram also did not contain any information about which ECUs communicated with each other which is an important aspect of vehicleLang models. This information was gathered using internal Scania tools which provided exact details about ECUs' messages.

3.1.2 First Model

Following the process described in the previous section, a first model of the in-vehicle network was created. This model contained half of the buses connected to the central gateway and the ECUs connected to those buses. This can be seen as approximately half of the in-vehicle network.

Hardware in the network was modeled using the corresponding assets found in vehicleLang such as ECUs being `ECU` assets, gateways being `GatewayECU` assets, CAN-buses being `CANNetwork` assets and so on. Communication in the network was modeled as follows:

- Every `ECU` was associated with its own `Firmware`, `VehicleNetworkReceiver` and `TransmitterService` assets.
- Every ECU's `TransmitterService` was associated with its own `ConnectionlessDataflow` asset.
- Every `VehicleNetworkReceiver` was associated with every `ConnectionlessDataflow` asset it should receive messages from.
- `ConnectionlessDataflow` assets were associated with every `CANNetwork` asset where their messages/data were suppose to be sent.
- ECUs connected on several CAN-buses had one pair of `TransmitterService` and `ConnectionlessDataflow` assets for each CAN-bus they were connected to. This was to create a separation of messages sent by an ECU on different buses.

The attacker's entry point to the network was a connection to the vehicle's OBD-II port. This was chosen as the entry point as it is a common entry point used for attacks discussed in the litterateur [10, 7].

Once this model was created, a peer review session was held with the creator of vehicleLang to evaluate if the language had been used as intended and modelled aspects of the network correctly. A diagram of the model was created to ease presentation of the model. This diagram, the vehicleLang model and the attack paths generated from it were subsequently used in semi-structured interviews held with Scania's cyber security experts.

3.1.3 Second Model

The second model created after the interviews built on and extended the first model. Before the second model was created, simple changes were made to vehicleLang based on feedback gathered from the interviews and further discussions with experts. These changes consisted of changing the way some attack steps were connected and adding a new defence. Verification of the changes was done using unit tests and integration tests, the same way vehicleLang's original implementation was verified. The reason behind implementing these changes was for the results of the validation process to be more substantial, especially the Turing test. Avoiding a situation where vehicleLang could have performed better in the Turing test with a few simple modifications was necessary to ensure the results were relevant.

Several aspects of the network were added to the second model. Firstly, the rest of the buses connected to the central gateway and their respective ECUs were added, with their communication being modelled the same way as in the first model. Sensors in the network were added to the model using the `PhysicalMachine` asset. UDS was added with its communication being model as such:

- Each `ECU` asset was associated with its own `UDSService` asset.
- Each `UDSService` asset was associated with its own `ConnectionOrientedDataflow` asset. This was to separate the dataflow of diagnostic messages and functional messages, and because UDS uses a client-server model for its communication.
- All `ConnectionOrientedDataflow` assets were associated with the same `NetworkClient` asset, and one `CANNetwork` which the `NetworkClient` was associated with. This was done to model that one client can communicate with many UDS services.

Some sensitive data which may be of interest to a potential attacker was added to the model using the `Data` asset. ECUs generating this data were given separate
`TransmitterService` and `ConnectionlessDataflow` assets only used by the added `Data` asset.

The attacker's entry points into the network were also changed in the second model. The `access` attack step on the `NetworkClient` was set as an entry point since UDS is an open standard [9]. The second entry point gave the attacker network layer access on one of the network's buses. This change was based on feedback received during the interviews. The experts mentioned that being able to connect to the OBD-II port in the first model implied that the attacker had physical access to the vehicle. This would, in turn, give the attacker access to simpler attack paths, such as cutting wires or immediately connecting to components. As it is not clear if such physical attacks are in the scope of vehicleLang (discussed in Chapter 4 and 5), it was deemed best to change the entry points so they did not imply physical access to vehicles.

3.2 Validation Process

The methods used to validate vehicleLang are presented in this section, including arguments for the chosen methods based on literature.

3.2.1 Interviews

The first model and two attack paths generated from it were used in the interviews held with Scania's cyber security experts. The main goal of these interviews was to not only validate vehicleLang but also understand *why* the experts had their opinions about the language. Thus, the interviews were not solely aimed at answering the problem statement of the report, but also examining if vehicleLang can be changed to perform better.

Semi-structured interviews (SSI) were chosen as the format based on several factors. The first factor was the number of security experts available. With only four experts available a structured survey was deemed inadequate as it would not provide qualitative data or enough quantitative data. Other factors were based on the work of Adams [29] who lists situations were SSIs are well suited. Two situations coincided with the context of this study, these were:

1. *"If you need to ask probing, open-ended questions and want to know the independent thoughts of each individual in a group."* [29]

2. *"If you are examining uncharted territory with unknown but potential momentous issues and your interviewers need maximum latitude to spot useful leads and pursue them."* [29]

The first situation fits well with the aim of understanding experts' opinions about vehicleLang. Open-ended questions combined with probing of their answers facilitates this sort of discussion. The second situation coincides with the fact that it is unknown beforehand both what the experts think and why. Some may have found issues with the language so it would be important to probe them on these issues and understand what they deemed to be incorrect and why.

Three SSIs lasting between one and one and a half hours were held with three different cyber security experts. Two interviews were split into two sessions because of experts' available time. All three interviews followed the same structure (complete interview guide is available in Appendix A):

1. Brief introduction to the project and aspects of vehicleLang.
2. Description of the session's aims and what aspects to disregard in their evaluation.
3. Presentation of first attack path with explanations of every attack step. This attack path showed an attacker shutting down an ECU on a bus *not* connected to the OBD-II port.
4. Questions about the expert's opinion on attack path and probing of their answers.
5. Same process as previous two steps but with second attack path. This attack path showed an attacker installing malicious firmware on an ECU on a bus *not* connected to the OBD-II port.
6. Description of vehicleLang's general modelling aspects (assets, associations, defences etc.).
7. Presentation of model using diagram.
8. Questions about the expert's opinion on the model and probing of their answers.

One expert was not able to answer questions about the second attack path due to time constraints.

Aspects which experts were asked to disregard were those considered outside the scope of the study, or not directly related to vehicleLang. One example is TTC values as these had not been implemented in vehicleLang yet. Another example is the visual aspects of the attack paths as these were generated by the simulation tool and not vehicleLang.

The creation of the interview guide followed the suggestions given by Adams in [29]. The interview guide was changed after the first interview to improve the clarity of questions and more seamlessly lead experts into expressing opinions relevant to the aim of the interviews. Changing the interview guide based on feedback is also suggested by Adams in [29].

3.2.2 Turing Test

vehicleLang falls into a category of software known as expert systems (ES) as it exhibits characteristics of these types of systems. More specifically, vehicleLang's objective is to reason and come to conclusions about complex problems in a specific domain that would generally require human expertise. Its conclusions should be comparable to that of a human expert in the same domain. These objectives coincide with the characteristics of ES described by O'Keefe and O'Leary in [30].

In [30], O'Keefe and O'Leary outline different methods for verifying and validating ES, and when it is suitable to use a method based on a few variables. One variable to consider is at what stage of the system's development the validation is being performed. As vehicleLang was not being further developed and the study focused on its validation, the system was considered to be in a late stage of development. One method they recommend for later stages of development is the Turing test.

According to O'Keefe and O'Leary, Turing tests are especially useful when:

1. "*It is difficult for the developer to assess output on a case study as correct, or otherwise, or make judgements about how it differs from a human expert.*" [30].

2. "*The system must be validated against multiple experts and there is variation between the performance of the experts.*" [30].

The first point corresponded well with this study as the author was neither a cyber security expert or experienced in the automotive domain. The second point was applicable for several reasons, namely the experts' level of experience and their backgrounds. As vehicular cyber security is a relatively new

domain, the background of experts in terms of education and previous experience can differ substantially. This can, in turn, lead to two experts working on the same problem producing two different solutions. In some cases, these two solutions may even be equally valid. Based on these two points and vehicleLang's stage of development, a Turing test was chosen as the main validation method for this study.

The Turing test was designed such that one cyber security expert would rate attack paths created by other experts and vehicleLang, without knowing who had created which paths. Six attack goals were chosen which described the component under attack and what the goal of the attack was. For example, one attack goal could state that the attack should lead to one specific ECU being unable to communicate on its bus. The attack goals were chosen in such a way that they would cover different types of components, different attacks implemented in vehicleLang and different parts of the network. The reasoning behind this was to validate as many aspects of vehicleLang as possible and to prevent a situation where one incorrectly implemented part of the language would have an influence on all the results.

Another aspect which had to be considered for the Turing test was the experts' entry points to the network and the state of the vehicle under attack. The experts were given the same entry points as the attacker in vehicleLang to ensure a fair comparison. However, there is no explicit way to specify the state of a vehicle in vehicleLang, such as current speed or state of the engine. Therefore, the second model of the network was created with the assumption that the vehicle would be standing still with the ignition turned on. This meant that things such as the defence `operationModeProtection`, which prevents ECUs from entering diagnostics mode while moving, were omitted from the model. These same assumptions about the vehicle were given to the experts when they were tasked with creating attack paths. They were also instructed that they did not have physical access to the vehicle.

Lastly, a limitation had to be placed on experts' ability to use Scania exclusive information and tools. They were instructed that they could use all information available to them as Scania employees to plan their attacks, such as how ECUs communicated or where certain information was present in the network. However, they were not allowed to use their privileges as Scania employees to perform attacks, for example, use their credentials to access cryptographic keys. The reasoning behind these limitations was based on what information was available in the vehicleLang model and how the language is intended to be used. vehicleLang is intended to be used for security analysis of systems which users develop or maintain. As such, for models to be accurate

they should contain information about the system that is not publicly available. This, in turn, means that the attacker has access to all this information in the simulated attacks. However, the attacker was, in this case, not assumed to be an insider and was therefore not given the same access privileges as an insider. Because of this, allowing experts to search for information about the network while not allowing them to use their employee credentials would promote a fairer comparison between them and vehicleLang.

Three one-on-one sessions were held with cyber security experts where they were tasked with creating attack paths for the six attack goals. The attack goals were also simulated using the second vehicleLang model. A final session was held where the attack paths were presented to and evaluated by a cyber security expert. This expert evaluator had not participated in the previous sessions. Participants of the previous sessions were asked to not discuss the attack goals or their solutions with the evaluator. This was to ensure that the evaluator had no previous knowledge of the created attack paths. Attack paths created by experts and vehicleLang were adapted into an ad-hoc common format to prevent the evaluator from distinguishing which paths were created by vehicleLang. The common format attack paths for each attack goal were presented in randomised order and the evaluator was asked to rate every attack path on a one to five Likert scale. The rating represented how much the evaluator agreed with the statement *"this attack path is reasonable and correct"*, with a rating of five indicating that they strongly agreed.

Chapter 4

Results

This chapter presents results gathered from the interviews and Turing test described in Chapter 3.

4.1 Interviews

Results from the semi-structured interviews are presented in this section. To improve readability, results relevant to the presented attack paths and model are divided into two sections.

4.1.1 Attack Paths

All three experts were asked to rate how realistic they perceived the attack paths to be. The definition of realistic given to experts was *"how well the attack steps correspond with the steps from the equivalent attack in the real world"*. They were given five rating options which were:

- Very realistic
- Realistic
- Somewhat realistic
- Not so realistic
- Not at all realistic

Two experts answered that the first attack path was "not so realistic" and one answered "somewhat realistic". Further questioning and probing of their answers showed that there were several reasons for their ratings. The attack path

used the attack step `busOffAttack` to reach the step `shutdown` on an ECU. Experts mentioned that this was generally not correct as they could not see any manufacturer designing their ECUs to shut down if put in a bus-off state (ECU stops communicating with its bus because of errors). The safety implications of such a design could be catastrophic depending on the function of the ECU. They suggested a more realistic option where ECUs in a bus-off state would still run in a "safe mode" where a minimum set of functions would be provided until the issue could be resolved.

Another aspect brought up by the experts was the implicit assumption of physical access to the vehicle as the entry point was the OBD-II port. They explained that physical access to the vehicle opened up the possibility for other attack paths which would allow the attacker to easier achieve their goal. One example that was mentioned was cutting the wires of an ECU. Physical access could also affect the attack path taken for remote attacks. For example, an attacker could connect a device to the bus which contained their target ECU. This device could then be used to communicate with the bus remotely effectively removing the steps which involved getting access to other parts of the network via the OBD-II port.

There was also a general consensus among the experts that the attack path did not contain enough information for them to give it a higher rating. One example was the exact nature of the attack step `busOffAttack`. The core principle behind this attack is to force ECUs into a bus-off state by incrementing their error counters. Incrementing an error counter can be achieved by sending error frames (a sort of error message in CAN), or tricking the ECU that an error has occurred which makes it generate error frames. How viable these two options are, according to the experts, depend on where in the network the attack is being performed and if there are any Gateway ECUs between the attacker and the target component. This is because error frames and the messages which Gateway ECUs forward are not on the same layer in the protocol stack. Error frames work on lower layers and generally do not propagate up the protocol stack. vehicleLang does not take any of these aspects into account which meant that the presented attack path did not contain this information. One expert also mentioned that the configuration of ECUs' CAN controllers, which cannot be specified in vehicleLang, could impact the viability of an attack path. Another expert's overall opinion was that the attack path could be used to discover potential attacks, but that it would still require the analysis of an expert to be deemed feasible/realistic. The main reason behind this being the lack of important information in the attack path.

The second attack path was only rated by two experts. One gave it a "some-

what realistic" rating whereas the other gave it a "not at all realistic" rating. Despite the disparity between their ratings, the two experts made similar comments about the second attack path. One point both experts brought up was the type of messages used in the attack. According to them, installing new firmware on an ECU would likely be done using UDS diagnostic messages as this is functionality provided by the standard. Diagnostic messages are not treated the same way by the network as regular messages, such as CAN messages, namely that these messages are always routed to their destination. This means that an attacker does not have to compromise Gateway ECUs between their entry point and their target since they will always forward their diagnostic messages. Therefore, if the assumption was that the attacker was using diagnostic messages the attack path would be shorter as most steps could be ignored.

A related point the experts brought up was that firmware upload via UDS is protected with a UDS service called Security Access. This is an authentication protocol which uses a challenge-response scheme with cryptographic keys to protect certain UDS services. If an attacker was to use UDS to upload malicious firmware they would have to find a way to gain Security Access, such as gaining access to the cryptographic key. Security Access is not implemented in vehicleLang.

Points made about physical access and lack of detailed information were brought up for the second attack path as well. With physical access, the attacker could physically connect themselves to the target ECU and install their own firmware thus bypassing most steps in the attack path. Details about the target ECU's hardware and architecture would be necessary for the experts to determine the difficulty of creating malicious firmware. One example that an expert brought up was that firmware for a Linux based ECU would be easier to make, because of the known architecture and available tools, compared with firmware for an ECU with obscure hardware and architecture.

4.1.2 Model

All three experts were asked to rate how accurate they perceived the presented model to be. The definition of accurate given to experts was *"how many high-level aspects of the in-vehicle network are captured by the model"*. They were given five rating options which were:

- Very accurate
- Accurate
- Somewhat accurate
- Not so accurate
- Not at all accurate

Two experts rated it as "somewhat accurate" and one expert rated it as "accurate". Generally, the experts did not find many faults with the model. Instead, things they would like to see added for better accuracy were discussed. One expert brought up many aspects of the model which were related to messages in the network, the first being a lack of specifying protocols used. Not only did the previous discussion about differentiating diagnostic messages apply here, but also other protocols such as ones used for message authentication codes (MAC), or others which could be used as defences for certain attacks. There was also no way of assigning priority to CAN messages which is a significant aspect of the protocol's functionality.

Another issue they found was that the defence `firmwareValidation` was a too broad and ambiguous term. It could include many different defense mechanisms that should not be seen as equal. One expert gave the example of a secure boot protocol which validates the boot software, not the firmware which is considered application software, and asked if that was included in `firmwareValidation`. Their suggestion was to divide it into several defences to make it less ambiguous.

The experts also found the modelling of ECUs and their firmware to be too simplistic in that it treats every ECU and its firmware the same (same assets). They argued that ECUs' hardware cannot be seen as equal since some are only a little more sophisticated than a microcontroller whereas some can be compared to PCs. This difference in hardware can give an attacker different attack capabilities once they have compromised an ECU. The difference in hardware is reflected in the software in that some ECUs have simple firmware whereas some have fully-fledged operating systems. This can have an impact on the attack surface of the firmware and the attack capabilities of the attacker once the firmware is compromised. These points also relate to the previous discussion about ECUs' architectures influencing the difficulty of creating malicious firmware. Moreover, they mentioned that two identical ECUs with the same firmware may have different functionality as the parameters of firmware are set depending on customers' orders. For example, if the customer has not ordered

autonomous driving for their vehicle then this functionality is disabled in the firmware via parameters. This cannot be reflected in the vehicleLang model.

The experts had some suggestions for things that may be interesting to add to the model/vehicleLang. One idea was being able to specify the base load of a bus. This could help indicate how sensitive buses are to extra traffic which in turn could be used to indicate how vulnerable a bus is to Denial of Service attacks. Another idea was being able to specify where components were physically placed in vehicles, or how difficult it would be to get physical access to them. This was in relation to attacks which required physical access to certain components, such as connecting to a certain ECU and updating its firmware.

4.2 Turing Test

Table 4.1 shows the results gathered from the evaluator's ratings of attacks paths.

	Expert A	Expert B	Expert C	vehicleLang
Attack 1	5	5	3	5
Attack 2	4	4	2	2
Attack 3	1	4	4	2
Attack 4	5	5	2	5
Attack 5	3	2	3	2
Attack 6	4	3	2	1
Mean	3.7	3.8	2.7	2.8
Median	4.0	4.0	2.5	2.0

Table 4.1: *Results from Turing test*

There are some notable results in Table 4.1. The first being that vehicleLang received a lower average rating than two of the experts and a lower median rating than all three experts. The second notable result is that vehicleLang's ratings are polarizing as it either received a "strongly agree" rating or "disagree"/"strongly disagree" ratings. It never received a rating of three or four. Finally, the results indicate that two of the experts performed similarly well whereas one expert performed noticeably worse. Possible reasons for this are discussed in Section 5.1.2.

Chapter 5

Discussion

Chapter 5 discusses the results presented in Chapter 4 and what conclusions can be drawn from them in relation to the report's problem statement. Discussions about the study's limitations and aspects related to ethics and sustainability are also presented. Finally, suggestions for future work are given.

5.1 Main Findings

This section highlights the main findings of the semi-structured interviews and the Turing test. The findings are critically analysed and placed in relation to the report's problem statement.

5.1.1 Interviews

Examining the comments made by cyber security experts during the semi-structured interviews shows that there are some overarching themes in their feedback. When discussing the presented attack paths and model there were four major points of discussion: lack of detailed information for each attack step, too simplistic modelling of ECUs' hardware and software, physical access to vehicles and diagnostic messages.

When discussing experts' comments it is important to view them in the context of vehicleLang's goals and indented functionality. The goal of vehicleLang is to facilitate security analysis of in-vehicle networks by performing attack simulations on high-level models of networks. The main results generated from these simulations are attack paths and the probability of an attacker successfully executing an attack path, mostly indicated by `TTC` values. Moreover, vehicleLang is a MAL-based language which can be viewed as defining

attacks steps and their connections as graphs. As such, there is an inherent limitation placed on the level of detail provided in models and attack paths. The nature of the language places its focus on *if* attack steps can be executed and *how likely* they are to be used by an attacker. *How* attack steps are performed, such as exact messages sent or timing of messages, can be seen as outside the scope of vehicleLang. This view of attacks clashes with the view cyber security experts have as they are, generally, more focused on exact details of attacks. This is not surprising as cyber security requires meticulous attention to details since even small mistakes or oversights can have major consequences. Therefore, these opposing views must be balanced as experts' feedback is used to further develop vehicleLang. It is also outside the scope of this report to decide what level of detail is sufficient for vehicleLang to serve its purpose.

Given the described context, comments made about the lack of details in attack steps may be seen as non-issues. Recall the experts' example of the attack step `busOffAttack` in the first attack path. If the goal of vehicleLang is only to show that it would be possible to perform a bus-off attack at this point in the attack path then the level of detail is sufficient. This is because no expert stated that it would be impossible to perform this step only that the possibility of success depends on the method used. However, if the analysis made by vehicleLang is to be truly comparable to that of an expert, it should be able to distinguish which method to use based on where in the network the attack is performed. In either case, adding the ability to specify how CAN-controllers are configured and how ECUs respond to these attacks would be beneficial. There is a defense present in vehicleLang named `busOffProtection` but this models a specific countermeasure to the bus-off attack found in the literature.

Comments made about the too simplistic modelling of ECUs' hardware and software are well founded and should not be disregarded. Given that differences in hardware and software have a significant impact on ECUs' security, it should not be dismissed as too detailed for vehicleLang's purpose. There also seems to be an indication that the creator of vehicleLang knew this as the documentation mentions assets such as `OperatingSystem`, `Application` and specific types of ECUs and architectures, although these are left unimplemented.

Discussions about the implication of physical access to vehicles reveal that vehicleLang has not clearly defined its scope. As the goal of vehicleLang is to model *cyber* attacks, one could assume that attacks which require physical access would not be considered. Especially as the rebuttal given by vehicle manufacturers when Koscher *et al.* [7] presented their work was that attackers

with physical access could achieve their goals without resorting to cyber attacks. However, the two papers presenting vehicleLang [5, 15] do not mention any limitations placed on attacks requiring physical access. On the contrary, there are attack steps named `physicalAccess` in several assets. Some attacks described in the reports even explicitly state that physical access to the network layer is required. The author of this report believes the creator of vehicleLang focused on modelling cyber attacks described in the literature, some requiring physical access to vehicles, without considering the implications of giving an attacker this access. Therefore, a clear limitation of vehicleLang's scope should be made and the language must be changed to reflect this limitation.

Lastly, interviews and sessions with experts where they were tasked with creating attack paths indicate that diagnostic protocols, such as UDS, have an impact on vehicles' security. This is because diagnostic protocols combined with legislation are designed to make vehicles' networks more open to the public. For example, to allow independent workshops to perform maintenance on vehicles or to ensure compliance with environmental regulations. Since diagnostic messages use a different protocol compared with functional messages, they are treated differently by the network. One example being that they are always routed to their target destination. The requirement of making the networks open to outsiders and diagnostic protocols being treated differently makes them a potential security risk, which experts are aware of. As such, this is something that should be reflected in vehicleLang.

As was mentioned in Section 3.1.3, diagnostic messages were added to the second model in the form of `UDSService` assets. This, in turn, means that experts' comments about the attack paths and model lacking diagnostic messages may be more related to the model itself, as opposed to vehicleLang. However, inspection of the `UDSService` asset combined with correspondence with the creator of vehicleLang has shown that this asset is not fully developed. The asset does not model much of the functionality available in the UDS standard, such as updating firmware. There is also no implementation of Security Access which is a critical service in terms of security. As such, experts' comments about lack of diagnostic messages cannot be opposed by simply stating that the model was missing the `UDSService` assets. The `UDSService` asset should be further developed and adding other diagnostic protocols to vehicleLang should be considered.

As an aside, other issues with vehicleLang mentioned during the interviews are not as substantial. This is because they will not require much research, verification or validation to be implemented. For example, ECUs not shutting down when put in a bus-off state was fixed before the Turing test

with the introduction of a `offline` attack step. Another example is dividing `firmwareValidation` into several defenses as it will only require new attack steps replacing it.

Not much focus has been placed on the ratings experts gave during the interviews for two reasons. The first being that the ratings are subjective and highly affected by how experts interpret questions. This is highlighted in the second attack path where two experts made the same comments but gave different ratings. The second reason is that asking experts to give ratings was used to lead into open-ended questions. This is suggested by Adams in [29] to more seamlessly enter into discussions which the interviewer is interested in.

5.1.2 Turing Test

Before discussing vehicleLang's ratings it is necessary to analyse and compare the ratings of the experts, as this is the data which vehicleLang will be compared against. The most notable result in terms of experts' ratings is that two of the experts performed similarly well whereas one performed noticeably worse. Expert C's lower average and median ratings may be a result of them having less experience with Scania's network and with cyber security. Expert C mentioned during the creation of the attack paths that, compared with the other experts, they had been employed the shortest amount of time at Scania and had the least experience working with vehicular cyber security. Assuming better knowledge of the network and more experience in vehicular cyber security should yield better ratings, it is not surprising that Expert C received the lowest ratings. The fact that the two more experienced experts performed similarly well in terms of average and median rating may indicate two things. Firstly, having more experience does, in fact, result in better ratings. Secondly, the experts are indeed relying on their experience and knowledge in their analyses as opposed to guessing.

The individual ratings on each attack for Expert A and B are similar with paths for half of the attacks receiving the same rating and the average difference between ratings being 0.83. If Attack 3 is excluded, the average difference between their ratings is 0.40. Creating attack paths with limited time, as the experts were instructed to do, is an error-prone process as it can be difficult to gather all the information needed for a thorough analysis. Small errors in attack paths can influence their ratings as they may rely on false assumptions, such as an ECU not having a certain defense mechanism. As such, the significant difference between Expert A and B's ratings on Attack 3 may be due to such errors. A rating of one, which Expert A received, also indicates that

the evaluator discovered something they knew was incorrect about the attack path. This could indicate an error in assumptions. Therefore, an argument can be made that if experts had been given more time to perform a thorough analysis, likely the case in a non-experimental setting, the average difference between their ratings would be closer to 0.40 than 0.83. In conclusion, the average and median ratings of Expert A and B should provide a suitable baseline for comparing vehicleLang to experienced experts.

One important note to make is that the ratings in Table 4.1 are based on the subjective opinion of the evaluator. Because of this, the results must be analysed to determine if the evaluator's ratings are a suitable measure for comparing vehicleLang to experts. A possible concern is that the evaluator's ratings are arbitrary, meaning that they are unable to consistently determine how reasonable and correct attack paths are. The two previously discussed points about Expert C's lesser experience and Expert A and B's similar ratings suggest that this is not the case. Assuming less experience results in more unreasonable or incorrect attack paths and vice versa, the results show the evaluator was able to distinguish reasonable/correct attack paths from unreasonable/incorrect ones. The evaluator's ratings of vehicleLang should, therefore, give a good indication of how well the language performs compared to experts with different levels of experience.

The average and median ratings of vehicleLang suggest that its generated attack paths, and thus its analysis of the network, are not comparable to that of an experienced expert. The results also indicate that its analysis is slightly worse than that of a less experienced expert. This is due to its lower median rating and because one can assume that Expert C would receive a higher average rating if they were given more time to perform their analysis. The distribution of vehicleLang's ratings coincides with the opinions gathered from experts during the interviews. Their underlying opinions were that there are aspects of in-vehicle networks that are not modelled or require more detailed modelling. This implies that what is implemented is not incorrect. With this in mind, it is not surprising that vehicleLang mostly received ratings of five, two or one meaning that it either performed very well or poorly. The evaluator explained that a rating of two meant that they could not state that the path was incorrect, but they found it difficult to determine how steps would be performed for it to be reasonable. This was mostly due to presented attack paths not considering important aspects of the network. Given that similar comments were made about vehicleLang's modelling in the interviews, half of the paths receiving a rating of two correlates well. The paths which received a rating of five may be seen as an indication that vehicleLang can perform well if the model covers

all aspects of the network with sufficient detail. However, it is unreasonable to draw such a conclusion with only two attack paths receiving a rating of five.

5.1.3 Relation to Problem Statement

Recall the first sub-question derived from the problem statement presented in Section 1.3:

Can vehicleLang be used to accurately model the in-vehicle network of a truck?

Given the results of the semi-structured interviews discussed in Section 5.1.1, the answer to this question is that vehicleLang, in its current state, does *not* accurately model the in-vehicle network of a truck. The main reasons for this are its lack of fully implemented assets representing diagnostic messages and its simplistic modelling of ECUs' hardware and software. Making defences more specific, such as dividing `firmwareValidation` into several forms of validation, should improve the accuracy of the model as well. The author has also noticed some potential inconsistencies in vehicleLang during the study, but these are small considerations compared with the previous points. If vehicleLang is further developed to include the suggested changes in Section 5.2, it may be able to accurately model in-vehicle networks.

Recall the second sub-question derived from the problem statement presented in Section 1.3:

Can models of in-vehicle networks created using vehicleLang, combined with attack simulations, be used for security auditing?

Based on the results from the Turing test, vehicleLang's models combined with attack simulations *cannot* be used for security auditing. It can in some cases perform an accurate analysis by generating reasonable and correct attack paths. However, the results of the Turing test suggest that this does not apply to a majority of cases. As such, the language's current analysis is not consistent or accurate enough to be used for security auditing of in-vehicle networks.

5.2 Suggested Changes to vehicleLang

This section presents suggested changes to vehicleLang based on the results of the validation process.

5.2.1 Diagnostic Protocols

The non-modelled aspect of in-vehicle networks with the most significant impact on security is diagnostic protocols. Therefore, future versions of vehicleLang need to implement these protocols for accurate modelling to be achieved. A good starting point may be to fully implement the already present asset `UDSServices`. This should include all functions available in the UDS standard, including the ability to protect services with Security Access, and modelling of attacks found in the literature. Support for UDS services can differ between manufacturers which needs to be taken into account. Balancing options for customising UDS services in models, such as disabling some or protection via Security Access, while keeping the modelling process scalable will be important. Next, other protocols may be research and implemented, or adding generic diagnostic messages not tied to certain protocols can be considered.

5.2.2 Hardware and Software

Modelling of ECUs' hardware and software needs to be improved to achieve better accuracy of models. As for software, a logical place to begin would be to implement the already present assets `OperatingSystem` and `Application`, which should model attack steps and defenses facilitated by their increased complexity compared with `Firmware`. modelling differences in hardware can be done in several ways. One example could be to represent different ECUs with separate assets, such as `SimpleECU` and `X86ECU` to differentiate their architectures. Another example could be assets representing architectures which are associated with `ECU` assets. Which approach will yield the best results will have to be investigated further. In either case, a thorough investigation into software and hardware commonly found in ECUs will be required. Implementing these two aspects of ECUs should enable modelling of more sophisticated ECU thus differentiating them from simpler ECUs.

Enabling modelling of differences in the configuration of ECUs with the same hardware and software may be fruitful as well. Some examples brought up by experts were configurations of ECUs' CAN-controllers and parameters of their firmware. How significant the impact of these differences in configurations is in terms of security will have to be investigated further.

5.2.3 Physical Access

As mentioned in Section 5.1.1, a clear definition must be made if attacks which require physical access are in the scope of vehicleLang. Depending on the

decision, different changes should follow. If physical access is excluded then all attacks which rely on this access need to be removed. If they are included then implementing attack paths facilitated by this access must be done, such as connecting directly to ECUs or cutting cables. In the latter case, it will likely be beneficial to add functionality specifying how difficult it would be for an attacker to achieve physical access to assets. One way of doing this could be to combine `TTC` values and defenses. For example, having defenses such as `OpenAccess`, `InsideCabin` or `InEngineBay` with appropriate `TTC` values. With this approach, vehicleLang could be used to model attacks where the attacker relies on the vehicle being left unsupervised for a certain time period.

5.2.4 Alternative Paths

When experts were tasked with creating attack paths they utilized some paths that are not considered in vehicleLang. The first being the ability to purchase spare parts, a certain ECU for example, and using that as a means to reverse engineer the software. It is currently unclear what effects, if any, the inclusion of this path would have on vehicleLang. One possibility could be that the attacker can gain `access` on the asset `MessageID` by analysing the communication of a spare part. In either case, the possibility of an attacker buying components as spare parts should be kept in mind as the language is further developed.

Another path experts utilized which is not considered in vehicleLang is social engineering. Vehicle manufacturers commonly have affiliated workshops which provide vehicle service to customers. As these workshops are affiliated with manufacturers, they may have access to information or tools which can be utilized by an attacker. One example could be workshops having access to keys used for firmware validation as they provide firmware updates to customers. These workshops provide an interesting attack path if the attacker can utilize social engineering. Especially as an attacker is more likely to have access to a workshop compared to a manufacturer's offices or factories.

5.3 Limitations

There are several limitations of the study that need to be considered. One limitation which affected most aspects of the study was that it was performed in cooperation with only one vehicle manufacturer. This limited the study to modelling of a single in-vehicle network and placed a limitation on the number of experts available for the validation process. There were two main reasons

for this limitation, the first being time available. Performing the study with two manufacturers would require roughly double the amount of work thus putting it outside the scope of a master's thesis project. The second reason is confidentiality. As Scania did not want information about their network to be shared with other manufacturers, opinions of other experts in the domain outside Scania could not be used.

Another limitation was that only one evaluator was used for the Turing test. This limitation is solely based on a limited number of experts being available. It would likely have been beneficial to use at least two evaluators to reduce the subjectivity and bias of ratings. One possible solution may have been to task one of the experts which created attack paths to instead rate attack paths. However, it was deemed more beneficial to compare vehicleLang's paths to as many experts as possible.

The results of the Turing test may have been affected by the fact that the evaluator was shown attack paths generated by vehicleLang during the interviews. Having seen examples of attack paths may have made it easier for the expert to distinguish which paths were generated by vehicleLang, thus influencing the rating they gave. However, attempts were made to limit the chance of this occurring by utilizing the ad-hoc common format for attack paths. The reason for the evaluator seeing attack paths generated by vehicleLang beforehand was a change in the report's problem statement during the study. The original problem statement included examining if vehicleLang could be further developed to better fulfill its goals. As the evaluator was a more senior expert it was deemed important to receive their input on this matter. Some time after the interviews had taken place, the problem statement shifted to only focusing on validating vehicleLang. Had the initial problem statement of the report been as described in Section 1.3, it may have been beneficial to not interview the evaluator to prevent it from influencing the Turing test. However, after the Turing test, the evaluator was asked if they had been able to distinguish attack paths created by vehicleLang the answer to which was "no". Although, their answer does not reveal if their participation in the interviews had a subconscious effect on their ratings.

Since vehicleLang's goal is to provide analysis comparable to that of a cyber security expert, there are few other sources than experts to base validation of the language on. However, basing the conclusions on subjective ratings and opinions of experts places an implicit limitation. Both during the interviews and Turing test, experts expressed difficulty in giving a quantifiable rating of what was presented. As previously discussed, this is less of an issue with the ratings given during the interviews. Using two evaluators for the Turing test

could have mitigated this limitation based on what was discussed in the previous paragraph. Future works which validate vehicleLang may find it fruitful to use experts in different ways. One possibility could be to generate attack paths with vehicleLang and then ask experts to perform these attacks as a form of validation.

5.4 Sustainability and Ethical Aspects

Any discussion regarding cyber security has a direct connection to discussions of sustainability and ethics. Computerized systems and digital information play a large role in society and individuals' lives and tools and techniques from the cyber security realm are not virtuous by nature. Aspects of sustainability and ethics related to the results of this report may not be significant by themselves, but in the context of the THREAT MOVE project there are many aspects to consider.

The author would argue that it is generally good to attempt to secure vehicles from cyber attacks in terms of sustainability. From an economic perspective, much of today's economy relies on the consistent transport of people and goods facilitated by vehicles. If vehicles' ability to reliably perform transport was undermined by cyber attacks, massive economic loss would be certain. One example could be an attack which causes a traffic jam in a major city during morning rush hour. This type of attack could lead to hundreds of hours of lost productivity because of commuters arriving late for work. Another example could be a targeted attack against a company that relies on providing service at customers' locations, such as a locksmith, thus causing them to lose business.

These types of attacks are important to prevent from a societal perspective as well. Many important services that lay the foundation for a modern society rely on transport facilitated by vehicles, such as police, fire brigade, ambulances and postal services. Targeted attacks against these services can cause economic and physical damage, especially if coordinated with other attacks such as criminal activity or terror attacks. One can also assume the perspective of an individual in wanting to secure vehicles against cyber attacks. Attacks targeting individuals can be used to perform theft or extortion through denial of service, or injury and even death via crashes. One can imagine that remote cyber attacks on vehicles are an attractive option for high priority targets, such as politicians or journalists.

From an environmental perspective, securing vehicles from insider threats is important to ensure that environmental regulations are followed. Tuning

of ECUs with the intention of increasing power output is prevalent both for cars and heavy vehicles. A common side-effect of ECU tuning is increased emissions of greenhouse gases which can result in the vehicle not complying with environmental regulations. These types of activities can be modeled as cyber security threats, but with the caveat being that the attacker is the owner of the vehicle thus giving them better access to it. One of the intended use cases of vehicleLang is to secure vehicles against unwanted modifications made by their owners to, among other things, ensure compliance with regulations.

There are some ethical aspects to consider regarding the creation of a tool like vehicleLang. Firstly, a modelling and attack simulation tool like vehicleLang does not necessarily have to be used to prevent cyber attacks. If an adversary obtained a high-level specification of a vehicle network, it may be possible for them to use vehicleLang as a tool to plan cyber attacks. Even though vehicleLang does not provide exact details of attacks, it may make the process of finding a feasible attack more efficient for the adversary. However, one can argue that since the adversary needs insider information to create an accurate model and cyber security expertise to perform an attack, an attack would likely occur regardless of them utilising vehicleLang.

Secondly, when creating a tool that is intended to perform as well as a human expert one has to consider the ethical aspects related to automating jobs. There is no judgement on this aspect that is generally accepted, and it is outside the scope of this discussion to make such a judgment. However, the author believes that vehicleLang will not replace cyber security experts in the foreseeable future. Instead, vehicleLang will be utilized by experts to make their work more efficient and help with decision making. As such, this discussion should not currently apply to vehicleLang.

5.5 Future Work

This section gives suggestions for future work based on the results and limitations of the study.

5.5.1 Implementation of Suggested Changes

The logical next step for vehicleLang's development would be to implement the changes suggested in Section 5.2. This will likely require a similar approach as the one used in the creation of the first version of vehicleLang [5]. A literature review should be performed where the suggested changes are studied, followed by implementation, verification and validation.

5.5.2 Probabilistic Aspect

As mentioned in Section 1.4, the probabilistic aspect of vehicleLang has not been implemented and was therefore not considered in this study. A natural progression from this report would be work focused on implementing this aspect. As data in this area is lacking, future works will have to attempt to gather this data through various means, such as interviews, experiments and literature reviews. Appropriate methods for validating the probabilistic aspect will also have to be considered.

5.5.3 Different Manufacturers

A limitation of this study was that it was performed in cooperation with only one vehicle manufacturer. A validating study conducted with a different manufacturer could be fruitful for several reasons. Firstly, creating a vehicleLang model of a different real-world in-vehicle network may reveal new aspects of networks that currently cannot be modeled in vehicleLang. Secondly, gathering input from more cyber security experts could reveal other aspects that are incorrect or suggestions for future implementations. Finally, a similar study with a different manufacturer could serve as a validation of this study if it generates similar results and conclusions. This is, however, assuming that vehicleLang has not been modified before such a study is conducted.

5.5.4 Broaden Scope of vehicleLang

The initial report tasked with creating the first version of vehicleLang limited itself to aspects related to in-vehicle networks [5]. In turn, a similar limitation was placed on the study in this report. Future works should be performed where other aspects of networks in vehicles are researched, implemented and validated. Some examples of these aspects are vehicle-to-everything communication, infotainment systems and over-the-air updates. If the decision is made to include attacks which rely on physical access, changes required because of this fall under this category as well.

Chapter 6

Conclusion

Based on the discussion in Chapter 5, vehicleLang is not applicable as a threat modelling language used for security auditing of in-vehicle networks. Some significant aspects of in-vehicle networks cannot be modelled, or cannot be modelled with sufficient detail using vehicleLang. This hinders its ability to perform security auditing of networks, through attack paths generated from attack simulations, that are comparable to that of a domain expert. Its lack of a clearly defined scope in terms of attacks which require physical access to vehicles is also a contributing factor. However, the results of the study show that vehicleLang is in some instances capable of producing more than adequate results. This suggests that the concept and theory behind vehicleLang can be used to achieve its intended goals, only that it needs to be further developed to do so. This report has presented suggestions for future development of vehicleLang based on comments made by domain experts during the study. These suggestions should provide a clear next step in the language's development process and by extension the THREAT MOVE project.

Bibliography

[1] Robert N. Charette. “This Car Runs on Code”. In: *IEEE Spectrum* (Feb. 2009).

[2] Charlie Miller and Chris Valasek. *Remote Exploitation of an Unaltered Passenger Vehicle*. Aug. 2015.

[3] foreseeti. *About us*. URL: https://www.foreseeti.com/about-us/.

[4] Vinnova. *THREAT MOVE (Hotmodellering och -simulering för fordons-IT)*. URL: https://www.vinnova.se/p/threat-move-hotmodellering-och--simulering-for-fordons-it/.

[5] Sotirios Katsikeas. “vehicleLang: a probabilistic modeling and simulation language for vehicular cyber attacks”. MA thesis. KTH Royal Institute of Technology, 2018.

[6] Tianxiang Huang et al. “On the Security of In-Vehicle Hybrid Network: Status and Challenges”. In: *Information Security Practice and Experience*. Ed. by Joseph K. Liu and Pierangela Samarati. Cham: Springer International Publishing, 2017, pp. 621–637. ISBN: 978-3-319-72359-4.

[7] Karl Koscher et al. “Experimental Security Analysis of a Modern Automobile”. eng. In: IEEE Publishing, 2010, pp. 447–462. ISBN: 9781424468942.

[8] *Bosch Automotive Electrics and Automotive Electronics Systems and Components, Networking and Hybrid Drive*. eng. 5th ed.. Bosch Professional Automotive Information. 2014. ISBN: 3-658-01784-8.

[9] *Road vehicles – Unified diagnostic services (UDS) – Part 1: Specification and requirements*. Standard. International Organization for Standardization, Mar. 2013.

[10] F. Sagstetter et al. “Security Challenges in Automotive Hardware/Software Architecture Design”. In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2013, pp. 458–463. DOI: 10.7873/DATE.2013.102.

[11] NXP Semiconductors. *The Gateway to Secure Connected Vehicles*. URL: https://blog.nxp.com/automotive/the-gateway-to-secure-connected-vehicles.

[12] T. Sommestad, M. Ekstedt, and H. Holm. “The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures”. In: *IEEE Systems Journal* 7.3 (Sept. 2013), pp. 363–373. ISSN: 1932-8184. DOI: 10.1109/JSYST.2012.2221853.

[13] P. Johnson et al. “pwnPr3d: An Attack-Graph-Driven Probabilistic Threat-Modeling Approach”. In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. Aug. 2016, pp. 278–283. DOI: 10.1109/ARES.2016.77.

[14] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. “A Meta Language for Threat Modeling and Attack Simulations”. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: ACM, 2018, 38:1–38:8. ISBN: 978-1-4503-6448-5. DOI: 10.1145/3230833.3232799. URL: http://doi.acm.org/10.1145/3230833.3232799.

[15] Sotirios Katsikeas et al. *Probabilistic Modeling and Simulation of Vehicular Cyber Attacks: An Application of the Meta Attack Language*. Non published draft. [Accessed on 2018-09-04]. URL: http://autosec.se/wp-content/uploads/2018/05/carLang-2.pdf.

[16] Zhendong Ma and Christoph Schmittner. “Threat Modeling for Automotive Security Analysis”. In: *Advanced Science and Technology Letters* 139 (2016), pp. 333–339.

[17] Adi Karahasanovic, Pierre Kleberger, and Magnus Almgren. “Adapting Threat Modeling Methods for the Automotive Industry”. In: *15th ESCAR Conference*. Berlin, Germany, 2017. URL: http://publications.lib.chalmers.se/records/fulltext/252083/local_252083.pdf.

[18] Christoph Schmittner et al. “A Case Study of FMVEA and CHASSIS As Safety and Security Co-Analysis Method for Automotive Cyber-physical Systems”. In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. CPSS ’15. Singapore, Republic of Singapore: ACM, 2015, pp. 69–80. ISBN: 978-1-4503-3448-8. DOI: 10.1145/2732198.2732204. URL: http://doi.acm.org/10.1145/2732198.2732204.

[19] Stijn van Winsen. “Threat modelling for future vehicles: on identifying and analysing threats for future autonomous and connected vehicles”. MA thesis. University of Twente, 2017.

[20] H. Holm et al. “P^{2}CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language”. In: *IEEE Transactions on Dependable and Secure Computing* 12.6 (Nov. 2015), pp. 626–639. ISSN: 1545-5971. DOI: 10.1109/TDSC.2014.2382574.

[21] M. Ekstedt et al. “securiCAD by foreseeti: A CAD Tool for Enterprise Cyber Security Management”. In: *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*. Sept. 2015, pp. 152–155. DOI: 10.1109/EDOCW.2015.40.

[22] Stephen Checkoway et al. “Comprehensive experimental analyses of automotive attack surfaces”. In: *USENIX Security Symposium*. San Francisco. 2011, pp. 77–92.

[23] X. Li et al. “Connected Vehicles’ Security from the Perspective of the In-Vehicle Network”. In: *IEEE Network* 32.3 (May 2018), pp. 58–63. ISSN: 0890-8044. DOI: 10.1109/MNET.2018.1700319.

[24] J. Takahashi et al. “Abnormal vehicle behavior induced using only fabricated informative CAN messages”. In: *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. Apr. 2018, pp. 134–137. DOI: 10.1109/HST.2018.8383901.

[25] S. Parkinson et al. “Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (Nov. 2017), pp. 2898–2915. ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2665968.

[26] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. “Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures”. In: *Reliability Engineering & System Safety* 96.1 (2011). Special Issue on Safecomp 2008, pp. 11–25. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2010.06.026. URL: http:

//www.sciencedirect.com/science/article/pii/S0951832010001602.

[27] Yelizaveta Burakova et al. "Truck Hacking: An Experimental Analysis of the SAE J1939 Standard". In: *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, 2016. URL: https://www.usenix.org/conference/woot16/workshop-program/presentation/burakova.

[28] P. Murvay and B. Groza. "Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol". In: *IEEE Transactions on Vehicular Technology* 67.5 (May 2018), pp. 4325–4339. ISSN: 0018-9545. DOI: 10.1109/TVT.2018.2795384.

[29] W.C. Adams. "Conducting Semi-Structured Interviews". In: *Handbook of Practical Program Evaluation: Fourth Edition*. Wiley Blackwell, 2015, pp. 492–505. ISBN: 9781119171386.

[30] Robert M. O'Keefe and Daniel E. O'Leary. "Expert system verification and validation: a survey and tutorial". In: *Artificial Intelligence Review* 7.1 (Feb. 1993), pp. 3–42. ISSN: 1573-7462. DOI: 10.1007/BF00849196. URL: https://doi.org/10.1007/BF00849196.

Appendix A

Interview Guide

This appendix presents the interview guide used during the semi-structured interviews. Questions concerning attack paths were asked twice, once for each attack path presented. As the interviews were semi-structured they included probing questions and potentially discussed topics not mentioned in the interview guide.

A.1 Attack Paths

1. If you where to rate how realistic this attack path is *overall*, would you rate it as:

   - Very realistic
   - Realistic
   - Somewhat realistic
   - Not so realistic
   - Not at all realistic

   Realistic meaning how well the attack steps correspond with the steps of the equivalent attack in the real world.

2. What is your reasoning behind the rating you gave? What aspects influenced your answer the most?

3. Are there any attack steps you believe ignore *critical details* that could affect the path? For example, steps requiring certain credentials, special hardware, the ECU/vehicle to be in a certain state etc

4. Are there any attack steps you think are *unnecessary*? For example, there is a different way to reach a future attack step that is simpler/less time consuming.

5. Are there any defences present in the actual network that are not present in the attack path? Think of things that would make it harder for an attacker to perform some attack step.

6. Assume the attacker **does not** have physical access, is the attack feasible?

7. Assume the attacker has physical access to the vehicle, is there another way to perform the same attack thanks to this access?

8. Do you wish to add anything?

A.2 Model

1. If you were to rate how accurate the model is, would you rate it as:

   - Very accurate
   - Accurate
   - Somewhat accurate
   - Not so accurate
   - Not at all accurate

   Accurate in this context refers to how many high-level aspects of the in-vehicle network is captured by the model.

2. What is your reasoning behind the rating you gave? What aspects influenced your answer the most?

3. Overall, do you believe there are any high-level aspects of the network that have been ignored by the model?

4. Are there any critical assets or parts of ECUs you believe have been left out by the model? This includes both hardware and software.

5. Are there any critical aspects about CAN-busses and their communication you believe has been left out by the model?

6. How significant is the difference between ECUs (on a high-level) in relation to cyber security? Hypothetical example could be ECUs running software with different levels of sophistication, think full OS versus simple firmware

7. Do different parts of the system used different kinds of CAN-buses? If so, do different types of CAN-buses have an impact on the (cyber) security of the network?

8. Anything to add?

TRITA -EECS-EX-2019:72