

POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell' Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in Telecommunications Engineering



A Cybersecurity-by-Design Methodology and Tool for Vehicular Networks

Supervisor:

PROF. STEFANO ZANERO

Assistant Supervisor:

STEFANO LONGARI

Master Graduation Thesis by:

ANDREA CANNIZZO

Student Id n. 872281

Anno Accademico 2017-2018

Contents

Abstract	9
Sommario	10
1 Introduction	14
2 Motivation and Background	18
2.1 Problem Statement	18
2.2 State of the art	19
2.2.1 Risk Analysis	19
2.2.2 Architectures Evaluation	20
2.3 Goals and Challenges	22
3 Approach	23
3.1 Overview	23
3.2 Details	24
3.2.1 Attack Trees	24
3.2.2 Risk Function	32
3.2.3 A Priori Risk Analysis	40
3.2.4 Topology Based Risk Analysis	42
3.2.5 Countermeasures Generation	49
4 Implementation Details	54
4.1 Tool overview	54
4.2 Details	55

4.2.1	Input Files and Scripts	55
4.2.2	Algorithms	60
4.2.3	Outputs	63
5	Experimental Validation	69
5.1	Goals	69
5.2	Dataset	70
5.3	Test Execution and Results	70
5.3.1	A Priori Risk Analysis	70
5.3.2	Topology Based Risk Analysis	80
5.3.3	Countermeasures Proposal	81
6	Conclusions and Future Works	95

List of Figures

1.1	Active Lane Assist and Adaptive Cruise Control	15
2.1	Security requirements determination process	20
2.2	Digital I/O channels appearing on a modern car	21
3.1	Approach Work Path	25
3.2	General Attack Tree Hierarchy	26
3.3	Steering Attack Tree Example	27
3.4	Feasibility computation example. In red, the action feasibility defined as described above, and in yellow the derived values for steps, methods and objectives.	38
3.5	Algorithm sequence process (“a priori”)	41
3.6	Topology example: three buses coloured in blue and multiple components connected to them (red for attack surfaces and green for other ECUs)	43
3.7	A visual representation of the whole risk analysis process.	47
3.8	Algorithm sequence process (“topology based”)	48
3.9	Gateway Insertion Example: a gateway has been added (coloured pink) between the “Apps and Internet” component and the rest of the network.	50
3.10	Countermeasure Generation Process	51
4.1	Logical representation of “trees.xml”	62
4.2	Logical representation of “steps.xml”	62

4.3	Logical representation of the merging result between “trees.xml” and “steps.xml”	62
4.4	Flow chart about countermeasures generation algorithm	64
4.5	Console Screenshot Example	65
4.6	Topology Solution Output Example	66
5.1	Range Rover Topology representation	74
5.2	Jeep Topology representation	74
5.3	Audi Topology representation	75
5.4	RangeRover Optimal Solution with Max-Distance Constraints	87
5.5	Audi Optimal Solution with Max-Distance Constraints	88
5.6	Jeep Optimal Solution with Max-Distance Constraints	88
5.7	Audi Solution with Max-Distance and Most-Ring Constraints	89
5.8	RangeRover Optimal Solution without Constraints	89
5.9	Audi Optimal Solution without Constraints	90
5.10	Jeep Optimal Solution without Constraints	90
5.11	Audi intermediate solution after first step	91
5.12	Audi intermediate solution after second step	93
6.1	Google’s first self-driving concept	96

List of Tables

3.1	Attack severity classification scheme (from [1])	33
3.2	Rating of attack requirements aspects (derived from [1])	35
3.3	Attack Requirements to Action Feasibility Table (derived from [1])	36
3.4	Feasibilities derivation rules	37
3.5	No safety related severity risk table	38
3.6	Safety related severity risk table	39
5.1	Miller and Valasek results regarding the security of three vehicles: “-” means least hackable, “++” means most hackable)	81
5.2	Risk Evaluation Results regarding the security of three vehicles	81

Listings

4.1	Excerpt from components.xml relative to the actions parameters of three specific components: PSS, BUS and ALA.	56
4.2	Excerpt from steps.xml relative to all possible actions to obtain the step with ID equal to 16	57
4.3	Excerpt from trees.xml relative to possible methods to obtain the “Steer” attack	58
4.4	Excerpt from a topology.xml example file made of three buses: CAN A, CAN B, and CAN c.	59
4.5	Output Tree excerpt relative to the attack “ACC DoS”	67
4.6	Output Components excerpt relative to five specific units	68
5.1	Range Rover topology file (XML)	71
5.2	Jeep topology file (XML)	72
5.3	Audi topology file (XML)	73
5.4	A priori Safety Risks Ranking excerpt (first 8 attacks)	76
5.5	A priori Privacy Risks Ranking excerpt (first 10 attacks)	77
5.6	A priori Operational Risks Ranking excerpt (first 11 attacks)	78
5.7	A priori Financial Risks Ranking excerpt (first 11 attacks)	79
5.8	Attacks Ranking regarding three vehicles (first 5 attacks)	82
5.9	Buses Actions Importance in RangeRover	82
5.10	Paths Values in Audi	83
5.11	Countermeasures algorithm proceeding	85
5.12	Attacks Risks Variation after first step	92
5.13	Attacks Risks Variation after second step	94

Abstract

The rapid innovation in vehicle related technologies recently led the automotive industry to start focusing on the security of cars and internal networks, admitting the dangerousness of cyberattack even in a field not usually considered related to computer science up until some years ago. Since it is a new field in continuous evolution, much research still needs to be done in order to produce tools for the development of secure infrastructures and networks towards the security by design paradigm. In this thesis work we propose a methodology and its implementation through a tool to help analysts while designing and assessing the security of vehicle on-board networks. The tool first performs a risk based analysis on a specific network topology and propose a set of values to the analyst that help understanding the strong and weak points of the given architecture, then it evaluates the global security level of the topology assigning a grade related to the risk and finally, through those values, proposes multiple countermeasures to improve the architecture in the prevention against the most dangerous attacks.

Sommario

L'autoveicolo negli ultimi anni ha certamente subito una forte spinta innovativa. Questo è il frutto di grandi investimenti che l'intera industria automotive più di altre ha voluto fare, per migliorare la sicurezza, la comodità e l'usabilità dei veicoli a quattro ruote. I primi sistemi di cruise control si sono evoluti in tecnologie intelligenti in grado di moderare automaticamente la velocità a seconda della prossimità dell'automobile davanti e di frenare autonomamente in caso di emergenza (Adaptive Cruise Control); i sensori che avvisavano il guidatore dell'attraversamento delle righe di corsia si sono integrati in moderni sistemi che automaticamente apportano correzioni allo sterzo per mantenere l'autoveicolo in corsia (Active Lane Assist). Parallelamente a queste evoluzioni sono entrate nel abitacolo tecnologie di telecomunicazioni wireless dapprima destinate solo all'elettronica di consumo. Se prima infatti si poteva al massimo trovare esclusivamente una connessione Bluetooth con il sistema di infotainment della macchina e il ricevitore del segnale GPS, oggi in diverse automobili troviamo la rete WiFi o persino la connessione GSM. È chiaro come l'orizzonte di tutto ciò sia da una parte la guida autonoma (ormai pronta a sbarcare sulle strade) e dall'altra l'IoT (Internet of Things): i veicoli si scambieranno autonomamente tra di loro informazioni sul traffico e sulla potenzialità di scontro con altri veicoli, comunicheranno con altri apparati lungo le strade come i sistemi di pedaggio, effettueranno automaticamente chiamate di emergenza in caso di incidente inviando tutte le informazioni necessarie quali coordinate GPS e altre implementazioni che solo in parte oggi possiamo immaginare. Per questo motivo,

in molti casi dobbiamo considerare le nuove automobili non semplicemente come “macchine intelligenti” ma più come “computer su ruote”.

Tuttavia, l’introduzione di nuove tecnologie come sempre comporta nuovi rischi alle macchine e agli uomini e nel nostro specifico caso si tratta di quelli che oggigiorno vengono chiamati “rischi cyber”. Inoltre nello scenario che noi trattiamo, relativo alle automobili quindi, questi rischi vanno considerati ancora più pericolosi a causa della natura dei sistemi che possono colpire: essendo infatti le macchine sia dei computer (schermi, dati, telecomunicazioni) che dei sistemi fisici (motore, sterzo, freni) ed essendo questi due aspetti non isolati ma interfacciati tra di loro, possono essere classificate come sistemi cyberfisici e per questa ragione richiedono una attenzione maggiore per le implicazioni di sicurezza che potrebbero generarsi in caso di manomissione intenzionale che andrebbero a mettere in pericolo le persone all’interno e nei pressi del veicolo.

Nel passato la sicurezza informatica, nella maggior parte dei casi, non era una preoccupazione nella fase di progettazione delle architetture di rete per le automobili. Difatti i primi studi di cyber security atti a fornire una valutazione globale di queste reti hanno meno di 10 anni, finché i primi standard definiti da organismi internazionali quali SAE hanno alimentato l’interesse dei produttori a prendere in considerazione la sicurezza informatica fin dalle primissime fasi di progettazione degli autoveicoli.

Nel corso degli anni diversi ricercatori hanno dimostrato la possibilità di portare a termine attacchi informatici alle automobili con eventuali implicazioni in termini di sicurezza. Quasi la totalità degli attacchi si struttura in due fasi: inizialmente ottenere l’accesso all’architettura di rete interna per poter inviare messaggi sul bus di sistema e poi forgiare messaggi su misura che inneschino risposte specifiche da parte dei componenti in ascolto. Dapprima gli studi hanno mostrato quali e quante cose si potessero fare ottenendo l’accesso al bus tramite l’interfaccia di comunicazione per la diagnostica (OBD - II): frenare improvvisamente, disabilitare i freni, sterzare, accelerare, mostrare messaggi sul display e molto altro. Poi i ricercatori

hanno voluto investigare la possibilità di perpetrare questi attacchi anche attraverso diverse superfici di attacco alcune delle quali accessibili persino da remoto (lettore CD, Bluetooth, rete WiFi, connessione GSM) ottenendo esiti positivi. Si può immaginare quindi la pericolosità di questi attacchi relativamente ai danni che potrebbero comportare oltre al basso livello di sicurezza riscontrato che spinge ancora oggi a nuovi studi della materia.

Risulta chiaro a questo punto quanto ci sia necessità di apportare contromisure in fase di progettazione che in qualche modo prevengano tali attacchi. Ogni esperto di sicurezza tuttavia ha bene in mente il fatto che la sicurezza totale di un sistema non è mai ottenibile e che quindi le contromisure da adottare vanno valutate secondo il solito trade off tra costi e benefici dove la variabile benefici è rappresentata dalla diminuzione globale del rischio. Ciò che è necessario è quindi attribuire una priorità alle contromisure da adottare che tengano conto degli assets più sensibili da proteggere. Per rispondere a questa esigenza viene di solito svolta quella che nel settore è chiamata analisi del rischio (un processo di identificazione dei possibili attacchi con annessa una valutazione della pericolosità) e vi sono alcuni studi e documenti già pubblicati che espongono come applicarla nel contesto del autoveicolo. Tuttavia ciò che manca allo stato dell'arte è una metodologia di analisi del rischio che tenga conto anche delle diverse architetture di rete delle automobili, una analisi del rischio applicata alla specifica topologia della macchina presa in considerazione. Per esempio una analisi del rischio che tenga conto del fatto che la macchina presenti o meno determinate interfacce di comunicazione o determinati sistemi di "guida autonoma" quali la frenata d'emergenza o il cruise control. Questo è il primo contributo che proponiamo di apportare con questo lavoro di tesi: una metodologia e una implementazione (tramite la programmazione di uno tool) di analisi del rischio basata sulla topologia di rete. L'importanza di questo strumento la si può comprendere a pieno solo scoprendo il secondo contributo che questa tesi vuole dare; l'automazione di un processo di valutazione globale del rischio su diverse vetture permette di testare e valutare velocemente diverse configurazioni della topologia di

rete e di conseguenza permette di trovare velocemente le contromisure che decrementino il valore di rischio maggiormente. E questo è quindi il secondo contributo che proponiamo: un algoritmo di generazione automatica di contromisure basato sulla diminuzione del rischio globale di una specifica architettura.

Per raggiungere questi obiettivi siamo partiti inizialmente dagli studi esistenti e dalle metodologie già realizzate che abbiamo modificato per adattarle al nostro bisogno. Siamo partiti quindi dalla definizione delle possibili minacce attraverso la costruzione di alberi d'attacco che rappresentano i possibili scenari dell'attaccante mostrando come più azioni di basso livello vengono integrate per ottenere obiettivi di alto livello. In questo modo abbiamo creato una connessione fra l'architettura (le azioni di basso livello sono state infatti mappate sui diversi componenti) e il rischio (gli obiettivi di alto livello sono stati valutati secondo la loro pericolosità) che ci permettesse di attribuire un valore globale di rischio a una determinata topologia. Poi abbiamo costruito l'algoritmo di generazione delle contromisure il quale ha come perno del suo funzionamento proprio la valutazione globale del rischio; ad ogni step infatti prova tutte le possibili combinazioni di spostamenti di un componente da un bus all'altro della topologia oltre a provare possibili inserimenti di gateway e alla fine applica la contromisura che maggiormente diminuisce il valore di rischio. L'algoritmo termina quindi con una soluzione di topologia ottimale che tiene anche conto di eventuali vincoli di progettazione previsti dal produttore.

Una volta sviluppato il tool che implementasse queste metodologie, lo abbiamo applicato alle topologie di tre autoveicoli in circolazione e ne abbiamo constatato la funzionalità anche paragonando i risultati ad altre valutazioni precedentemente fornite da esperti del settore.

Questa tesi vuole essere quindi un punto di partenza per la progettazione di nuovi e più precisi strumenti che aiutino i produttori in fase di progettazione a trovare le contromisure prioritarie per produrre automobili sempre più sicure.

Chapter 1

Introduction

In last years vehicles and all automotive industry has received a strong innovative boost, more than many other technology industries. At first with the introduction of new safety systems like collision avoidance or lane assist (figure 1.1) and later with the integration of wireless communication system like WiFi, GSM or those proposed to implement the V2X paradigm, the car has exceeded the number of one hundred millions lines of code (more than a space shuttle or a Boing 777) [2]. For this reason, in many cases we have to consider new road vehicles not just like “smart cars” but more like “computers on wheels”. As always, the introduction of new technologies brings new risks to machines and humans, in this case mostly increasing the so called “cyber risks”. Moreover, in the specific scenario of vehicles these risk may be considered even more dangerous due to the nature of systems they can affect: being cars both computers and physical systems they can be classified as cyberphysical systems, and for this reason deserve even more attention due to the safety implication that could be generated by flaws, that may lead to put in danger people inside or around the vehicle.

In the past, cybersecurity was not a design-phase concern for most vehicle networks: as a matter of fact, the very first document about a comprehensive cybersecurity assessment of such networks is less than 10 years old [3,4]. Successivly, the appearance of the first standards by SAE [5] fueled the interest

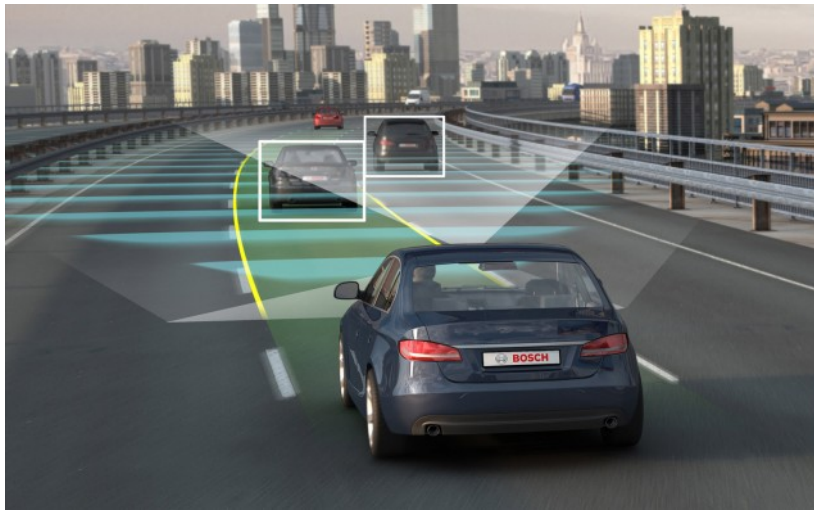


Figure 1.1: Active Lane Assist and Adaptive Cruise Control

of manufacturers for embedding security considerations into the design phase of new vehicles and nowadays new analysis tools are rising.

Moreover, we need to consider that the technology used today to protect vehicles are very often in use for ten years or more, so any design observation that is applied today needs to ensure that it can last up until 2030 without having to be physically changed.

If we consider the whole set of attacks that have been proven possible by multiple researchers [3, 4, 6, 7] we easily understand the importance of security in vehicles. In fact, in these studies, researchers can manage to put at risk the privacy and safety of the driver, passengers and people surrounding the vehicle also demonstrating the eventual damage that this type of attacks could cause. For instance they have been able on different cars to control the braking, the steering, the acceleration and to show messages on the display by only accessing the OBD-II port of the vehicle. Subsequently they also demonstrate the possibility to access the car internal network remotely through attack surfaces like the Bluetooth, Wi-Fi, Cellular or other communication interfaces.

For these reasons, while designing new vehicles, connected and full of cyberphysical systems, it is extremely important to focus on security especially

when considering the electronics architecture. The design phase of the ECUs and all electronic parts of a vehicle comprehends multiple steps, from the design of the single unit, both from a hardware and software point of view, to the layout of the on board network topology composed of all the communicating units and sensors; the focus of this thesis work is on the latter one. The main known analyses about this subject have been done by Miller and Valasek [8], by the european project EVITA [1] and the SAE guidebook SAE-J3061 [5]. The latter one proposes guidelines for a secure design of on board networks and it highlights the importance of risk analysis in the design phase but has not been thought to produce a practical methodology to evaluate an existing architecture. On the other hand, Miller and Valasek have been known for their proof of concept attacks first through the OBD-II port of a vehicle [6] and then through external connections such as Wifi and cellular networks [9]. Apart these two extremely significant papers they also proposed an analysis of the on board network of multiple vehicles, which considered mostly the risks involved with the layouts of the networks and the ease, for an attacker, to reach safety critical ECUs. Although the analyses done are complete and well described, what is not proposed in their work is a way to systematize the process of risk analysis for automotive vehicles. This has been partially solved by Ruddle et al. [1] in their paper for the EVITA project: they came out with a methodology to produce risk analyses for the automotive industry. This type of risk analysis provides risks associated to each attack but the main limit is that it is not topology based: it does not depends on the specific car, it gives the same results for each different scenario.

Our proposal focuses mainly on three different contributions:

1. An improved risk analysis methodology based on the one by Ruddle et al., that maintains part of the core concepts of the well known project but enables the possibility to create an ad-hoc topology based analysis for each network.
2. For each different network, an algorithm to propose optimal solutions

that focuses on changing the layout of the topology and inserting firewalls if needed.

3. An implementation of the aforementioned methodologies to systematically and automatically retrieve the scores of each network and discover the optimal solutions, using as inputs only the description of the topology itself, without having to create each attack path for each topology.

We have finally analyzed tests performed with the developed tool on three different real architecture topologies (the 2014 Jeep Cherokee, the 2014 Audi A8 and the 2010 RangeRover Sport) to evaluate the program outputs and we have verified results meaningfulness compared to considerations made by Miller and Valasek in [8].

The rest of this work is structured as follows: chapter 2 presents the problem and the goal of the study alongside a background section, chapter 3 presents the whole methodology, focusing first on the attack trees and risk function used to retrieve the scores for each attack, and then on the algorithms and methods used to obtain from those scores the whole topology based risk analysis and solution proposals for the specific network. Chapter 4 describes the implementation of the proposed methodologies in a semi automatic tool while chapter 5 presents the results of our work, comparing them with other previously evaluations. Finally chapter 6 discusses the results we found and concludes the thesis with an analysis of possible future works.

Chapter 2

Motivation and Background

2.1 Problem Statement

Every security specialist knows that no system is invulnerable and that every securing problem is a risk management problem with the usual trade off between risks and costs. In fact preventing every single attack against a system is more than often infeasible either due to specification requirements or in terms of costs. These notions also apply to automotive systems and led us to look for a risk analysis methodology that allows us to find the most risky attacks. Based on these analysis outputs we could propose specific prioritized countermeasures. Prioritization is the keyword of this work: having said that we cannot deal with every possible attack, we need to find which countermeasures have to be done at first in order to minimize the risk.

Vehicle internal networks are quite various, having different kinds and amounts of ECUs and different ways of interconnecting them. It is easy to understand how the topology of a vehicle influences the risk related to attacks. A vehicle with no wireless communication interface and no cyber-physical system is clearly less attackable than a system full of external connections and automatic driving tools. Therefore, threat analysis and risk assessment process (TARA) needs to be topology based in order to consider the influence of the architecture on it.

2.2 State of the art

In this section we present the main studies and works that have been produced regarding risk analysis in the automotive sector and evaluations on car internal architecture topologies.

2.2.1 Risk Analysis

EVITA Project

Ruddle et al. produced a paper [1] for the EVITA European project in which they also described a methodology to produce risk analyses for the automotive on board networks. The way they propose it is by using a technique that relies on attack trees (a conceptual representation of the necessary steps to perform an attack) and that enables the analyst that is assessing the security of the network to retrieve rankings through a risk function, so that it is possible for her to analyze a set of possible attacks basing on a common measurement system. In this way the analyst can understand which are the more risky attacks and therefore the ones that have higher priority in being prevented. There are three specific elements of Ruddle et al. proposal that we focus on improving: the first is that it is not topology oriented, it can be used to describe the attack path on a general layout but it has not been thought to change the risk of specific attacks in specific network layouts. The second is that it requires the analyst to describe, for each attack, the whole attack path by hand to retrieve the final attack score. Finally, it does not easily provide solutions to the highlighted vulnerabilities.

SAE

The document J3061 produced by SAE International [5] is a strong cybersecurity guidebook for vehicle systems throughout the entire development lifecycle process. The authors of the document firmly state the necessity of a Threat Analysis and Risk Assessment (TARA) process in the design phase

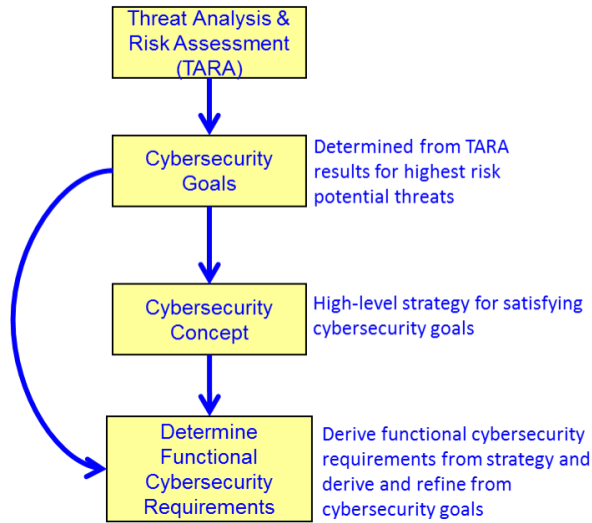


Figure 2.1: Security requirements determination process

in order to identify cybersecurity goals and then requirements as described in figure 2.1. The work also cites the EVITA Risk Analysis [1] as one of the valid methods to implement the TARA process.

2.2.2 Architectures Evaluation

Miller and Valasek

As we already said above, these two researchers in one of their works [8] proposed an analysis of the on board network of multiple vehicles, which considered mostly the risks involved with the layouts of the networks and the ease, for an attacker, to reach safety critical ECUs. They used the information retrieved by the analysis of the vehicle to propose a ranking that mainly considered the architecture of the network, the amount of external attack surfaces and the dangerousness and amount of cyberphysical controls that the vehicle had such as cruise control and lane assist. These analyses are well described and very extensive since they considered very different topologies, however the main limit is that when they evaluated the architecture of the network (one of the three main factors) they attributed values

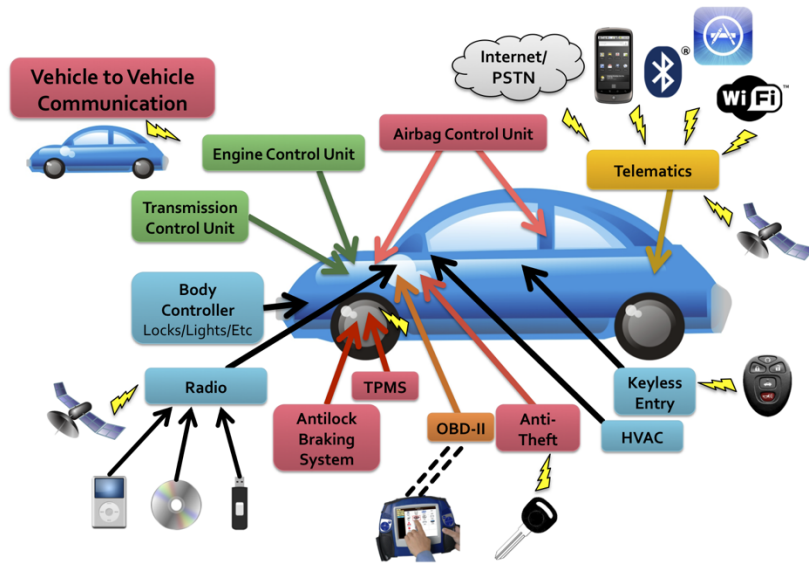


Figure 2.2: Digital I/O channels appearing on a modern car

only based on their personal consideration and knowledge. Having done this way, they did not provide a standard method to establish the riskiness of an architecture topology. By the way we used their topologies and rankings as a valid and trusted benchmark to compare our evaluations.

Checkoway et al.

Concerning architectures but not focusing on topologies, Checkoway et al. produced a paper [4] that analyzes a large number of attack surfaces. This work has been an important step towards the study of the subject because besides providing a general framework about what are the possible entry points to direct an attack against a car (figure 2.2), it also developed surfaces evaluations regarding attack range, attack costs and vulnerability level: a necessary starting point for every consideration about architecture topologies.

2.3 Goals and Challenges

The main two goals of our work are:

- **Topology Based Risk Analysis:** define and implement through a tool a methodology to assess the threats and analyze the risk of a system considering both the set of possible attacks and the actual internal architecture topology of the vehicle.
- **Automatic Countermeasures Proposition:** define and implement through a tool a methodology to automatically generate countermeasures that decrease the global risk of the architecture.

These two goals opens several challenges to overcome. For instance in order to calculate the risk we need to define a risk function and so decide the parameters that determine the risk, then how this risk function is influenced by the topology. Moreover all these processes need a mapping between components of the architectures and possible attacks in order to establish a connection point (influence point) between the two data structures. Regarding the countermeasures proposition we need to define what types of countermeasures can be generated, what prioritization mechanism (optimization function) and then also which types of constraints can limit this process.

Chapter 3

Approach

3.1 Overview

Our aim is to devise a topology-based risk analysis methodology for automotive on-board networks and an algorithm that can automatically derive security solution proposals to improve the security of a vehicle. The set of proposed solutions might involve both movements in the network layout in order to protect the most important ECUs by increasing their distance from attack surfaces, as well as the possibility to insert gateways or other segregation mechanisms that strengthen the security of a specific subnetwork by shielding it from attack surfaces. In order to obtain both the risk analysis and the proposed improvements, we need an automated risk evaluation algorithm. In fact solutions proposals will be generated by an algorithm that at each step will look for the countermeasure that would lead to the highest decrease of the score output by the risk evaluation methodology.

This evaluation methodology and implementation must take into account an expansible set of known attacks to automotive on-board networks. In order to do so, we assign to each attack a grade called “risk value” that is used to classify its dangerousness and create a ranking. This score is calculated through a risk function that mainly considers the attack “severity” (the damage that the attack can cause) and its “feasibility” (the easiness of

implementation). The latter is obtained taking into account, through the development of custom attack trees, all possible combinations of single steps that may enable said attack. The whole process is automated and can be updated incrementally, supporting the analysts and designers at each step.

3.2 Details

Figure 3.1 may help in understanding the work path in a top down approach which sums up what has been said above. Starting from our goal that are the final “Risk Values” of the attacks, we understood that we need a “Risk Function” to calculate them. Subsequently we understood that the “Risk Function” needs some input values to elaborate the result and some of these parameters are the “Feasibility Values”. Finally we realized that in order to get these values that are related to attacks we need to model somehow the threats: “Attack Trees”. This final step will be our starting point.

Next subsections are structured as follows: at first in 3.2.1 we will describe the attack tree designing process, then in 3.2.2 we will present the risk function and in 3.2.3 how it is applied to attack trees in the context of an “a priori” (not considering the topology) risk analysis. Subsequently in 3.2.4 we will show how the risk function is combined with an actual system architecture to provide a “topology based” risk analysis and finally in 3.2.5 we will explain how countermeasures are generated taking advantage of the risk analysis.

3.2.1 Attack Trees

We choose to model attack and threat scenarios through attack trees [10] in order to propose a systematic method to support risk analysis. As a matter of fact, this model shows how low-level actions interact to achieve high-level objectives through hierarchical diagrams, providing a methodical way of describing the potential attacks. It is a simple way to describe complex processes, such as cyberattacks, dividing them into small building blocks

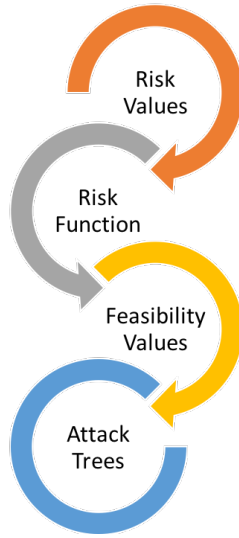


Figure 3.1: Approach Work Path

which can be modularly assembled. In particular, each parent node in the tree represents an action that can be completed through children nodes.

We designed very structured trees with a precise level separation organized in this order:

$$Goal \longrightarrow Attack \longrightarrow Method \longrightarrow Step \longrightarrow Action$$

The root of the tree is always an abstract **goal** (level 0) which the attacker wants to obtain. The goal is the final purpose of the attacker, what drives him to act and it does not consider in any way the means through which it has to be obtained, which is represented by the **attack** (level 1). Each attack can be achieved through different **methods** (level 2) which represent all the ways in which a specific attack can be performed. A method to be implemented requires a set of **steps** (level 3) which are all the elements that must be present or the intermediate objectives that have to be accomplished to execute that specific method. Finally, even a step can be obtained through multiple **malicious actions** (level 4) which are the basic element of the whole tree.

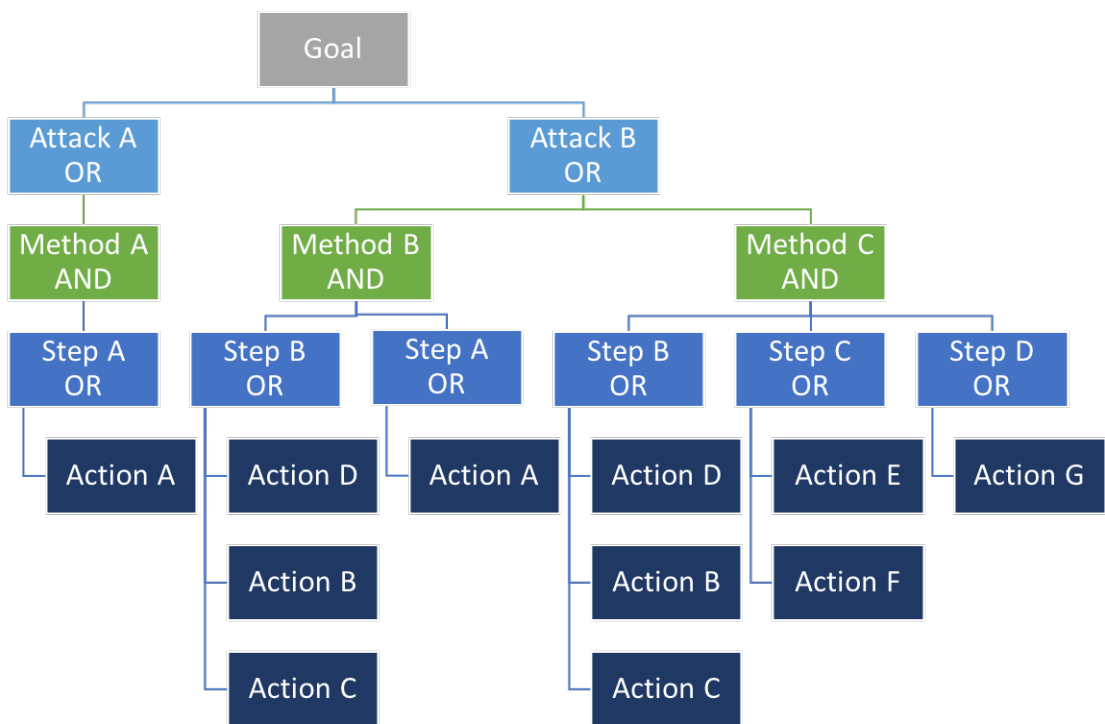


Figure 3.2: General Attack Tree Hierarchy

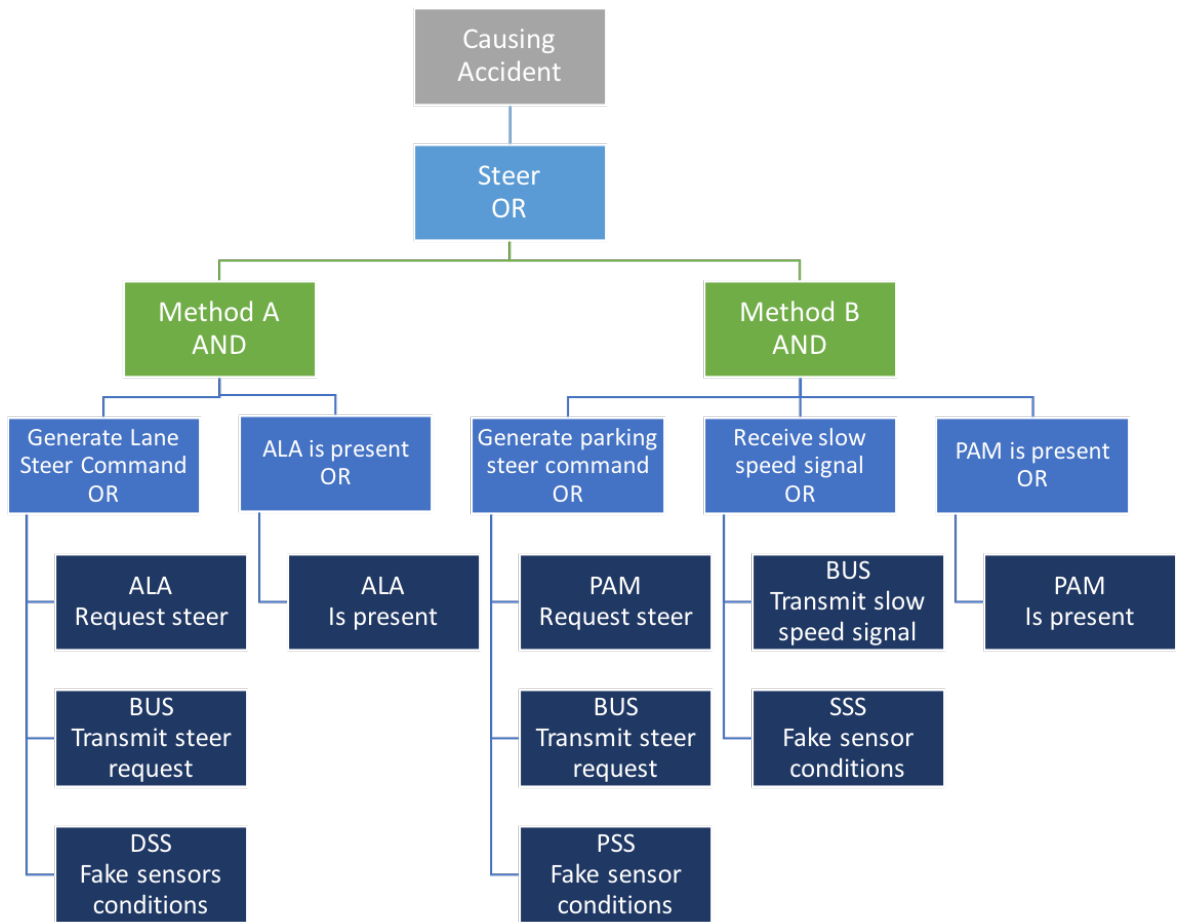


Figure 3.3: Steering Attack Tree Example

As we already said, this hierarchical model needs to be very structured and must maintain precise relationships among different levels. In fact we chose that the relationship between a node and his child nodes could only be of two types: logical **OR** or **AND**. In particular, starting from the higher levels, an **attack** is a logical **OR** combination of **methods**, a **method** is a logical **AND** combination of **steps** and finally a **step** is a logical **OR** combination of **malicious actions**. These rules have to be always respected while building attack trees otherwise it would not be possible to enable a successive automatic risk calculation.

Figure 3.2 shows the hierarchy structure described above whereas figure 3.3 presents, as a simplified example, the modelling of an attacker who has the goal of causing an accident. In this example, the only way to cause an accident is to steer the car while the driver is not supposed to (although this one is not the only possible attack able to cause an accident, we made this supposition for sake of simplicity). To obtain this attack there are two methods, that we called “Method A” and “Method B”, that are alternatives (thus in a logical **OR** relationship). Method A consists of generating a “Lane Steer Command”, but it requires an ALA (Active Lane Assist) component to be present, so these two steps are in a logical **AND** relationship as both are necessary. In order to generate a steer command the attacker can either force the ALA to request the vehicle to steer for example through malware injection, impersonate the ALA in the bus and send himself the forged request or simulate the environmental conditions which would require to turn, making so that the ALA acts in an incorrect manner without even reaching it. As these three Actions are equivalent, they are in an **OR** relationship. On the other hand, Method B consists of generating a “Parking Steer Command” but at the same time the PAM (Parking Assist Module) control unit needs to receive a “Slow Speed Signal”; in fact in most cases PAM module requests are ignored if the speed is over a defined threshold. Moreover this method obviously requires the presence of the PAM module. In order to generate a parking steer command the attacker can either force the PAM to request

the vehicle to steer, impersonate the PAM in the bus and send himself the forged request or simulate the environmental conditions which would require to turn, making so that the PAM acts in an incorrect manner without even reaching it. Whereas, to make the PAM receive the signal that the car is stopped or proceeding slowly the attack can directly transmit the information on the bus impersonating the sensor or manipulate the sensor to simulate a slow speed state.

Another important feature and rule that we can notice from figure 3.3 is that every malicious action (leaf nodes) is mapped on a precise possible component of the vehicle. This means that smallest building blocks used to build an attack tree can only be picked up from a set of possible actions on possible components.

Attack trees allow the model to be strongly modular, making it easy to add new methods by just combining steps, or by adding new ones. If new information arises (or new methods are identified) the model can be updated easily and updates can be propagated throughout the tree chain and throughout the other steps of our methodology. Also, the high level of abstraction of the root “goal” component makes it possible to comprehensively address the set of basic threats to vehicles through only five trees.

We list now the set of attacks that we identified divided in the five goals:

- Causing an accident
 - Adaptive Cruise Control DoS
 - Collision Avoidance DoS
 - Active Lane Assist DoS
 - Disable Brakes
 - Brake
 - Activate Air Bag
 - Turn Off Headlights
 - Lock Steering

- Steer
- Overheat Engine
- Speed Up
- Turn Off Engine
- Engage Seat Belt
- Vehicle theft
 - Remote Acceleration
 - Key less Ignition
- Ransomware
 - Lock Engine
 - Lock Brakes
 - Lock Doors
- Information theft
 - Eavesdrop On Conversations
 - Capture Cameras Images
 - Get Address Book
 - Track Position
- Insurance fraud
 - Tamper Odometer
 - Hide Crash Info

While we do not claim to provide a complete coverage of all possible (or indeed, even all known) attacks, it should be noted that our main purpose is to propose a methodology through which we can enable automation. Through this generalization therefore we propose a set of the major known attacks

and the known paths to implement them, so that it is possible just by implementing our proposal as it is to know how topologies are affected by the most known and dangerous attacks, but we do not claim to provide the complete set of known automotive attacks. To model these threats and imagine many possible attack paths we studied in deep the bibliography produce by expert researchers. We have indeed found many attack methods ideas and proofs of concept in [3], [6] and [9]; most of the methods focused on two main steps: access the internal bus network through a communication interface and then send messages on the bus to trigger specific actions of the listening components.

Differences from EVITA proposition

The idea of using attack trees in a vehicle environment, per se, is not novel: the EVITA project also explored it while making their risk analysis [1]. Our attack tree has a consistent strength in relation to the ones of the EVITA project though: in their trees there is no structured and clear division between AND and OR levels. Although at first sight this could seem an advantage in terms of flexibility, it leads to situations in which high final objectives and initial actions are on the same level. Our methodology is instead more structured, without actually creating any limitations: although it may require to create one more step than the EVITA one to complete an attack tree, it makes possible to structure the same attacks on the same level of detail. Also, being more structured, it enables a semi automatic analysis through a tool we propose later in the paper.

More importantly, the novelty of our methodology is mapping the leaf nodes (low-level actions) onto the components of the network architecture. For instance EVITA does not differentiate among proximity sensors, light sensors and speed sensors enclosing them in the category “In car sensors”, but our final objective, a topology based risk analysis, would be impossible without distinguishing those components.

3.2.2 Risk Function

Since our final goal is to highlight the most critical nodes of a vehicular network, and to propose design improvements, we need to devise a method and a function to calculate the risk of each attack patch in the tree. In particular, we need a function that takes as input variables of the the attack and outputs a risk value. Since it is one of the methodologies recommended by [5], we use a slightly modified version of the function and ratings proposed in [1]:

$$R = riskF(A, S, C)$$

Where the inputs are:

- **Severity** (S), a parameter that represents the potential damage that an attack can cause. It is further broken down in four aspects or types of effect (Safety, Privacy, Financial and Operational), each one with a parameter value that can range from $S0$ (no threat) to $S4$ (very significant threat). The four severity levels are defined as in Table 3.1 as proposed in [1].
- **Controllability** (C), a parameter that has to be considered only when the severity vector includes a non-zero safety component, and which represents the potential for the driver to confine the severity of the outcome. Four different levels of controllability are considered, from $C1$ (avoidance possible through human response) to $C4$ (situation impossible to influence).
- **Attack Feasibility**¹(A), a parameter that describes how easy is to complete the attack on a scale from $A0$ (impossible) to $A5$ (very easy).

All parameters are related to each specific attack (i.e., level 1 of the attack tree). Parameters S and C can be defined a priori for the attack, whereas

¹This parameter was called “attack probability” in [1], but we decided to change the name for clarity, as this is evidently not a probability.

Table 3.1: Attack severity classification scheme (from [1])

Severity Value	Safety	Privacy	Financial	Operational
0	No injuries.	No unauthorized access to data.	No financial loss.	No impact on operational performance.
1	Light or moderate injuries.	Anonymous data only (no specific driver or vehicle data).	Low level loss.	Impact not discernible to driver.
2	Severe injuries (survival probable). Light or moderate injuries for multiple vehicles.	Identification of vehicle or driver. Anonymous data for multiple vehicles.	Moderate loss. Low losses for multiple vehicles.	Driver aware of performance degradation. Indiscernible impacts for multiple vehicles.
3	Life threatening (survival uncertain) or fatal injuries. Severe injuries for multiple vehicles.	Driver or vehicle tracking. Identification of driver or vehicle, for multiple vehicles.	Heavy loss. Moderate losses for multiple vehicles.	Significant impact on performance. Noticeable impact for multiple vehicles.
4	Life threatening or fatal injuries for multiple vehicles.	Driver or vehicle tracking for multiple vehicles.	Heavy losses for multiple vehicles.	Significant impact for multiple vehicles.

instead parameter A is derived through a two-step process: “Action Feasibility Definition” and “Attack Feasibility Derivation”. Once these two phases are completed the process goes on with the “Risk Function Application”.

Action Feasibility Definition

In the first step, we evaluate for each leaf node a set of **attack requirements**², i.e. the effort needed for the malicious action to be successful. We use a similar methodology to [1], considering five factors for each malicious action:

- Elapsed time: the total amount of time to identify a vulnerability, develop an attack and perform it.
- Specialist Expertise: the general knowledge required by the attacker in relation to these types of attack.
- Knowledge of the system under investigation: the specific knowledge of the system needed by the attacker.
- Window of opportunity: the time available to the attacker to access the exploitable target.
- IT hardware/software or other equipment: all kind of equipment needed to perform the attack.

We evaluate each of these factors according to Table 3.2, adding up the requirement values (the higher the value, the higher the difficulty or challenge for the attacker due to that factor). The result, evaluated according to Table 3.3, can be correlated to attack feasibility (of course, the higher the attack requirements, the lower the feasibility).

As an example, let us calculate the attack feasibility of a malicious action where a connected unit of a vehicle is remotely reflashed with a malicious firmware:

²Called “attack potential” in [1]

Table 3.2: Rating of attack requirements aspects (derived from [1])

Factor	Level	Comment	Value
Elapsed Time	≤ 1 day		0
	≤ 1 week		1
	≤ 1 month		4
	≤ 3 months		10
	≤ 6 months		17
	> 6 months not practical	The attack path is not exploitable within a timescale that would be useful to an attacker.	19 ∞
Expertise	Layman	Unknowledgeable compared to experts or proficient persons, with no particular expertise.	0
	Proficient	Knowledgeable in being familiar with the security behaviour of the product or system type.	3
	Expert	Familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, principles and concepts of security employed, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc.	6
	Multiple experts	Different fields of expertise are required at an Expert level for distinct steps of an attack.	8
Knowledge of system	Public	e.g. as gained from the internet.	0
	Restricted	e.g. Knowledge that is controlled within the developer organization and shared with other organizations under a non-disclosure agreement.	3
	Sensitive	e.g. Knowledge that is shared between discreet teams within the developer organization, access to which is constrained only to team members.	7
	Critical	e.g. Knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need-to-know basis and individual undertaking.	11
Window of Opportunity	Unnecessary/unlimited	The attack does not need any kind of opportunity to be realized because there is no risk of being detected during access to the target of the attack and it is no problem to access the required number of targets for the attack.	0
	Easy	Access is required for ≤ 1 day and number of targets required to perform the attack ≤ 10 .	1
	Moderate	Access is required for ≤ 1 month and number of targets required to perform the attack ≤ 100 .	4
	Difficult	Access is required for > 1 month or number of targets required to perform the attack > 100 .	10
	None	The opportunity window is not sufficient to perform the attack (the access to the target is too short to perform the attack, or a sufficient number of targets is not accessible to the attacker).	∞
Equipment	Standard	Readily available to the attacker.	0
	Specialized	Not readily available to the attacker, but acquirable without undue effort. This could include purchase of moderate amounts of equipment or development of more extensive attack scripts or programs.	4
	Bespoke	Not readily available to the public because equipment may need to be specially produced, is so specialized that its distribution is restricted, or is very expensive.	7
	Multiple bespoke	Different types of bespoke equipment are required for distinct steps of an attack.	9

Table 3.3: Attack Requirements to Action Feasibility Table (derived from [1])

Values	Attack Requirements	Action Feasibility
0-9	Basic	5
10-13	Enhanced-Basic	4
14-19	Moderate	3
20-24	High	2
≥ 25	Beyond High	1

- Elapsed time: some months are required to evaluate the ECU vulnerabilities and write the malicious firmware, 17
- Specialist Expertise: an expert is required, 6
- Knowledge of the system under investigation: supposing a reverse engineering process there is no requirement in relation to system knowledge, 0
- Window of opportunity: there's no specific requirement as the vehicle just has to be turned on, 0
- IT hardware/software or other equipment: some tools may be required to analyze and reflash the ECU, 4

So the attack requirement value is 27, which corresponds to a rating "High" and to a feasibility of 2 (on our scale of 0 to 5).

By applying this analysis to all malicious actions we obtain the attack feasibility for each leaf node in the tree.

Attack Feasibility Derivation

In order to apply the risk function and calculate the risk value of a specific attack we need the attack feasibility at level 1 of the tree, however we have only defined action feasibilities at level 4 of the trees (leaf nodes). So we need a method to automatically calculate attack feasibilities starting from the action ones and to do this we can combine children level values to derive each parent level value in the tree. We therefore follow two simple rules: for

Table 3.4: Feasibilities derivation rules

	Children	Feasibility
Attack	OR of 'Methods'	MaxA(Children Methods)
Method	AND of 'Steps'	MinA(Children Steps)
Step	OR of 'Actions'	MaxA(Children Action)
Action		$A(\text{Action}) = \{1 - 5\}$

OR nodes, the combined feasibility is the highest of the children feasibility values (as the attacker will choose the easiest path), whereas for an AND node the parent node feasibility is the same as the lowest value among its children as described in table 3.4. These two rules are very simple however we have to mention that we are no more considering “probabilities” but “feasibilities” so we are not forced to use values between 0 and 1, furthermore it is an approach really similar to the one applied by [1]. In this way, following a bottom-up process, we derive the combined feasibility of each step until reaching level 1 attack feasibility.

A practical example of this attack feasibility derivation process is visible in Figure 3.4.

Risk Function Application

At this point, we can apply the methodology described in [1] to combine S , C and A into a qualitative risk value, ranging from $R0$ (no risk) to $R8$ (unacceptable/extreme risk). We obtain one risk value for each attack and each risk category: safety risk, privacy risk, operational risk, financial risk. For the first category, Table 3.6 is used, while for the risks not connected to safety, Table 3.5 is used (since controllability does not influence the outcome). Furthermore, for both tables this rule is valid: if a severity level of an attack is 0, the risk level of the attack related to that severity will be 0 (if an attack doesn't have privacy related consequences, there is no privacy risk about that attack).

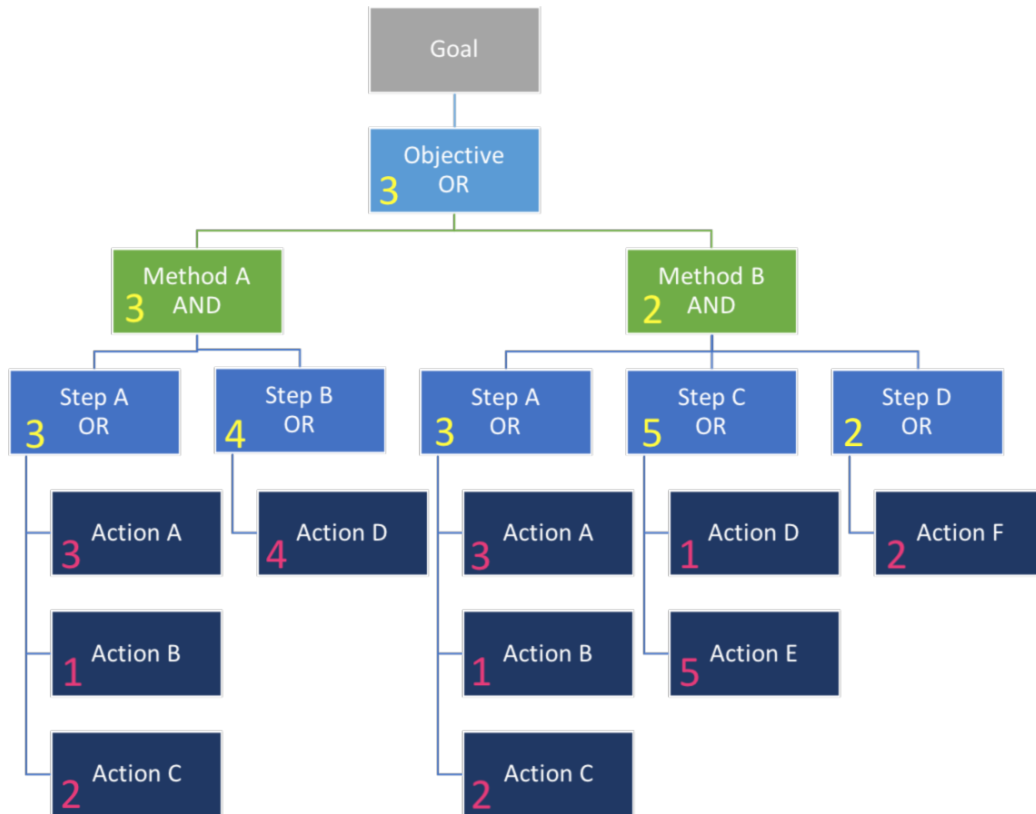


Figure 3.4: Feasibility computation example. In red, the action feasibility defined as described above, and in yellow the derived values for steps, methods and objectives.

Table 3.5: No safety related severity risk table

Security Risk Level(R)		Attack Feasibility				
		A=1	A=2	A=3	A=4	A=5
Non-Safety Severity	S=1	R0	R0	R1	R2	R3
	S=2	R0	R1	R2	R3	R4
	S=3	R1	R2	R3	R4	R5
	S=4	R2	R3	R4	R5	R6

Table 3.6: Safety related severity risk table

Control- liability	Safety-related Severity	Attack Feasibility				
		A=1	A=2	A=3	A=4	A=5
C=1	S=1	R0	R0	R1	R2	R3
	S=2	R0	R1	R2	R3	R4
	S=3	R1	R2	R3	R4	R5
	S=4	R2	R3	R4	R5	R6
C=2	S=1	R0	R1	R2	R3	R4
	S=2	R1	R2	R3	R4	R5
	S=3	R2	R3	R4	R5	R6
	S=4	R3	R4	R5	R6	R7
C=3	S=1	R1	R2	R3	R4	R5
	S=2	R2	R3	R4	R5	R6
	S=3	R3	R4	R5	R6	R7
	S=4	R4	R5	R6	R7	R8
C=4	S=1	R2	R3	R4	R5	R6
	S=2	R3	R4	R5	R6	R7
	S=3	R4	R5	R6	R7	R8
	S=4	R5	R6	R7	R8	R8

To clarify with an example, let us suppose that for the attack shown in Figure 3.4 severity and controllability are defined as follows:

- Safety Severity: 3
- Operational Severity: 4
- Privacy Severity: 0
- Financial Severity: 2
- Controllability: 3
- Attack Feasibility: 3

If we apply tables 3.5 and 3.6 to the values above we obtain risk values (divided in four respective categories) relative to the specific attack. The result is the following:

- Safety Risk: 4
- Operational Risk: 3
- Privacy Risk: 0
- Financial Risk: 1

3.2.3 A Priori Risk Analysis

With the terms “a priori” and “topology based” we want to differentiate between an analysis that doesn’t take the topology into account and an analysis that does. In fact the two approaches will have different inputs, different aims and different outputs. An “a priori” risk analysis, not considering a precise car model or system architecture, is more generic and less precise. Its only goal is to identify the main risks to be aware while designing a system and rank them to prioritize countermeasures. It will not be able to automatically understand if an attack is actually performable on the specific car which is being designed; this could be done through a “topology based” risk analysis that will consider instead the actual architecture topology of the system. An “a priori” analysis applied to different cars will have the same outcome since the topology will not influence calculations whereas a “topology based” analysis applied to different cars will provide different outcomes. Having different outcomes means that we will be able to evaluate and compare architectures.

Complete Process

The process of the “a priori” risk analysis is exactly the one described until now and corresponds to applying the risk function to all attacks present in the attack trees after having defined or derive all parameters. Figure 3.5 may help synthesizing how the process work: Malicious Actions of the attack trees are passed through an attack requirement analysis as a first step and from this phase the Action Feasibility is defined. The generated Action Feasibilities are then used as inputs of the attack trees to derive Attack Feasibilities. Finally,



Figure 3.5: Algorithm sequence process (“a priori”)

The Attack Feasibilities alongside the Severity and Controllability of each attack are passed through the Risk Function to obtain the set of outputs that we present in next subsection.

Outputs

At the end of the topology-based risk analysis process, we obtain as outputs:

- The **attack trees** generated from original ones, annotated with updated feasibilities and with risk values for each attack.
- A ranking of the **most dangerous attacks**, by safety, privacy, financial or operational risk value.
- A ranking of the **most sensitive malicious actions**, showing the actions most involved in dangerous attacks: they can be ranked by an “importance value” obtained by adding together all the risk values of attacks in which the malicious action is involved, multiplied by the feasibility value of the action and by the feasibility value of the method

in which it appears (to give a higher importance to malicious actions that are easier to implement or that appear more frequently).

3.2.4 Topology Based Risk Analysis

The core novelty of our approach lies in the topology-based risk assessment. All previous research, to the best of our knowledge, always focused on analysing the risk either ignoring the actual architecture of the vehicle, or at best assuming it to be known a priori and fixed, a sort of “black box” that influences the output but is not a parameter under assessment.

We aim to perform a topology based analysis where we consider only attacks that are actually achievable given the topology under assessment and considering the impact of the topology on the feasibility of those attacks, with the final aim of proposing structural countermeasures.

In order to achieve this, we need to “map” the risk function discussed in Section 3.2.2 onto the topology model.

The first step is to provide a structure to model the architecture. In most cases, in-vehicle communications are based on different buses, on which signals are transmitted using different protocols (the typical choice is CAN [11] but even LIN [12] is quite common). In this thesis we do not consider the specific differences among such protocols, although the modularity of the analysis and of the developed tool allows future extensions in this direction and requirements can be simulated with constraints.

In most such architectures, there are several ECUs connected to a small number of different buses. Some ECUs are connected to more than one bus, acting as gateways. It is natural to represent such topologies using graph models, in particular star graphs. In these graphs we distinguish two different kinds of existing components: either an ECU is connected with external surfaces and is therefore defined as an attack surface or it is not and is then considered as a normal ECU. Finally, we map all buses to the attack tree leaf nodes that are involved in attacks focusing on a bus (i.e. DoS of CAN Bus) so that the action involves only the specific bus that is being attacked.

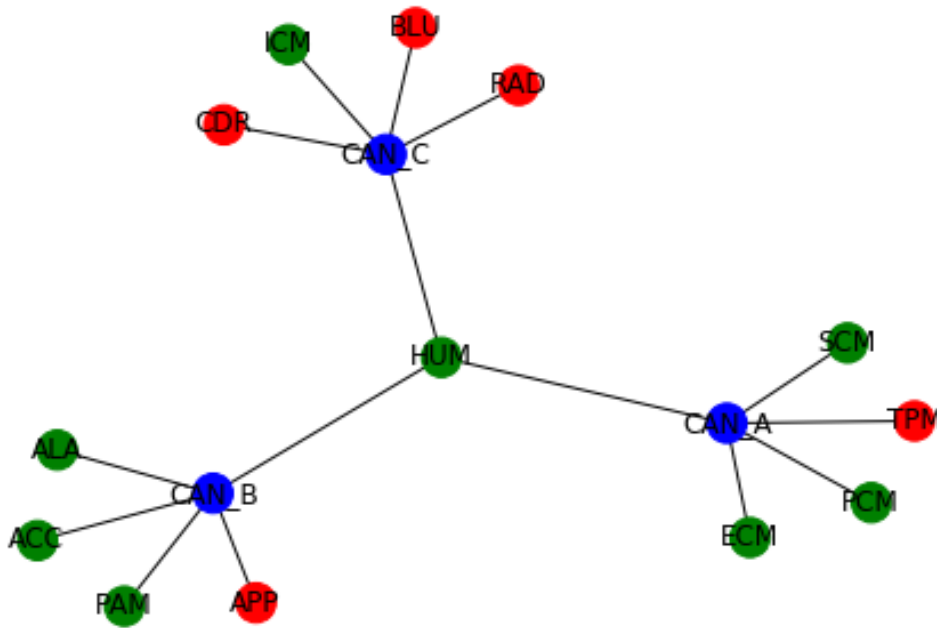


Figure 3.6: Topology example: three buses coloured in blue and multiple components connected to them (red for attack surfaces and green for other ECUs)

An architecture topology model example is provided in figure 3.6.

In order to map risks onto the topology, our key intuition is that the only input of risk functions (attack feasibility, severity and controllability) that can change due to the topology is the attack feasibility, as controllability and severity are related directly to the higher levels of the attack tree. Therefore, if we can link attack feasibility at leaf node level to the topology of the network, we can apply the very same process described in Section 3.2.2 to obtain a risk value that is influenced and changes along with the topology.

The only difference we implement when working on the topology based risk analysis in relation to the risk function is that instead of using the tables proposed in section II-B to obtain the risk value, which only accept finite output values, we use the linear functions that represent said tables: for non

safety risk values (table 3.5) we use

$$R_x = A_x + S_x - 3$$

for safety related risk values (table 3.6) we use

$$R_s = A_s + S_s + C - 4$$

Values are then capped at a maximum of 8 and a minimum of 0 if required, to follow the tables. As in previous “table” version for both functions is valid this rule: if a severity level of an attack is 0, the risk level of the attack related to that severity will be 0. Moreover is also valid this other rule: if the feasibility of an attack is 0, then all risk values of the attack are 0. This second rule wasn’t considered in “a priori” risk analysis because it wasn’t permissible that a feasibility could be 0 whereas now this is a possible case.

The impact of topology on attack feasibility can be twofold. The most trivial impact is that, in some architectures, an attack cannot be performed because of the absence of a specific required component. For instance, any attack which has a step where malware should be re-flashed on the Parking Assist Module of a vehicle cannot be performed if this specific ECU is not present. We handle this trivial case by setting to 0 (impossible) the feasibility of any action related to components not found in the topology.

More interestingly, as suggested by Miller and Valasek in [8], the proximity of an ECU to attack surfaces influences the feasibility. For example, observe Figure 3.6: an injection on the CAN-C bus can happen through multiple attack surfaces, such as Bluetooth, CD reader and radio receiver. This is more feasible, for an attacker, than injecting content on a bus that is not connected to the outside. In order to account for this, we divide components in two categories: attack surfaces and simple components. Attack surfaces are all those components that offer an opportunity to the attacker to access the internal network. Not all attack surfaces are equally dangerous, so we rank them along three dimensions:

- *Cost*: a value from 1 to 3 describing the cost and the effort necessary to break into the component and take control (1 means “high cost”, 3 means “low cost”)
- *Surface*: a value from 1 to 3 related to the amount of possible new attack steps that can be done if the attacker obtains control of the component (1 means few, 3 means many)
- *Range*: a value from 1 to 3 describing the necessary physical distance from the vehicle to access the surface (1 means “in car”, 3 means “remotely exploitable”)

Once these values are set, they are added up and the result is transformed in a number between 1 and 3, 1 if the sum is between 1 and 5, 2 if the value is 6, 3 if the value is higher, obtaining a “dangerous parameter” value for every surface.

Established “dangerous” parameters of the surfaces it is possible to write a function for the update of the action feasibilities based on the distance between component and surfaces. To do so we define and use these values and functions:

- d_x : distance between component and surface x defined as number of minimum hops in the topology graph to reach one node starting from the other one (minimum value is 1).
- a_x : “dangerous” parameters of the surface x (one for each attack surface of the topology).
- w : is a value called “influence weight” whose objective is to define how much the topology should influence the action feasibility update (1 = high influence, 5 = low influence).
- I_w : through a function defined as

$$I_w = \left(\sum_{x=1}^N \frac{a_x}{d_x} \right) / w$$

we produce the value that is going to update the action feasibility. it lowers each dangerous parameter value in relation to how far it is from the considered component, sums all of the resulting values and then divides the result for w , lowering the impact of the topology on the risk assessment if required.

- K : malicious actions of the analyzed component (i.e. for Head Unit currently the two feasible actions are Root Access and Malware injection).
- p_y : feasibility of action y of the component analyzed:

$$p_y = p_y + I_w - \frac{4}{w} \text{ for each } y \text{ in } K$$

The function increments the action feasibilities of all those actions concerning components that have a high number of surfaces in their proximity considering the distance and considering the dangerous level of the surfaces. The factor $-\frac{4}{w}$ is used to maintain feasibilities around original values and avoid a saturation of feasibilities values to 5.

Complete Process

Figures 3.7 and 3.8 may help understanding how the whole risk analysis process work: Malicious Actions are passed through an attack requirement analysis as a first step, not considering the topology of the specific vehicle. From this process the Action Feasibility is derived. The topology of the vehicle is then taken into consideration, deleting the impossible actions (making their feasibility 0) and updating the Action Feasibility of the remaining. The generated Action Feasibilities are then used as inputs of the attack trees of all the considered attacks, generating Attack Feasibilities for the specific topology. Finally, The Attack Feasibilities alongside the Severity and Controllability of each attack are passed through the Risk Function to obtain the set of outputs that we present in next subsection.

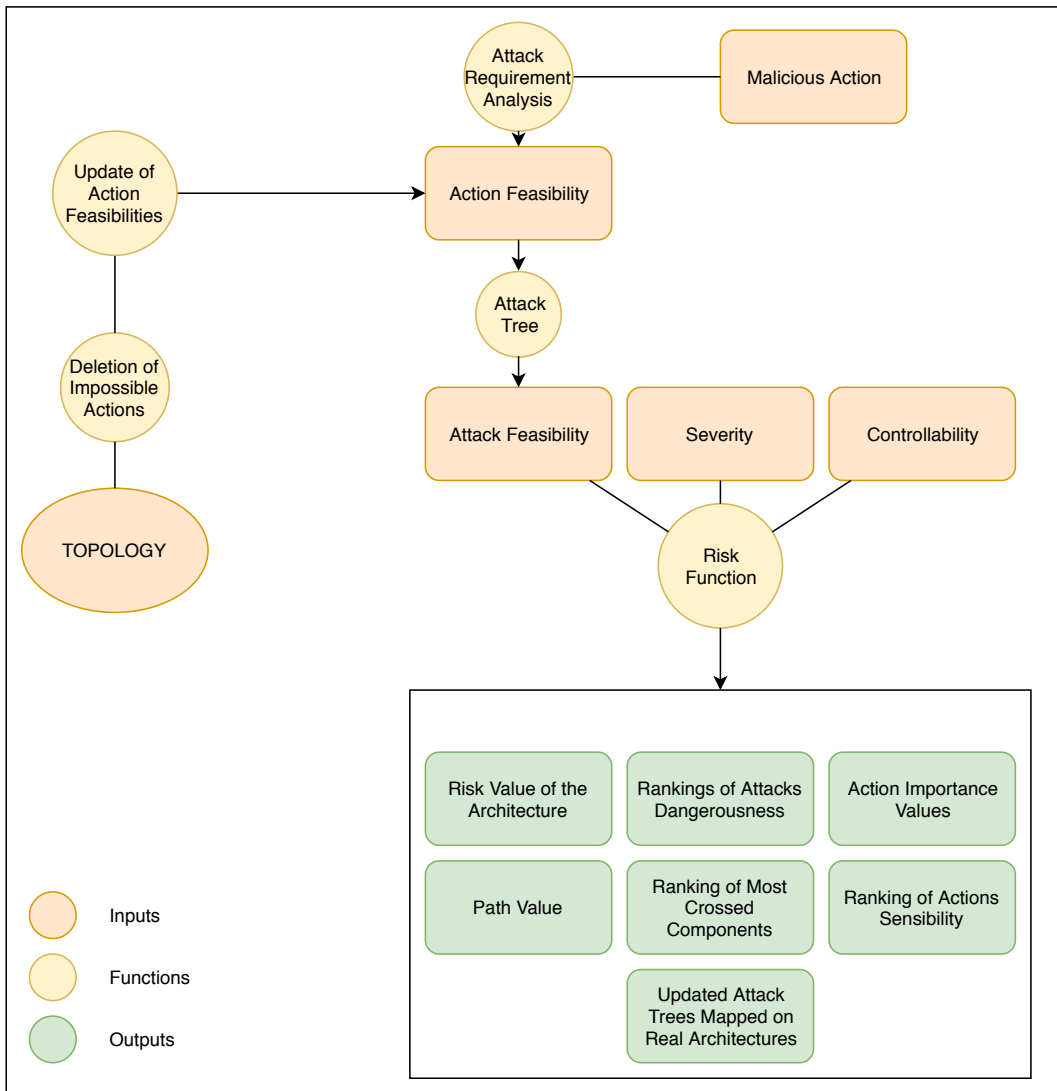


Figure 3.7: A visual representation of the whole risk analysis process.



Figure 3.8: Algorithm sequence process (“topology based”)

Outputs

At the end of the topology-based risk analysis process, we obtain as outputs:

- The **attack trees** generated from original ones, mapped on the topology of the vehicle buses, annotated with updated feasibilities and with risk values for each attack.
- A ranking of the **most dangerous attacks**, by safety, privacy, financial or operational risk value.
- A ranking of the **most sensitive malicious actions**, showing the actions most involved in dangerous attacks: they can be ranked by an “importance value” obtained by adding together all the risk values of attacks in which the malicious action is involved, multiplied by the feasibility value of the action and by the feasibility value of the method in which it appears (to give a higher importance to malicious actions that are easier to implement or that appear more frequently).
- A ranking of the components **most traversed by attack paths**, ob-

tained by sorting the components according to the number of paths they are traversed by, from each attack surface to all components involved in malicious actions.

- A global **risk value** for the architecture, that can be obtained by adding together all risk values (safety, privacy, financial, operational) of each attack. It is a good approximation of an overall risk evaluation of the architecture.

3.2.5 Countermeasures Generation

Besides offering all of the indicators listed in Section 3.2.4, which can be analysed by the designer to implement bespoke defence mechanisms and evaluate their impact (as briefly described in 3.2.5), our approach can also automatically suggest countermeasures for the proposed network topology. As we already stated, the distance between attack surface and target ECU plays a crucial role in defining the risk, so changes of the topology could better the architecture evaluation.

In order to help architecture designers to improve topologies and decrease the global risk, we developed an algorithm that, starting from the original topology, proposes changes trying to reach a local optimum for the risk value. In particular, at each step the algorithm looks for the action that decreases the risk value the most, among two types of architectural countermeasures: topological changes or insertion of security gateways (i.e., firewall components).

The algorithm tries, first, all possible shifts of a component from a bus to another. After each movement, it recomputes the global risk value of the architecture. After trying all possible changes (separately), the algorithm saves the one that produces the maximum reduction of risk.

In a second phase, the algorithm tries to replace each component with a gateway, and to attach the component to that gateway. In this way, it increases the distance amongst components, so that attack surfaces have less

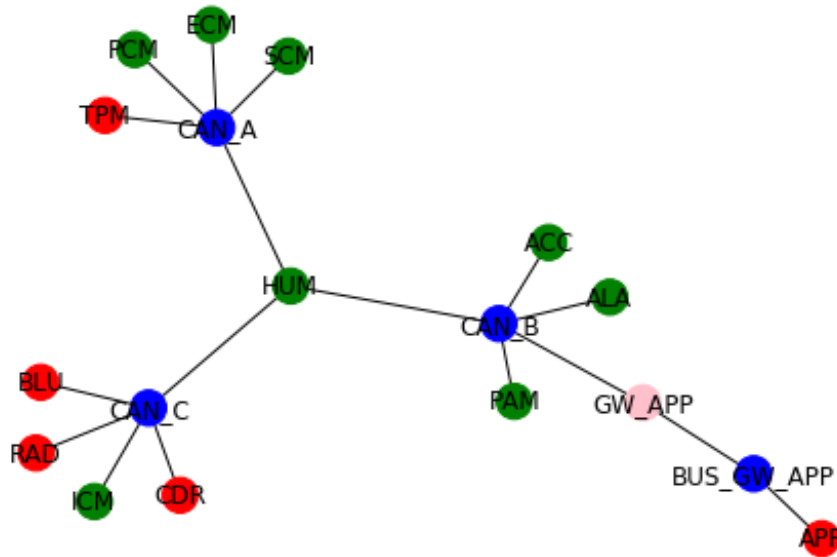


Figure 3.9: Gateway Insertion Example: a gateway has been added (coloured pink) between the “Apps and Internet” component and the rest of the network.

influence in increasing attack feasibility. After trying all possible insertions, the algorithm saves the one that produces the maximum reduction of risk. In Figure 3.9 we show an example of gateway insertion in the topology previously presented in Figure 3.6.

Finally, the algorithm proposes the action among all the ones attempted that produces the largest decrease of risk value. After having confirmed this first best action, the topology is updated, and the algorithm can be rerun to find the “next best” action to perform. The complete process is the one described in figure 3.10.

As the knowledgeable reader might have noticed, many of the structural changes randomly proposed by such an algorithm would violate design constraints. For instance, the algorithm may propose to move the Adaptive Cruise Control ECU far away from Speed Sensor for risk reasons relative to the bus in which ACC is communicating but this distance between ACC

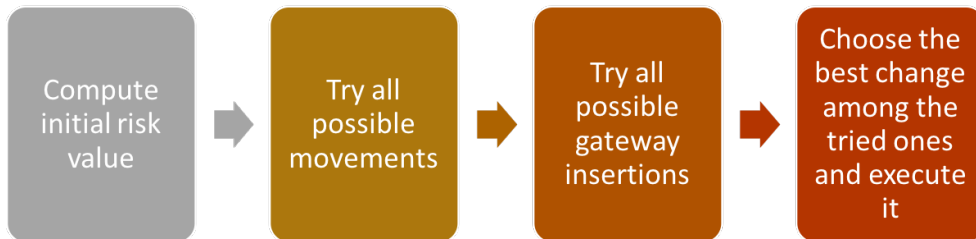


Figure 3.10: Countermeasure Generation Process

and Speed Sensor could generate an unacceptable latency. To overcome this problem we add to the algorithm a list of requirements that have to be verified before proposing any change. The algorithm accepts two kind of requirements:

- **DisMax, A, B:** distance between component A and component B must be lower than this value
- **DisMin, A, B:** distance between component A and component B must be greater than this value

This list of requirements is completely programmable by the designer or by the analyst. One of the requirements that we choose to add by default is the impossibility to have two consecutive linked gateways as it would be useless.

Pruning

The evolution of the algorithm often generates topologies that need to be fixed, therefore after each step the algorithm checks and corrects any problem that has been produced amongst these:

- **Useless Bus:** if a bus becomes empty due to all components connected to it being moved to another subnetwork the bus is removed.

- **Useless Gateway:** if a bus is removed leaving a gateway connected only to one single component, the gateway is removed.
- **Gateway Chain:** if the only components that are communicating on a bus are two gateways, one can be removed alongside the bus inbetween them.

Termination and cost based regularization

After each termination the countermeasures algorithm moves a maximum of one component or inserts only one gateway. The process can then be run again on the new topology until there is no movement or gateway that decreases the risk value of a topology, which makes it a local minimum.

Supposing that the algorithm is run until its local minimum and without any distance constraint inserted the final result is a network in which every component is shielded by a gateway and "owns" a bus. To avoid this solution which is clearly not optimal in real world scenarios due to time and cost limitations it is possible to add distance constraints amongst all components, which leads to faster and more realistic final configurations.

Analysis of the risk analysis outputs

Apart the automatic countermeasures proposals described above it is also possible for an analyst to consider the outputs of the risk analysis to understand which other countermeasures to apply and how to apply them. It is not possible to push the risks to a value of 0 mainly for economic reasons but the analysis of the outputs makes it possible to focus on the most important ones.

Considering the **Attack Rankings** it is possible to understand which attacks have to be prioritized in terms of countermeasures.

Analyzing **Action Rankings** an analyst can understand which are the actions that are mainly applied in attacks, considering both the number of attacks who apply that action both the dangerousness of those attacks.

Through these information it is possible to derive if some other countermeasures are required for a specific action that is involved in many attacks. For instance bus injection is one of the most common actions and the analyst can consider if, for a specific bus, it may be required to improve cryptography or if an intrusion detection system [13] could be a solution.

Finally considering **Path Value Ranking** it is possible to understand which are the components that are mostly traversed by attacks paths, still considering the dangerousness of each attack which involves that component. This gives an interesting view on the components that are key nodes of the architecture, from an attacker point of view. The analyst can then decide for example to implement the software security of said component, or implement a firewall to ensure its security.

Chapter 4

Implementation Details

4.1 Tool overview

As introduced before, we developed a tool to perform the risk analysis and countermeasures proposals in a semi automatic way. The program takes as input three files (XML) and combines them to build a complete attack tree. These three files are “components.xml”, “steps.xml”, “trees.xml” and contain respectively: a list of actions on components, a list of steps, a list of attacks and methods. Then the tool imports a topology file (XML) that can be chosen by the user among different loaded topologies and subsequently it performs the analysis logging the outputs on the console and producing some output files. Finally it starts the countermeasures generation that can be carried forward step by step or directly until the terminal configuration that is the local optimum point. The main procedure code is contained in a script (Python) named “main.py” that calls functions and algorithms defined in another file (Python) named “functions.py”.

In building the tool we have used these coding languages and libraries:

- **Python**: an object oriented programming language that we used to write scripts, functions and algorithms.
- **XML (eXtensible Markup Language)**: a markup language that

we used to write data files (input and output files) in a structured and hierarchical way.

- **NetworkX**: a python supporting library to create, update and analyse graph structures (useful for topologies).
- **xml.etree.ElementTree**: a python supporting library to parse .xml files and handle them as trees.
- **xml.dom**: a python supporting library to dynamically access and update the content, structure, and style of an .xml document.
- **matplotlib.pyplot**: a python plotting library that can produce a variety of different types of figures (useful to print topology graphs).

4.2 Details

4.2.1 Input Files and Scripts

components.xml

This file contains the list of all the components available in the tool to build topologies and steps of the attack trees. Each component has an “ID” parameter made of three upper-case characters and a set of malicious actions (sub-levels of the component) virtually performable on the specific component. A “name” parameter and an “action feasibility” parameter (which is defined as described in 3.2.2) are associated with each action. In listing 4.1 is shown an excerpt from the code related to the component PSS that is Proximity Sensors module, the BUS and the Active Lane Assist module.

steps.xml

This file contains the list of all the steps that can be used to build methods of the attack trees. Each step has an “ID” parameter made of two digits and a set of malicious actions (sub-levels of the step) that can be alternatively

Listing 4.1: Excerpt from components.xml relative to the actions parameters of three specific components: PSS, BUS and ALA.

```
<component id = "PSS"> <!-- Proximity sensors-->
  <action name= "MalwareFlashed" rank="2"> </action>
  <action name= "Manipulate" rank="3"> </action>
  <action name= "DoS" rank="4"> </action>
</component>
<component id = "BUS"> <!-- Generic bus that could be CAN -->
  <action name= "DiagnosticInj" rank="2"> </action>
  <action name= "Inject" rank="2"> </action>
  <action name= "Listen" rank="4"> </action>
  <action name= "DoS" rank="4"> </action>
</component>
<component id = "ALA"> <!-- Active lane assist module -->
  <action name= "MalwareFlashed" rank="2"> </action>
  <action name= "DoS" rank="3"> </action>
  <action name= "Present" rank="99"> </action>
</component>
```

performed to complete the specific step. The actions are identified through the “ID” parameter of the component and the “action” parameter that needs to correspond to one of the “name” parameters of the actions in components file. In listing 4.2 is shown an excerpt from the code related to the step 16.

trees.xml

This file contains the list of all the attack methods related to each attack in the tree. Each attack has a “name” parameter, five integer parameters (safety severity, privacy severity, financial severity, operational severity, controllability) and a set of methods (sub-levels of the attack) that can alternatively performed to complete the specific attack. Then each method is composed by a set of steps identified through the “ID” parameter and described through the “name” sub-tag. In listing 4.3 is shown an excerpt from the code related to the attack “Steer”, the same tree described in figure 3.3.

Listing 4.2: Excerpt from steps.xml relative to all possible actions to obtain the step with ID equal to 16

```
<step ID= "16">
  <name> Generate acceleration command </name>
  <component id= "ACC" action= "MalwareFlashed">
    Request acceleration
  </component>
  <component id= "ECM" action= "MalwareFlashed">
    Enter acceleration command
  </component>
  <component id= "PCM" action= "MalwareFlashed">
    Enter acceleration command
  </component>
  <component id= "BUS" action= "Inject" to="ECM">
    Transmit acceleration request
  </component>
  <component id= "BUS" action= "Inject" to="PCM">
    Transmit acceleration request
  </component>
  <component id= "PSS" action= "Manipulate">
    Fake proximity sensors conditions
  </component>
  <component id= "PSS" action= "MalwareFlashed">
    Fake proximity sensors conditions
  </component>
  <component id= "BUS" action= "Inject" to="ACC">
    Insert fake proximity data
  </component>
</step>
```

Listing 4.3: Excerpt from trees.xml relative to possible methods to obtain the “Steer” attack

```
<attack name="Steer" Ss="4" Sp="0" Sf="4" So="4" C="3">
  <method>
    <step ID="14">
      <name> Generate lane steer cmd </name>
    </step>
    <step ID="34">
      <name> ALA is present </name>
    </step>
  </method>
  <method>
    <step ID="05">
      <name> Receive slow speed signal </name>
    </step>
    <step ID="15">
      <name> Generate parking steer cmd </name>
    </step>
    <step ID="35">
      <name> PAM is present </name>
    </step>
  </method>
</attack>
```

Listing 4.4: Excerpt from a topology.xml example file made of three buses: CAN A, CAN B, and CAN c.

```
<data>
  <bus id="CAN_A" type="BUS">
    <component id="HUM"> </component>
    <component id="SCM"> </component>
    <component id="PCM"> </component>
    <component id="ECM"> </component>
    <component id="TPM"> </component>
  </bus>
  <bus id="CAN_B" type="BUS">
    <component id="HUM"> </component>
    <component id="ACC"> </component>
    <component id="PAM"> </component>
    <component id="ALA"> </component>
    <component id="APP"> </component>
  </bus>
  <bus id="CAN_C" type="BUS">
    <component id="HUM"> </component>
    <component id="ICM"> </component>
    <component id="CDR"> </component>
    <component id="BLU"> </component>
    <component id="RAD"> </component>
  </bus>
</data>
```

topology.xml

This file contains the topology of the specific architecture considered. There is the list of buses present in the architecture and (as sub-levels) the list of every component that communicates through that specific bus. Each bus is identified through “ID” string parameter and a “type” parameter describing the bus category (wireless / wired). Then each component is identified through an “ID” parameter using the same reference system used in “components.xml” and “steps.xml” files. In listing 4.4 is shown the complete code of the same topology example described in figure 3.6.

main.py

This file contains the main procedure code of the tool. It is a sequence of python statements and function calls modularly assembled in order to easily modify or update the program. All functions called from the main code are defined in an other file called “functions.py” which is imported at the beginning of the code. The main code basically follows this sequence:

1. Import libraries
2. Import input files: components, steps, trees
3. Merge the three input files to build a unique tree
4. Import topology file (selected by the user among a set of architectures)
5. Check data structure integrity
6. Build topology graph
7. Update action feasibilities (topology based)
8. Calculate attack feasibilities
9. Calculate initial risk values
10. Run countermeasures algorithm
11. Calculate final risk values
12. Generate outputs

4.2.2 Algorithms

In this section we present the key points behind the implementation of the two most particular algorithms. Both of them are coded through one or more python functions in the file named “functions.py” and they are called from the main code. The two algorithms that we chose to highlight are:

- Attack Tree Building and Buses Mapping
- Countermeasures Generation

Attack Tree Building and Buses Mapping

The aim of this algorithm is to integrate information contained in the four XML input files concerning attack trees and topology in order to create a single data structure: a unique attack tree. The program considers the data structure contained in the file named “trees.xml” as the main tree to which append branches defined in the others two files (“steps.xml” and “components.xml”). At first the tool merge steps structures with the main tree structure by appending the first ones to the corresponding step ID parameter in the main tree (the second one). In figure 4.3 is shown the final data structure as result of the merging of figure 4.1 with figure 4.2. Then, once the topology is loaded and topology graph is build, the tool updates action feasibilities contained in “components.xml” file by inspecting distances described in the graph and sets to 0 all feasibilities concerning components that are not present in the topology.

Another operation executed in this phase is buses mapping. In fact, some actions in the tree are referred to the component BUS but architecture topologies have many buses so the tool needs to know which of the buses is the one it has to consider at each step. In order to facilitate this operation all actions concerning buses are coded with an additional information: a “to” parameter that indicates where to find the bus. For instance the last action in listing 4.2 is an injection on the bus relative to the Adaptive Cruise Control; then the tool will look in the topology graph for the bus in which ACC is communicating and will substitute the id BUS with the actual component bus id that could be CAN A for instance.

Finally the program appends these updated action feasibilities to the main tree and moves on to the risk calculation.

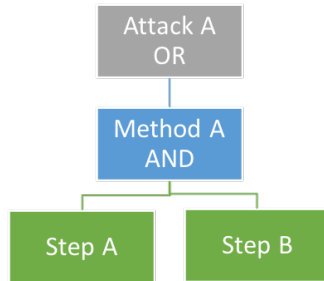


Figure 4.1: Logical representation of “trees.xml”

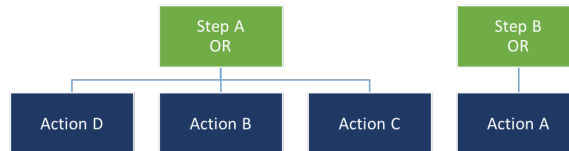


Figure 4.2: Logical representation of “steps.xml”

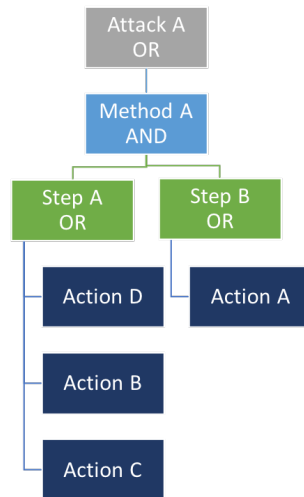


Figure 4.3: Logical representation of the merging result between “trees.xml” and “steps.xml”

Countermeasures Generation

The aim of this algorithm is to obtain an optimal configuration through incremental changes that step by step improve the topology from the point of view of security risk. The algorithm at the beginning evaluates the initial risk value and initialize the variable “minimum risk” with this value then it tries all possible changes (all possible movements of a component from a bus to an other one and all possible gateways insertions). After each topology change it checks the constraints and optimize the topology (pruning), then it updates action feasibilities and recomputes attack feasibilities and risk values. If the new risk value is less than the “minimum risk” value it updates the minimum risk value with this new value and it saves this movement as the one to perform at the end (supposing constraints not violated). After each risk calculation it restores the initial topology in order to try an other modification. At the end the algorithm applies definitely the movement that produces the minimum risk value. If the initial risk value is equal to the final risk value the algorithm terminates as there is no better topology to be generated without violating a constraint, otherwise if the final risk is less than the initial risk the algorithm recursively calls itself to find the next best change. In figure 4.4 a scheme is shown of the algorithm as previously described.

4.2.3 Outputs

In this subsection we describe how the outputs presented in section 3.2.4 are shown to the user.

- **Console:** through the terminal the tool prints out the ranking of the most dangerous attacks as in the example screenshot of figure 4.5, the ranking of the most sensitive malicious actions, the ranking of the most traversed component and the global risk value. Moreover it shows the coloured topology graph of the initial architecture and of the all intermediate configurations until the optimal solution as the example

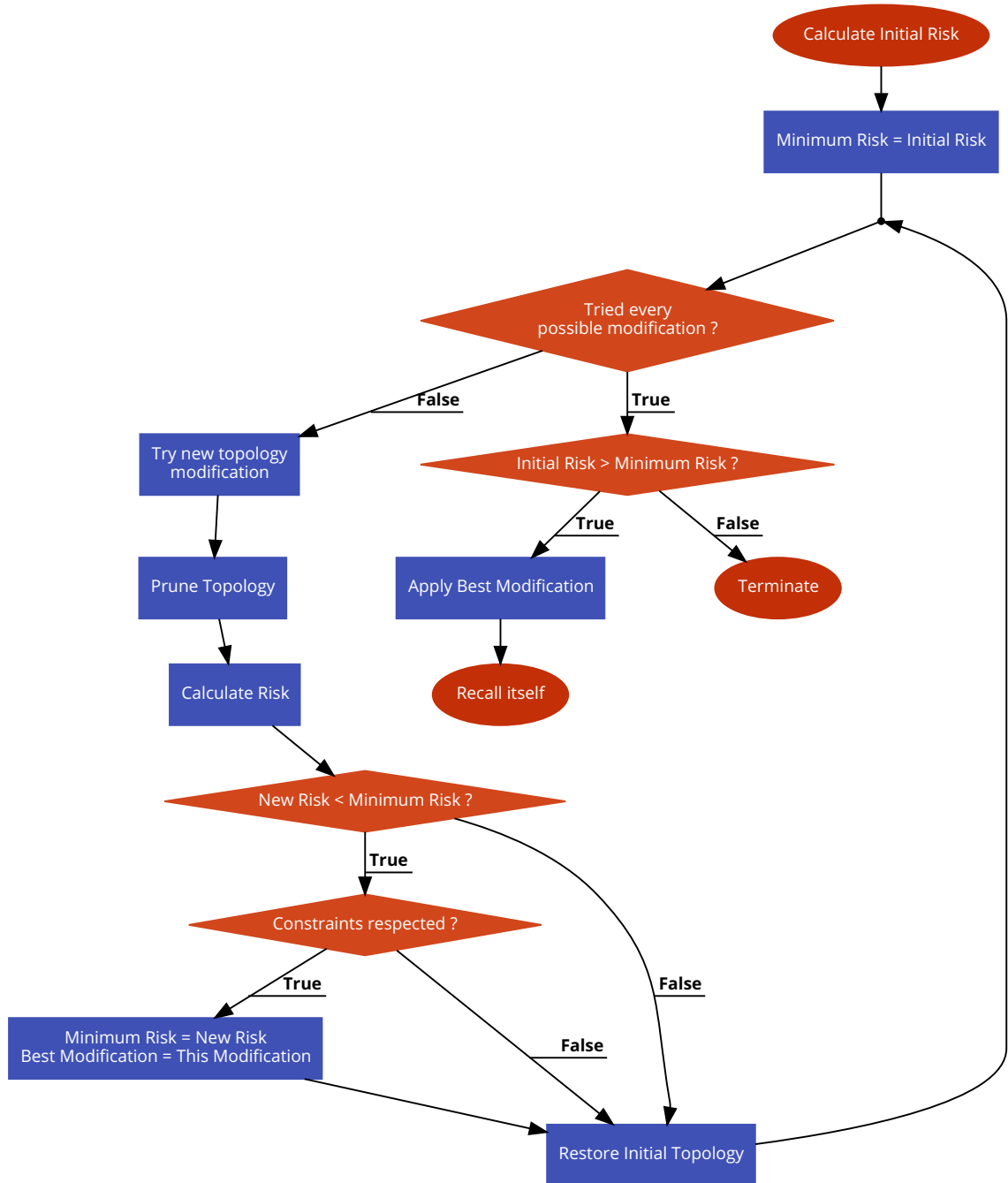
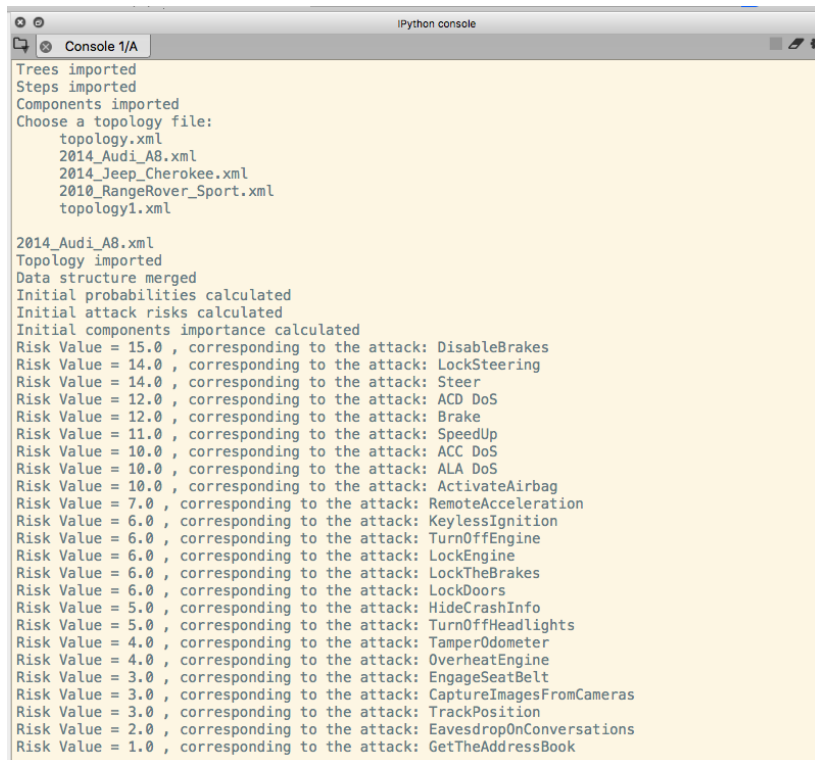


Figure 4.4: Flow chart about countermeasures generation algorithm



```
IPython console
Console 1/A
Trees imported
Steps imported
Components imported
Choose a topology file:
  topology.xml
  2014_Audi_A8.xml
  2014_Jeep_Cherokee.xml
  2010_RangeRover_Sport.xml
  topology1.xml

2014_Audi_A8.xml
Topology imported
Data structure merged
Initial probabilities calculated
Initial attack risks calculated
Initial components importance calculated
Risk Value = 15.0 , corresponding to the attack: DisableBrakes
Risk Value = 14.0 , corresponding to the attack: LockSteering
Risk Value = 14.0 , corresponding to the attack: Steer
Risk Value = 12.0 , corresponding to the attack: ACD DoS
Risk Value = 12.0 , corresponding to the attack: Brake
Risk Value = 11.0 , corresponding to the attack: SpeedUp
Risk Value = 10.0 , corresponding to the attack: ACC DoS
Risk Value = 10.0 , corresponding to the attack: ALA DoS
Risk Value = 10.0 , corresponding to the attack: ActivateAirbag
Risk Value = 7.0 , corresponding to the attack: RemoteAcceleration
Risk Value = 6.0 , corresponding to the attack: KeylessIgnition
Risk Value = 6.0 , corresponding to the attack: TurnOffEngine
Risk Value = 6.0 , corresponding to the attack: LockEngine
Risk Value = 6.0 , corresponding to the attack: LockTheBrakes
Risk Value = 6.0 , corresponding to the attack: LockDoors
Risk Value = 5.0 , corresponding to the attack: HideCrashInfo
Risk Value = 5.0 , corresponding to the attack: TurnOffHeadlights
Risk Value = 4.0 , corresponding to the attack: TamperOdometer
Risk Value = 4.0 , corresponding to the attack: OverheatEngine
Risk Value = 3.0 , corresponding to the attack: EngageSeatBelt
Risk Value = 3.0 , corresponding to the attack: CaptureImagesFromCameras
Risk Value = 3.0 , corresponding to the attack: TrackPosition
Risk Value = 2.0 , corresponding to the attack: EavesdropOnConversations
Risk Value = 1.0 , corresponding to the attack: GetTheAddressBook
```

Figure 4.5: Console Screenshot Example

showed in figure 4.6.

- **Tree.xml:** this output file is a version of the complete and unique attack tree generated from the merging of the different files with all the feasibilities and risk values updated. An excerpt of example of this file is shown in listing 4.5.
- **Components.xml:** this output file is a version of the input components file updated with the importance and rank values and cleaned by removing components not present in the topology. An excerpt of example of this file is shown in listing 4.6.

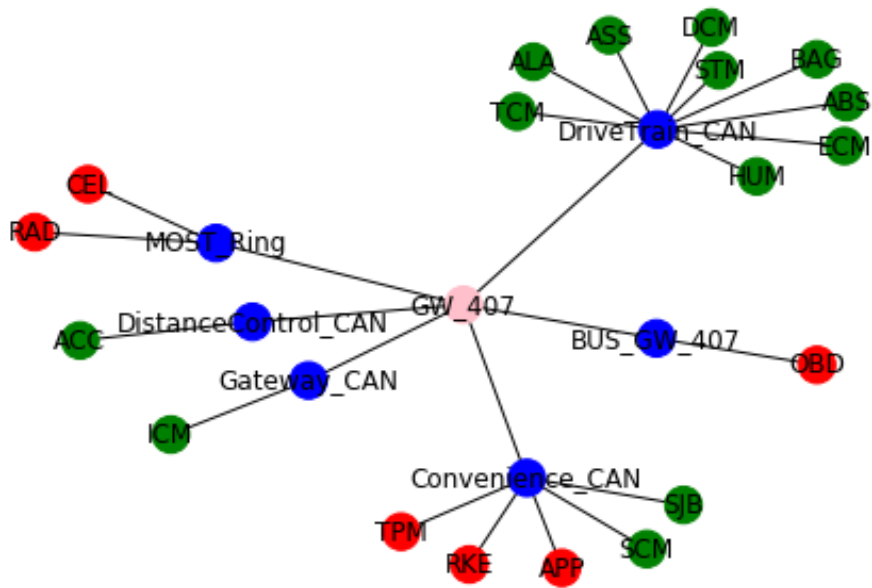


Figure 4.6: Topology Solution Output Example

Listing 4.5: Output Tree excerpt relative to the attack “ACC DoS”

```

<attack C="1" Rf="3.375" Ro="2.375" Rp="0" Rs="2.375"
  Sf="3" So="2" Sp="0" Ss="2" name="ACC DoS" Feasibility="3.375">

  <method Feasibility="3.375">
    <step ID="06" Feasibility="3.375" to="ACC">
      <name> Prevent data receiving </name>
      <component action="DoS"
        id="CAN_HS" Feasibility="3.375" to="ACC">
        Flood the bus
      </component>
    </step>
  </method>

  <method Feasibility="2.1666666666666665">
    <step ID="25" Feasibility="2.1666666666666665">
      <name> Disable ACC </name>
      <component action="MalwareFlashed"
        id="ACC" Feasibility="1">
        Disable
      </component>
      <component action="RootAccess"
        id="HUM" Feasibility="2.1666666666666665">
        Request disabling
      </component>
      <component action="MalwareFlashed"
        id="HUM" Feasibility="1.1666666666666665">
        Request disabling
      </component>
      <component action="MalwareFlashed"
        id="ICM" Feasibility="1.1666666666666665">
        Request disabling
      </component>
      <component action="Inject"
        id="CAN_HS" Feasibility="1.375" to="ACC">
        Transmit disabling request
      </component>
    </step>
    <step ID="32" Feasibility="5">
      <name> ACC is present </name>
      <component action="Present"
        id="ACC" Feasibility="5">
        ACC is present
      </component>
    </step>
  </method>
</attack>

```

Listing 4.6: Output Components excerpt relative to five specific units

```
<component id="PAM">
  <action importance="325.4351851851852"
    name="MalwareFlashed" rank="2.333333333333335">
  </action>
  <action importance="0"
    name="DoS" rank="3.333333333333335">
  </action>
  <action importance="0.0"
    name="Present" rank="5">
  </action>
</component>
<component id="ICM">
  <action importance="332.93750000000006"
    name="MalwareFlashed" rank="2.333333333333335">
  </action>
  <action importance="0"
    name="DoS" rank="3.333333333333335">
  </action>
</component>
<component id="DCM">
  <action importance="130.4875"
    name="MalwareFlashed" rank="2.2">
  </action>
  <action importance="0"
    name="DoS" rank="3.2">
  </action>
</component>
<component id="ITM">
  <action importance="0"
    name="MalwareFlashed" rank="1.4000000000000001">
  </action>
  <action importance="0"
    name="DoS" rank="2.4000000000000004">
  </action>
</component>
<component id="LSS">
  <action importance="28.000000000000001"
    name="MalwareFlashed" rank="1.4000000000000001">
  </action>
  <action importance="48.000000000000003"
    name="Manipulate" rank="2.4000000000000004">
  </action>
  <action importance="0"
    name="DoS" rank="3.4000000000000004">
  </action>
</component>
```

Chapter 5

Experimental Validation

In this chapter we will present our results concerning the application of the tool on three different architecture topologies: the 2014 Jeep Cherokee, the 2014 Audi A8 and the 2010 RangeRover Sport. The reason behind these choices is the necessity to find benchmarks for our results. These three topologies have been studied by Miller and Valasek in [8] where they made a general evaluation of the security of their networks focusing on the network topology, the number of cyberphysical systems and the number of attack surfaces. We analyse first the a priori risk analysis results, then the topology based risk analysis results and finally the generated countermeasures proposals for all architectures.

5.1 Goals

While executing our tests, we are mainly interested in verifying the meaningfulness of three results:

- **A Priori Risk Analysis:** we don't have a valid benchmark to evaluate these results since the analysis contained in the paper written by Ruddle et al. [1] for the EVITA project proposed different attack trees. However we will verify results consistency and meaningfulness and we will analyze how they will evolve in the context of the topology based

analysis.

- **Topology Based Risk Analysis:** to evaluate these results we will compare them with the analysis contained in [8] by Miller and Valasek.
- **Countermeasures Generation:** since our tool is the first proposition that generates automatically these type of solutions, we don't have any other benchmarks to compare these results, so we will limit to explain some considerations about the reasonableness of the optimal solutions.

5.2 Dataset

The dataset consists of the input files mentioned in subsection 4.2.1 where the topology file changes in different tests and it is picked from the aforementioned three topologies: the 2014 Jeep Cherokee, the 2014 Audi A8 and the 2010 RangeRover Sport. We choose these architectures for a specific reason: the risk analysis evaluation provides a set of values that can be useful by themselves but are also extremely clearer if compared with other results, specially while providing a proof of the validity of our own results. So we choose the topology that scored best (RangeRover), the worse (Jeep) and one in between (Audi) in Miller and Valasek tests to confront their outcomes with our ones. Listings 5.1, 5.2 and 5.3 show the three topologies represented using the components available in the tool and coded in XML files as previously described. Figures 5.1, 5.2 and 5.3 show the graph representation of the topologies.

5.3 Test Execution and Results

5.3.1 A Priori Risk Analysis

Concerning the risk analysis applied to our attack trees without considering any topology implication we obtained the results showed in listings 5.4, 5.5, 5.6 and 5.7. They are excerpt of the console outputs about safety risk attack

Listing 5.1: Range Rover topology file (XML)

```
<data>
  <bus id="CAN_MS" type="BUS">
    <component id="SJB"> </component>
    <component id="DCM"> </component>
    <component id="RKE"> </component>
    <component id="ICM"> </component>
    <component id="HUM"> </component>
    <component id="PSS"> </component>
    <component id="RAD"> </component>
    <component id="OBD"> </component>
    <component id="TPM"> </component>
  </bus>
  <bus id="CAN_HS" type="BUS">
    <component id="ECM"> </component>
    <component id="ABS"> </component>
    <component id="ACC"> </component>
    <component id="SJB"> </component>
    <component id="TCM"> </component>
    <component id="ICM"> </component>
    <component id="STM"> </component>
    <component id="ASS"> </component>
    <component id="OBD"> </component>
    <component id="TPM"> </component>
  </bus>
</data>
```

Listing 5.2: Jeep topology file (XML)

```

<data>
  <bus id="CAN_C" type="BUS">
    <component id="ABS"> </component>
    <component id="ACC"> </component>
    <component id="TCM"> </component>
    <component id="PAM"> </component>
    <component id="STM"> </component>
    <component id="ALA"> </component>
    <component id="PCM"> </component>
    <component id="ESM"> </component>
    <component id="ICM"> </component>
    <component id="PSS"> </component>
    <component id="OBD"> </component>
    <component id="RAD"> </component>
    <component id="RKE"> </component>
    <component id="BLU"> </component>
    <component id="TPM"> </component>
    <component id="CEL"> </component>
    <component id="APP"> </component>
    <component id="HUM"> </component>
  </bus>
  <bus id="CAN_IHS" type="BUS">
    <component id="DCM"> </component>
    <component id="ICM"> </component>
    <component id="OBD"> </component>
    <component id="RAD"> </component>
    <component id="BLU"> </component>
    <component id="CEL"> </component>
    <component id="APP"> </component>
    <component id="HUM"> </component>
  </bus>
  <bus id="LIN" type="BUS">
    <component id="ICM"> </component>
    <component id="PCM"> </component>
    <component id="DCM"> </component>
    <component id="STM"> </component>
    <component id="ITM"> </component>
    <component id="LSS"> </component>
  </bus>
</data>

```


Listing 5.3: Audi topology file (XML)

```

<data>
  <bus id="DriveTrain_CAN" type="BUS">
    <component id="ECM"> </component>
    <component id="ABS"> </component>
    <component id="BAG"> </component>
    <component id="TCM"> </component>
    <component id="ALA"> </component>
    <component id="STM"> </component>
    <component id="ASS"> </component>
    <component id="OBD"> </component>
  </bus>
  <bus id="Convenience_CAN" type="BUS">
    <component id="DCM"> </component>
    <component id="RKE"> </component>
    <component id="SCM"> </component>
    <component id="SJB"> </component>
    <component id="TPM"> </component>
    <component id="OBD"> </component>
  </bus>
  <bus id="Gateway_CAN" type="BUS">
    <component id="ICM"> </component>
    <component id="OBD"> </component>
  </bus>
  <bus id="DistanceControl_CAN" type="BUS">
    <component id="ACC"> </component>
    <component id="OBD"> </component>
  </bus>
  <bus id="MOST_Ring" type="BUS">
    <component id="HUM"> </component>
    <component id="OBD"> </component>
    <component id="RAD"> </component>
    <component id="CEL"> </component>
    <component id="APP"> </component>
  </bus>
</data>

```

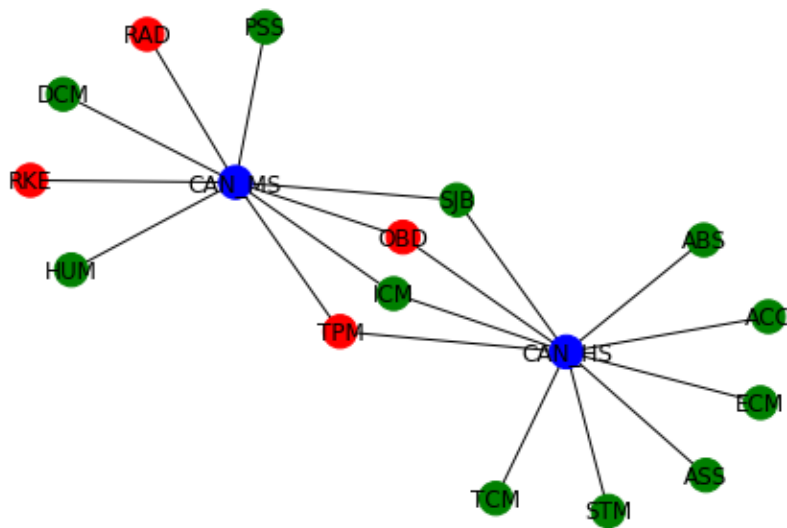


Figure 5.1: Range Rover Topology representation

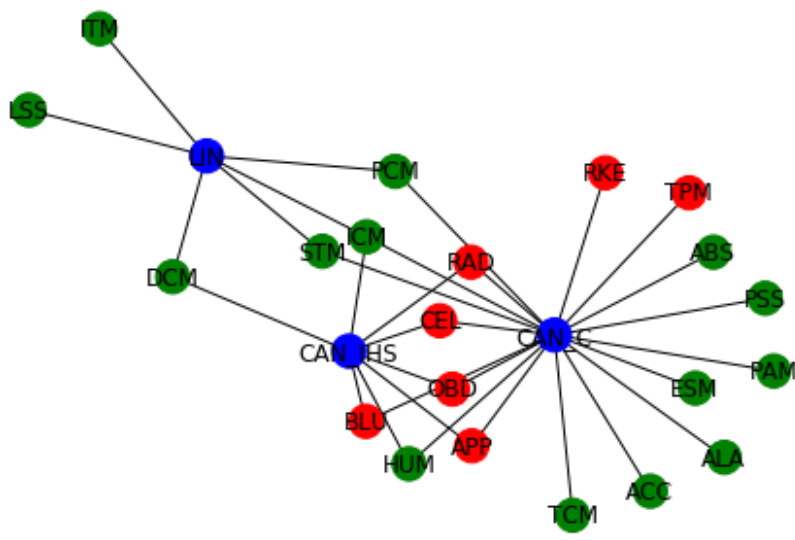


Figure 5.2: Jeep Topology representation

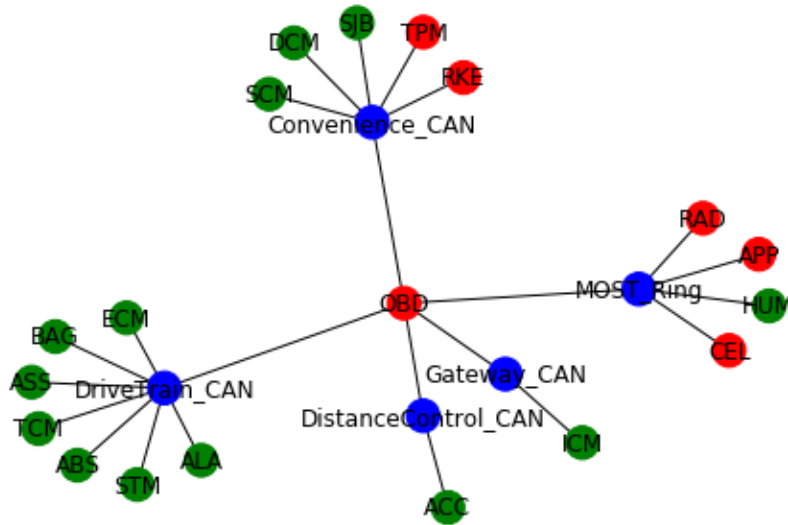


Figure 5.3: Audi Topology representation

ranking, privacy risk attack ranking, operational risk attack ranking and financial risk attack ranking respectively. From these results we first ascertain the correctness of the algorithm in calculating the risk given severity, feasibility and controllability parameters. Moreover we notice the meaningfulness of the results because of different reasons: for instance in the privacy risks ranking (5.5) the only attacks with a risk value different from 0 are the four attacks relative to information theft; this observation is coherent with the fact that attacks not oriented to steal information don't represent a privacy risk. Similarly in the financial risk ranking (5.7) the attacks at the top of the ranking are two attacks relative to frauds since, even if they are not the only ones that cause financial damages, they have very easy methods to be implemented. Finally we generally recognize a good consistency in the resulting ordering of the risks even if this consideration can not be scientifically proved. For instance in the safety risks ranking (5.4) there is a strong difference between the attack "Disable Brakes" valued 7 and "ALA DoS" (Deny of service of active lane assist) valued 3; in fact even if they have a

Listing 5.4: A priori Safety Risks Ranking excerpt (first 8 attacks)

Rs = 8	
Rs = 7	
	DisableBrakes
	Feasibility: 4
	Severity: 4
	Controllability: 3
Rs = 6	
	LockSteering
	Feasibility: 4
	Severity: 3
	Controllability: 3
	Steer
	Feasibility: 3
	Severity: 4
	Controllability: 3
Rs = 5	
	ACD DoS
	Feasibility: 4
	Severity: 3
	Controllability: 2
	Brake
	Feasibility: 3
	Severity: 3
	Controllability: 3
	ActivateAirbag
	Feasibility: 3
	Severity: 3
	Controllability: 3
Rs = 4	
	SpeedUp
	Feasibility: 3
	Severity: 3
	Controllability: 2
Rs = 3	
	ALA DoS
	Feasibility: 4
	Severity: 2
	Controllability: 1

Listing 5.5: A priori Privacy Risks Ranking excerpt (first 10 attacks)

Rp = 8	
Rp = 7	
Rp = 6	
Rp = 5	
Rp = 4	
Rp = 3	
	CaptureImagesFromCameras
	Feasibility: 2
	Severity: 4
	TrackPosition
	Feasibility: 2
	Severity: 4
Rp = 2	
	EavesdropOnConversations
	Feasibility: 2
	Severity: 3
Rp = 1	
	GetTheAddressBook
	Feasibility: 2
	Severity: 2
Rp = 0	
	RemoteAcceleration
	Feasibility: 2
	Severity: 0
	KeylessIgnition
	Feasibility: 2
	Severity: 0
	TamperOdometer
	Feasibility: 4
	Severity: 0
	HideCrashInfo
	Feasibility: 4
	Severity: 0
	ACC DoS
	Feasibility: 4
	Severity: 0
	ACD DoS
	Feasibility: 4
	Severity: 0

Listing 5.6: A priori Operational Risks Ranking excerpt (first 11 attacks)

Ro = 8	
Ro = 7	
Ro = 6	
Ro = 5	
Ro = 4	
	DisableBrakes
	Feasibility: 4
	Severity: 3
	Brake
	Feasibility: 3
	Severity: 4
	LockSteering
	Feasibility: 4
	Severity: 3
	Steer
	Feasibility: 3
	Severity: 4
	SpeedUp
	Feasibility: 3
	Severity: 4
	LockEngine
	Feasibility: 3
	Severity: 4
	LockTheBrakes
	Feasibility: 3
	Severity: 4
Ro = 3	
	RemoteAcceleration
	Feasibility: 2
	Severity: 4
	KeylessIgnition
	Feasibility: 2
	Severity: 4
	ACC DoS
	Feasibility: 4
	Severity: 2
	ACD DoS
	Feasibility: 4
	Severity: 2

Listing 5.7: A priori Financial Risks Ranking excerpt (first 11 attacks)

Rf = 8		
Rf = 7		
Rf = 6		
Rf = 5		
	HideCrashInfo	
	Feasibility:	4
	Severity:	4
Rf = 4		
	TamperOdometer	
	Feasibility:	4
	Severity:	3
	ACC DoS	
	Feasibility:	4
	Severity:	3
	ACD DoS	
	Feasibility:	4
	Severity:	3
	ALA DoS	
	Feasibility:	4
	Severity:	3
	DisableBrakes	
	Feasibility:	4
	Severity:	3
	LockSteering	
	Feasibility:	4
	Severity:	3
	Steer	
	Feasibility:	3
	Severity:	4
Rf = 3		
	RemoteAcceleration	
	Feasibility:	2
	Severity:	4
	KeylessIgnition	
	Feasibility:	2
	Severity:	4
	Brake	
	Feasibility:	3
	Severity:	3

similar implementation simplicity, the first one is much more dangerous and less controllable.

5.3.2 Topology Based Risk Analysis

Concerning the topology based risk analysis applied to the three real architecture that we chose, our tool produced multiple results, as follow:

1. **Risk Values:** Table 5.2 shows the calculated risk values concerning the four risk categories and the total value for each architecture. This results is the best one in showing how our results are consistent with the ones of Miller and Valasek (Table 5.1 shows how the two researchers evaluated them) as the overall value of the worst architecture is much greater than the best one. This means that our methodology and tool takes both the the presence of surfaces or cyber physical systems and the topology well into account.
2. **Attacks Ranking:** Listing 5.8 shows the five most dangerous attacks for each specific network. Due to the high potential damage of some attacks they tend to have high rankings, but in relation to the position of different units inside the car network they actually are more or less considered. This shows the influence of the topology on the attacks ranking. By the way, we remind to the reader that the influence is a parameter that can be tuned according to the needs or considerations of the analyst.
3. **Actions Importance** Listing 5.9 shows the dangerousness of different attacks on specific buses (in the RangeRover topology). In relation to the ECUs available on a specific network this output helps understand which are the most endangered ones and which kinds of attacks can be more disruptive if implemented. In this specific case CAN_HS is more important to protect due to the number of cyberphysical systems like ABS and Adaptive Cruise Control that are connected to it.

Table 5.1: Miller and Valasek results regarding the security of three vehicles: “-” means least hackable, “++” means most hackable)

Car	Attack Surface	Network Architecture	Cyber Physical
2008 Range Rover Sport	-	-	-
2014 Audi A8	++	-	+
2014 Jeep Cherokee	++	++	++

Table 5.2: Risk Evaluation Results regarding the security of three vehicles

	RangeRover	Audi	Jeep
Safety	23.916	37.866	49.583
Privacy	0.000	12.466	14.333
Financial	25.000	37.358	54.500
Operational	32.000	42.358	60.500
Total	80.916	130.050	178.916

4. **Most Traversed Components Ranking:** Listing 5.10 shows the path values ranking of the components in the Audi topology. It describes how many attacks traverse each component considering the specific network topology. Since OBD is the core of the topology, it is the most traversed component as it also analysed from the tool.

5.3.3 Countermeasures Proposal

Differently than with risk analysis, there are no analyses of countermeasure proposals known to the authors, which makes the comparison of our solutions with already validated state of the art proposals impossible, therefore we aim to describe this results and show their feasibility without making comparisons with any previous work. For clarity we still use the same three topologies used for the risk analysis results section.

The final solutions we found optimal have been obtained after inserting multiple constraints on the networks and are visible in 5.4, 5.5 and 5.6. Said constraints are a limit on the maximum distance between every component and the HUM and OBD to ensure that the final solution does not become a

Listing 5.8: Attacks Ranking regarding three vehicles (first 5 attacks)

```
RangeRover
Risk Value = 13.125 , corresponding attack: DisableBrakes
Risk Value = 12.125 , corresponding attack: LockSteering
Risk Value = 9.5    , corresponding attack: Brake
Risk Value = 8.5    , corresponding attack: SpeedUp
Risk Value = 8.125 , corresponding attack: ACC DoS

Audi
Risk Value = 14.625 , corresponding attack: DisableBrakes
Risk Value = 13.625 , corresponding attack: LockSteering
Risk Value = 10.625 , corresponding attack: Steer
Risk Value = 9.625  , corresponding attack: ACC DoS
Risk Value = 9.625  , corresponding attack: ALA DoS

Jeep
Risk Value = 18.0   , corresponding attack: DisableBrakes
Risk Value = 17.0   , corresponding attack: LockSteering
Risk Value = 15.5   , corresponding attack: Steer
Risk Value = 13.5   , corresponding attack: Brake
Risk Value = 13.0   , corresponding attack: ACC DoS
```

Listing 5.9: Buses Actions Importance in RangeRover

```
CAN_MS
DiagnosticInj = 35.16406249999999
Inject        = 54.12239583333331
Listen        = 0
DoS           = 0

CAN_HS
DiagnosticInj = 61.700737847222214
Inject        = 348.9468315972222
Listen        = 0
DoS           = 380.162109375
```

Listing 5.10: Paths Values in Audi

OBD	has a paths value of: 583.0
MOST_Ring	has a paths value of: 447.0
DriveTrain_CAN	has a paths value of: 306.0
Convenience_CAN	has a paths value of: 301.0
HUM	has a paths value of: 144.0
CEL	has a paths value of: 139.0
APP	has a paths value of: 135.0
RKE	has a paths value of: 123.0
RAD	has a paths value of: 119.0
TPM	has a paths value of: 119.0
DistanceControl_CAN	has a paths value of: 84.0
ECM	has a paths value of: 42.0
ACC	has a paths value of: 42.0
ALA	has a paths value of: 30.0
ICM	has a paths value of: 30.0
Gateway CAN	has a paths value of: 30.0
ABS	has a paths value of: 24.0
SCM	has a paths value of: 18.0
TCM	has a paths value of: 18.0
DCM	has a paths value of: 12.0
STM	has a paths value of: 0
SJB	has a paths value of: 0
BAG	has a paths value of: 0
ASS	has a paths value of: 0

sequence of single components interconnected by gateways (results maximum distance is 6 for the Audi and 5 for the other topologies). The final risk value has lowered for all three networks, going from 178,9 to 105 for the Jeep, from 130 to 87,6 for the Audi and from 80,9 to 65 for the RangeRover.

These solutions are not necessarily feasible as we anticipated before, more constraints may be required between multiple components to ensure that the network could be operable in real world scenarios. Not knowing the actual requirements for these specific network we propose a solution with one of the only constraints we know is probably required, which is that the MOST ring of the Audi should remain with the same components on it as those components are probably not compatible with other architectures. In figure 5.7 it is possible to see how the algorithm (with maximum distance constraint set to 5) modifies the structure of the network following this constraint about MOST ring. The final risk value is 114, so it is still possible to optimize the network but significantly less than without constraint.

Solutions without constraints are completely meaningless since they build chains of components one behind the other with an excess of gateways insertion as shown in figures 5.8, 5.9 and 5.10.

Finally, as the algorithm makes one step at a time, we implemented the possibility for the analyst to see the development of the countermeasures step by step, giving him the chance to stop at the solution he reputs most valid. In this way the analyst can avoid to let the algorithm reach solutions that are unfeasible in real world scenarios. Consider for example 5.11: this is the first step proposed by the algorithm for the Audi architecture, which represents the insertion of a gateway inbetween the OBD component and the other part of the network, and decreases the risk value from 130.05 to 114.75. The decrease of risk for each possible attack is visible in listing 5.12. If the analyst reputs this an optimal solution he can simply stop. The second step proposed by the algorithm (once accepted the first one) is the move of the Head Unit Module to the bus named DriveTrain CAN as shown in figure 5.12 and it decreases the risk value from 114.75 to 110.68. The decrease of

risk for each possible attack is visible in listing 5.13.

As a last note regarding the algorithm, it often chooses to apply a gateway instead of moving a component, which is a predictable behaviour due to the gateway adding distance from every other component. This solution may be too expensive in some cases, making it preferable to apply movements. For this reason it could be worth adding a cost function that makes it possible to change the importance of one action over the other. Currently the only preference towards movements comes when two potential solutions one of each kind have the same risk value, in which case the movement is chosen instead of the gateway. We leave different possible improvements to future works.

We finally append this log file excerpt (5.11) that could help in explaining the process of gateway insertion decision of the algorithm during the first step when applied to the Audi topology:

Listing 5.11: Countermeasures algorithm proceeding

```
2018-07-04 18:59:50,665 Trying to insert a GW in ECM
2018-07-04 18:59:50,675 MAX Distance Violation: HUM ECM 6
2018-07-04 18:59:50,682 Constraints violated
2018-07-04 18:59:50,694 Trying to insert a GW in ABS
2018-07-04 18:59:50,705 MAX Distance Violation: HUM ABS 6
2018-07-04 18:59:50,712 Constraints violated
2018-07-04 18:59:50,724 Trying to insert a GW in BAG
2018-07-04 18:59:50,736 MAX Distance Violation: HUM BAG 6
2018-07-04 18:59:50,742 Constraints violated
2018-07-04 18:59:50,754 Trying to insert a GW in TCM
2018-07-04 18:59:50,765 MAX Distance Violation: HUM TCM 6
2018-07-04 18:59:50,771 Constraints violated
2018-07-04 18:59:50,784 Trying to insert a GW in ALA
2018-07-04 18:59:50,795 MAX Distance Violation: HUM ALA 6
2018-07-04 18:59:50,804 Constraints violated
2018-07-04 18:59:50,820 Trying to insert a GW in STM
2018-07-04 18:59:50,831 MAX Distance Violation: HUM STM 6
2018-07-04 18:59:50,837 Constraints violated
2018-07-04 18:59:50,849 Trying to insert a GW in ASS
2018-07-04 18:59:50,861 MAX Distance Violation: HUM ASS 6
2018-07-04 18:59:50,867 Constraints violated
```

2018-07-04 18:59:50,884 Trying to insert a GW in OBD
2018-07-04 18:59:50,944 New risk = 114.75
2018-07-04 18:59:50,957 Trying to insert a GW in DCM
2018-07-04 18:59:50,968 MAX Distance Violation: HUM DCM 6
2018-07-04 18:59:50,974 Constraints violated
2018-07-04 18:59:50,986 Trying to insert a GW in RKE
2018-07-04 18:59:50,999 MAX Distance Violation: HUM RKE 6
2018-07-04 18:59:51,005 Constraints violated
2018-07-04 18:59:51,018 Trying to insert a GW in SCM
2018-07-04 18:59:51,029 MAX Distance Violation: HUM SCM 6
2018-07-04 18:59:51,035 Constraints violated
2018-07-04 18:59:51,048 Trying to insert a GW in SJB
2018-07-04 18:59:51,066 MAX Distance Violation: HUM SJB 6
2018-07-04 18:59:51,075 Constraints violated
2018-07-04 18:59:51,092 Trying to insert a GW in TPM
2018-07-04 18:59:51,105 MAX Distance Violation: HUM TPM 6
2018-07-04 18:59:51,111 Constraints violated
2018-07-04 18:59:51,123 Trying to insert a GW in OBD
2018-07-04 18:59:51,169 New risk = 114.75
2018-07-04 18:59:51,182 Trying to insert a GW in ICM
2018-07-04 18:59:51,193 MAX Distance Violation: HUM ICM 6
2018-07-04 18:59:51,200 Constraints violated
2018-07-04 18:59:51,272 Trying to insert a GW in ACC
2018-07-04 18:59:51,283 MAX Distance Violation: HUM ACC 6
2018-07-04 18:59:51,290 Constraints violated
2018-07-04 18:59:51,312 Trying to insert a GW in OBD
2018-07-04 18:59:51,357 New risk = 114.75
2018-07-04 18:59:51,370 Trying to insert a GW in HUM
2018-07-04 18:59:51,380 MAX Distance Violation: HUM SCM 6
2018-07-04 18:59:51,385 MAX Distance Violation: HUM ECM 6
2018-07-04 18:59:51,391 MAX Distance Violation: HUM TCM 6
2018-07-04 18:59:51,396 MAX Distance Violation: HUM ABS 6
2018-07-04 18:59:51,401 MAX Distance Violation: HUM STM 6
2018-07-04 18:59:51,406 MAX Distance Violation: HUM ACC 6
2018-07-04 18:59:51,415 MAX Distance Violation: HUM ALA 6
2018-07-04 18:59:51,422 MAX Distance Violation: HUM ICM 6
2018-07-04 18:59:51,427 MAX Distance Violation: HUM DCM 6
2018-07-04 18:59:51,432 MAX Distance Violation: HUM SJB 6
2018-07-04 18:59:51,438 MAX Distance Violation: HUM BAG 6
2018-07-04 18:59:51,443 MAX Distance Violation: HUM ASS 6
2018-07-04 18:59:51,448 MAX Distance Violation: HUM TPM 6

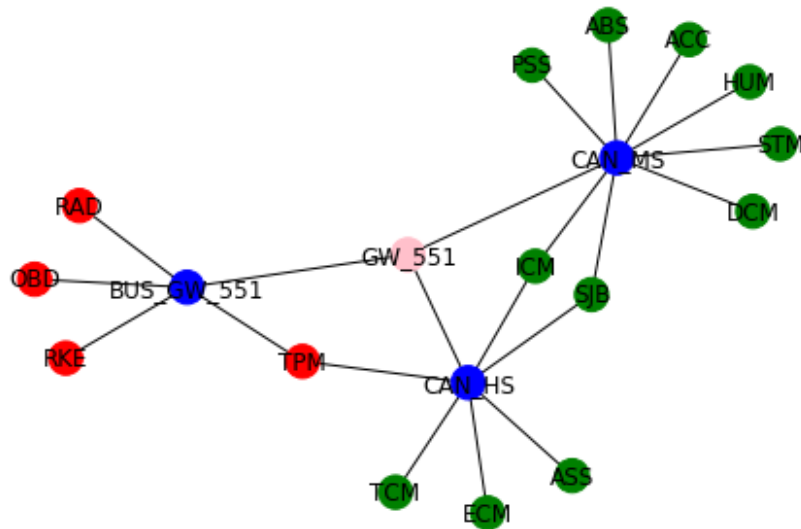


Figure 5.4: RangeRover Optimal Solution with Max-Distance Constraints

```

2018-07-04 18:59:51,453 MAX Distance Violation: HUM RKE 6
2018-07-04 18:59:51,459 Constraints violated
2018-07-04 18:59:51,471 Trying to insert a GW in OBD
2018-07-04 18:59:51,522 New risk = 114.75
2018-07-04 18:59:51,535 Trying to insert a GW in RAD
2018-07-04 18:59:51,578 New risk = 125.84999999999997
2018-07-04 18:59:51,591 Trying to insert a GW in CEL
2018-07-04 18:59:51,635 New risk = 125.84999999999997
2018-07-04 18:59:51,648 Trying to insert a GW in APP
2018-07-04 18:59:51,698 New risk = 122.45476190476191
2018-07-04 18:59:51,712 ----> I decided to insert a GW in OBD
  
```

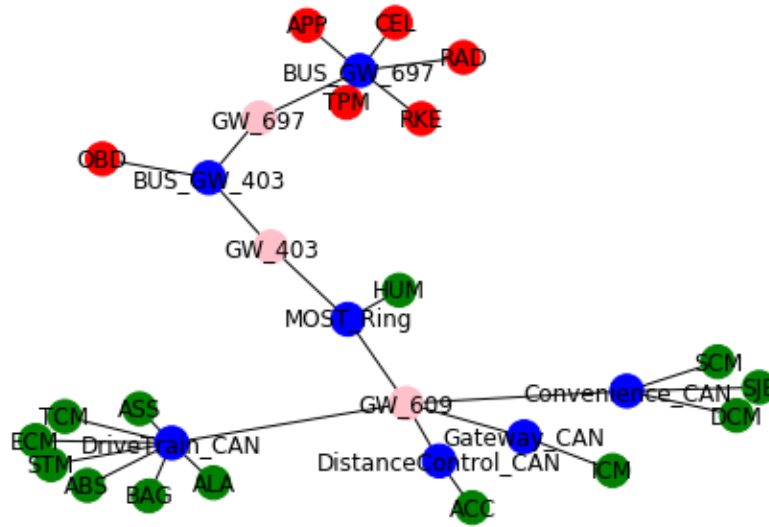


Figure 5.5: Audi Optimal Solution with Max-Distance Constraints

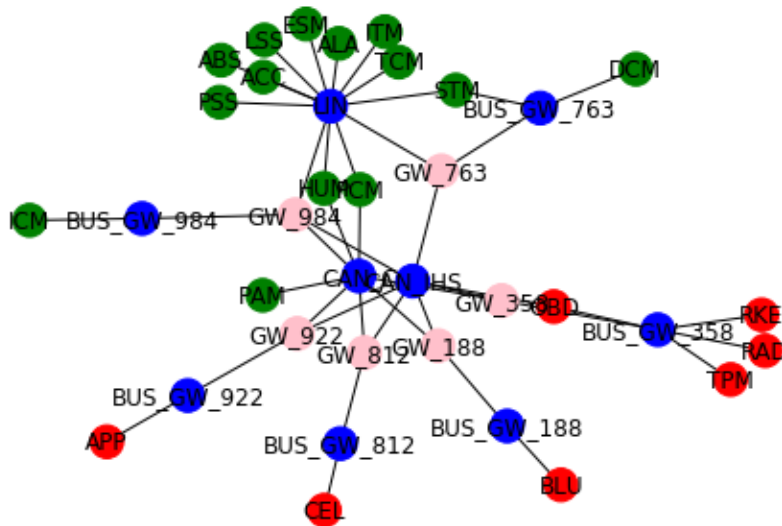


Figure 5.6: Jeep Optimal Solution with Max-Distance Constraints

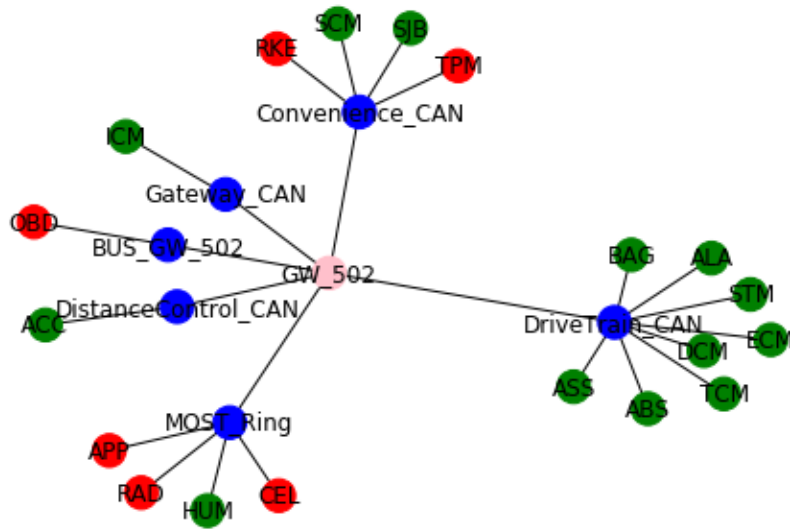


Figure 5.7: Audi Solution with Max-Distance and Most-Ring Constraints

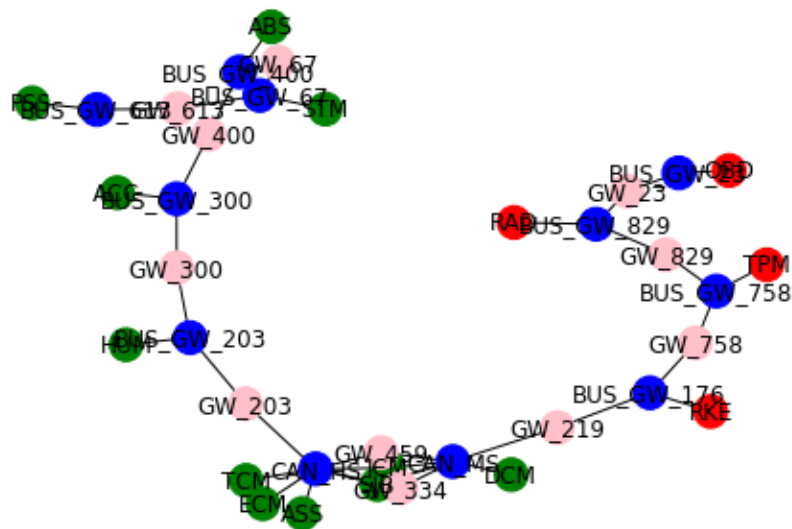


Figure 5.8: RangeRover Optimal Solution without Constraints

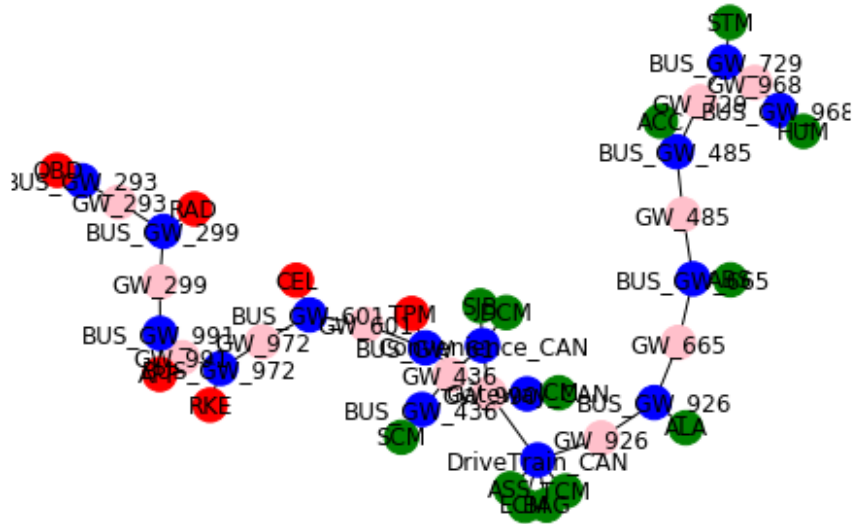


Figure 5.9: Audi Optimal Solution without Constraints

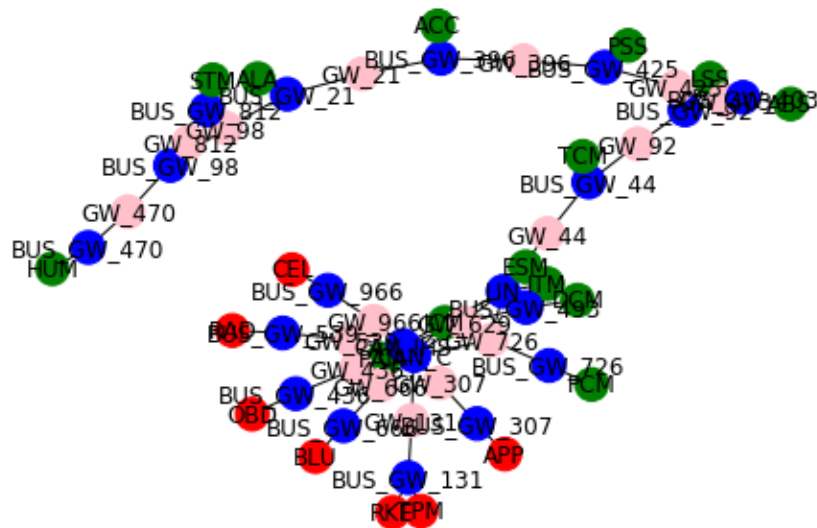


Figure 5.10: Jeep Optimal Solution without Constraints

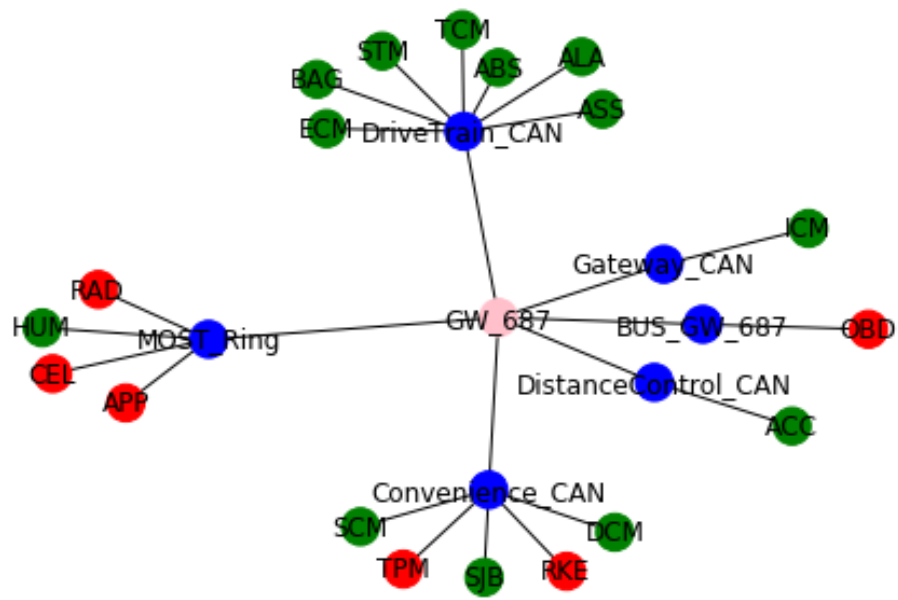


Figure 5.11: Audi intermediate solution after first step

Listing 5.12: Attacks Risks Variation after first step

```
I have created a GW in OBD
Risk Variation = 1.125 , corresponding attack: RemoteAcceleration
Risk Variation = 1.125 , corresponding attack: ACC DoS
Risk Variation = 1.125 , corresponding attack: ALA DoS
Risk Variation = 1.125 , corresponding attack: DisableBrakes
Risk Variation = 1.125 , corresponding attack: Brake
Risk Variation = 1.125 , corresponding attack: ActivateAirbag
Risk Variation = 1.125 , corresponding attack: LockSteering
Risk Variation = 1.125 , corresponding attack: Steer
Risk Variation = 1.125 , corresponding attack: SpeedUp
Risk Variation = 1.125 , corresponding attack: LockDoors
Risk Variation = 0.75 , corresponding attack: KeylessIgnition
Risk Variation = 0.75 , corresponding attack: OverheatEngine
Risk Variation = 0.75 , corresponding attack: LockTheBrakes
Risk Variation = 0.6 , corresponding attack: TurnOffHeadlights
Risk Variation = 0.4 , corresponding attack: LockEngine
Risk Variation = 0.2 , corresponding attack: EavesdropOnConversations
Risk Variation = 0.2 , corresponding attack: CaptureImagesFromCameras
Risk Variation = 0.2 , corresponding attack: GetTheAddressBook
Risk Variation = 0.2 , corresponding attack: TrackPosition
Risk Variation = 0.0 , corresponding attack: TamperOdometer
Risk Variation = 0.0 , corresponding attack: HideCrashInfo
Risk Variation = 0.0 , corresponding attack: ACD DoS
Risk Variation = 0.0 , corresponding attack: TurnOffEngine
Risk Variation = 0.0 , corresponding attack: EngageSeatBelt
```

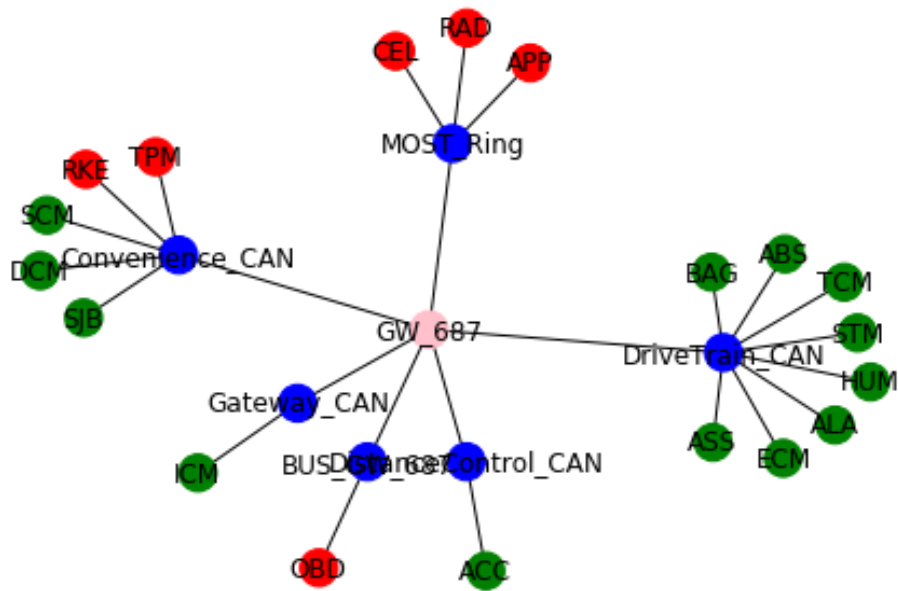


Figure 5.12: Audi intermediate solution after second step

Listing 5.13: Attacks Risks Variation after second step

```
Ho spostato HUM in DriveTrain_CAN
Risk Variation = 1.4    , corresponding attack: TurnOffHeadlights
Risk Variation = 0.933 , corresponding attack: LockEngine
Risk Variation = 0.466 , corresponding attack: CaptureImagesFromCameras
Risk Variation = 0.466 , corresponding attack: GetTheAddressBook
Risk Variation = 0.466 , corresponding attack: TrackPosition
Risk Variation = 0.33  , corresponding attack: EavesdropOnConversations
Risk Variation = 0.0   , corresponding attack: RemoteAcceleration
Risk Variation = 0.0   , corresponding attack: KeylessIgnition
Risk Variation = 0.0   , corresponding attack: TamperOdometer
Risk Variation = 0.0   , corresponding attack: HideCrashInfo
Risk Variation = 0.0   , corresponding attack: ACC DoS
Risk Variation = 0.0   , corresponding attack: ACD DoS
Risk Variation = 0.0   , corresponding attack: ALA DoS
Risk Variation = 0.0   , corresponding attack: DisableBrakes
Risk Variation = 0.0   , corresponding attack: Brake
Risk Variation = 0.0   , corresponding attack: ActivateAirbag
Risk Variation = 0.0   , corresponding attack: LockSteering
Risk Variation = 0.0   , corresponding attack: Steer
Risk Variation = 0.0   , corresponding attack: OverheatEngine
Risk Variation = 0.0   , corresponding attack: SpeedUp
Risk Variation = 0.0   , corresponding attack: TurnOffEngine
Risk Variation = 0.0   , corresponding attack: EngageSeatBelt
Risk Variation = 0.0   , corresponding attack: LockTheBrakes
Risk Variation = 0.0   , corresponding attack: LockDoors
```

Chapter 6

Conclusions and Future Works

This thesis focused on threat assessment in the design phase of vehicular internal networks aimed to identify possible countermeasures to be adopted. We proposed a methodology to make a security risk analysis on automotive architectures which is able to propose risk values for different attack scenarios (described as attack trees) differentiating the results according to the different topologies and assigning a global risk evaluation to each of them. Then we also proposed an algorithm that suggests countermeasures related to the specific architecture reaching a local optimum with the aim of minimizing the global risk evaluation of the topology.

Both of the propositions have been implemented through a simple python tool which we hope can help analysts while studying the architecture of a vehicle, whether while designing the network of a new vehicle or assessing the network of an already existing one.

The methodology results, obtained through the application of the tool to three real topologies, have been compared with already known analyses from known experts of the field to prove their correctness and they also showed the hints richness that can be deduced. On the other hand, countermeasure proposals, not having any already existing work to which we could compare them, have been analysed and appear to be functional to the scope of the work leading to coherent local optimal solutions.



Figure 6.1: Google's first self-driving concept

However we still recognize some limitations in our works:

- **Attack Trees are not exhaustive.** In fact we didn't consider some new scenarios like attacks to the eCall [14], attacks to the eToll [15], traffic manipulation attacks or remote autonomous drive. Moreover the attack methods that we hypothesized are not the only ones possible and they could also evolve very much in next years. So future works may consider to improve our database adding different known attacks and completing the list with less known ones.
- **Every component can act as a gateway.** One of the weakest assumptions that we made in our methodology is that every component, if hacked, could potentially be a gateway if it is communicating on two different buses. This is not always true since a component could only listen on a bus so future works may improve the tool by better investigating this possibility in order to restrict this assumption.
- **Simplified Gateways.** The only effect that causes the insert of a gateways in a topology is the distance increase among components. Future works may implement in the algorithm an automatic generation of firewall rules for a better shielding of the networks segments.
- **Local optimum.** Final solutions obtained by the countermeasures

generation algorithm do not reach a global optimum but only a local optimum. This means that the final topology configuration is just the incremental result of small adjustments made step by step starting from the initial configuration. Future works may consider the design of an algorithm that identifies the global optimal topology structure only starting from the components list and requirements.

- **Costs.** Our countermeasures algorithm doesn't differ countermeasures considering the costs: every countermeasure implicitly requires the same expense. Future works may consider to improve the tool by embedding cost evaluations in the decision phase of the algorithm.

Furthermore, next studies may consider a further development of the tool developing different countermeasures which may complement the ones we already propose and it may be interesting to study the outputs of the tool changing the base values or changing how the tool considers the importance of some attacks and solutions.

New technologies will arrive (6.1) and new risks will arise. The more cars will be connected, the more attackers will have access points to enter vehicles networks. For these reasons in the future, automotive industry will have to be very efficient in concentrating costs and investments where is more important knowing that it's not possible to easily prevent all attacks. Approaches like the one described in this thesis could facilitate this process, approaches that aim to prioritize countermeasures following mainly the risk minimization.

Bibliography

- [1] A. Ruddle, D. Ward, B. Weyl, S. Idrees, Y. Roudier, M. Friedewald, T. Leimbach, A. Fuchs, S. Gürgens, O. Henniger *et al.*, “Security requirements for automotive on-board networks based on dark-side scenarios,” EVITA Deliverable D.2.3, 2009.
- [2] R. N Charette, “This car runs on code,” vol. 46, 02 2009.
- [3] K. Koscher *et al.*, “Experimental security analysis of a modern automobile,” May 2010. [Online]. Available: <http://www.autosec.org/pubs/cars-oakland2010.pdf>
- [4] S. Checkoway *et al.*, “Comprehensive experimental analyses of automotive attack surfaces,” August 2011. [Online]. Available: <http://www.autosec.org/pubs/cars-usenixsec2011.pdf>
- [5] SAE International. Cybersecurity guidebook for cyber-physical vehicle systems. [Online]. Available: <https://www.sae.org/standards/content/j3061/>
- [6] C. Valasek and C. Miller, “Adventures in automotive networks and control units,” August 2013. [Online]. Available: http://www.ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf
- [7] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, “A stealth, selective, link-layer denial-of-service attack against automotive networks,” in *Detection of Intrusions and Malware, and Vulnerability Assessment - 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings*, ser. Lecture Notes in Computer Science, M. Polychronakis and M. Meier,

- Eds., vol. 10327. Springer, 2017, pp. 185–206. [Online]. Available: https://doi.org/10.1007/978-3-319-60876-1_9
- [8] C. Miller and C. Valasek, “A survey of remote automotive attack surfaces,” August 2014. [Online]. Available: <http://illmatics.com/remote%20attack%20surfaces.pdf>
- [9] —, “Remote exploitation of an unaltered passenger vehicle,” August 2015. [Online]. Available: <http://illmatics.com/Remote%20Car%20Hacking.pdf>
- [10] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner, “Toward a secure system engineering methodology,” in *Proceedings of the 1998 Workshop on New Security Paradigms*, ser. NSPW '98. New York, NY, USA: ACM, 1998, pp. 2–10. [Online]. Available: <http://doi.acm.org/10.1145/310889.310900>
- [11] National Instruments, “Controller area network (CAN) overview.” [Online]. Available: <http://www.ni.com/white-paper/2732/en/>
- [12] —, “Introduction to the local interconnect network (LIN) bus.” [Online]. Available: <http://www.ni.com/white-paper/9733/en/>
- [13] H. M. Song, H. R. Kim, and H. K. Kim, “Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network,” January 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7427089/>
- [14] R. Oorni and A. Goulart, “In-vehicle emergency call services: ecall and beyond,” *IEEE Communications Magazine*, vol. 55, no. 1, pp. 159–165, January 2017.
- [15] G. S. et al, “Automated toll cash collection system for road transportation,” *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 2, pp. 216–224, February 2015.