UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE & ENGINEERING DEPARTMENT

PhD Thesis

in Computers and Information Technology

Security in Automotive Linux

Jan Alexandru Văduva

Scientific Supervisor:
Prof. dr. ing. Răzvan-Victor RUGHINIŞ

BUCHAREST

2021

University POLITEHNICA of Bucharest

Abstract

Security in Automotive Linux

by Jan Alexandru Văduva

Scientific Supervisor

Professor Răzvan-Victor RUGHINIŞ

Department of Computer Science

Modern automotive approaches are composed from a multitude of highly complex modules with extensive functionalities which are interconnected through a wide variety of network options, most of the information regarding them being kept secret by the automotive industry. In recent years it was proven that this type of perspective becomes highly problematic when exposing those modules to the outer world transforming them in high-paying targets exposed to remote cyber attacks. Worst case scenario would be when the attacker obtains full access to critical systems around the car such as engine, breaking or steering.

This thesis proposes a change of perpective where multiple stakeholders, crowd knowledge and vital security are at the core of design for the architecture. In order to meet this objective we proposed a new architecture model with core and edge devices which have different security requirements and maintainance support. For the devised architecture we submit security hardening functionalities in order to enhance the new security by design paradigm. The first solution we present is a based on the ARM`s TrustZone technolgy, this is followed by a binary analysis component and a host intrusion detection system. In addition we bring forward a technique able to mitigate certain attacks and ensure the integrity of critical applications based on an the source code analysis of updates before applying them. Finally we evaluated the proposed implementations and demonstrated their capabilities to alleviate prospective attack vectors available inside the automotive operating system.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I struggle to write acknowledgements because I consider words are not able to do justice to the level of gratitude I have for those supporting me along the process of writing this thesis, which proved really time consuming and I believe the end results benefited greatly from the input of a community of professors, colleagues, friends and family. It is an honour for me to have the opportunity to put in writing my thanks to these people who supported and inspired me throughout the whole process.

I am deeply thankful to my adviser Răzvan Rughiniș, my thesis comitee members Răzvan Deaconescu, Emil Slușanschi, Catalin Leordeanu and other collaborators from University Politehnica of Bucharest Nicolae Țăpuș, Mihai Chiroiu, Dragos Niculescu, Alexandru Radovici, Ștefan-Dan Ciocîrlan who constantly pointed me towards the right direction.

I am thankful to my Systems group colleagues Daniel Rosner, Iulia Florea, Ioana Culic, Eduard Stăniloiu, Răzvan Nițu, Flavia Oprea, Octavian Grigorescu, Cristian Trancă, Lucian Mogoșanu, Florin Stancu, Mihai Carabaș, Cristian Pătru which supplied me with useful research information and comments on the published work.

I am fortunate to also work with Daniel Pirjan, Ovidiu Mihalachi, Bogdan Mirea, Andreea Guran, Cristi Irimia, Dragos Ducu, Tudor Ciochină, Lucian Perisoara who have been a constant source of encouragement and experience. Also I have to thank Daniel Bornaz who had the patience to mold my Linux and engineering skills and helped me along the process.

I thank to all the students with whom I collaborated, exchanged ideas and spent a wonderful time. I am really happy to have meet them, there a quite a great bunch of people, really hard to mention them all, but I can name here the UPBDrive group where we try to build a Formula Student racing car, USO and RL groups, Radu Chișcariu, Vlad Pașca or Ștefan Dascălu.

I would also like to thank my wife, parents, brother and even my new born child which provided their unfailing support and encouragement throughout my years of study, research and writing for the thesis.

Many others helped develop ideas in this process, provided support or advice. For all of them I have to thank for making all this enjoyable even through the hardest moments.

Chapter 1. Understanding automotive security

In this chapter I will define and describe the context and technologies used. It will also contain the main research questions and contributions provided for the domain of automotive security, its threats, vulnerability and countermeasures.

1.1 Context

Here we discuss the changes the open source involved to various industries and companies together with its advantages:

- Minimal cost for research and development for new innovation
- Advantage for better development productivity
- Integration or involvement of users or concerned parties in the development process
- Better product quality and customer satisfactions
- Provide access to cooperate for internal and external innovations

Also we provide an argument for the role of open source in the automotive bussines and in the same time the advantages such transition could have for all the involved shareholders:

Understand how the car works: The automotive industry offers us some incredible products with a lot of interconnected electronic components and little information on how they work. Learning how the components interact within each other or with the outside world we will be better at determining and repair issues.

Work on car electrical systems: Vehicles are becoming more electrical than mechanical, they are usually known only by the dealerships and other automotive shareholders. Understanding how these electronic components work from that information or the information the automotive manufacturers outsource can help get around the problems and provide tools in order to investigate them.

Modify the car: Understanding the way the car works helps not only with being able to provide improvements but also add extra support for third-party peripherals which could work seamlessly.

Discover undocumented components: Most of the components available in cars are undocumented or disabled. Learning about how to use them is beneficial and enhances the functionality and potential of the car.

Validate the security of the vehicle: A few years back until the 2010 article presenting exploits of a car, vehicle safety did not account for any security threats. After that event everyone started looking at a car like at any desktop system. This also means we require validating and auditing our cars since they are the ones caring our whole family around cities and outside them at a whole range of speeds. Knowing more about how to hack your car, helps everyone learn about their vulnerabilities and in the end help with improving safety standards.

Help the automotive industry: The gathered knowledge provides guidelines for discovering vulnerabilities and also solutions for protecting this ecosystem. This can translate as benefits for automotive industry helping them provide implementations for security practices and other researchers as well by providing a way of communicating their findings.

1.2 Thesis scope

The scope of this thesis is to analyse the implications and impact of security metrics applied to the automotive industry, one where until 2010 security was not considered a concern. When discussing about security metrics we discuss about more than one category. There is more than one way of classifying them. One such method is the accountability of the maturity level for the processes contributing to the system security, another is represented by the degree of presence for a certain security characteristic.

Beside the information security metrics, other domains such as the financial one were able to make progress on quantitative methods for risk measurement, something that for information security would mean realistic assumptions and reliable results. By advancing the field of security measures and metrics is beneficial to areas from design and implementation to operations. However, security measurement is a hard to solve problem. Further evidence is represented by the fact that, Enterprise-Level Security Metrics, was included in the Hard Problem List maintained by the INFOSEC Research Council.

We need to do for information security what quants did for the financial industry. We need to be able to understand, score, package, quantify, measure and be also able to trade the risks in an efficient manner similar to how the financial markets proceed. A quick resolution to this problem will not be something we are expecting, also when this will happen most likely there will be a couple of aspects that will remain as low hanging fruits. Some of the factors that have an impact of this progress are:

- No good estimators available for a system security.
- Too much reliance on subjective and human generated inputs.
- Extended and sometimes false means of measurements.
- Lack of proper understanding of the insides of security mechanisms.

However, lately for the automotive business the paradigm started to change since UNECE WP.29 started making the certification process for a cybersecurity management system mandatory and basically forced the industry to implement ISO/SAE 21434 [130]. All this because there is no mathematical formula that could be used to obtain the trust level required by each and every one so working with smaller components which could be formally validated or semiformal depending on the component requirements. So the problem revolves around onto which components the application should run depending on the needed security level.

1.3 Research questions

As already stated in numerous research papers and even in the media lately the vehicle available today is very different than the one we had 10 years ago, not only that but the technology involved around it changed and was incorporated inside. The internet reached the vehicle and a large number of services will be available inside our own car in the following years.

As indicated by [20] the vehicles available on the road today have between 30 and 100 computers inside also called ECUs in the automotive industry. This is even bigger than most startups and small companies but with no security support. The number of lines inside the vehicles today revolves around hundreds of millions and since we can have a vulnerability available for each 10 lines of codes and the fact that these vehicles will be connected through the internet to the infrastructure and other vehicles. This makes security imperative and as documented in a security

report since 2016 until 2019 the number of security issues reported inside vehicles was increased seven times.

To match these challenges, we propose these research questions:

- Q1 How prepared is the automotive industry for moving from security by obscurity to security by design?
  - a Why do we require the shift towards security by design?
- Q2 How would an automotive digital architecture where we could apply security by design with multiple stakeholders, crowd knowledge and vital security with integrated new services and technologies would look like?
  - a What type of sensors and services can be connected to such as an architecture?
  - b How can we make this integration process easier to understand by non-technical background individuals?
- Q3 What improvements could be employed for enhancing existing security by design solutions in the automotive industry?

1.4 Thesis contributions

This thesis started with an analysis validating the problems available inside the existing vehicles due to the security by obscurity paradigm that was promoted so far in the industry. We validated these problems by constructing, based on Raspberry Pi hardware, a platform which could be used as an open platform which facilitate the interaction with the vehicle.

On this platform we managed to integrate Wyliodrin web platform in order to facilitate the integration of new sensors and functionalities. Since we wanted to construct a testbed which simulate a secure by design architecture we arrived to a secondary problem of the field, the fact that the change from security by obscurity to security by design would require the interaction with technologies for groups of individuals which do not have a technical background. We managed to solve this with the help of Wyliodrin which was also used as a mechanism to extend the functionalities in a manner which is easy to understand and program by such individuals. We took the platform even more far and integrated an Android companion in order to facilitate the interaction and also incorporate inside the power of the smartphones in the ecosystem.

Since all this functionalities prove to be quite a handful for the Raspbian OS, optimization was required, for that I used a Yocto Project based Linux distribution. Of course I chose as staring point the Automotive Grade Linux (AGL) distribution which also has hardware support for Raspberry Pi and added the required Wyliodrin support on top of that. All we were required to do if a new functionality was provided was to make sure the OS also had the underlying dependencies integrated inside.

At this point we managed to establish the open digital architecture that we could use as a testbed. Now I have to secure the ecosystem. In order to do this we have a look at the secure automotive software update system and have taken a look at the trusted execution probes mechanism for introspection of the system implemented through TrustZone SMC calls which proved not being industry graded for our purposes. We proposed a security hardening where we employ the binary analysis and a host intrusion detection system responsible for anomaly detection. To that we also added static code analysis where we provide a mechanism of scoring the impact to the attack surface of any update employed to the system. With all this we managed to construct a

secure by design minimum architecture testbed and the details of the work will be presented in the following sections.

To summarize the contributions and match them to the research questions defined in previous chapter we provide the following summary:

A1 With the help of the work carried out in [3] with the Raspberry Pi hardware we present the state of the systems as they are, how easily a vulnerability can be exploited and the consequences of doing that also what it would mean to obtain security by design as described in state of the art [211].
   a. An analysis of the state-of-the-art [211] over the process of standardization that is being done at the moment presented a serious lack of security and a lack of awareness over the impact of new technologies on the automotive industry.

A2 Starting from the work from A1 where we developed a centralized system based on Raspberry Pi, a widely available hardware platform, we began to construct an automotive digital architecture where we mimic multiple stakeholders, crowd knowledge and integrated a number of new services and technologies. On the hardware platform we configured a custom Yocto Project based Linux distribution [118] and [119] in order to have full control over the hardware capabilities and we added on top extensions for easier control and customization. For extensibility and ease of use we stopped over Wyliodrin and we added support for extra sensors to control as part of [117], we added support for speech recognition as part of [115] and in order to easily control all this we defined an Android companion with [116].
   a. We integrate new types of sensors and services with the help of the work developed in contributions [115], [116] and [120] described above.
   b. We facilitate the integration process for stakeholders with various background with the help of the contributions available in [117], [118] and [119] also described in details in A2 section.

A3 We investigated the secure automotive software update system and have taken a look at the trusted execution [111] probes mechanism for introspection of the system implemented through TrustZone SMC calls which proved not being industry graded for our purposes. We proposed a security hardening solution with both a binary analysis solution [109] and a host intrusion detection system [110], both providing an enhancement for both static and dynamic introspection of the system responsible for anomaly detection. To that we also added static code analysis with [121] where we provide a mechanism of scoring the impact to the attack surface of any update employed to the system.

1.5 Thesis structure

In this section, part of Chapter 1, we present the structure of the thesis and a brief overview of how we addressed our objectives in each chapter and what contributions were included in them together with how the chapter influences the objectives of the thesis.

This thesis is structured in seven chapters, which each one addressing one of the four main objectives: identifying how prepared is the industry for a shift towards security by design from security by obscurity; identify how a digital architecture which integrated multiple sensors and services could apply security by design and how this could be integrated inside the process of

development for non-technical background individuals; identify improvements which could be done in order to extend existing security of the system.

In Chapter 2 we provide a survey of existing literature on scientific research and security standards developed and applicable in the automotive industry and how they have implications towards all the industry processes. Based on this we presented how a secure software development lifecycle could look and their implications on all phases of development for a reference hardware architecture. This work is covered in one article.

In Chapter 3 we hijack the ECU systems inside the vehicle and connect them to the outside world enabling the possibility to access the vehicle remotely in a secure way through a user-friendly interface using open source software and the Raspberry Pi hardware platform. This work is covered in one published article.

In Chapter 4 we integrate Yocto Project and provide a custom configurable operating system to the hardware platform, one more fit to the automotive requirements and integrate the Wyliodrin web platform in order to easily integrated new hardware support and also help non-technical individual get a grasp of computer science theoretical notions with a lower learning curve and be able to provide applications related to their needs. This work is covered in one article and two books that were published while working for the PhD thesis.

In Chapter 5 we bring support for a service providing vocal commands as a way to extend the supported services and provide an Android companion support not only to facilitate the interaction with the testbed easier but also provide extensibility for it. This work covers three articles that I published.

In Chapter 6 we take what we built in previous chapters and provide for mechanisms for securing this ecosystem that we built. We employ trusted execution, binary and system call analysis for both build time and runtime evaluation of the operating system and also we provide a mechanism for updates analysis before applying them to the system. All these covering four of the published articled that we include in this thesis.

Chapter 7 represents the conclusion of this work providing an overview of all the contributions and defining future exploratory space and how the present work could be improved.

Chapter 2. State of the art

In this chapter we discuss the main scientific and standardization work carried our around the automotive domain, an analysis of the most significant research programs developed under the European Commission umbrella which provided the basis for the research carried out in the automotive industry. Based on this research we present a use case for an automotive specific Software Development Lifecycle process involving a modified TARA methodology applied to the development for a reference hardware architecture where security and safety plays a central role.

Looking at the automotive industry over the history, its existence safety was considered one of the most important characteristics, leaving security almost unaddressed - especially for the software intensive components using security through obscurity as the default go to option.

Today we look at vehicles as only a component of the ecosystem of connected mobility and smart cities and in this context the connected vehicle is considered the most critical component with a myriad of attack points available on it but also from backend systems and infrastructure. So much so that UNECE regulation considered cybersecurity on top of the agenda for automotive industry for over 60 markets as it becomes type-approval relevant.

As described by the HoliSec project [127] security engineering represents an engineering discipline concerned with the security of a system, this includes all processes from system design to implementation and maintenance; however, as they themselves mention, data security mechanisms for safety-related embedded systems are the key point for automotive security and requires research and investigations.

With the problems available solutions arrived and proper internal architectures were designed, most of them while keeping in mind the fact that internal ECUs inside the car are price limited. Since there were so many solutions and good article written onto the protection mechanism side I decided to present them in Figure 2.1 for readability purposes:

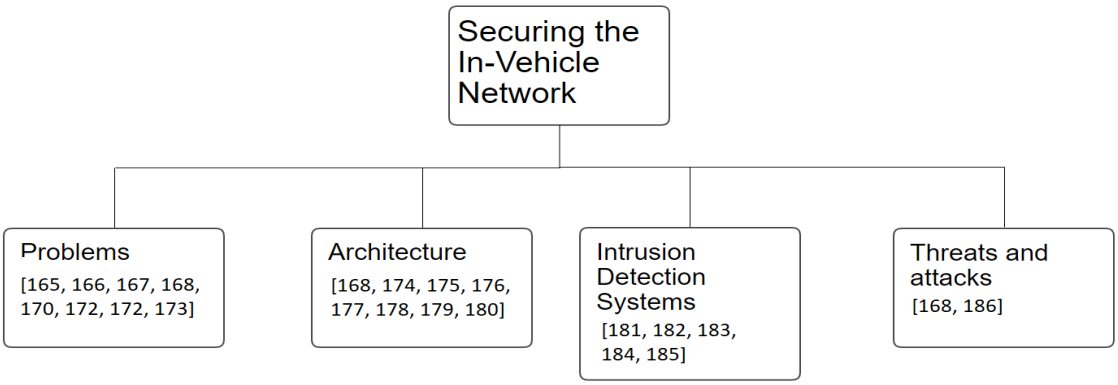

Figure 2.1. Protection mechanisms for in-vehicle networks

The summary of the papers was that traditional mechanisms which provided good results in other industries could also be used successfully in the automotive business. Authentication codes could be used for ensuring traffic integrity while firewalls could filter external traffic. Intrusion detection systems could provide information over unusual activities and even prevent them, Certificates can be used in order to identify vehicles and other objects and many other option could be consider viable protection mechanism. Also although only two reference papers were mentioned on the attached figure the number of attacks on vehicles increased sevenfold since 2016.

2.1 Standardization literature review

General and established methods are applied to a sizeable number of use cases and applications, methods such as the framework developed by EVITA project [128]. Standardization work is also in progress in multiple vehicular subdomains from security frameworks for V2X services [129] or cybersecurity engineering [130] or secure software development life cycle [131] and others.

For example **National Highway Traffic Safety Administration (NHTSA**) defined the US Department of Transportation traffic safety regulations specifics [132] where the five levels of driving automation were designed. Another US based society which became international now is **SAE International**, its standards SAE J3061 [134], SAE J1939 [135], SAE J1709 [136] define the best practices in the industry regarding security. **AUTomotive Open System Architecture (AUTOSAR)** with its AUTOSAR WP-X-SEC [133] standard defines software interoperability between ECUs in a secure and safe manner.

The activities in the Intelligent Transportation System (ITS) are spread around various groups which collaborate between them such as **European Committee for Standardization (CEN)** which provides CEN/TC 278 [145], **European Telecommunications Standards Institute (ETSI)** which provides a multitude of standards: TR 102 893 [137], EN 302 665 [138], TS 102 731 [139], TS 102 941 [140], TS 102 940 [141], TS 103 097 [142], TS 102 165 [173] which will be discussed more in depth in the following lines and **Car2Car Communication Consortium (C2C-CC)** which collaborated closely with ETSI and provided ETSI TC ITS [143] and ISO/TC 204 [144]. Last but not least **Institute of Electrical and Electronics Engineers (IEEE)** defines the communication between the vehicles themselves but also with the infrastructure itself with the help of standards such as IEEE 1609 [153] which represents the Wireless Access in Vehicular Environment (WAVE) and the IEEE AVB [155] Ethernet network technology standards such as IEEE 802 11.p [154], IEEE 802.1AS [157], IEEE 802.1BA [126], IEEE 802.1Qav [158], IEEE 802.1Qat [159] and of course the traditional WLAN technology standards such as IEEE 802.11 a, b, g, n.

Functional safety is covered by **International Electrotechnical Commission (IEC)** and its IEC 61508 [146] and the ISO/IEC 27034 [147] overview and concepts over security techniques and also by **International Organization for Standardization (ISO)** and its ISO 26262 [149] standard together with a handful of others: ISO/DIS 13400 [150], ISO 15118 [151], ISO 15765 [152] defining communication inside the vehicle exception being ISO/TC 22 [148] handling the external communication and control area in collaboration with ETSI TC ITS.

A closer look at the European Commission programs for research and innovation within the ITS domain we can observe that most of their results affect ETSI and also ended up inside ISO for standardization. From this least we can enumerate projects like:

- **CARONTE (Creating an Agenda for Research on Transportation sEcurity)** [160] which tries classify research for security in transportation.
- **HEAVENS (HEAling Vulnerabilities to Enhance Software Security and Safety)** [161] provided the baseline architecture for the ITS domain and its use cases and the security requirements derived from them, these results arrives in standards such as SAE J3061, AUTOSAR WP_X_SEC and NHTSA.
- **HoliSec (HOLIstic Approach to Improve Data SECurity)** [127] which provided models, methods, processes, tools and guidelines for reaching the goal of “security by

design" while also providing seminars and workshops for increasing knowledge and awareness on the security topic.

- **SeFram** [162] a continuation of SIGYN I and II provided an in-depth analysis of secure remote diagnostics, resulting in a formally verified protocol with impact in ETSI ITS specifications.
- **SESAMO (SEcurity and Safety MOdelling)** [163] addressed the converging issues of security and safety in embedded systems with high impact in the standard definitions.
- **EVITA (E-safety Vehicle Intrusion protected Applications)** [164] focused hardware solutions for the protection of intra-vehicle communication (V2X), resulted in the development of HSM, an ECU cryptographic co-processor, which is now the standard in hardware design.

This makes us arrive to the conclusion that for the automotive domain standards and established methods provided by projects are leading the way while in the same time providing information about the lack of security in automotive systems.

2.2 Technical dynamics and challenges

The vehicular domain standards apply for manufactures available in multiple countries with different legal stipulations which could also be conflicting. This is why we require general security solutions which can adapt depending on the context. Also since inside the automotive domain we have a large variety of technologies, protocols and services with different time, resources and security constrains. We will discuss in the following lines the complexity of this system capable of providing QM (Quality Management), ASIL-B and ASIL-D (Automotive Safety Integrity Level) processing, including multi dimension redundancy and the challenges faced.

The review of above mentioned work discussing various sections of the automotive domain provided us with the following list of challenges that must be considered by a threat model:

- Trust and privacy: for instance the owner of the vehicle is not trusted by the manufacturer, neither the driver by the owner;
- External communication: DSRC (IEEE 802.11p), WLAN (IEEE 802.11a,b,g,n), cellular communication (GSM, GPRS, 3G, 4G, 5G), Bluetooth or NFC;
- Hardware and Operating System (OS) security: preference for cheap "off-the-shelf" hardware, many OSs running directly on the microprocessors;
- Authentication, interoperability and mobility: since long distances are traveled in short periods of time maintenance and servicing should remain possible;
- The in-vehicle network: one euro ECUs with various network types such as Controller Area Network (CAN), Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST), FlexRay and Automotive Ethernet or even Wireless Sensor Networks (WSNs);
- Interplay of security and safety: an insecure vehicle cannot be safe since a vulnerability can affect safety and vice-versa;
- Real-time requirements: security must be designed in order to hot disturb real-time applications from the vehicle;
- ECU origins: third party ECUs providing both hardware and software solution without a well-defined security architecture;

- Software update and maintenance: a vehicle lasts for approximately 20 years so the lifetime security mechanisms and solutions should last the same amount of time, also compatibility is important between different vendors;
- Legal requirements: different countries might have country specific requirements and legislation, nevertheless all vehicle should be able to participate in the future communication systems.

An applicable methodology that meets all the above considerations is the Threat Analysis and Risk Assessment (TARA) [197] technique.

2.3 Secure Software Development Life Cycle

In this chapter an analysis for the area of Security Development Lifecycle (SDL) is presented with emphasize on automotive electrical and/or electronic systems adaptations according to standards applicable to the domain such as ISO 27034 [188], SAE J3061 [131] or ISO/SAE DIS 21434 [130].

Inspired by the above mentioned metrics and processes I adapted a similar process, tailored for the automotive specific systems and compatible with SAE J3061 Cybersecurity Secure Software Development Life Cycle (SDLC) Guidelines and ISO/SAE 21434: Road vehicles — Cybersecurity engineering compliant, a standard providing guidelines for harmonizing the automotive cybersecurity development processes based on ASIL levels introduces by ISO 26262 [149]. The process consists of three phases: concept, product development and production & operation.

Every SDL process takes advantage of code reviews and static analysis in order to detect security defects. Performing threat modeling and risk assessment are mandatory during concept phase in addition to dynamic testing during product deployment phase. Next we will briefly present several tools that can be used in various stages of the SDL.

Static analysis examines the source code and tests its weaknesses without executing it. The thesis work of Patrik Hellström [194] represents an analysis of both commercial and open source static analysis tools and provides a state-of-the-art work for this field including tools such as Fortify, Ounce, Coverity, Klockwork Insight and CodeSonar.

Regarding threat modeling and risk assessment according to HEAVENS security model [161] and SAE J3061 [134], TARA [197] is preferred with applied TMEA and ATA for threat analysis and a combination and EVITA [128] and HEAVENS [161] for risk assessment providing in the end a better architectural understanding.

Dynamic testing including penetration and fuzz testing examines the source code and tests its weaknesses while executing it. Here the following tools got highlighted CERT Basic Fuzzing Framework (BFF), SDL MiniFuzz File Fuzzer and SDL Regex Fuzzer.

2.4 Insights in Security architecture

As mentioned in the previous sections, based on the standards and automotive domain specification we have identified the most applicable software mechanisms for automotive security, some of them already validated on other industries.

When discussing operating systems in automotive industry we need to keep in mind that we are discussing multiple options and even more than one bundles on the same hardware. As part of this work we focus only on Linux and Android since AUTOSAR and RTOS based ones are usually smaller in dimensions and some of them are even formally verified in order to meet ASIL-D requirements. So for each available operating system option we need to make sure the kernel is protected against the following exploit classes:

- Stack-based attacks
- Buffer overflows
- Return-oriented programming
- Control-flow integrity attacks.

In order to protect against the above mentioned exploits the following hardening mechanisms:

- Stack protectors
- Address-space layout randomization
- Ensure no memory page must be writable and executable at the same time

A closer look at the types of memory these two operating systems interact with, based on the properties relevant for the automotive domain such as confidentiality, integrity and availability (CIA) reveals the following:

- Universal Flash Storage (UFS): it is the default option for booting the hardware specific binaries for the reference hardware presented previously. Meets the last two properties integrity and availability while confidentiality needs to be investigated based on the selected hardware. Secure boot support need to be enabled.
- NOR Flash: can be used by first stage bootloader in order to minimize the boot time due to UFS startup time delay. Similar to UFS it meets integrity and availability while confidentiality needs to be investigated based on the selected hardware. Secure boot support need to be enabled when used.
- LPDDR4/5: mainly used for sharing information between VMs or with AUTOSAR. In case the shared data is temporary CIA is not required but if the data is processed then CIA might be required. Cryptographic data and HSM memory support might be required in order to ensure confidentiality.

Other types of memory such as SD Card and USB are not supported and therefore not recommended to be used.

An analysis of the access control mechanisms reveals that SELinux is the preferred mechanism used by Android while for Linux there is no mechanism yet defined, although Automotive Grade Linux (AGL) workgroups try to standardize SMACK, but using SELinux for both would be mutually beneficial taking into consideration the Linux knowledge outside to embedded based domain. A short overview of the two mechanisms reveals the following:

- SMACK (Simplified Mandatory Access Control Kernel)
    - Governed by a central policy
    - Designed with embedded systems in mind and for securing Internet-connected devices.

    - Simple configuration and maintenance.
- SELinux
    - Complex due to high granularity.
    - No access by default and rules required
    - Rules must be created and loaded into the operating system to specify allowable access rights

Other security measures could involve a secure service oriented architecture with firewall support, packet filtering and traffic shaping. Domain isolation should also be employed based on address spaces, virtual machines and hardware cored together with isolation classification between different applications features all guarded by intrusion detection system.

This work presents demonstrated security threats and their implication with an emphasis on the importance of security in the vehicular domain and how various aspects were addressed as part of a number of research projects. A reference architecture was designed and proposals and analysis was made regarding the mechanisms for securing it. We discussed traditional solutions which proved efficiency in other domains of application and even new specific solutions were presented.

The goal for this work was to present a synthesis of the field with the hope that a better understanding of the vehicular domain will be provided and others will be inspired to further the research because there is a lot to be investigated regarding the internal security of a vehicle, making the provision of safer and more secure cars and traffic the paramount goal.

Chapter 3. Vehicle safety

In this chapter I will present the first steps of this thesis where we have a look at the current vehicles, a large network of electronic computing units (ECU) connected not only between them but also with the outside world. We interact with the automotive technologies, present their vulnerabilities and increase the security and safety by providing relevant information through a user-defined interface. In the same time, we document the processes, infrastructure, technologies and hardware components that we used in this process. We do this exercise in order to better understand how the security and attack surface of a vehicle looks like.

3.1 The context of vehicle safety

Here we base our work on the general overview of the current vehicle hacking context and we aim to provide explanations of why this project is relevant, how the solution is significant, what is our approach towards the problem and the use cases we have in mind.

The concept of car hacking has become an important component of a vehicle, even FBI issued warnings related to the risk's car hacking imply. The modern vehicle includes over 100 electronics devices connected via multiple industry specific networks, according to an IEEE Spectrum report [1] from 2009. This is similar or even more powerful than a personal computer and every component can be acessed by establishing a conection with the available networks, communicating with the motor, the head unit display and others. As documented in [2] "the automotive industry has churned out some amazing vehicles but has released little information on what makes them work. Understanding how the vehicle communicates will help you diagnose and troubleshoot car problems. As vehicles have evolved, they have become less mechanical and more

electronic. Unfortunately, these systems are typically closed off to mechanics. Whilst dealerships have access to more information than you can typically get, the auto manufacturers themselves outsource parts and require proprietary tools to diagnose problems."

The automotive industry from both European Union and United States arrived to the same conclusion, that of following the same protocols and standards regardless of the implementation. This enabled the usage of generic support and collaboration so further development happened on the base functionality of a vehicle.

3.2 The available technologies

In order to start the implementation for this project the first step required was the one which required the understanding of vehicle specific protocols and interfaces. The first one that we are going to present here is the OBD port, an interface which became mandatory on all available vehicles. It was introduced first in USA in 1996, then in 2003 in Europe and by 2008 in China and is now the primary choice for communication with the vehicle.

In a vehicle the OBD port performs self-diagnostics in order to provide the information required by the repair shops for identifying problems at the car. This time of information is available through diagnostic trouble codes (DTC), messages which allow fast malfunction check and identification from ECUs all around the car.

The other important element available in an automobile is the CAN serial protocol, developed by Bosch and used extensively in automotive. The OBD-II socket offers support for this protocol in order to provide diagnosis. It is described in ISO 11898 [12] and ISO 15765 [2] and it became the de facto standard for vehicles in 2001 for European market and 2008 for the US one.
As an internal structure we have a number of different embedded systems which communicate through CAN. Their implementation is different and there may be multiple networks and types of communications which include different bus speeds from low speed to high speed.

As part of the hardware support we have used not only a Raspberry Pi board but also some extra components such as a GSM chipset called SIM5218 an OBD interfacing microcontroller called ELM327 and an USB2CAN device. Since there are so many components that compose the software stack more details are available inside the paper together with a a diagram for faster walkthrough. A short presentation of the software stack is also visible in the next sections of the chapter as well.

3.3 Project architecture

Here we describe the currently available features of the projects, the ones that are implemented so far and present the ways they are put together to work in order to provide the documented results. From a project evolution perspective, we can describe four stages:

1. The initial deployment: includes the Raspberry Pi operating system (OS) setup, Digital Ocean droplet and Ansible deployment
2. Basic integration: includes OBD/CAN, SMS MT/MO, GSM 3G AT, S-GPS/A-GPS, Wi-fi and Bluetooth added support.
3. Development: where we did the Python API , database and authentication system implementation

4. End-to-End Testing: with Android client, web user interface and real-life testing support added on top.

This stage started with the selection of the setup. For Raspberry Pi development board, we used the Raspbian operating system, the one officially supported by the Raspberry Pi Foundation. Inside we installed Ansible support as the configuration manager for all the software and services installed and configured inside the OS image which resides on the development board SD card.

With the software available and configured accordingly the hardware setup can be connected inside the vehicle using the ELM327 device through the on-board diagnosis port. We will use this setup together with the development source code in order to validate our use cases.

With the complete hardware setup and in place the user is able to connect to the smartphone application or the web server interface and register for an account. With the account available the pairing is necessary in order to associate both the device and the account and after this some initial testing is carried out in order to validate the full duplex communication between the two.

All these steps are essential not only for the use cases validation and testing but also for the solution next stages implementation. Although not all are available from the first project interactions they are mandatory for the final solution setup.

In order to meet the security requirement of the project the solutions that were used here were carefully selected and configured in order o meet with best practices from the security field associated with the industry.

All wireless communication is protected with WPA2 using 256-bit keys and authentication done based on a username and password combination with a minimum requirement for the password length of eight characters. This one is actually the most basic form of security since it is quite mainstream and heavily validated.

Moving further we tackle the next subject where security is important, and that is the SMS part, where commands are sent towards the vehicle.

Another important piece of the puzzle that needs to be solved is represented by the RESTful API security where beside security also consistency and integrity of the data need to be supported. This is ensured with features such as: user authentication, user authorization, encryption, role based access control, data validation, data integrity and consistency.

Besides this we need to ensure the security of the server that contains the support for RESTful API and the database. It needs to be protected with authentication mechanisms over the HTTPS functionality which provides encryption with SSL certificates help. We also have support for a firewall there with default DROP policy on all packages that do not correspond to the defined rules. We defined there, exceptions for the SSH protocol (22/TCP), RESTful API and the Web User Interface exposed over HTTPS (443/TCP), also, on Raspberry Pi all incoming ports are closed for safety measurements.

Last but not least the vehicle setup, here we are not talking about security but more about safety of the car, which we ensure with the help of redundancy for both the Wi-fi communication and GSM or GPS module.

Chapter 4. IoT open education

In this chapter I will discuss how we can bridge the knowledge gap for automotive stakeholders through the use of technology and describe ways of making it accessible and its impact accountable. I will present a set of tools which could be used by both technical and non-technical individuals since the shift of paradigm towards security by design needs to be understood at all levels.

Since there is a lack of awareness over technology`s impact on automotive we propose several methodologies to help raise awareness for all the stakeholders, like regulatory institutions, the general public, and engineers involved in the automotive industry. For that the first step is to present a set of technologies and how they could be applied in order to reduce the knowledge gap.

In a similar manner we employ these technologies to further develop the automotive digital architecture that we started in previous chapter and not only provide information regarding the state of the vehicle in a safe and secure context but also provide mechanisms for multiple stakeholders to diagnose and troubleshoot various aspects of the car while keeping in touch with the shift from mechanical to electronic and from enclosed to open and outsourced.

4.1 Wyliodrin a technology enabler

Nowadays automotive contains a lot of electrical and electronic systems that run software and are connected to the Internet [201]. This trend change means that computer skills are required for the interaction with the vehicle. This is applicable for a large number of categories of stakeholders which are lacking the technical skills. The trend change from software, hardware and technology skills being a developer activity towards general public is also visible other industries from arts to mechanical and chemical engineering. Even non-technical universities introduced computer science classes where the learn how to interact and construct an infotainment system or similar devices. Such examples are architecture, civil engineering and even film universities [202] [203] where they encourage students to include state-of the-art technologies in everyday activities. The fact that they implemented this in a successful way paves the way for automotive as well.

When interacting with software, hardware and technology concepts in general is not easy work due to a number of reasons such as: hard to comprehend concepts, a stip learning curve in general and huge infrastructure to interact with, containing multiple ECUs and sensors available. Mostly there is the difficulty in applying the technology concepts into their everyday activities. Wyliodrin can help with these issues by guiding stakeholders in a hands-on manner to build the systems they require in order to help them relate better with technology.

Wyliodrin is a platform which allows stakeholders to develop and interact with systems that are related to their field of work. With the help of this platform stakeholders can see how a subsystem from the vehicle is designed, identify the programming skills required to build them and how they can interact with it. The plaform provides support for both software and hardware and provided a good candidate for the requirements of a secure by design architectures which is both safe and easy to interact with by all stakeholders available in the industry.

When teaching computer skills and programming, the approach varies depending on the stakeholder age and background. While some have the capacity to understand and apply abstract concepts into practical examples, others require to be engaged during the process. As a result, we provide a wide range of materials and platform funtionalities to choose from when designing.

Scratch is a visual block-based programming language, extensively used for teaching programming for teens and children. It was developd inside MIT Media Lab and since then was used largely to build applications by dragging and dropping blocks on a dashboard like in a puzzle avoiding missconfigurations. All these blocks are converted on standard programming languages. It can be integrated in 3rd—party platforms or from withing the Scratch Studio IDE, a web platfrom.

Another similar platform is Blockly, developed by Google which is as well converted into standard programming languages such as Javascript, Python, Dart or PHP code. The advantage of this language is the fact that it permits the generation of new blocks and integrate them back into the platform [204].

Arduino is a microcontroller board mainly developed around the ATmega328 microcontroler which also comes together with its own IDE. It is the most used hardware and software platform capable of integrating multiple sensors and actuators under its 14 digital input/output pins out of which 6 support PWM outputs and 6 analog inputs. The board is also equipped with a USB and ICSP header for programming, a power jack, a reset button and a 16Mhz crystal oscillator. It is widely used by hobbyists and begginers around the world for learning dn prototyping not only because of the tamper proof and cheap development board and easy to use development environment but also because of the multitude of libraries available which expose functions for controlling pins and connected peripherals. These advantages alone made Arduino a good candidate for the university environment and in particular for the automotive digital architecture constructed with the help of Wyliodrin.

Last by not least we used the Raspberry Pi a small single-board computer developed for educational purposes and powering most of the do it yourself electronic community projects. The Raspberry Pi Fundation, the foundation governing the development of the Raspberry Pi development board aims at promoting computer science education in undeveloped regions with the help of a affordable and easy to use platform.

With the leasons learned from the Arduino world, Rasperry Pi also comes equipped with a large 26 respectively 40 pins GPIO headers a large database of demonstrators making it popular with the educational sector that Arduino also addressed [205] [206].

The Grove extension boards represents a kit which represents a complementary component that could interact with Arduino and Rasperry Pi boards and provides interaction with Grove compatible electrical components; sensors and actuators which could be connected directly to the Grove available pins in such a way that incorrect connections and errors such as a short-circuit could appear [207].

The above described architecture and framework was designed with educational purposes in mind and it was optimized for the maximum engagement and best overall group progress also taking into account the complexity of the developed applications and grading obtained by the stakeholders for completing the tasks. The results obtained after interaction with the above described framework are compared with the results obtained previous year by the same stakeholders in a similar context.

The stakeholders are students at mechanical engineering with an authomotive background and they did the same exercises using the Wyliodrin demonstrator as they had previous year. The difference between using Wyliodrin demonstrator setup and previous year direct interaction with the sensors directly was visible, the advantage of using Wyliodrin being that it abstracts and simplifies the interaction[209][210]. Comparing the two scenarios: last year the projects were not

completed, focus being lost on various aspects since with the provided solution everyone managed to finish the same projects and demonstrated higher engagement rates.

The fact that the framework also enabled the transition from a visual programming language such as Blockly to a Python one was really helpful and enabled them to learn a high level programming language. As an example we started the interaction only with the Blockly usecases generate Python code from that. Next step was to generate a faulty Python code and provide the means to solve them. By the end of the interaction, 80% of the stakeholders were able to write complex Python applications which are able to interact with multiple sensors. By the time of the exam 70% of them were able to control and exchange data between the connected peripherals while interacting with a web interface which was a hige leap from the previous year experience.

With this project we managed to design a platform which enabled stakeholders to aquire engineering skills and apply them effortlessly in their activities. This type of approach for automotive industry proved to be beneficial and in the transition from a paradigm such as security by obscurity towards security by design would make Wyliodrin a good example of how to simplify and enable a smoother conversion.

4.2 Yocto Project applied in the industry ecosystem

Yocto Project was initially release in March 2011 and in quickly became the standard for providing customized and predictable Linux distributions and binaries. It was used as part of this thesis for the same purpose because we required predictability and a Linux distribution fitting to our needs which only packed the minimum requirements for the job.

Our purpose was to obtain a system easily customizable where we could include functionality for the features and sensors we could bring on top of the base system, porting the functionality to an entirely new hardware platform and basically every functionality we could think of. For a better understanding of the way Yocto Project was used as part of this work I will try to present a bit its architecture and available support in the sections to come.

The workflow briefly states the fact that developers need to state the architecture, patches, policies and configuration details before building. When the build process starts, source code is fetch and downloaded from specific location defined inside the metadata. With the source code available locally the build engine applies the patches on top, configures and compiles the software. It then installs it in a temporary location before packaging it in the preconfigured format recognized by the selected package manager. After that QA and sanity checks are carries out to validate the build process and when everything looks good the results are binary package feeds are generated in order to create the hardware system and SDK images.

On top of everything presented in the previous subchapter, the industry started adding metadata support in order to configure their own version of the product similarly on how things are happening inside the Linux kernel with kernel modules, the difference here being that the support is included in layers with different functionalities, be them BSP layers with support for various hardware platforms or layers which include various functionalities such as real-time, virtualization and security support.

Besides the above presented layers there are a lot more layers available with support developers could use. Some of them under the Yocto Project umbrella other maintained by private companies or individuals. For a newcomer to this project all this could prove a lot, even

more so then the interaction with the Linux kernel source code but the Linux Foundation and the maintainers of the project strive to help with this.

Besides the work of providing functionalities and support as part of layers both the industry and the open source community saw the value of the project and started providing reference distributions support. One of the first ones was the OpenStack mentioned above but there are also a lot of alternatives which provide not only for ways of interaction with the Linux distribution but also new tools support and configuration options for the core functionality.

Some of these distributions are Carrier Grade Linux (CGL) a Linux distribution which started as a reference documentation for the telecom industry, a guideline for the functionalities a carrier grade operating system based on Linux should have, now is a combination of Yocto Project layers which could be used as staring point and ease the process a lot for a company.

Another similar initiative, started in a way from the CGL initiative, because it was initiated by the same manager of Linux Foundation, is Automotive Grade Linux (AGL) which provides a reference Linux distribution one which arrived even in cars such is Toyota and is in continuous development by all the major players in the automotive industry.

This was the support I started to use, since it also supported the Raspberry Pi board when I had a decision to make regarding an operating system which could be configured according to my fluctuating requirements. It proved to be the best decision since the end result was not only easy to configure but also the end result used a lot less resources, which were used with adding new functionalities for ensuring a true open, configurable system, a simulation of a secure by design digital automotive architecture which was used as my testbed for this thesis.

Chapter 5. Industrial prototyping

In this chapter I will continue the discussion on bridging the knowledge gap for automotive stakeholders through the use of technology and present sets of prototypes as ways of making it accessible and its impact accountable. The roles of prototypes is be handled by both technical and non-technical individuals as already emphasized on previous chapter.

As already stated, there is a lack of awareness over the impact of technology on automotive and we provided tools and prototypes in order to cope with this forming knowledge gap and help raise awareness for all the stakeholders such as: regulatory institutions, engineers and the general public involved in the automotive industry. As second step of this process is to present a set of prototypes combining new technologies and applying them in order to achieve same or even better results in industrial scenarios. A quick look at these prototypes and we can easily conclude the presented technological application could also be applied in automotive in order to reduce the knowledge gap.

5.1 Speech recognition service

The work for the speech recognition service enabled inside the automotive context started from the research carried inside the paper where we designed and develop an Intelligent Haptic Robot-Glove (IHRG) suitable for the rehabilitation of patients which have been diagnosed with a cerebrovascular accident. The design and implementation of the IHRG solution was carried out in our laboratory. This is suitable considering that for the patients will feel more comfortable having their medical exercises at home. The proposed IHRG has a great advantage - the possibility to specify vocal commands, helping the patient make an extensive number of medical exercises.

In the article we aimed at restoring the motric functions lost after a major trauma as a result of strokes and cerebrovascular diseases (CVA) where as a result, a part of the body controlled by the affected area of the brain cannot function properly [213]. We extended that implementation for the automotive vehicle speech recognition functionality which instead is able to provide interaction with various ECUs from the vehicle such as the infotainment system, lights or others.

Lately, the available literature presented numerous techniques and concepts that permit evaluation of the field with different implementation although the initial article begun as an application for the medical sector this is not a requirement. Then numerous studies [214][215] have allowed the reproduction of the human hand kinematics as much as possible with the help of kinematic structures development. Now the same core can be applied in a similar manner as a main support functionality for the driver, which is required to keep his hands on the steering wheel and his eyes at the road.

The design of the solution is made based on the same Raspberry Pi hardware platform with the help of the Arduino Mega 2560 extentions which simulate an ECU controller that determines the activity of small engines. In order to make the specific voice recognition computation the Raspberry Pi hardware platform was used. Using a microphone, the driver can send vocal commands to the equipment which will infer actions being triggered for a specific period of time [216][217].

Extensive testing have been done and the results provided a very good detection rate, from 200 tests, 97,5% accurately returned the correct result for simple one word commands. However a good detection rate of 95% was also observed for more complex constructs. One of the most important factors influencing the recognition process is the noise. Whenever the noise is is kept to a minimum level, close to zero, the results of speech recognition are at a good quality; whether the noise level is increasing, the level of accuracy for the speech recognition decreases exponentially almost. Another important factor for the detection accuracy is related to the use of the Raspberry Pi hardware platform and its audio quality support which prove to have a smaller detection rate then when the tests were carried on a personal computer.

In order to improve the precision of the results we can try modify the recorded values as input or by changing the Viterbi score. Another way for improving the accuracy is through the training phase of the acoustic model. With the data gathering increase, a point of equilibrium is reached where the accuracy remains unchanged, this is consider the point of maximum.

In order to provide an acoustic model usable by more drivers, it will be compulsory for each driver to contribute with their own input into the system. It will be interesting at that point to compare our solution to a similar solution already available in Automotive Grade Linux (AGL) called Alexa which is provided by Amazon and is also integrated inside the Yocto environment.

A very good detection was obtained for simple and also for complex commands. Different other aspects were also analyzed in order to identify their influence over the recognition process.

5.2 Wyliodrin Android companion

The embedded systems market is constant growing. They are present everywhere around us in various devices, from usual devices to automobiles or smart heating systems. The current paper describes the process of development of an Android application that allows the user to interact with embedded system through the Internet, using a mobile phone. This way, the users can use values read from phone sensors in their projects and they can control them remotely in real-time.

Internet of Things (IoT) is a concept which aims to connect to the Internet as many devices as possible and to allow controlling them remotely, using special applications. The growth of the embedded systems number will cause a difficulty in how they can be controlled if each type of device will have a different mode of interaction and control.

Devices such as Raspberry Pi and Intel Galileo help developing IoT by offering anyone interested a simple way of developing embedded systems that will achieve the desired functions. Connecting these devices to the internet is the easy part, the challenge is of course developing specific applications for controlling remotely each one of them.

Wyliodrin is a web platform that offers users a simple way of developing complex software applications which can be executed on embedded devices without requiring advanced programming knowledge. Applications created using this system run on devices independently, without user interaction [218] [219].

Because smartphones are so widely used, it only felt natural to try and capture the smartphone user`s attention towards IoT and embedded systems development. Combining your smartphone with Wyliodrin represents the perfect starting point for remote controlling embedded systems via Internet. Also, the mobile application can use all the functions provided by the mobile devices, such as reading phone sensors values and send to embedded systems.

The current paper describes the development process of an Android application that communicates with embedded systems which execute applications created using Wyliodrin. This mobile application will send user actions and data from the phone, such as sensors values, to the embedded systems.

The application obtained during the process described is functional and can be used to control embedded devices and allows the user to interact with them. As previously described, the first screen of the application is the login screen. In this screen, the user can choose to scan a QR code that will allow him to connect in the system.

After the authentication was successful, the user is redirected to the main screen, where he can view the list of saved dashboards and can choose to create a new one or to delete one by long pressing it. If the user chooses a dashboard from the list or he presses the New dashboard button, the application will redirect him to the dashboards screen

. The list of elements that can be added to the dashboard can be seen in a sliding menu, in the left side of the screen. After choosing one element from the list, the user must choose the properties for the new element. To edit or delete an element, the user must long press it and the application will display a popup menu.

The dashboard can be used to control elements such as a simple heating system. The dashboard shows the current temperature in the habitaclu using a thermometer and the history of the received values using a graph. The user can set the desired temperature in the room using a seek bar and he can turn the system on and off using a toggle button.

To sum it up, Wyliodrin COMPANION is an Android application that allows interaction with embedded systems remotely via Internet; it can send data and commands to systems and receive information that can be viewed in one of the available forms.
The application offers a number of elements used for sending commands (such as buttons, toggle buttons), sending data (such as phone sensors, seek bar) and display received information (such as thermometers, speedometers, graphs). The application uses the XMPP protocol in a way that provides a quick and safe communication between mobile and embedded devices, like it happens in the Wyliodrin system [218].

Chapter 6. Secure automotive digital architecture

In this chapter we have investigated the mechanisms that could be implemented on the automotive digital architecture developed so far, in order to provide a multi-level security system, summarizing the work that was published in [109], [110], [111] and [121].

We started this work by taking a look at the trusted execution environment, described in subchapter 6.1 with details. Ideally we would want to run each and every application in such an environment and guarantee for the security of the execution but unfortunately there are a number of penalties of time and performance which are not considered acceptable in all contexts. With the help of [111] sprobes mechanism we manage to do an introspection of the system registers and memory pages through TrustZone SMC calls in order to prevent malicious interactions. Unfortunately the obtained performance results and the limitations of the solution which required continuous interrogation of the system together with the penalties of the trusted environment in general proved not being industry graded enough for our purposes.

To counter the limitation of the previous solution for those application which cannot be executed in a trusted execution, we chose a systemic approach, proposing a security hardening solution with the help of [109] and [110] which we described in subchapter 6.2 of the thesis. First we provide a host intrusion detection system [110] which is a more discrete option of analysis based on the interrogation of system calls triggered by an application together with a binary analysis solution [109] which has a low overhead being called only one time for a binary putting it one of the multiple categories available. Only one of those categories are benign the rest are malicious for simplifying the post analysis process in case of vulnerabilities.

Now that we have a secure system we also want to make sure any updated provided do not impact in a negative way the security so we also added static code analysis with [121] where we provide a mechanism of scoring the impact to the attack surface of an update employed. This part of the work is described in subchapter 6.3 with further details regarding implementation, results and evaluation of the solution. This too is a one-time type of analysis with low overhead to the system and supplementary validation of security, really useful because over-the-air updates are a common thing in automotive.

6.1 Trusted execution

We will move further and try to describe one by one each article work and will try to provide for each one the results and evaluation fitting to our context. In order to provide secure updates we need to have hardware HSM security as enforced by ISO 26262 [149]. On top of the hardware we need to build our trusted execution support. In order to provide a trusted execution one of the options provided by the industry is to use the complex multi-stage hypervisor based architecture, similar to the one described in the state of art, where the hypervisor is used to validate interactions. This comes with a huge overhead, so the alternative is probes mechanisms which provides lower overhead. In the following lines of this subchapter we are going to investigate this novel alternative solution to the trusted execution problem.

On core systems, mechanisms are needed to maintain security throughout the life of the vehicle. There is the problem of escalating privileges on such a system and that is why we start the presentation with the solution that uses the trusted execution support called TrustZone in ARM architectures and the idea of protecting the operating system from rootkits with a minimum overhead.

For this we have defined a number of 5 rules from S1 to S5 that must be implemented and observed in order to protect the system. We used an introspection mechanism based on SMC calls provided by TrustZone to analyze in secure world the safety of instructions affecting registers and memory pages, without malicious interaction on the analysis process. This mechanism was validated in a synthetic environment by a group of researchers and because the initial results proved to be encouraging we decided to implement it on a Linux 4.9 kernel and latest arm-trusted-firmware.

The results of the implementation of the sprobes mechanism showed an overhead of about 10% for a system with 279 tasks run and a number of 5611 extra instructions executed, however such an implementation would not have been accepted in the Linux kernel because it modifies core components that are extremely sensitive to specific changes such as ours and unfortunately the overhead of time and performance, namely 2 seconds rear view camera of standards such as NHTSA (National Highway Traffic Safety Administration) is high compared to the requirements of the automotive field.

We cannot block an attacker's way of manipulating the kernel from inside. Although we might limit his possibilities of gaining access to root. This implementation with the SMC calls proved to be problematic, breaking some functionality and not adding the impressive layer of protection with little overhead. Although the Linux kernel is not perfect, and many consider it a megalithic and deprecated it is certainly not a defenseless system and it prove very hard to be cracked. A polling or interrupt kind of mechanism for protecting changes in registers so the handling could be done automatically when instructions is executed at the lowest levels could constitute further improvements.

Although the SPROBES mechanism sounds like a good option that could very well be a part of the Linux kernel functionalities, most likely this would not be possible. Since it would require a number of changes which would also prove challenging for the Linux community. Although in theory I enjoy the SPROBES elegant proposal, in practice it could prove to post quite some problems, making a hypervisor solution a sturdier and safer approach.

Although with limitation which are not acceptable for the automotive, SPROBES proves to be a good starting point for a real-time Linux kernel security solution. This was validated already in Samsung KNOX Android products. This type of solution proves to be useful especially for

embedded devices based on ARM architecture with TrustZone support. Any other hardware isolation mechanism would have to implement a similar solution, particular for the hardware specifications.

6.2 Security hardening

After we enhanced security to kernel and firmware level with [111], it is time to move up in the architecture onto the operating system application space where we present the work that was published in two research articles [109] and [110] both involving artificial intelligence in order to assess the security of an operating system. One is responsible for the analysis at runtime of system calls and the other of the static features interacting with a binary with the purpose of identifying and classifying uncertainty and inconsistency in operating systems. Both works are part of a systemic approach for security hardening the cyber defense of our system and detecting attacks and exploits.

Largely, the main technique for detecting anomalies inside a system is represented by "misuse detection" where a digital signature of an attack is drawn after it had taken place and included as an update inside the security appliance. Although very effective against known threats, this strategy is not suitable for catching a zero-day attack vector. Better results are provided when building a baseline of "normal behavior" and flag any activity not fitting into the baseline. For a detection system based on sequences of system calls, the intrusion detection system captures any abnormal sequence and marks it as illegitimate. The same applies for the problem of classification for binaries based on benign and various malware classes as a vector of numerical features using API calls generated when the application calls a function available in a different module.

The work presented here is based on a large literature body and it tries to tackle the problem of anomaly existence inside an operating system either under the form of an intrusion or a malware activity. The malware family includes viruses, trojans, root kits, worms, bots, backdoors, adware, spyware, ransomware and it represents a fast-growing threat for the available operating systems. It represents a multi-billion-dollar market which has an exponential increase in the amount of malware activities as reported by antivirus companies.

Because the previous solution did not fully meet the time and performance requirements required by the industry, we moved on to a HIDS solution based on system calls, an adaptive security solution that builds a baseline for what normal/benign behavior means and signals any deviation. This solution is built to keep the security of the database containing security policies because it is a major point of vulnerability.

This proved vulnerable especially in 2019 when the database maintained by Honda allowed operators to see which systems in the network were vulnerable to security flaws, exposing the system data of 134 million employees. A similar thing happened with the improperly configured database in India, which allowed the disclosure of personal and vehicle information of 452 thousand users and the list goes on.

We use for implementation an instance of the MySQL database that we monitor in a container. Here we generate benign samples using mysqlslap and malicious sample which includes malicious models such as service fingerprinting, sql injection or brute-force attacks using sqlmap. Thus we manage to capture 4,732,254 malicious system calls and 35,351,981 benign system calls.

These samples are then parsed using the tf-idf method (term frequency – inverse document frequency, an algorithm used in NLP successfully) to build a dictionary of system calls to which we assign as weights frequency, to which is added thread ID, dynamically linked libraries, return value or system call duration. All these properties form a rare matrix that represents the state of the container at a certain point in time.

The decision core is made based on the Random forest algorithm which has proven to be very efficient in such classification problems. In the analysis, speed was an important quality and we noticed that it is influenced by the size of the event buffer (trace complexity). The best results were obtained for a buffer with a size of 250 system calls: 95% accuracy and 0.4 ms execution time.

As the size of the buffer increases, the number of samples available for the learning module will decrease accordingly, affecting the overall accuracy. For example, while a size 250 buffer is used, a total of 160,336 malicious plus benign/normal monsters result. While T1 performed better, T2 achieved comparable accuracy using fewer learning samples than T1.

The next step is to analyze the binaries in order to identify the existence of different types of malware within the operating system because such a system (similar to the Android environment) allows adding applications on top of it and this has led to attacks like ransomware WannaCry or NotPetya (both based on the EternalBlue exploit on the implementation of the Server Message Block (SMB) protocol allowing the execution of arbitrary code on target computers) that affected thousands of servers of transport companies such as Deutsche Bahn, Maersk or FedEx. For this we analyzed the classification efficiency of random forest and neural networks algorithms. Initially it is necessary to build a representative dataset, so we build a Cuckoo sandbox to capture the behavior of the binaries and provide the results as input to the algorithms one by one, at the end we compare their efficiency in the analysis.

Other Linux-specific ransomware attacks: Tycoon (and for Windows, exploits unsecured remote desktop protocol (RDP) connections), QNAPCrypt (exploits SOCKS5 proxy connections), or Erebus (exploits security issues such as DirtyCOW - allows processes to write to read only files due to the exploitation of a race condition from the Linux kernel, function from copy-on-write, hence the name - or certain versions of Apache 1.3.36, PHP 5.1.4).

In this case the dataset includes 10,000 monsters of which 5022 are benign, 4978 malware of which 901 are synthetically generated by us. Following the analysis, 7101 features are extracted that could help identify malware applications, features that we managed to reduce to 78 using both an automatic and a manual analysis. Among the features are a number of dynamic libraries, addresses, dimensions or functions (but also the entropy of each section of the executable).We can list access to Binder libraries (on the transaction side, reply, release), Widget libraries, networking libraries or interactions with standard libraries such as libc or openssl (libssl).

Following this representation in the form of a set of characteristics extracted both in the static analysis of the binary and while running, the classification process is performed with the two algorithms mentioned above dividing the data set into 80% training and 20% testing. Cross-validation is used in order to evaluate the chosen models and in the end the random forest algorithm is the one that presents the best results (97% compared to 96%). As a side note: the use of only static characteristics, although decreases the accuracy percentage, keeps it at a value that can be taken into account (95%).

The present study proves the applicability of machine learning techniques towards developing stronger security hardening based on the above developed tools. For both tools we

manage to construct good starting points with the datasets that could prove useful for future work on the subject.

For the application analysis system we managed to find some good accuracy with the Random Forest algorithm, constructed a good model which could be scripted and also the number of extracted features could provide some better insight about malicious software on a deeper analysis even if done manually.

For the HIDS system we could improve the classification rate by trying to capture the sequence of system calls (using manifold learning [71], frequent pattern mining [72] or time series approaches [73]), or by capturing their arguments [74], [75] and clustering them. Intuitively, a certain sequence of system calls could improve the classification problem, but this needs further analysis.

6.3 Static code analysis

In order to maintain a secure environment we require a mechanism of assessing the security of applied updates. For this we propose an analysis of the attack surface and a mechanism of scoring the impact to the update based on that. In the following lines I will describe this solution which I prototyped in article [121] as an example.

Most vulnerabilities are created by calling a function from a library in a wrong way or calling a vulnerable function [199]. This project aims to automate the discovery of these types of vulnerabilities by using a deep learning approach. The dataset used to approach this problem is composed of different programs containing vulnerabilities. We use static analyzers on them in order to produce labels which we use in order to train our assessment tool, in order to provide a vulnerability classification efficiency and gather different programming styles.

Having the system protected by the above described solutions, we turned our attention to the analysis of the updates that are added over the stabilized system. We aim to improve the results obtained for detecting code vulnerability by implementing a method of detecting vulnerability on C/C++ code, based on the normalization of the project source code lexicon with the help of abstraction and then classify using deep learning algorithms if the code impacts the attack surface of the system or not.

The proposed process has four stages that include: building a code scrapper from GitHub, static analysis to produce tags used to anonymize the code, training deep learning algorithms and analyzing their results.

We use clang-extra-tools to analyze AST (Abstract Syntax Tree) context nodes when doing semantic analysis. For each node analyzed we keep the place where it is declared for the first time (to differentiate between the same variable names with different domain/purpose) with the help of the ASTContext class offered by clang. We gather all the C++ blocks: Variable declarations, Type declarations, Label declarations, Namespace declarations, Namespace aliases, builtin template declarations, enum constants etc. We keep the granularity at the file level, so we can identify illegal API calls. At the end of the transformation we convert the file into a token dictionary arranged in the parsing order that can be used for analysis based on the defined algorithms, that we configure to identify the optimal configurations. In the case of Bidirectional LSTM I arranged the result in the form of a huge matrix with different smoothness until I discovered the optimal one in terms of time and accuracy but also based on compatibility with future releases of LLVM and Clang.

We manage to improve the detection of vulnerabilities with the help of the built data set, the anonymization process and the implementation of the evaluation based on binary crossentropy Adam. We achieve this by comparing several types of neural networks to identify the most efficient approach in terms of accuracy and speed, including: short-term memory (LSTM), bidirectional LSTM, multilayer perceptron and convolutional neural networks (CNN). Of these, we obtained the best 96% accuracy with the LSTM bidirectional algorithm, but the others were also quite close.

In order to verify how well the implementation behaves, we analyze it on the SARD dataset where we normalize the files to analyze only the vulnerable/good use of the APIs, which means that we will not provide as input the rest of the files to our normalization source code. After transforming the code into tokens, we divide the data set into 2 groups: 30% testing and 70% training. We also turn all non-zero labels into label 1, which means vulnerable. There are 128.198 examples in the dataset. The 96% result is a 6% improvement over the results obtained by VulDeePecker on the SARD dataset, which includes programs containing 126 vulnerabilities identified by a CWE ID.

There are many open-source compilers that would support modifications in order to allow us to normalize the programs. We used clang because we can derive a tool much faster with it, but GCC also works. For an easier time making the tools we can just parse the AST generated by a compiler and use it as input with not much information loss. The way we designed this program is to make it possible to add line number to the vulnerability in the future in order to integrate it into an IDE.

A second conclusion we can draw is the rigidity of utilities available in production and the difficulty of modifying and understanding the mechanisms of these utilities. Such a task greatly increases the production time of derivative products based on them and such a tool could help patch faster the occurrence of vulnerabilities across production chain.

Chapter 7. Conclusions

This thesis focuses on defining an exploratory space for a secure-by-design automotive architecture, built by starting from an available vehicle architecture where security was not considered as required. We include in this thesis original contributions distributed in five chapters as follows: in chapter 2 we discuss the state of the art providing an overview of how an automotive central computing unit supporting security by design would look like; in chapter 3 we exploit a vehicle in order to expose its vulnerabilities and provide upgrade functionalities in chapters 4 by adding support for Wyliodrin and Yocto Project for easier interaction and in chapter 5 we extend the architecture with Android and add a speech recognition service as a way to extend the functionality of the system; in chapter 6 we enhance the security with the help of trusted execution security hardening mechanisms based on binary and system call analysis and static code analysis of the updates before applying them.

Our results published in the fields of IoT open education, industrial prototyping, automotive safety and security received great feedback from the scientific community. The exploratory space for the field of automotive security proves to be comprehensive and versatile with a growing number of contributions up to this date. In the following lines I will try to provide some conclusions to my work.

7.1 Thesis overview

The work for this thesis started when we tried to learn more about today vehicles, the field is very closed source and kind of elitist, no access to resources were provided and the traditional vendors for the OEMs were also off limit. We were interested to switching from proprietary code to open source in order to study the security problems in a controlled manner.

The first problem encountered was the difficult access to the interfaces, sensors or control elements of a car. The INVICTUS solution [3] was created, a solution which manages to connect a Raspberry Pi to the CAN network of a car using an ELM327 controller for communication with the OBD-II interface present on the car. The motivation was to bring the most used prototyping platform, Raspberry Pi, in connection with the automotive field. We managed to connect it to the internet and the GSM network making the vehicle itself connected to the internet. A lot of reverse engineering was required in order to provide this.

Next, we ran into the problem of the ease of development of this hardware platform, which for the time being requires a physical presence. So with [117] we came up with the solution to connect the board to the Wylodrin online development platform. Working on this platform, we have reached the secondary problem that does not directly affect you, but affects the community: not all automotive enthusiasts have programming and electronics skills. So in addition to connecting to the Wyliodrin platform, we created an extension to help people connect additional sensors and devices to the Raspberry Pi without the need for electronics knowledge, and we used the block programming option for people with no advanced programming skills to use. The motivation was to use an open-source solution to help as much of the automotive community as possible.

However, the Raspberry Pi is a platform only covers one use case of automotive digital platform that we presented in the state of the art [211], an architecture embodying multiple operating systems options and mixed security levels. Thus we came to the problem with bringing an industrial product into your architecture and in the same time keeping the evolution made previously. The solution came through the work [120] that manages to integrate an industrial device, which has connections similar to the extension for Raspberry Pi, within the Wyliodrin platform.

We can say that at this point we had solved the problem of the hardware platform for interacting with the ECUs within the automotive domain. The next problem we had was with choosing an operating system. We found the solution by using Yocto Project [118] and [119]. It is the most used distribution on hardware platforms, supported by Intel and almost everyone, we have the Automotive Grade Linux operating system as a starting point for our work and it offers the predictability and configurability we require for the job.

Once we have chosen the operating system, we moved forward to developing a higher control and provide accessibility to the user. We have probed the field of voice command recognition through the paper [115], but we consider that the biggest benefit is the integration of the mobile phone with the Wyliodrin application through the paper [116]. This way we could access the data of the automotive device sensors from the mobile phone, no longer requiring a physical connection to the OBD-II interface, or searching through incomplete information provided by the device monitor, or connecting to the centralizer provided previously, or even accessing a web page. The advantage being the access to a new powerful use case in our automotive digital platform. We also provided control of the car within the application, but especially a new path to features that also take advantage of mobile phone data, or devices related to it, such as unlocking and starting

the vehicle depending on the location of the mobile device, or income data from the fitness bracelet. The motivation was to create a complete automotive architecture which could be under our control, open sourced and with full support for monitoring the interactions.

A new stop in our journey now that we have managed to fully develop a system from the bottom up open source, solving the closed source problem that we started with. We realize that there is no more the case for security through obscurity in our testbed and automotive field in general, and now we need to make sure that what we have done, this transition from proprietary code to open source has security. So we have to go back over the whole system in the same bottom up approach and provide security and hardening of the device in mind.

We start this as low as we can, disregarding the hardware platform for the beginning, because most of the attacks at this level are side-channel and if the person is already in the car to do that, it doesn't make much sense in our case. So we start from the level of the operating system and consider that one of the big problems in automotive systems is the escalation of privileges, which in our mixed world scenario is fundamental, is one think to turn on the lights and another to control the steering. Thus we took care of the protection of the operating system against rootkits through the work [111]. Sprobes [111] is a theoretically concept developed for Linux kernel version 2.6 and validated in a synthetic environment. We also used them and implemented them in an automotive specific scenarios, unfortunately the result proved to not be convincing enough for the industrial context that we required.

Going further through the development, one step was to update the code. The problem is how sure we are that the new uploaded code is not somehow vulnerable. We addressed this issue in [121], which analyzes the vulnerabilities of a source code using a token code transaction and tested three types of machine learning algorithms to classify the vulnerability, reaching an accuracy of 96.27%. We wanted a confirmation that the new code that you will upload on the platform does not increase the attach surface and had no security impact as such.

We can now consider the operating system secured, but we still have to secure external services code available in binary form, something common in the automotive. The problem is how we can check binary applications on which we do not have access to the source code. The solution comes through the work [109]. We iterate on the previous work [121], but this time we try different ML algorithms to detect vulnerable binaries reaching 95% using random forest. The motivation is to check binary for vulnerabilities.

The accuracy of 95% is not the same as 96.27%. Moving forward we decided that we also require a solution to protect running processes and especially the vulnerable security policy database. Through paper [110] we propose a HIDS based on system calls using a decision tree. The motivation is to improve the system. In the end we have the architecture presented in [211] providing security by design.

7.2 Contributions

The work presented in this thesis is elaborated in the field of operating systems security with applicability into the automotive industry. The thesis contains original contributions to the fields of IoT open education, industrial prototyping, automotive safety and security, enabling further research and development for the automotive industry.

While I was working on the thesis, especially during the hand-on prototyping I observed not only the fact that a certain set of technical skills were required for developing these systems. A summary of skills together with how they can be integrated inside the conventional embedded system education is provided in the articles [115], [116], [117], [118], [119] and [120] presented in chapters 6 and 7. Here you can also find the technology choices and the practical implementations.

When discussing automotive we imagine a close source and very elitist community. With the work we did in [120] we managed to demonstrate that even industrial hardware can become accessible to not so technical individuals and all that in a more secure manner, while still permit being programed with the help of graphical programming languages so that non-technical people be able to interact and configure them according to their needs and be able to understand the technology. We are hoping to be able to do the same for the automotive field as to provide not only a more secure environment but also one that is more accessible for the non-technical user to interact with. All that because in the era of autonomous and flying vehicles everyone should be able to understand the technology around and even customize their rides according to their needs and desires. My intent is to provide a proper framework and corresponding metrics and methodology in order to comply with both the educational and security sides of the work.

In the first chapter I introduce the notions, agents, factors and attributes involved and used in the automotive field that are taken into consideration for this work. I present the state of the art and the general characteristics of the automotive security field, also by the end of the chapter I present a couple of security models which can be implemented in already available frameworks for identifying security requirements of automotive systems. We do a comparison of the tools, methods and processes available for threat and risk assessment.

Contribution1: Automotive safety

Article - Invictus: Open-Source Solution for an Intelligent Vehicle Security System

This is an exploration into safety criticality and reliability aspects of a vehicle internal networks architecture and presents a functional architecture, use-case studies and hacking experiments over one of the most used network protocols inside the vehicle, the CAN network. I focused my work here on CAN based ECU exploits since from our analysis this is the most spread communication inside the vehicle, but in this chapter we also move past that work and discuss exploits available in all the internal and external communication protocols. I also discuss the widespread use of Linux exploits which we have on In-vehicle-infotainment (IVI) and Autonomous Drive (AD) parts of the vehicle. This part is not overlooked but much more research was done on this side and we only review the most relevant ones for the automotive industry. I do not discuss other OSs available such as QNX or Android because those are not part of our work but discuss CAN for example because the most powerful Linux based ECUs interact with it and even control the vehicle, driver, passengers and pedestrian safety. Through these practical experiments that we carried out or documented we manage to provide a real time measurement, one that we can compare with the results we obtain through our provided solutions.

Contribution2: Automotive security

Article - Adrem: System call based intrusion detection framework
Article - Applications of machine learning in malware detection
Article - Vulnerability analysis pipeline using compiler based source to source translation and deep learning
Article - Observations over SPROBES mechanism on the TrustZone architecture

Moving further in the next section I discuss runtime solutions. In chapter 4 I discuss machine learning based solutions which we validated in articles [109] and [110] where we study classifications. In article [110] we discuss how we can use system calls in order to identify malicious activity and in article [109] we discuss how interaction with various system libraries can help classify malware in categories. With these two solutions we provide a means of identifying and classifying malicious interactions through the datasets we used and provided in our work.

In chapter 5 we have another type of runtime assessment solutions, ones which is not an external solution but based on the internal support of the hardware and its functionalities such is the case for article [111] or the available source code as for the work described in article [121]. For the work in [111] I used ARM TrustZone support and extended Linux kernel and arm-trusted-firmware in order to provide a new type of probes, secure ones called sprobes which inspect registers and pages in order to protect the operating system against rootkits or similar threats and keep the operating system free of contamination even when the contamination is already packaged inside the system. While for the work in [121] I constructed a source code database where the source code content was abstracted in order to apply on top machine learning algorithms which are trained to detect CVE or CWE issues under the form of vulnerabilities of the source code.

Contribution3: IoT open education

Article - Teaching computer engineering concepts to non-technical students
Book - Learning Embedded Linux Using the Yocto Project
Book - Linux: Embedded Development

Interaction with vehicle On-Board Diagnostics (OBD-II) provides access to vehicles to an entire category of individuals and this is only one of the entry points inside the vehicle architecture. We need to make sure that these types of interactions, from individuals complementary to the automotive industry, are not putting in danger the drivers, passengers and the other traffic participants. Help non-technical people interact with embedded devices, acquire computer engineering skills by interacting with programming with the help of Scratch and Blockly plus a Raspberry Pi hardware extension. The technical aspects revolve around the use of Wyliodrin Lab, an in-house developed extension platform and a web interface for controlling the student's workspace, inspecting their progress and grade them. With the help of the two books I teach technical people how to interact with Yocto Project, a reference collaboration project which helps developers create and interact with custom Linux-based operating systems regardless of the hardware architecture. We discover not only what the Yocto Project is and how to use it but also how the Yocto Project influences industries such as automotive, IoT and open source industry in general.

Contribution4: Industrial prototyping

Article - Hardware design and implementation of an Intelligent Haptic Robotic Glove
Article - Android application that controls Internet of Things devices Wyliodrin COMPANION

Design and implement a robotic glove in order to help people which suffered an accident recover motricity in their fingers. The glove is activated by predefined speech commands and is design in order to assist medical staff help patients recover quicker, from here we extrapolated a solution which simulates an automotive ECU connected to various sensors controlled with the same

speech recognition framework. Develop an Android app in order to control and interact with embedded devices remotely a companion to the Wyliodrin Studio IDE, used for programming embedded devices by non-technical students with Scratch and similar graphic programming languages.

In the conclusion of my thesis, I discuss how all these contributions are beneficial for a more robust operating system and automotive domain. Discuss about how for a truly secure ecosystem we need solutions applied to multiple levels of interaction of the systems, ones that provide metrics which can be observed, validated and improved over time. Dynamic solution which can not only ensure but also maintain the security of a vehicle, since it stays on the roads for at least 10 or 20 years.

7.3 Future work

In the journey to change an ecosystem with focus on security by obscurity to one providing security by design there are multiple directions of future contributions to be made. We grouped our contributions in four categories called IoT open education, industrial prototyping, automotive safety and automotive security and there are tons of directions to be followed but the main ones involved the improvement to the automotive architecture that we designed.

We want first to focus on making the process of integration automatic in order to facilitate the support of new components and services. This can be done with the already available technologies and hardware support and has the role of simplifying the interaction but does not bring much added value improving mainly on the state of the art.

A valuable contribution that we would want to pursue is to intrinsically analyse the data from the testbed that we constructed. Bringing more automotive specific support would be a good valuable contribution for that, also extending the testbed in the process.

More punctual and also insightful contributions would provide for a mechanism of clustering the thread information as part of the HIDS study, we could employ here similar algorithm with what we used for the binary analysis where we defined classes of malicious and benign support. Similar contributions could be provided from extending the static source code analysis components where we could enable the integration of more complex inputs. More details are available in each chapter.

References

[1] Robert N. Charette, “This car runs on code,” IEEESpectrum, Feb 01, 2009. [Online]. Available: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code. [Accessed Aug 16, 2019].

[2] T.C.ISO/TC-22. Iso 15765: Road vehicles – diagnostic communication over controller area network (docan). ISO, 2004-2016.

[3] Cristian Lupu, Alexandru Jan Văduva, Răzvan Rughiniș. “Invictus: Open-source solution for an intelligent vehicle system,” 2016 15th RoEduNet Conference: Networking in Education and Research (RoEduNet NER, (15):1–5, 2016.

[4] Alexandru Radovici, Ioana Culic, “Online Platform for Embedded Devices”, The 10th International Scientific Conference eLearning and software for Education Bucharest, April 24-25, 2014.

[5] Lukas Zöscher, Jasmin Grosinger, Raphael Spreitzer, Ulrich Muehlmann, Hannes Gross, Wolfgang Bösch, “Concept for a security aware automatic fare collection system using HF/UHF dual banc RFID transponders”, 45th European Solid-State Device Research Conference (ESSDERC), 2015.

[6] Sastry Duri, Marco Gruteser, Xuan Liu, Paul Moskowitz, Renald Perez, Moninder Singh, Jung-Mu Tang, “Framework for security and privacy in automotive telematics”, WMC '02 Proceedings of the 2nd international workshop on Mobile commerce, 2002.

[7] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, Ahmad-Reza Sadeghi, “Smart keys for cyber-cars: secure smartphonebased NFC-enabled car immobilizer”, CODASPY '13 Proceedings of the third ACM conference on Data and application security and privacy, 2013.

[8] Răzvan Tătăroiu, Dan Tudose, “Remote Monitoring and Control of Wireless Sensor Networks”, 17th International Conference of Control Systems and Computer Science CSCS17, vol. 1, pp. 187-192, 2009.

[9] S. Craig, The Car Hacker’s Handbook – “A Guide for the Penetration Tester”, San Francisco: No Starch Press, 2016.

[10] Dr. Charlie Miller, Chris Valesek, “Remote exploitation of an Unaltered Passenger Vehicle”, Aug 10, 2015.

[11] Sasan Jafarnejad, Lara Codeca, Walter Bronzi, Raphael Frank, Thomas Engel, “A Car Hacking Experiment: When Connectivity meets Vulnerability”, IEEE Globecom Workshops (GC Wkshps), 2015.

[12] T. C. ISO/TC 22, ISO 11898: Road vehicles -- Controller area network (CAN), ISO, 1993-2015.

[13] E. Electronics, “ELM327 OBD to RS232 Interpreter Manual,” ELM Electronics – Circuits for the Hobbyist, 2016.

[14] SIMCom, SIM5218 Serial AT Command Manual, 1.21 ed., Shanghai: SIM Tech, 2011.

[15] E. Electronics, “ELM327 OBD to RS232 Interpreter Manual,” ELM Electronics – Circuits for the Hobbyist, 2016.

[16] X. Ge, H. Vijayakumar, and T. Jaeger. “SPROBES: Enforcing kernel code integrity on the trustzone architecture”, In Proceedings of the 2014 Mobile Security Technologies (MoST) workshop, 2014.

[17] R. Toulson and T. Wilmshurst, “Fast and Effective Embedded Systems Design: Applying in ARM mbed”, Oxford: Newnes, 2012
[18] CAN in Automation. “Challenges in automotive applications”, 2005. [Online]. Available: http://www.cancia.org/applications/passengercars/challenge.html. [Accessed Aug 16, 2019].
[19] Jon Erickson, “Hacking: The Art of Exploitation”, 2nd edition, No Starch Press, Inc., 2007.
[20] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage “Experimental Security Analysis of a Modern Automobile”, USENIX Security Symposium, 2010.
[21] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, , Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno, “Comprehensive Experimental Analyses of Automotive Attack Surfaces”, USENIX Security Symposium, 2011.
[22] F. Simonot-Lion, “In-car embedded electronic architectures: how to ensure their safety”, presented at the 5th IFAC Int. Conf. Fieldbus Systems and Their Applications (FeT 2003), Aveiro, Portugal, 2003.
[23] P. Kleberger, T. Olovsson, and E. Jonsson, “Security aspects of the in-vehicle network in the connected car,” in Intelligent Vehicles Symposium (IV), 2011 IEEE. IEEE, 2011, pp. 528–533.
[24] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaˆaniche, and Y. Laarouchi, “Survey on security threats and protection mechanisms in embedded automotive networks,” in Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on. IEEE, 2013, pp. 1–12.
[25] M. Wolf, A. Weimerskirch, and T. Wollinger, “State of the art: Embedding security in vehicles,” EURASIP Journal on Embedded Systems, vol. 2007, no. 1, p. 074706, 2007.
[26] T. Garfinkel, M. Rosenblum et al., “A virtual machine introspection-based architecture for intrusion detection.” in NDSS, 2003.
[27] P. D. Noble, “When virtual is better than real [operating system relocation to virtual machines],” in Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on. IEEE, 2001, pp. 133–138.
[28] L. Litty, H. A. Lagar-Cavilla, and D. Lie, “Hypervisor support for identifying covertly executing binaries.” in USENIX Security Symposium, 2008, pp. 243–258.
[29] M. I. Sharif, W. Lee, W. Cui, and A. Lanzi, “Secure in-vm monitoring using hardware virtualization,” in Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009, pp. 477–487.
[30] X. Jiang and X. Wang, “out-of-the-box monitoring of vm-based highinteraction honeypots,” in Recent Advances in Intrusion Detection. Springer, 2007, pp. 198–218.
[31] M. Rosenblum and T. Garfinkel, “Virtual machine monitors: Current technology and future trends,” Computer, vol. 38, no. 5, pp. 39–47, 2005.
[32] K. Nance, B. Hay, and M. Bishop, “virtual machine introspection,” IEEE Computer Society, 2008.
[33] B. D. Payne, M. Carbone, M. Sharif, and W. Lee, “Lares: An architecture for secure active monitoring using virtualization,” in Security and Privacy, 2008. SP 2008. IEEE Symposium on. IEEE, 2008, pp. 233–247.

[34] A. Seshadri, M. Luk, N. Qu, and A. Perrig, “Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses,” ACM SIGOPS Operating Systems Review, vol. 41, no. 6, pp. 335–350, 2007.
[35] R. Riley, X. Jiang, and D. Xu, “Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing,” in Recent Advances in Intrusion Detection. Springer, 2008, pp. 1–20.
[36] X. Zhang, L. van Doorn, T. Jaeger, R. Perez, and R. Sailer, “Secure coprocessor-based intrusion detection,” in Proceedings of the 10th workshop on ACM SIGOPS European workshop. ACM, 2002, pp. 239–242.
[37] N. L. Petroni Jr, T. Fraser, J. Molina, and W. A. Arbaugh, “Copilota coprocessor-based kernel runtime integrity monitor.” in USENIX Security Symposium. San Diego, USA, 2004, pp. 179–194.
[38] J. Wang, A. Stavrou, and A. Ghosh, “Hypercheck: A hardware-assisted integrity monitor,” in Recent Advances in Intrusion Detection. Springer, 2010, pp. 158–177.
[39] PaX Team, “Documentation for the PaX project - overall description,” 2008. [Online]. Available: https://pax.grsecurity.net/docs/pax.txt. [Accessed Aug 16, 2019].
[40] S. Andersen and V. Abella, “Data execution prevention. Changes to functionality in microsoft windows xp service pack 2, part 3: Memory protection technologies,” 2004.
[41] OpenBSD, “OpenBSD Innovations,” 2008. [Online]. Available: http://www.openbsd.org/innovations.html. [Accessed Aug 16, 2019].
[42] H. Shacham, “The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86),” in Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007, pp. 552–561.
[43] T. Wang, K. Lu, L. Lu, S. Chung, and W. Lee, “Jekyll on ios: when benign apps become evil,” in USENIX Security Symposium, 2013, pp. 559–572.
[44] ARM Inc., “ARM Security Technology Building a Secure System using TrustZone Technology,” 2009. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html. [Accessed Aug 16, 2019].
[45] ARM Inc., “ARM OS use of translation tables”, 2018 [Online]. Available: https://developer.arm.com/products/architecture/cpu-architecture/a-profile/docs/100940/latest/os-use-of-translation-tables. [Accessed Aug 16, 2019].
[46] J. Winter, “Trusted computing building blocks for embedded linux-based arm trustzone platforms,” in Proceedings of the 3rd ACM workshop on Scalable trusted computing. ACM, 2008, pp. 21–30.
[47] Azab, Ahmed M., et al. "Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world." Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014.
[48] ARM Inc., “SMC CALLING CONVENTION System Software on ARM Platforms”, 2018 [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.den0028b/ARM_DEN0028B_SMC_Calling_Convention.pdf. [Accessed Aug 16, 2019].
[49] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In Recent Advances in Intrusion Detection (RAID), Toulouse, France, October 2000.

[50] Christopher Kruegel, Darren Mutz, Fredrik Valeur, Giovanni Vigna, On the Detection of Anomalous System Call Arguments, University of California, Santa Barbara, Department of Computer Science, 2003.
[51] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin, Intrusion detection by machine learning: A review, Elsevier, 2009.
[52] Warrender, C., S. Forrrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on Security and Privacy, Oakland, CA, 1999, pp. 133–145.
[53] Liu, Z., G. Florez, and S.M. Bridges. A comparison of input representations in neural networks: A case study In intrusion detection. In: Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, 2002.
[54] Liao, Y.H. and V.R. Vemuri. Use of K-nearest neighbor classifier for intrusion detection. Computers & Security, 2002.
[55] Chebrolu, S., A. Abraham, and J.P. Thomas. Feature deduction and ensemble design of intrusion detection systems. Computers & Security, 2005.
[56] Gideon Creech, Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks, PhD dissertation, The University of New South Wales, Canberra, 2013.
[57] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Logstaff, “A Sense of Self for Unix process”, Proceedings of 1996 IEEE Symposium on Computer Security and Privacy 120-128, 1996.
[58] Amr S. Abed, Charles Clancy, David S. Levy, Intrusion Detection System for Applications Using Linux Containers, Department of Electrical & Computer Engineering, Virginia Tech, Blacksburg, VA, USA, 2015.
[59] Ye, N., X.Y. Li, Q. Chen, S.M. Emran, and M.M Xu. Probabilistic techniques for intrusion detection based on computer audit data. IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans, 2001.
[60] Lee, W. and W. Fan. Mining system audit data: Opportunities and challenges. SIGMOD Record, 2001.
[61] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security (JCS), 6(3):151{180, 1998.
[62] Scikit-learn “Scikit-learn User guide”, 2017 [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed Mar, 2017].
[63] Docker, “Docker containers: How secure are they?” 2013 [Online]. Available: http://blog.docker.com/2013/08/containers-docker-how-secure-are-they. [Accessed Mar, 2017].
[64] Sysdig, “Sysdig Documentation”, 2017 [Online]. Available: https://docs.sysdig.com/. [Accessed Mar, 2017].
[65] Sysdig, “Sysdig vs dtrace vs strace: A technical discussion”, 2017 2017 [Online]. Available: https://sysdig.com/blog/sysdig-vs-dtrace-vs-strace-a-technical-discussion/. [Accessed Mar, 2017].
[66] Scikit-learn “Scikit-learn Getting Started”, 2017 [Online]. Available: https://scikit-learn.org/stable/getting_started.html. [Accessed Mar, 2017].
[67] Github, “Test_db repository”, 2017 [Online]. Available: https://github.com/datacharmer/test_db. [Accessed Mar, 2017].

[68] MySQL, “Mysqlslap User manual”, 2017 [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/mysqlslap.html. [Accessed Mar, 2017].
[69] Sqlmap, “Sqlmap User manual”, 2017 [Online]. Available: http://sqlmap.org. [Accessed Mar, 2017].
[70] Al-Sakib Khan Pathan, The State of the Art in Intrusion Prevention and Detection, CRC Press, Taylor & Francis Group, LLC, ISBN 978-1-4822-0351-6, 2014
[71] Scikit-learn “Scikit-learn User guide”, 2017 [Online]. Available: http://scikit-learn.org/stable/modules/manifold.html. [Accessed Mar, 2017].
[72] Apache, “Frequent pattern mining – RDD-based API”, 2017 [Online]. Available: https://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html. [Accessed Mar, 2017].
[73] Ali Jalali and Sujay Sanghavi. Learning the Dependence Graph of Time Series with Latent Factors, University of Texas at Austin, 1 University Station Code:C0806, Austin, TX 78712 USA, Jun 9, 2011.
[74] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna. On the detection of anomalous system call arguments. In European Symposium on Research in Computer Security, Gjvik, Norway, October 2003.
[75] G. Tandon and P. Chan. Learning rules from system call arguments and sequences for anomaly detection. In ICDM Workshop on Data Mining for Computer Security (DMSEC), pages 20-29, 2003.
[76] Sin Yeung Lee, Wai Lup Low, Pei Yuen Wong. Learning Fingerprints for a Database Intrusion Detection System. In 7th European Symposium on Research in Computer Security (ESORICS), 2002.
[77] Niels Provos. Improving host security with system call policies. In USENIX Security Symposium, Washington, DC, USA, August 2003.
[78] David Wagner and Drew Dean. Intrusion detection via static analysis. In IEEE Symposium on Security and Privacy, Oakland, CA, May 2001.
[79] R. Sekar, M. Bendre, P. Bollineni, and D. Dhurjati. A fast automaton-based method for detecting anomalous program behaviors. In IEEE Symposium on Security and Privacy, 2001.
[80] T. Garnkel, B. Pfar, and M. Rosenblum. Ostia: A delegating architecture for secure system call interposition. In USENIX Security Symposium, Washington, DC, USA, August 2003.
[81] H. Feng, J.T. Gin, Y. Huang, S. Jha, W. Lee, and B. P. Miller. Formalizing sensitivity in static analysis forintrusion detection. In IEEE Symposium on Security and Privacy, 2004.
[82] Debin Gao, Michael K. Reiter, and Dawn Song. Gray-box extraction of execution graphs for anomaly detection. In ACM conference on Computer and Communications Security (CCS), pages 318{329, Washington, DC, October 2004.
[83] Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. Linux J. 2014(239), 2 (2014)
[84] Yeung, D.Y., Ding, Y.: Host-based intrusion detection using dynamic and static behavioral models. Pattern Recogn. 36(1), 229–243 (2003)
[85] Warrender, C., S. Forrrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on Security and Privacy, Oakland, CA, 1999, pp. 133–145.

[86] Taesoo Kim, Nickolai Zeldovich, Practical and effective sandboxing for non-root users. In: 2013 USENIX Annual Technical Conference (USENIX ATC '13), USENIX Association, 2013.
[87] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In Recent Advances in Intrusion Detection (RAID), Toulouse, France, October 2000.
[88] Towards data Science, "The Random Forest Algorithm," 2017 [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499cd. [Accessed Aug 16, 2019].
[89] XGBoost, "XGBoost Documentation," 2017 [Online]. Available: http://xgboost.readthedocs.io/en/latest/model.html. [Accessed Aug 16, 2019].
[90] Felan Carlo C. Garcia and Felix P. Muga II. Random forest for malware classification. CoRR, abs/1609.07770, 2016.
[91] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. CoRR, abs/1603.02754, 2016.
[92] S. Hahn, M. Protsenko, and T. Muller. Comparative evaluation of machine learning based malware detection on android. In M. Meier, D. Reinhardt, and S. Wendzel, editors, Sicherheit 2016 - Sicherheit, Schutz und Zuverl•assigkeit, pages 79{88, Bonn, 2016. Gesellschaft fur Informatik e.V.
[93] M. S. Alam and S. T. Vuong 2013. Random forest classification for detecting android malware. IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, 2013 pp. 663-669, 2013.
[94] B. Sun, Q. Li, Y. Guo, Q. Wen, X. Lin, and W. Liu. Malware family classification method based on static feature extraction. 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 507-513, 2017.
[95] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In Recent Advances in Intrusion Detection (RAID), Toulouse, France, October 2000.
[96] Jonathan Crowe. New "ovidiy stealer" malware makes credential theft cheap and easy, 2017 [Online]. Available: https://blog.barkly.com/ovidiy-stealer-credential-theft-malware. [Accessed Aug 16, 2019].
[97] Ali Jalali and Sujay Sanghavi. Learning the Dependence Graph of Time Series with Latent Factors, University of Texas at Austin, 1 University Station Code:C0806, Austin, TX 78712 USA, Jun 9, 2011.
[98] Anton Ivanov and Orkhan Mamedov. The return of mamba ransomware, 2017 [Online]. Available: https://securelist.com/the-return-of-mamba-ransomware/79403/. [Accessed Aug 16, 2019].
[99] Andy Green. "Dnsmessenger: 2017's most beloved remote access trojan (rat)," 2017 [Online]. Available: https://blog.varonis.com/dnsmessenger-2017s-beloved-remote-access-trojan-rat/. [Accessed: 17- Nov- 2017].
[100] Proofpoint, "Adylkuzz cryptocurrency mining malware spreading for weeks," 2017 [Online]. Available: https://www.proofpoint.com/us/threatinsight/post/adylkuzz-cryptocurrency-mining-malware-spreading-for-weeks-via-eternalblue-doublepulsar. [Accessed Aug 16, 2019].

[101] Sandbox, “Free Automated Malware Analysis Service powered by Falcon Sandbox,” 2017 [Online]. Available: https://reverse.it/. [Accessed: 17- Nov- 2017].
[102] VirusShare, “VirusShare Research,” 2017 [Online]. Available: https://virusshare.com/. [Accessed: 10- Jan- 2018].
[103] Freewareles, “Freeware Files Free Software Downloads and Reviews,” 2017 [Online]. Available: https://freewareles.com/. [Accessed: 26- Nov- 2017].
[104] CNET, “Windows PC Software Free Downloads and Reviews,” 2017. [Online]. Available: http://download.cnet.com/windows/. [Accessed: 25- Nov- 2017].
[105] M. Baykara and B. Sekin. A novel approach to ransomware: Designing a safe zone system. In 2018 6th International Symposium on Digital Forensic and Security (ISDFS), pages 1-5, March 2018.
[106] Q. Chen and R. A. Bridges. Automated behavioral analysis of malware: A case study of wannacry ransomware. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 454-460, Dec 2017.
[107] Cockoo, “Cockoo Sandbox,” 2010. [Online]. Available: https://cuckoosandbox.org. [Accessed: 14- Nov- 2017].
[108] CNET, “CNET News,” 2017. [Online]. Available: https://www.cnet.com/news/wannacry-wannacrypt-uiwix-ransomware-everything-you-need-to-know. [Accessed: 14- Nov- 2017].
[109] JA Văduva, VR Pașca, IM Florea, R RUGHINIȘ. Applications of Machine Learning in Malware Detection. In 2019 eLearning & Software for Education Vol. 2, April 2019.
[110] JA Văduva, RE Chișcariu, I Culic, IM Florea, R RUGHINIȘ. ADREM: System Call Based Intrusion Detection Framework. In 2019 eLearning & Software for Education Vol. 1, Jan 2019.
[111] JA Vaduva, S Dascalu, IM Florea, I Culic, R Rughinis. Observations over SPROBES Mechanism on the TrustZone Architecture. 2019 22nd International Conference on Control Systems and Computer Science (CSCS), pages 317-322, May 2019.
[112] Ravinder R. Ravula, Kathy J. Liszka, and Chien-Chung Chan, Learning Attack Features from Static and Dynamic Analysis of Malware, International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, pages 109-125, 2011.
[113] Mcafee, “Reports: a good decade for cybercrime,” 2010 [Online]. Available: http://www.mcafee.com/us/resources/reports/rp-good-decade-for-cybercrime.pdf. [Accessed Aug 16, 2017].
[114] Messagelabs, “Reports: 2010 forensics evaluation,” 2011 [Online]. Available: http://www.messagelabs.com/mlireport/MLI_2011_01_January_Final_en-us.pdf. [Accessed Aug 16, 2017].
[115] N Popescu, D Popescu, A Cozma, AJ Vaduva. Hardware design and implementation of an Intelligent Haptic Robotic Glove. 2014 International Conference and Exposition on Electrical and Power Engineering (EPE), pages 174-177, October 2014.
[116] EA Stoican, A Radovici, A Văduva. Android application that controls Internet of Things devices Wyliodrin COMPANION. 2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER), pages 233-237, September 2015.
[117] JA VADUVA, A RADOVICI, I CULIC. Teaching computer engineering concepts to non-technical students. 2019 " CAROL I" National Defence University Publishing House.
[118] Alexandru Vaduva. Learning Embedded Linux Using the Yocto Project. Ed. Packt Publishing Ltd, June 2015.

[119] Alexandru Vaduva. Linux: Embedded Development. Ed. Packt Publishing Ltd, September 2016.
[120] I Culic, A Radovici, L Moraru, C Radu, JA Vaduva. Porting JerryScript to NXP Rapid Prototyping Kit, 2020 eLearning & Software for Education, April 2020.
[121] JA Vaduva, S Dascalu, I Culic, A Radovici, R Rughinis. Vulnerability analysis pipeline using compiler based source to source translation and deep learning. 2020 eLearning & Software for Education, April 2020.
[122] NSA, "NSA's Center for Assured Software. Juliet Test Suite C/C++," 2019. [Online]. Available: https://samate.nist.gov/ SARD/around.php#juliet_documents. [Accessed Jun 8, 2019].
[123] CERN, "Guide for common Vulnerabilities using C language," 2009. [Online]. Available: https://security.web.cern.ch/security/ recommendations/en/codetools/c.shtml. [Accessed Jun 8, 2019].
[124] M. Stevanovic. Advanced C and C++ Compiling. Apress, 2014. ISBN 9781430266686.
[125] C. manning Publications. Deep Learning with Python, Francois Chollet, 2018: Python. Deep learning with Python. Bukupedia, 2018.
[126] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pp. 1097–1105, 2012.
[127] Christian Sandberg, HoliSec Automotive Security and Privacy Holistic Approach to Improve Data Security, March 2003.
[128] EVITA Project, "E-safety Vehicle Intrusion Protected Applications (EVITA)". [Online]. http://www.evita-project.org/. [Accessed Nov 14, 2016]
[129] ETSI. Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA). Technical Report TR 102 893, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Mar. 2010.
[130] ISO/SAE DIS 21434. Road vehicles — Cybersecurity engineering. Standard. International Organization for Standardization, under development.
[131] SAE J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. Standard. Vehicle Cybersecurity Systems Engineering Committee, 2016.
[132] NHTSA. National Highway Traffic Safety Administration (NHTSA). [Online]. http://www.nhtsa.gov. [Accessed Dec 19, 2016].
[133] AUTOSAR Consortium. AUTomotive Open System ARchitecture. [Online]. https://www.autosar.org/. [Accessed Nov 14, 2016].
[134] SAE J3061 Cybersecurity Guide-book for Cyber-Physical Automotive Systems. Standard. Vehicle Electrical System Security Committee, 2016.
[135] SAE International. 'Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document Superseding J1939 JUN2012'. In: SAE International, Aug. 2013. [Online]. https://saemobilus.sae.org/content/J1939_201308. [Accessed Nov 14, 2016].
[136] SAE International. 'European Brake Fluid Technology - J1709_200607, cancelled Jul 2006'. In: SAE International, Jul. 2006.
[137] ETSI. Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA). Technical Report TR 102 893, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Mar. 2010.

[138] ETSI. Intelligent Transport Systems (ITS); Communications Architecture. European Standard EN 302 665, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Sept. 2010.
[139] ETSI. Intelligent Transport Systems (ITS); Security; Security Services and Architecture. Technical Specification TS 102 731, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Sept. 2010.
[140] ETSI. Intelligent Transport Systems (ITS); Security; Trust and Privacy Management. Technical Specification TS 102 941, v1.2.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, May 2018.
[141] ETSI. Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management. Technical Specification TS 102 940, v1.3.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, April 2018.
[142] ETSI. Intelligent Transport Systems (ITS); Security; Security header and certificate formats. Technical Specification TS 103 097 v1.3.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Oct. 2017.
[143] ETSI. CYBER; Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA). Technical Specification TS 102 165-1, v5.2.3. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Oct. 2010.
[144] ISO/TC 204: Intelligent transport systems. Standard ISO/TC 211 - ISO/TC 204 WG: GIS-ITS under development. International Organization for Standardization.
[145] CEN/TC 278 Intelligent transport systems. Standard under development. European Committee for Standardization.
[146] IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements. Standard IEC 61508-1:2010. International Electrotechnical Commission
[147] ISO/IEC 27034: Information technology — Security techniques — Application security — Part 1: Overview and concepts. Standard ISO/IEC 27034-1:2011. International Organization for Standardization.
[148] ISO/TC 22: Road vehicles. Standard under development. International Organization for Standardization.
[149] ISO 26262 road vehicles functional safety part 1–10. Standard. International Organization for Standardization, 2011.
[150] ISO/DIS 13400-1: Road vehicles — Diagnostic communication over Internet Protocol (DoIP) — Part 1: General information and use case definition. Standard. International Organization for Standardization.
[151] ISO 15118: Road vehicles — Vehicle to grid communication interface — Part 1: General information and use-case definition. Standard 15118-1:2019. International Organization for Standardization.
[152] ISO 15765: Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. Standard 15765-2:2016. International Organization for Standardization.
[153] IEEE. 1609 WG - Dedicated Short Range Communication Working Group. Oct. 2013. [Online]. https://standards.ieee.org/develop/wg/1609_WG.html. [Accessed Nov 14, 2016].
[154] 802.11p-2010 - IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular

Environments. Jun. 2010. [Online]. https://standards.ieee.org/standard/802_11p-2010.html. [Accessed Nov 14, 2016].
[155] L. L. Bello. 'Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging'. In: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA). Sept. 2014, pp. 1–8. DOI: 10.1109/ETFA.2014.7005251.
[156] IEEE-Standards Association. 802.1BA-2011 - IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems. Tech. rep. 2011.
[157] IEEE-Standards Association. P802.1AS - Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications. Tech. rep. 2011.
[158] IEEE-Standards Association. 802.1Qav-2009 - IEEE Standard for Local and metropolitan area networks– Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. Tech. rep. 2009.
[159] IEEE-Standards Association. 802.1Qat-2010 - IEEE Standard for Local and metropolitan area networks–Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). Tech. rep. 2010.
[160] CARONTE Project. Creating an Agenda for Research On Transportation sEcurity. URL: http://www.caronte-project.eu/ (visited on 14th Nov. 2016).
[161] HEAVENS: HEAling Vulnerabilities to ENhance Software Security and Safety. [Online]. http://www.vinnova.se/sv/Resultat/Projekt/Effekta/HEAVENS-HEAling-Vulnerabilities-to-ENhance-Software-Security-and-Safety/. [Accessed Nov 25, 2016].
[162] SeFram. Security framework for vehicle communication (SeFram). [Online]. https://research.chalmers.se/en/project/6225. [Accessed June 11, 2020].
[163] SESAMO. Security and Safety Modeling (SESAMO). [Online]. http://www.sesamo-project.eu. [Accessed Dec 19, 2016].
[164] EVITA Project. E-safety Vehicle Intrusion Protected Applications (EVITA). [Online]. http://www.evitaproject.org/. [Accessed Nov 14, 2016].
[165] D. K. Nilsson and U. E. Larson. 'Simulated Attacks on CAN Buses: Vehicle Virus'. In: Proceedings of the 5th IASTED International Conference on Communication Systems and Networks. AsiaCSN '08. Anaheim, CA, USA. Palma de Mallorca, Spain: ACTA Press, 2008, pp. 66–72. ISBN: 978-0-88986-758-1.URL:http://portal.acm.org/citation.cfm?id=1713277.1713292.
[166] M. Wolf, A. Weimerskirch and C. Paar. 'Security in Automotive Bus Systems'. In: Workshop on Embedded IT-Security in Cars. Bochum, Germany, Nov. 2004.
[167] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway et al. 'Experimental Security Analysis of a Modern Automobile'. In: 2010 IEEE Symposium on Security and Privacy (SP). IEEE, 2010, pp. 447–462. ISBN: 1424468949. DOI: 10.1109/SP.2010.34.
[168] T. Hoppe and J. Dittmann. 'Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy'. In: Proceedings of the 2nd Workshop on Embedded Systems Security (WESS). Salzburg, Austria, 2007.
[169] J. D. Howard and T. A. Longstaff. 'A Common Language for Computer Security Incidents'. In: Sandia Report: SAND98-8667 (1998). [Online]. http://www.cert.org/research/taxonomy_988667.pdf. [Accessed Nov 14, 2016].
[170] A. Lang, J. Dittmann, S. Kiltz and T. Hoppe. 'Future Perspectives: The Car and Its IP-Address — A Potential Safety and Security Risk Assessment'. In: Proceedings of the 26th

International Conference on Computer Safety, Reliability, and Security (SAFECOMP '07). SAFECOMP '07. Nuremberg, Germany, Sept. 2007, pp. 40–53.
[171] T. Hoppe, S. Kiltz and J. Dittmann. 'Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures'. In: Computer Safety, Reliability, and Security. Ed. by M. D. Harrison and M.-A. Sujan. Lecture Notes in Computer Science 5219. Springer Berlin Heidelberg, Jan. 2008, pp. 235–248. ISBN: 978-3-540-87697-7, 978-3-540-87698-4. URL: http://link.springer.com/chapter/10.1007/978-3-540-87698-4_21 (visited on 19th Dec. 2016).
[172] T. Hoppe, S. Kiltz and J. Dittmann. 'Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats'. In: Computer Safety, Reliability, and Security. Ed. by B. Buth, G. Rabe and T. Seyfarth. Lecture Notes in Computer Science 5775. Springer Berlin Heidelberg, Jan. 2009, pp. 145–158. ISBN: 978-3-642-04467-0, 978-3-642-04468-7. URL: http://link.springer.com/chapter/10.1007/978-3-642-04468-7_13 (visited on 19th Dec. 2016).
[173] D. K. Nilsson, U. E. Larson, F. Picasso and E. Jonsson. 'A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay'. In: Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems (CISIS'08). Ed. by E. Corchado, R. Zunino, P. Gastaldo and Á. Herrero. Vol. 53. Advances in Intelligent and Soft Computing. 10.1007/978-3-540-88181-0_11. Springer Berlin / Heidelberg, 2009, pp. 84–91. URL: http://dx.doi.org/10.1007/978-3-540-88181-0_11.
[174] M. L. Chávez, C. H. Rosete and F. R. Henríguez. 'Achieving Confidentiality Security Service for CAN'. In: Proceedings of the 15th International Conference on Electronics, Communications and Computers, 2005. CONIELECOMP 2005. Feb. 2005, pp. 166–170. DOI: 10.1109/CONIEL.2005.13.
[175] D. K. Nilsson, U. E. Larson and E. Jonsson. 'Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes'. In: Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th. 2008, pp. 1–5. DOI: 10.1109/VETECF.2008.259.
[176] A. Groll and C. Ruland. 'Secure and Authentic Communication on Existing In-Vehicle Networks'. In: 2009 IEEE Intelligent Vehicles Symposium. 2009, pp. 1093–1097. DOI: 10.1109/IVS.2009.5164434.
[177] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka and H. Imai. 'New Attestation-Based Security Architecture for In-Vehicle Communication'. In: IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. New Orleans, Louisiana, 2008, pp. 1–6. DOI: 10.1109/GLOCOM.2008.ECP.369.
[178] C. Szilagyi and P. Koopman. 'A Flexible Approach to Embedded Network Multicast Authentication'. In: 2nd Workshop on Embedded Systems Security (WESS). 2008.
[179] S. Schulze, M. Pukall, G. Saake, T. Hoppe and J. Dittmann. 'On the Need of Data Management in Automotive Systems'. In: 13. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS). Vol. 144. Gesellschaft für Informatik (GI). Münster, Germany, Mar. 2009.
[180] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille and D. Scheuermann. 'Car2X Communication: Securing the Last Meter - A Cost-Effective Approach for Ensuring Trust in Car2X Applications Using In-Vehicle Symmetric Cryptography'. In: 2011 IEEE Vehicular Technology Conference (VTC Fall). 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093081.

[181] U. E. Larson and D. K. Nilsson. ‘Securing Vehicles against Cyber Attacks’. In: CSIIRW ’08: Proceedings of the 4th annual workshop on Cyber security and information intelligence research. CSIIRW ’08. Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead. New York, NY, USA: ACM, 2008, 30:1–30:3. ISBN: 978-1-60558-098-2. DOI: 10.1145/1413140.1413174. URL: http://doi.acm.org/10.1145/1413140.1413174.

[182] T. Hoppe, S. Kiltz and J. Dittmann. ‘Adaptive Dynamic Reaction to Automotive IT Security Incidents Using Multimedia Car Environment’. In: Proceedings of the 4th International Conference on Information Assurance and Security (ISIAS ’08). Sept. 2008, pp. 295–298. DOI: 10.1109/IAS.2008.45.

[183] T. Hoppe, S. Kiltz and J. Dittmann. ‘Applying Intrusion Detection to Automotive IT — Early Insights and Remaining Challenges’. In: Journal of Information Assurance and Security 4.3 (2009), pp. 226–235. URL: http://www.mirlabs.org/jias/hoppe.pdf (cit. on pp. 57, 61, 63).

[184] M. Müter, A. Groll and F. C. Freiling. ‘A Structured Approach to Anomaly Detection for In-Vehicle Networks’. In: 2010 Sixth International Conference on Information Assurance and Security (IAS). Atlanta, GA, Aug. 2010, pp. 92–98. DOI: 10.1109/ISIAS.2010.5604050.

[185] M. Müter and N. Asaj. ‘Entropy-Based Anomaly Detection for In-Vehicle Networks’. In: 2011 IEEE Intelligent Vehicles Symposium (IV). Baden-Baden, Germany, June 2011, pp. 1110–1115. DOI:10.1109/IVS.2011.5940552.

[186] R. Brooks, S. Sander, J. Deng and J. Taiber. ‘Automobile Security Concerns’. In: Vehicular Technology Magazine, IEEE 4.2 (June 2009), pp. 52–64. ISSN: 1556-6072. DOI: 10.1109/MVT.2009.932539.

[187] G. Graham and B. Cohen. Microsoft Security Development Lifecycle Adoption. Microsoft, Edison Group. Sept. 2013.

[188] ISO/IEC 27034: Information technology — Security techniques — Application security — Part 1: Overview and concepts. Standard ISO/IEC 27034-1:2011. International Organization for Standardization.

[189] OWASP (Open Web Application Security Project). Open Software Assurance Maturity Model (SAMM), Version 1.0. Mar. 2009. URL: http://www.opensamm.org/downloads/ SAMM-1.0.pdf (visited on 2nd Dec. 2019).

[190] G. McGraw, S. Migues and J. West. BSIMM-V (Building Security In Maturity Model, Version 5.0). Oct. 2013. [Online]. http://www.bsimm.comm/download/dl.php. [Accessed Nov 22, 2019].

[191] OWASP (Open Web Application Security Project). Comprehensive, Lightweight Application Security Process (CLASP). [Online]. http://www.owasp.org. [Accessed Oct 28, 2019].

[192] Cisco Systems, Inc. Cisco Secure Development Lifecycle (CSDL). URL: [Online]. http://www.cisco.com/web/about/security/cspo/csdl/index.html. [Accessed Oct 28, 2019].

[193] Foundstone (McAfee). Secure Software Development Lifecycle (SSDLC). 2008. [Online]. http://www.mcafee.com/us/resources/data- sheets/foundstone/ds- secure- software-devlife-cycle.pdf. [Accessed Nov 22, 2019]

[194] P. Hellström. Master’s Thesis: Tools for Static Code Analysis: A Survey. 2009. DOI: LIU-IDA/LITHEX-A--09/003--SE.

[195] B. De Win, R. Scandariato, K. Buyens, J. Grégoire and W. Joosen. 'On the secure software development process: CLASP, SDL and Touchpoints compared'. In: Information and software technology 51.7 (2009), pp. 1152–1171.
[196] The STRIDE Threat Model. https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. Accessed: 2016-11-25 (cit. on p. 74).
[197] G. Macher, E. Armengaud, E. Brenner and C. Kreiner. 'A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context'. In: International Conference on Computer Safety, Reliability, and Security. Springer. 2016, pp. 130–141.
[198] H. Bidgoli. Handbook of Information Security, Threats, Vulnerabilities, Prevention, Detection, and Management. Handbook of Information Security. Wiley, 2006. ISBN 9780470051214.
[199] C. Cummins, P. Petoumenos, A. Murray, and H. Leather. Compiler fuzzing through deep learning. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pages 95–105. ACM, 2018.
[200] M. Dowd, J. McDonald, and J. Schuh. The art of software security assessment: Identifying and preventing software vulnerabilities. Pearson Education, 2006.
[201] Lea, Perry. Internet of Thing for Architects, Packt Publishing, 2018
[202] Baker, Jean; Smart Board in the Music Classroom, Music Educators Journal, 2007, Volume 93, Issue 5, pp. 18-19
[203] Smith, David; Smart Clothes and Wearable Technology, AI&Society: Journal of Knowledge, Culture and Communication, 2007, Volume 22, Issue 1, pp. 1-3
[204] Fraser, Neil; Ten things we've learned from Blockly, IEEE Blocks and Beyond Workshop, 2015
[205] Richardson, Matt; Wallace, Shawn. Getting Started with Raspberry Pi, Maker Media, Sebastopol, California, 2013
[206] Raspberrypi, "Raspberrypi Education," 2019 [Online]. https://www.raspberrypi.org/education/. [Accessed Feb 7, 2019].
[207] SeeedStudio "Grove System," 2019. [Online]. http://wiki.seeedstudio.com/Grove_System/. [Accessed Feb 7, 2019].
[208] Kasser, Joseph; Tran, Xuan-Linh; Matisons, Simon; Prototype Educational Tools for Systems and Software (PETS) Engineering, Engineering Education for a Sustainable Future: Proceedings of the 14th Annual Conference for Australasian Association for Engineering Education and 9th Australasian Women in Engineering Forum, 2003, pp. 532-543
[209] Crawley, Edward; Malmqvist, Johan; Ostlund, Soren; Brodeur, Doris; Edstrom, Kristina. Rethinking Engineering Education, Springer International Publishing, Switzertland, 2014
[210] Heydt, Gerald; Vittal, Bijay; Feeding our profession [power engineering education], IEEE Power and Energy Magazine, 2003, Volume 99, Issue 1, pp. 38-45
[211] Thomas Rosenstatter, A State-of-the-Art Report on Vehicular Security, April 2017.
[212] ARM Inc., "Interaction of Normal and Secure worlds," 2019. [Online]. https://developer.arm.com/documentation/100935/0100/Interaction-of-Normal-and-Secure-worlds-?lang=en. [Accessed Feb 17, 2020].
[213] G. Grebenstein, "A Method for Hand Kinematic Designers", ICABB, Venice, Italy, 2010.
[214] M.C. Carrozze, F. Vecchi, F. Sebastiani, G. Cappiello,S. Roccella, M. Zecca, R. Lazzarini and P Dario, "Experimental Analysis of an Innovative Prosthetic Hand with Proprioceptive Sensors," Proc. IEEEIntl. Conf. Rob. Aut., Taipei, Taiwan, pp. 2230-2235, 2003.

[215] A. Wege and G. Hommel, “Development and control of a hand exoskeleton for rehabilitation of hand injuries," IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, 2005.
[216] Beigi, Homayoon, “Fundamentals of Speaker Recognition”, New York: Springer, 2011
[217] S. E. Levinson, L. R. Rabiner and M. M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition", in Bell Syst. Tech. Jnl. v62(4), pp1035--1074, April 1983
[218] A. Radovici, I. Culic, “Open cloud platform for programming embedded systems”, Networking in Education and Research, RoEduNet,vol. 12, pp. 1-2, 2013.
[219] A. Radovici, I. Culic, I. Bocicor, A. Armean, M. Ene, “ Platformă educațională care permite programarea dispozitivelor embedded din Cloud – Wyliodrin”, Conferinţa Naţională de Învăţământ Virtual, 9th edition, vol. 9, pp. 57-61, 2013.
[220] Peter Saint-Andre, Kevin Smith, and Remko Troncon. XMPP: The Definitive Guide. 2009.