Article

A Connected and Autonomous Vehicle Reference Architecture for Attack Surface Analysis

Carsten Maple *, Matthew Bradbury, Anh Tuan Le and Kevin Ghirardello

WMG, University of Warwick, Coventry, UK
* Corresponding Author: cm@warwick.ac.uk

Academic Editor: name
Version 14th November 2019 submitted to Appl. Sci.

Featured Application: The reference architecture presented is to be instantiated with different components which is then used to analyse the attack surface of those components.

Abstract: Connected autonomous vehicles (CAVs) will be deployed over the next decade with autonomous functionalities supported by new sensing and communication capabilities. Such functionality exposes CAVs to new attacks that current vehicles will not face. To ensure the safety and security of CAVs it is important to be able to identify the ways in which the system could be attacked and to build defences against these attacks. One possible approach is to use reference architectures to perform an attack surface analysis. Existing research has developed a variety of reference architectures, but none for the specific purpose of attack surface analysis. Existing approaches are either too simple for a sufficiently detailed modelling or require too many details to be specified to easily analyse a CAV's attack surface. Therefore, we propose a reference architecture using a hybrid Functional-Communication viewpoint for attack surface analysis of CAVs, including the Devices, Edge, and Cloud systems CAVs interact with. Using two case studies, we demonstrate how attack trees can be used to understand the attack surface of CAV systems.

Keywords: Connected Autonomous Vehicles; Reference Architecture; Attack Surface Analysis

---

1. Introduction

In recent years interest in deploying connected autonomous vehicles (CAVs) on real road networks has been increasing [1]. In order to enable the applications that depend on connectivity [2] and autonomy [3], vehicle computer systems are becoming more complex and there are becoming a greater number of ways in which the vehicles can communicate with other devices, each other, nearby Edge infrastructure, and the Cloud. Such changes in complexity [4], connectivity, and levels of autonomy means that there are more ways in which a CAV can be attacked [5] and a successful breach carries greater impact.

Due to the safety ramifications, it is important to protect the security of vehicles and the systems they rely on. Security breaches could lead to vehicle theft, privacy leakage, or in the worst case lead to injury or death of occupants. Analysing these security threats in isolation is insufficient since vulnerabilities could be, and often are, exploited in combination to lead to escalated threats with the potential for greater harm.

Reference architectures can be used to help understand and analyse complex systems, specifying the entire system and any interactions. In addition to being a useful tool for analysis, a reference architecture can be used to assist in performing attack surface analysis, for example, as part of the system level analysis and design in SAE J3061 [6, Figure 7]. By using output from a threat modelling, the identified goals, resources, capabilities, motivations and presence of an attacker can be used with a reference architecture to help understand *how* an attack could be executed. However, a problem with using existing reference architectures for attack surface identification and analysis is that they are often either lack important details [7] in order to derive certain categories of attacks, or too complex [8] for vehicle manufacturers and CAV system designers to feasibly use (which will be elaborated on in Subsection 2.1 and Subsection 2.2). This paper addresses these issues by proposing a hybrid Functional-Communication viewpoint reference architecture for attack surface analysis. This reference architecture aims to balance the complexity-completeness trade-off, such that the model is sufficient complex to model a wide range of interactions but remains easy enough to practically use.

While many of the attacks against traditional vehicles could be modelled using this reference architecture, we target L3–L5 autonomous vehicles (which are described in Table 1). These are the new and emerging autonomous vehicles that are beginning to be deployed, and which will encounter new threats compared to L0–L2 vehicles [9]. These new threats may try to manipulate input sensor data [10] in order to affect how and where an autonomous vehicle drives, or may simply try to remotely take control of the vehicle's functions [11]. There is the potential for these attacks (and others [12]) to have a large impact due to the potential of leading to unsafe conditions for vehicle occupants and pedestrians [13]. As the way in which vehicles are designed and operated is changing at a rapid pace, this reference architecture aims to focus on the next 10 years [14] of autonomous vehicles and be flexible to facilitate future changes.

To demonstrate the effectiveness of using this reference architecture to perform attack surface analysis, we instantiate it with two different case studies. Using the interactions of components in the reference architecture and goals identified from a threat modelling, attack surfaces are derived. Performing the threat modelling to identify attacker goals, motivations, capabilities and resources is out of the scope of this paper as the attack surface defines how these goals can be reached, but does not aim to specify what these goals are. There exist many threat modelling approaches [6,15–18] that can be used as input to the reference architecture. In the first example of valet parking, the attacks against a vehicle parking itself in an autonomous car park are investigated. In the second example, a real world attack against Tesla vehicles is used to highlight the need to consider the Edge infrastructure in the security of CAVs.

We make the following contributions in this paper:

1. A reference architecture made up of 4 sub-architectures: CAVs, Devices & Peripherals, the Edge, and the Cloud formed of a hybrid Functional and Communication viewpoint.
2. A methodology to use the reference architecture to synthesis the attack surface in the form of attack trees.
3. Two case studies to demonstrate the applications of attack surface and attack tree analysis in deepening the security knowledge of the system.

The remainder of this paper is organised as follows. In Section 2 we present relevant related work, including examining existing vehicular reference architectures. Section 3 describes our proposed reference architecture, its components and relevant attack surfaces, and in Section 4 describes the methodology for using the attack surface; including using attack trees as a method to perform the analysis. In Section 5 two case studies of example applications are presented as instantiations of our reference architecture. The implications of the reference architecture is discussed in Section 6; and future work is presented in Section 7, before the paper concludes with Section 8.

2. Related Work

There has been much work conducted on analysing the threats that an autonomous vehicle will face [7,11,21–23]. The issue with existing work on threat analysis is that they did not consider a comprehensive ranges of components (i.e. CAV, devices, Edge, Cloud) that form the potential CAV operational contexts. This means that threats which use a combination of attacks against different components in specific orders can be missed. Reference architectures have been developed to aid in

Table 1. Levels of Vehicular Autonomy [19] from no autonomy (where the driver is in full control of the vehicle) at level 0 to level 5 where the vehicle is in full control.

Level	Name	Description	Example
0	None	The human driver is in full control.	Anti-lock Braking System
1	Driver Assistance	The human driver is assisted by a driver assistance system of steering or acceleration/deceleration using information about the driving environment. The human performs all other tasks.	Cruise Control
2	Partial	The human driver is assisted by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment. The human performs all other tasks.	Lane Centring
3	Conditional	The autonomous vehicle controls all aspects of driving, with the expectation that the human driver will respond appropriately to a request to intervene.	Traffic Jam Chauffeur [20]
4	High	The autonomous vehicle controls all aspects of driving, even if a human driver does not respond appropriately to a request to intervene.	Driverless Valet Parking [20]
5	Full	The autonomous vehicle is in full control and no human input related to driving is expected.	Driverless Cars

the design of products and services for autonomous vehicles, but have seldom been used to provide a wider view of composite threats. Those reference architectures that do exist, can suffer from being too broad, or are insufficiently detailed, for attack surface analysis. When too broad, they require specifying less pertinent details as part of the model, which detracts from performing the attack surface analysis. When insufficiently detailed, there are threats that cannot be analysed using the reference architecture. The remainder of this section will present related work on reference architectures used to model autonomous vehicle systems.

2.1. Reference Architectures

In order to better analyse how a system is structured *reference architectures* are used as an abstract way of specifying a system. A reference architecture is an approach to model a system and provide a consistent and standardised way to describe that system. This common model should be created such that it is able to describe a broad range of scenarios that the system can be used in. Reference architectures allow modularisation of a system into components and interfaces between these components to be defined. These features can be used to assist with system development in a scalable way (e.g., by multiple organisations [24]) and also facilitate testing of the system.

This paper will develop a reference architecture specifically for assisting in the attack surface identification and analysis in CAVs. We will present our reference architecture in the next section, but provide here an overview of existing reference architectures. This related work guides our own development, and assists us in identifying the shortcomings of existing schemes that are discussed in the next section.

A common feature of reference architectures is to decompose the system they are modelling into multiple viewpoints and then specify those viewpoints in detail. There are several different viewpoints that reference architectures can present, including:

- Functional: how the components work and what their tasks are
- Communication: how the components interact
- Implementation: how the components are implemented
- Enterprise: the relation between organisations and users
- Usage: concerns of expected usage of the system

- Information: the types of information handled by the system [25]
- Physical: the physical objects in the system and their connections

Of these viewpoints, presenting the Functional, Communication, and Implementation tend to be the most common as they cover what the system does and how the system interacts with itself and other systems. When developing a reference architecture, it is important to develop only the viewpoints necessary to describe the system to prevent a user of the reference architecture needing to provide additional unnecessary information.

2.1.1. Non-CAV Reference Architectures

Before exploring the existing CAV reference architecture it is useful to examine reference architectures for different fields. In doing so they raise interesting ideas for ways in which CAV reference architectures can be improved.

A common architectural framework for the development of interoperable industrial internet systems was presented in [26]. The Industrial Internet Reference Architecture (IIRA) is divided into four viewpoints, namely Business, Usage, Functional and Implementation. While the last two viewpoints are of utmost importance in the identification of a system's threats and vulnerabilities, as they are concerned with a system's functional requirements, interdependencies, and technological implementations. The IIRA also describes a system's business objectives and expected usage, both of which go beyond the scope of a reference architecture for attack surface analysis.

A Smart Grid Reference Architecture was developed in [27] which uses Business, Functional, Information, and Communication viewpoints. Explicit considerations of *information security*, are included (i.e., confidentiality, privilege escalation), however, the methodology of how to perform a security analysis of the system is not described. The systems described are complex and include many implementation details, including the scenario a component is operating in and what actions the component is involved with. From a security analysis perspective the reference architecture could be simplified (e.g., by removing business cases) to reduce the scope for which cyber security needs to be considered. This means that while the reference architecture states that it is useful for a cyber security analysis, due to it describing aspects of a Smart Grid which do not have cyber security considerations, performing a cyber security analysis is difficult. The conclusions from this are that reference architectures for cyber security analysis, should focus on the aspects of the system for which cyber security is relevant.

2.1.2. CAV and ITS Reference Architectures

A functional reference architecture for autonomous driving was introduced in [28], which provided a foundation for considering the functionality of an autonomous vehicle irrespective of its implementation. There are close relations between functional safety and security analysis in the automotive domain. The functional safety analysis relies on information taken from hazard identification, which can be influenced by security aspects such as the communication between the components or access to assets. On the other hand, the implemented countermeasures to address functional safety can determine the security level of the system. As a result, there are certain attempts to integrate security into (functional) safety analysis in CAV, such as SAE J3061 [6]. However, there is insufficient focus on CAV interactions to support using this model for an attack surface analysis. This is because the approach focuses on the vehicle only, and does not consider interactions with RSUs, other vehicles, the Internet, and other devices.

In [7] a security-focused risk assessment was performed for autonomous driving (AD). To achieve this the authors defined a reference architecture by synthesising from multiple academic and industrial AD sources to model select AD applications. The model was instantiated for different selected applications of interest and a risk assessment of the identified threats was performed. The authors note that their work does not attempt to perform an exhaustive specification of threats, but to provide ways to specify the system to aid in deriving the threats. The reference architecture and analysis of it performed in our work is similar to this paper, however, we argue that certain details are lacking from this model which prevents a sufficiently in-depth analysis of the attack surface.

A reference architecture for ITS infrastructure that focuses on business and organisational aspects of the system was presented in [29]. While the paper does not discuss technical considerations of an ITS system, the organisational aspects highlight certain areas of interaction which are of interest from a security perspective. One issue that was highlighted was that heterogeneous systems had trouble interacting due to different implementations by different suppliers. An adaptor was required to allow these systems to interact, which would be a component of the attack surface. The reference architecture raises the importance of service collaboration, for example, parking and guidance services will need to collaborate to ensure a car is not directed to a full car park. The interactions between these services will also form part of the attack surface.

A detailed and comprehensive reference architecture for cooperative and intelligent transport was developed in [8]. There are three components to this architecture, (i) Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT), (ii) Regional Architecture Development for Intelligent Transportation (RAD-IT), and (iii) Systems Engineering Tool for Intelligent Transportation (SET-IT). RAD-IT focuses on tools for regional ITS architectures and SET-IT focuses on assisting with developing "architectures for pilots, test beds and early deployments". They key component is ARC-IT when it is used to specify a Functional viewpoint[1] and Communication viewpoint[2]. The architecture is designed to be comprehensive, which is a benefit as the architecture can be used to specify interactions in detail. However, the additional detail adds additional complexity that makes the tool harder to use. There is need for a simpler model that can be easily analysed.

The CARMA project [30], which aims to investigate the distribution of the autonomous control functions throughout an ITS defines a three tiered architecture in terms of the CARMA CORE, CARMA EDGE and VEHICLE. The CARMA CORE layer acts as in a supervisory role of the distributed vehicle control functions (such as mission planning of the end-to-end vehicle trip). The majority of mid-level controls, such as improving the calculation of reference signals for vehicle control, are implemented in the CARMA EDGE. However, some of these mid-level controls are implemented in the VEHICLE layer. The CARMA system presents a model of a complex autonomous system that introduce a number of security concerns and challenges [31]. A reference architecture could be used to achieve an understanding of the attack surface thereby allowing a more holistic threat assessment.

ITS reference architectures have also been developed for other regions, such as Holland [32], the USA [33], and Europe [34]. However, these architectures suffer from the same problem that ARC-IT does, that they are intended to be very general and cover a wide range of considerations of intelligent transport systems. This lack of focus reduces their usability to undertake an attack surface analysis.

2.2. Requirements for Attack Surface Analysis

The extant reference architectures for CAVs variously consider analysis (of attacks and of risk), viewpoints, and features (autonomous vehicles, devices, edge and cloud). The reason that these architectures have different characteristics is that they serve different purposes. When analysing an attack surface, not all of the characteristics are required, indeed some are undesirable as they may be too detailed and complex, and as such not be effective for easy identification of the surface and associated threats. To be most effective, a reference architecture needs to have the essential characteristics and no more. For example, [8] considers the widest range of viewpoints, but this can hamper the security analysis. One example of this is that the information flow of the system is described in the Physical Viewpoint using entities from the Enterprise View. These information flows are also described in the Communications viewpoint. This repetition is helpful for system design within a single viewpoint, but

1 https://local.iteris.com/arc-it/html/viewpoints/functional.html
2 https://local.iteris.com/arc-it/html/viewpoints/communications.html

Table 2. Summary of CAV Reference Architectures where the purpose, viewpoints used and the components are identified with a ✓ if included or an ✗ if not included.

Reference Architecture	Analysis		Viewpoints							Considers			
	Attack	Risk	Functional	Communication	Implementation	Enterprise	Usage	Information	Physical	CAV	Devices	Edge	Cloud
Behere and Törngren [28]	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
Osório *et al.* [29]	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Dominic *et al.* [7]	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗
The Architecture Team [8]	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓
Passchier and van Sambeek [32]	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓
Heise [33]	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗
Begoña *et al.* [34]	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✓	✓
This Work	✓	✗	✓	✓	~	✗	✗	✗	✗	✓	✓	✓	✓

not security analysis across multiple viewpoints; a more focused reference architecture can simplify the process of performing a cyber security attack surface analysis.

The minimal viewpoints required for a cyber security attack surface analysis are Functional and Communications, as is it necessary to know what the system does and how it interacts. This allows what actions an adversary can perform and how an adversary's interactions with the system can produce the attack. Other components are necessary for other systems, for example, the Physical viewpoint is required to investigate cyber-physical attacks. Other viewpoints, such as the Implementation viewpoint are important to analyse attacks against specific systems. But to perform a more general attack surface analysis, the Functional viewpoint is sufficient. Other viewpoints (e.g., Enterprise and Usage) are useful in considering different types of security such as security management. Therefore, the Functional and Communications viewpoint can be focused on when performing a cyber security attack surface analysis.

A comparison of the existing and proposed reference architectures is provided in Table 2. Features that the reference architecture includes is indicated with a ✓, and features that are not included are indicated with a ✗, the following features are shown: (i) purpose of the reference architecture (Analysis), (ii) the viewpoints used (Viewpoints), and (iii) the areas the reference architectures consider (Considers). Our work partially considers the Implementation viewpoint as it can be implemented as a *virtual* component and is thus marked with a ~. Some of the existing reference architectures fail to focus on the wide range of interactions that a CAV could be involved with. Most reference architectures include Edge devices such as RSUs, but do not considering the wider range of interactions between CAVs, Devices & Peripherals, the Edge and the Cloud. Without considering all of these interactions, it will be impossible to analyse many current and emerging attacks, so a new reference architecture needs to specify these interactions.

2.3. Summary

There are many threats that have been identified for CAVs and there have been several reference architectures developed to analyse the attack surface of CAVs. However, the reference architectures tend to either be too broad and consider aspects of an ITS that do not need to be specified when considering the attack surface of CAVs, or lack sufficient detail to analyse certain types of threats. In the next section we will present a reference architecture formed of a hybrid functional-communication viewpoint to address the lack of reference architectures that balance ease of use with being sufficiently detailed.

3. A CAV Reference Architecture: Components and Related Attack Surfaces

The reference architecture presented in this work uses the Functional and Communication viewpoints combined into a single hybrid viewpoint. These are the minimal two viewpoints needed, as a threat agent would need to know what the CAV does and how the CAV can be interacted with to attack it. However, the Implementation is also an important viewpoint (as will be shown in Section 5), because a threat actor can take advantage of vulnerabilities in the implementation of a component. To resolve this in our reference architecture, the implementation can be considered as part of a functional component, or as a *virtual* functional component that exists and interacts with all components. Important virtual components that might exist include the Operating System and the hardware that the software is executing on (e.g., Electronic Control Units (ECUs)). The users of the system are considered when identifying the scenarios of interest in which the reference architecture will be instantiated with concrete components. Finally, how users and organisations interact may lead to security issues (e.g., resetting a password), but as these threats do not specifically relate to CAVs they are out of the scope of this paper.

The four sub-architectures that are presented were designed by identifying key components within CAVs and the ways it which they will interact. The sub-architectures for CAVs and Devices & Peripherals are presented in Figure 1. The two sub-architectures for the Edge and the Cloud are shown in Figures 2 and 3 respectively. These architectures are composed of various abstract components which need to be instantiated with concrete implementations to undertake an analysis of the architecture. For example, the *Sensors* component could be instantiated with GPS, LIDAR, tire pressure, and temperature sensors. These components should be instantiated with the desired concrete implementations that are required for a specific application. When analysing different applications, the reference architecture will be instantiated with a different set of components.

3.1. CAV Reference Architecture

The first of three reference sub-architectures is shown in Figure 1, and it specifies the abstract components for CAVs and the devices & peripherals that interact with the CAV. Certain components are not included in the diagram as they are implementation details. For example, how the components interact (internal communications, usually via the Controller Area Network (CAN)), how the components are implemented (usually as an ECU), or what operating system is used. These components are important to consider when analysing attacks, but they do not form the high level functionality of the system. For example, the telematics control unit subject to research in [22] contains multiple functional components in a single physical component. The remainder of this section will describe the components present in the architecture.

3.1.1. Wireless Communications

Vehicles are currently or expected to be equipped with multiple antennas in order to communicate over different wireless protocols. This includes antennas for (i) receiving audio over AM, FM or DAB radio, (ii) receiving and transmitting IEEE 802.11 WiFi, (iii) bidirectional V2X communications over IEEE 802.11p, and (iv) bidirectional cellular antennas (such as 4G). It may also be the case that Internet of Things (IoT) technologies such as IEEE 802.15.4 or ZigBee are included to facilitate interoperability with IoT networks. Many of the systems in the CAV will interact with the communications due to the need to coordinate with nearby vehicles, or to provide services to the vehicle's users. As communications are the primary way in which vehicles will exchange information, they

Example Attacks
• DoS V2X communications [35] • Eavesdrop • Replay • MiTM Intercept • Incorrect handling of malicious packets (e.g., DAB [36]) leading to RCE • Context information leakage (e.g. location, identity [37]) • Sybil Attacks [38] • Colluding to defeat agreement protocols [39] • Wormhole (Relay) Attack [40]

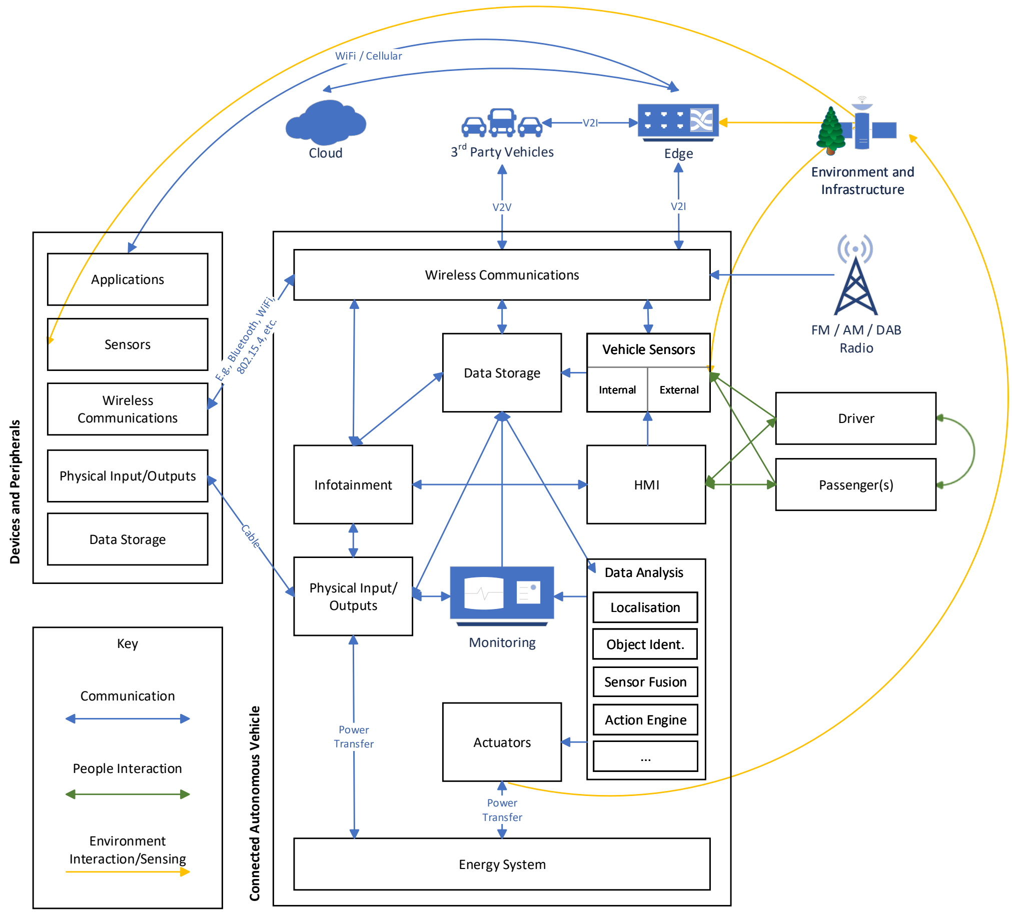


Figure 1. CAV and Devices & Peripherals Reference Architectures (Hybrid Functional-Interaction viewpoint)

will be the avenue through which most attacks are launched. These attacks may try to compromise or interfere the way in which packets are communicated, or compromise the components to which the packets are forwarded.

3.1.2. Physical Inputs and Output

Physical inputs and outputs that are contained within a vehicle include ports such as USB, OBD-II, audio connections, and others. Exploiting these ports is typically harder for an attacker as they would usually require physical access to the vehicle, however, due to the presence of additional devices that connect to these ports there are ways in which attacks can be performed via a remote connection.

Example Attacks
• Cause electrical damage [41] • Install malicious software (e.g., by firmware updates on CDs or USB sticks) [11]

With the presence of a USB port (depending on the protocol with which the hardware interprets the data), there is the possibility for an adversary to gain access to the vehicle's internal network [42]. Malicious USB sticks could be given out to people loaded with music or videos for free, with the intent of being plugged into the vehicle. When plugged in, malware could attempt to access the internal vehicle's CAN bus. Another approach is to fool users into connecting a device that resembles a USB stick but can repeatedly deliver a high voltage discharge that would destroy a vehicle's internal electronics [41].

3.1.3. Internal Communications (Virtual)

Example Attacks
• Send crafted packets [11,42–44] • DoS [43] • Eavesdrop [43]

As well as a communication system that allows a CAV to communicate with external devices (such as RSUs or 3rd party vehicles), they also have an internal communication system such as a CAN bus. This is used to connect the multiple components that form an implementation of the functions specified in the architecture. This communication network is not explicitly specified as a component, as it is implicit due to the components interacting. Vehicles may also use a different internal interaction (such as Ethernet) in the future and by under-specifying this implementation detail, the reference architecture is more generic.

Attacking this internal communication network can be performed by a direct connection to it, for example, via an OBD-II port. Alternatively, an attacker can gain access to this internal network via vulnerabilities in the components that connect to it. Once these components are compromised an attacker will have access to eavesdrop on messages sent [43], or the ability to inject malicious messages [42–44]. With access to the internal network of a vehicle many functional aspects of the vehicle can be controlled, including: the radio, instrument panel, the vehicle's body, engine, brakes, HVAC, and others [21]. A solution to these issues is to use encryption and authentication of messages [45], however, vehicles currently on the road act as if the CAN bus is a walled garden and do not attempt to encrypt or authenticate messages sent on the bus.

3.1.4. Sensors (Internal and External)

Example Attacks
• Induce misleading readings (Spoof, Replay, Delay) [10] • Blind, Jam [10] • Tamper (Disable, Replace)

Sensors are a key component of CAV systems. The vehicles will rely on their input to build a model of the world. Example sensors include: (i) Global Navigation Satellite System (GNSS) to be aware of a vehicle's position, (ii) wheel rotation senors to be aware of velocity, (iii) LIDAR to be aware of the relative position of other vehicles, (iv) parking cameras to assist drivers, and a variety of other sensors such as temperature, humidity and light. Sensors may also observe information *passively* about the occupants in the vehicle. As sensors are a way for the vehicle to obtain the state of the environment around it, if that data can be maliciously manipulated, then the vehicle may make incorrect decisions based on the manipulated data. Alternately, an adversary may attempt to eliminate the vehicle being able to use certain sensors, such as by jamming GNSS signals or producing too much LIDAR interference for the data to be useful [10]. Another approach may be for an adversary to place additional sensors on the vehicle exterior, or to subject the sensors to physical manipulation.

In certain systems, the vehicle's sensors may wirelessly communicate their data to the car (such as when monitoring tire pressure [46]). Most sensors are expected to be hardwired to the system due to high reliability requirements. Wireless sensors pose a greater security threat as there is a larger attack surface for an adversary to take advantage of. For example, the Tire Pressure Monitoring System (TPMS) leaks identity information about the vehicle by including unencrypted identifiers in the packets it sends. Due to the lack of authentication and validation, the system also is vulnerable to spoofing and replay attacks, where the vehicle could easily be fooled into believing the tire is flat even if it was not.

3.1.5. Data Storage

Example Attacks
• Violate Integrity (manipulate data) • Violate Confidentiality (extract data) • Violate Availability (delete data) • Violate Non-repudiation (delete logs) • Remote firmware update [22]

Vehicles will need to store data, including (i) the firmware and software used to run the car, (ii) maps and navigation information, (iii) music and videos for the entertainment system, and other information necessary for different use cases. This data will not be stored in a central location on the vehicle and will be stored in multiple locations. Data storage should also be segregated based on the purpose for the data. For example, music and video should not be stored in the same location as the vehicle's software, but implementation details may mean that this is not the case. Not all data will be stored locally, some will be present in the Cloud and only requested when required. Other data may be stored in the Edge, or even in other vehicles on the road.

3.1.6. Data Analysis

To make sense of the data obtained from external sources (such as the sensors) and the data stored locally in the vehicle, some sort of analysis will need to be performed on it. This analysis may use simple conditions to trigger actuators (e.g., if temperate rises above a threshold, then turn on the air conditioning), but more complicated techniques such as machine learning models will also be used. These machine learning models will be prevalent in CAVs due to the need for autonomy.

Example Attacks
• Induce bad analysis (e.g., adversarial ML [47]) • Obtain analysis • Malicious input to put analysis into infinite loop (DoS)

Localisation

One of the key pieces of knowledge for an autonomous vehicle is its location. Information such as from GNSS can be used to provide a fairly accurate location [48] as long as the vehicle is in an open area with few buildings blocking satellite signals. Other approaches such as dead reckoning are used to calculate the vehicle's current position based on a previously known position, the vehicle's speed, heading, and the travel time.

Object Identification

As part of autonomous driving it will be necessary for the CAV to be able to identify objects. These objects will include people, obstacles, road signs, and many other objects. Machine learning based algorithms will be used to perform visual identification. However, using machine learning can open the vehicle up to being attacked in new ways. One example is adversarial machine learning, where input manipulation can lead to unexpected results. For example, in [49] 3D printed objects were crafted to be misclassified by an object detection model. In one case a turtle was detected as a gun, such a detection could lead to unexpected behaviour in the vehicle. Alternate issues might include the vehicle failing to recognise another vehicle, such as when a Tesla was involved with a fatal accident when it attempted to drive under a truck [50]. An adversary manipulating the data provided to sensors, may affect the actions vehicles take.

Sensor Fusion

To improve accuracy of sensor input the data provided from sensors is usually fused, such as via a Kalman Filter [51]. By doing so the quality of the fused data should be higher than the individually sensed data. However, if manipulated sensor data is used then the fusion approach could produce less accurate or even inaccurate results [52]. In [53] spoofing sensor data was used to control a UAV, with the technique possibly extendable to other autonomous vehicles. Therefore, the sensor fusion technique needs to be aware of how to handle data provided by an attacker, such that it does not lead to incorrect actuations.

Action Engine

Once an autonomous vehicle has both determined its location and the road objects surrounding it, it may call on the Action Engine sub-module to decide what it must do next. Possible actions to be taken include interactions with other connected vehicles on the road and both short and long term driving decisions. RSUs or the Cloud, on the other hand, make use of the Action Engine to ensure that the vehicle control and planning systems are correct and safe, and to ensure that multiple vehicles on the road at the same time coordinate and are managed to move people and packages to their destinations in the most effective way.

3.1.7. Energy System

Example Attacks
• Overcharge battery to damage it • Drain power

The energy system both supplies energy (in the form of electricity) to the systems within a CAV and is also capable of being supplied with energy. Energy can be supplied back to the batteries through the use of regenerative breaking, solar panels, recharging cables, and other sources. The energy system is also tasked with maintaining the vehicle's batteries to ensure power is safely drawn from them. If the energy system is compromised then unsafe usages of electricity may follow which could lead to damage to the vehicle.

3.1.8. Actuation

Example Attacks
• Disable

This module contains any components that can perform an action with an impact on the physical world. This may include, applying the brakes, changing wheel speed, changing the angle the wheel is pointed in, operating the air conditioning, lowering or raising windows, locking and unlocking car doors, and others. If an adversary is not attempting to gain information about the vehicle or passengers, then actuating components are likely to be the key target. For example, an attacker may attempt to compromise a large number of vehicles in order to provide Theft as a Service (TaaS) [54]. Rather than steal cars, the thief will install malware on as many vehicles as possible. Then, when there is demand for a particular car the malware can give the thief access to the vehicle. The adversary who installed the malware may not even need to active the malware themselves, as they could provide a crafted key to the intended buyer.

3.1.9. Monitoring and Logging

Example Attacks
• No longer forensically valid • Extract data

Monitoring and logging are important aspects for CAVs in a number of scenarios, including: verifying that vehicles are functioning correctly, analysing past decisions made, and will be used to manage maintenance schedules. For example, if a CAV is in an collision the vehicle will need the ability to explain why it made the decisions before the collision. If an adversary is capable of accessing the diagnostics unit then it may rewrite decision making history, preventing reliable auditing.

3.1.10. Infotainment

Example Attacks
• Arbitrary code execution (via browser) [43] • Arbitrary code execution (via crafted audio/video files)

The infotainment system is used to manage the entertainment system within a vehicle (such as audio/video systems) and information systems (such as maps and navigation, phone, and car status). Infotainment systems are also likely to contain a web browser to facilitate access to the internet for both entertainment and information. An issue with infotainment systems is that they may process data from untrusted sources. If the data is maliciously crafted to take advantage of vulnerabilities in the system, then an attacker may be able to remotely execute arbitrary code.

3.1.11. Human-Machine Interface

Example Attacks
• Spoofing vehicle status • Intercept commands

A Human-Machine Interface (HMI) is any device or software which allows a person to *actively* interact with a machine. A passive observation of the occupants would be performed by the Sensors component. In vehicles HMI includes critical systems such as the steering wheel, accelerator pedal, break pedal, and gear controls. Less critical system include the controls on the dashboard and feedback mechanisms. An attacker may attempt to intercept the signals from the HMI to prevent the vehicle doing something other than requested. Alternatively, the attacker might use the HMIs to report statuses that are incorrect to attempt to get the driver or passengers to perform certain actions. For example, the adversary may turn on engine safety warnings (when there is no problem) to cause the driver to stop the car. The attacker could then use this opportunity to steal the vehicle, or perform other attacks, such as attaching a tracking sensor.

Note that HMI does not communicate directly with the actuators. There will need to be some data analysis performed that potentially adjusts the action performed. For example, an anti-lock breaking system would not always actuate the brakes in the way the driver requests.

3.2. Devices and Peripherals Reference Architecture

Vehicles may have a number of peripherals that interact with each other. Some examples of the kinds of devices and peripherals that may be present and in use are: (i) Car Keys, (ii) Smart phones, (iii) MP3 players, (iv) Bluetooth devices, (v) 3rd Party Navigation Systems, (vi) Dashcams, (vii) Portable games consoles, and others. These devices could either interact with the vehicle or simply be present within the vehicle. Some of these interactions may be relatively simple, such as accessing the vehicle's WiFi in order to connect to the internet via a cellular connection. Others may involve accessing the vehicle's storage, actuating the infotainment system, or controlling other aspects of the vehicle. These peripherals are additional vectors that attackers can take advantage of to attack the system. This may be by loading the device with malware to gain control [55], or interacting with the context of the inter-device communication [56].

It is also the case that some of these interactions may be unintended. For example, a passenger leaving their phone in an automated taxi may leak the journey history of the taxi if it is running a phone tracking service. This sort of leak could also be caused by an attacker intentionally attaching such a device to the vehicle.

Example Attacks
• Location tracking via sensor data (e.g., magnetometer [57]) • Data harvesting • Become internal attack vector for remote adversary • Malicious smartphone app interfering with CAN bus [55]

3.2.1. Applications

One of the key features of certain devices (such as smartphone) are the ability to run applications on it. Some vehicle manufacturers (such as Volkswagen [58]) are creating mobile apps that obtain information from the car or allow the app to control certain features (such as the infotainment system). If the phone is compromised then that malware may be able to affect the vehicle's systems via the app. The attacker may be able to leak data about the car, gain an internal vector to the vehicle's systems, or use the phone's connection to the cloud to attack the vehicle.

3.2.2. Sensors

Example Attacks
• Blind, Jam • Induce misleading readings (Spoof, Replay, Delay)

The devices within a vehicle may have their own sensors that reveal information about the state of the environment inside the vehicle, or about the vehicle itself. An adversary may wish to take advantage of these sensors to gain knowledge about the vehicle, which could be potentially useful in escalating the severity of other attacks.

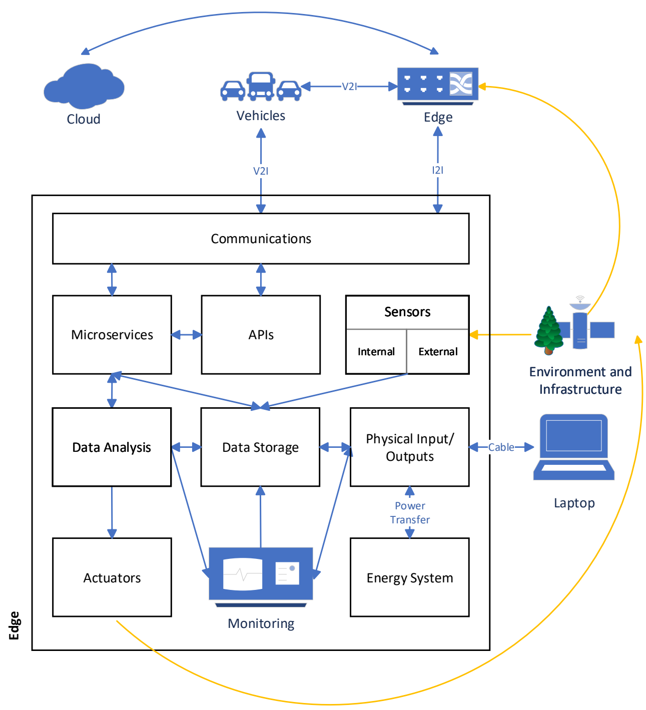


Figure 2. Edge Reference Architecture (Hybrid Functional-Interaction viewpoint)

3.2.3. Wireless Communications

The devices present in a vehicle are expected to communicate wirelessly. This may be with the cellular network, directly with the vehicle, or possibly with other devices in the vehicle. One example, is that vehicular privacy may be leaked due to the presence of devices in the vehicle. For example, WiFi devices will broadcast their MAC addresses periodically when looking for a device to connect to [59]. Bluetooth devices will also beacon their MAC address in order to find devices to connect to [56]. Both reveal identity information that could be used to track people in vehicles.

Example Attacks
• Relay Attack (Car Key Signal [40]) • Replay attack (e.g., unlock car using recorded signal) • Wireless protocols leak identity information about owner [59] • Facilitates tracking of person and vehicle [56]

3.3. Edge Reference Architecture

The Edge reference architecture specifies the interactions of components that occur between operations of the vehicle and the operations of the Cloud. This may include devices used to access a WAN (such as cellular base stations or WiFi hotspots). Edge devices must include some functionality that does not occur remotely but occurs close to where the vehicle is operating or at the boundary between the vehicle and the cloud. There is a wide range of scenarios that could be considered in the Edge reference architecture. The main example are Road-Side Units (RSUs) which are computing devices placed along road networks to support CAVs travelling along the roads. These devices will communicate with autonomous vehicles to assist their autonomous activities. Alternate pieces of infrastructure could also be considered as part of the Edge. For example, internet connected traffic lights, smart parking garages, and others, may need to interact with autonomous vehicles and actuate components to facilitate autonomous driving.

Example Attacks
• Modify hardware (Tamper) • Disable hardware

Certain components have been previously described (e.g., Sensors, Data Analysis) and will not be repeated as part of the Edge sub-architecture. Some components previously described will be repeated due to differences with the previous sub-architectures.

3.3.1. Communication

Example Attacks
• Edge Emulation [60] • DoS

Communication on the Edge has additional capabilities compared with CAVs and the Devices & Peripherals within them, as the Edge can be physically connected to a wide area network (WAN) rather than just wirelessly connected. Such physical connections might be provided by high bandwidth fibre, Ethernet and other communication approaches that require a physical medium. However, Edge nodes will still need to have wireless communication in order to facilitate V2I communication. This communication will include the technologies specified in vehicles to facilitate Dedicated Short-Range Communications (DSRC) (e.g., IEEE 802.11p and/or C-V2X). Other technologies might include non-vehicular specific cellular communications, WiFi and protocols to interact with IoT systems (e.g., IEEE 802.15.4).

3.3.2. Data Storage

Example Attacks
• Violate Integrity (manipulate data) • Violate Confidentiality (extract data) • Violate Availability (delete data)

Data storage at the Edge will typically be centralised in each device as a single piece of hardware. As the Edge is susceptible to tampering it is important to ensure precautions such as encrypting the entire disk is used to prevent a threat actor from removing, reading from, and then replacing the disk.

3.3.3. Actuators

Example Attacks
• Disable

Edge systems may potentially have the ability to actuate key pieces of infrastructure which can influence the environment (such as traffic lights, or barriers). Depending on what the actuator is, the Edge device(s) may be capable of having a large impact on the behaviour and security of vehicles. For example, a compromised Edge might claim a certain actuation state that is not true, such as claiming a traffic light is green when it is red.

3.3.4. Energy System

Example Attacks
• Disconnect power supply

The energy system being used to power the Edge system is important to consider as different kinds could be used. Typically Edge systems will be powered using mains power and the attacks on this system relate to removing access to this power. However, alternate power systems (such as via batteries and renewable energy like solar) may be used in areas where providing mains power is infeasible or too costly.

3.3.5. Physical IO

Example Attacks
• Privilege Escalation

The Edge will have Physical IO ports that allow technicians to connect directly to the Edge infrastructure. These ports should be protected using physical security mechanisms (such as locks) to protect against attacks. From a cyber security perspective the ports need to defend against attacks that occur once physical security is bypassed. This means that any user connecting via these ports should be correctly authenticated and forensic logs made about these connection attempts.

3.3.6. Monitoring and Logging

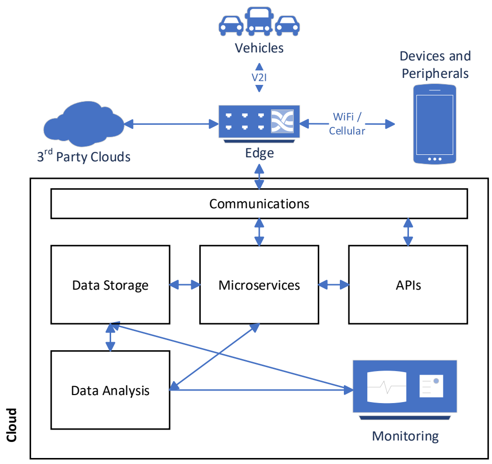


Figure 3. Cloud Reference Architecture (Hybrid Functional-Interaction viewpoint)

Example Attacks
• Delete/Modify logs

Records of actions taken by both the Edge and Cloud will need to be kept. This is to allow investigators to understand why a specific sequence of action occurred. They will also be needed to understand performance characteristics of the system.

3.3.7. Microservices

Example Attacks
• Malicious firmware deployment • Privilege Escalation

Microservices involve an application or services designed to provide functionality via a collection of loosely coupled services. These microservices each provide a single service compared to a monolithic model which provides multiple services at once. Benefits to this style of architecture include: improved scalability to a large number of users and increased resistance to certain attacks. A modular architecture is easier to test and develop, reducing the likelihood of bugs and vulnerabilities being present. Any services that are used internally do not need to be exposed to the wider internet, which reduces the attack surface compared to a monolithic application. However, while each individual microservice has a smaller attack surface, the inter-microservices communications become a possible avenues of attack.

3.3.8. Application Programming Interface (APIs)

Example Attacks
• Lack of user data validation (e.g., SQL injection) • Incorrect data disclosure

The APIs exposed by a service hosted on the Edge are used to access that service. APIs can be exposed in a number of ways, however, a common technique is to use RESTful APIs [61] that represent a request and response in JSON which is typically sent over HTTP(S). As APIs often involve user provided data, it is important to ensure that it is sanitised before being manipulated or used for a task. A lack of sanitation, or vulnerabilities in the parsing code of the request can lead to confidentiality or integrity violations. A common example of this kind of attack are SQL injections.

3.4. Cloud Reference Architecture

The interactions with CAVs and the Cloud, and the operation of the Cloud are important to consider with respect to the attack surface of autonomous vehicles. Much of the information that CAVs request will be provided by Cloud services and specific applications will require interaction with Cloud APIs for services to function. The Cloud reference architecture is intended to be a simplified representation of the key components that are important for CAVs. It is sufficiently detailed for an analysis of how attacks on the Cloud will impact a CAV, however, more detailed reference architectures and threat models should be used to analyse the Cloud in greater depth (such as [62–64]).

The remainder of this section will describe the components in the Cloud reference architecture. Certain components have been previously described in Subsection 3.3 (e.g., Monitoring and Logging, Microservices, APIs) and will not be repeated here.

3.4.1. Communication

The communication patterns that occur in the Cloud will be more complex due to the Cloud's need for scalability, high performance and high reliability. Rather than having a single connection to the wider networking infrastructure, the Cloud will have multiple gateways which utilise load balancing to improve performance. As the Cloud is internet connected, large services will be under attack from DDoS packet spam [65]. This means that firewalls and DDoS protection is an important part of the Cloud's communication infrastructure.

Example Attacks
• Jam or disconnect link • MiTM • DDoS

3.4.2. Data Storage

Cloud data storage will be different to both vehicular and RSU data storage, as it will be physically distributed across many different data centres. The data will also be replicated to ensure integrity and availability under hardware failures. This replication and distribution increases the attack surface of the data storage, as there are multiple sites to consider exploiting and the communication between sites to perform the replication could also be vulnerable to exploitation.

Example Attacks
• Insider attacks against data centre [66] • Hardware failures limiting availability • Unintended remote access

3.4.3. Data Analysis

The data analysis performed by the Cloud is going to be different from that performed by the vehicle, as the Cloud will have access to much more data over a longer time period. Therefore, the Cloud will have different objectives in terms of the analysis it produces from the data. For example, it may analyse historical data to better predict traffic patterns, which could be used to load balance road networks when a vehicle requests a route from its origin to its destination. An attacker may wish to gain this analysis (as it is likely to be very valuable) or impact the analysis so it outputs poor results (e.g., such that all vehicles are directed into a lower capacity road, leaving higher capacity roads free).

Example Attacks
• Privacy leakage of user information (Privacy Preserving Data Mining to protect it [67])

4. Methodology

In the previous section we presented the four components our reference architecture that can be used as an aid for the examination of cyber security threats and to develop appropriate strategies to address these threats. This reference architecture provides an abstracted view of the ecosystem, that allows developers of new products, services and infrastructure to see how their own contribution fits into this system of systems. To identify and mitigate attacks using the reference architecture, the users undertake three steps: instantiate the architecture with their particular use case; isolate the attack surface; and identify attack entry points in the boundary and internal interaction points. We explain each of these steps below.

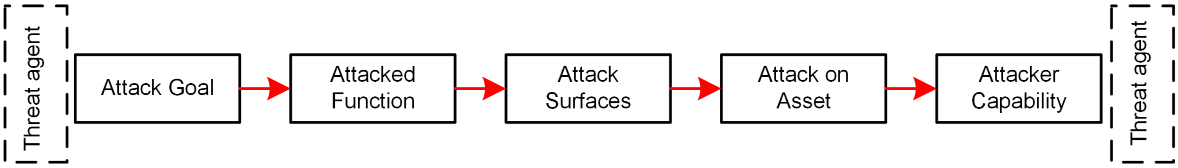


Figure 4. The process of analysing an attack goal when performing an attack surface analysis.

4.1. Instantiating the Reference Architecture

Thus far the *abstract* reference architecture has been presented, with abstract components such as Sensors. To use the reference architecture the abstract components needs to be instantiated with concrete components as required by the specific scenario of interest (as will be shown in Section 5). For example, the Sensors component could be instantiated with multiple sensors such as LIDAR, odeometry and temperature when an application needs the output from these sensors to perform its function. Not all components need to be instantiated, as the scenario may not involve certain systems within the vehicle. Only once the reference architecture has been instantiated with components the threats against those components can be identified. Using the reference architecture the threats posed by compromised components can be identified by the links specifying how the components interact.

4.2. Synthesis of Attack Surface

Once the system has been instantiated for a use case, attack surface analysis is used to identify a comprehensive set of feasible methods for adversaries to achieve their goals. Attackers can use or combine different attack paths to reach their desired goals. Where mitigations should be implemented can be identified by focusing on reducing the ability for an adversary to exploit critical attack surfaces. Attack goals can be obtained by systematically performing a threat modelling on the critical components or functionality of a system. There are a number of approaches to perform threat modelling [6], of which Microsoft's STRIDE is commonly used in the automotive security domain. A reference architecture is useful in conjunction with threat modelling, as it provides a methodology to identify the attack routes to achieve a goal that may not have been previously considered in the threat modelling. However, performing a threat modelling is out-of-scope for this work to ensure generalisation to arbitrary threat modelling techniques.

One effective method to describe attack surfaces are *attack trees*, which were first introduced in [15] to manage the large number of threats derived from comprehensive threat modelling in general security. Attack tress have since been employed in automotive security in a number of scenarios [68–70]. To create attack trees potential threat agents and their goals in compromising the system first need to be identified. For each attack goal, the relevant attack surfaces need to be specified that define possible paths to reach this goal. These paths can then be represented as an attack tree. At the end of this procedure, a list of attack trees which cover known goals, sub-goals and attack methods of potential threat agents are produced.

In this paper, we also employ attack trees to synthesise, manage, and control the attack surface. The process to perform the attack tree analysis is illustrated in Figure 4 and described below:

1. The goal(s) of the threat actor needs to be specified.
2. Using these goals, identify the component in the reference architecture that ultimately needs to be compromised for these goals to be achieved.
3. Identify the possible entry points to the system the threat actor could exploit.
4. Using the entry point(s) calculate the path(s) that an threat actor could take to reach the target component from an external interaction.
5. Considering a threat actor's capabilities, resources and presence, prune paths that the threat actor cannot exploit.

Evaluation of threat agents appear at both ends of an attack tree. At the beginning, goals are derived from threat agents' motivations. It is assumed that threat agents will only consider goals that follow from their motivations. For example, a thief has a motivation to increase their wealth, so a goal is to steal physical assets rather than cause damage. Each threat will require a specific capability to be carried out, such as: technique, skills, knowledge, equipment, presence, and others. Therefore, at the end of the procedure, the capability of threat agents also needs to be evaluated to check if achieving the goal is feasible. If achieving the goal is not feasible, then the attack tree needs to be pruned from the set of attack trees generated.

Existing work has been performed on identifying threat actors and their capabilities, goals, resources and motivations which should be used as input to this attack surface analysis. For example, a comprehensive library of threat agents for general information systems was provided by Intel [71] in their Threat Agent Risk Analysis (TARA) model. This library contains information of 22 threat agents and their 9 common attributes. However, many of the agents are inapplicable in to CAV security. For example, the TARA list was reduced to the seven most relevant agents in [7], which included: thief, owner, organised crime, mechanic, hacktivist, terrorist, and foreign government.

4.3. Identify Attack Entry Points at the Boundary and Internal Interaction Points

Attacks against a single component can have limited impact. Therefore, is it often the case that compromised components are used to aid in attacking another component, or multiple components are attacked simultaneously. These attacks are more complicated and take longer to perform, but can have a greater impact on the CAV. The motivations for an attacker to attack a component via another compromised component can be divided into two categories: (i) escalating attacker capability, and (ii) creating greater impacts. Achieving one of these categories (or both) can be obtained by sequential manipulation (attacking a component from another already compromised component), concurrent manipulation (attacking two components simultaneously), or a mixture of the two manipulations.

4.4. Summary

This section described the procedure to synthesis the attack surface of a system described using a reference architecture. To provide an insight into how to apply this technique, two case studies using it are explored in the next section.

5. Case Studies

In this section we present two different case studies to demonstrate how to use the proposed reference architecture for attack surface analysis. The procedure for creating these case studies is as follows:

1. Identify a scenario where cyber security is important.
2. Instantiate the reference architecture with concrete instances of components that are present in the scenario.
3. Use input from threat modelling to identify the goals, motivations, capabilities and resources of an adversary.
4. Use the instantiated reference architecture to build attack trees. This facilitate the cyber security analysis of the scenario by identifying how an adversary will perform attacks.
5. Finally, identify the ways in which the system can be changed to mitigate the attacks.

5.1. Driverless Valet Parking

The first case study is the driverless valet parking example from [20], where a driver wishes to leave their vehicle at a parking garage. Once the driver leaves the vehicle, they can request the vehicle to autonomously park itself by collaborating with the smart parking garage. The parking garage will allocate the vehicle a parking space and provide internal maps to aid the vehicle in locating its allocated space. When the driver wishes to retrieve the vehicle, a signal can be sent from a smart

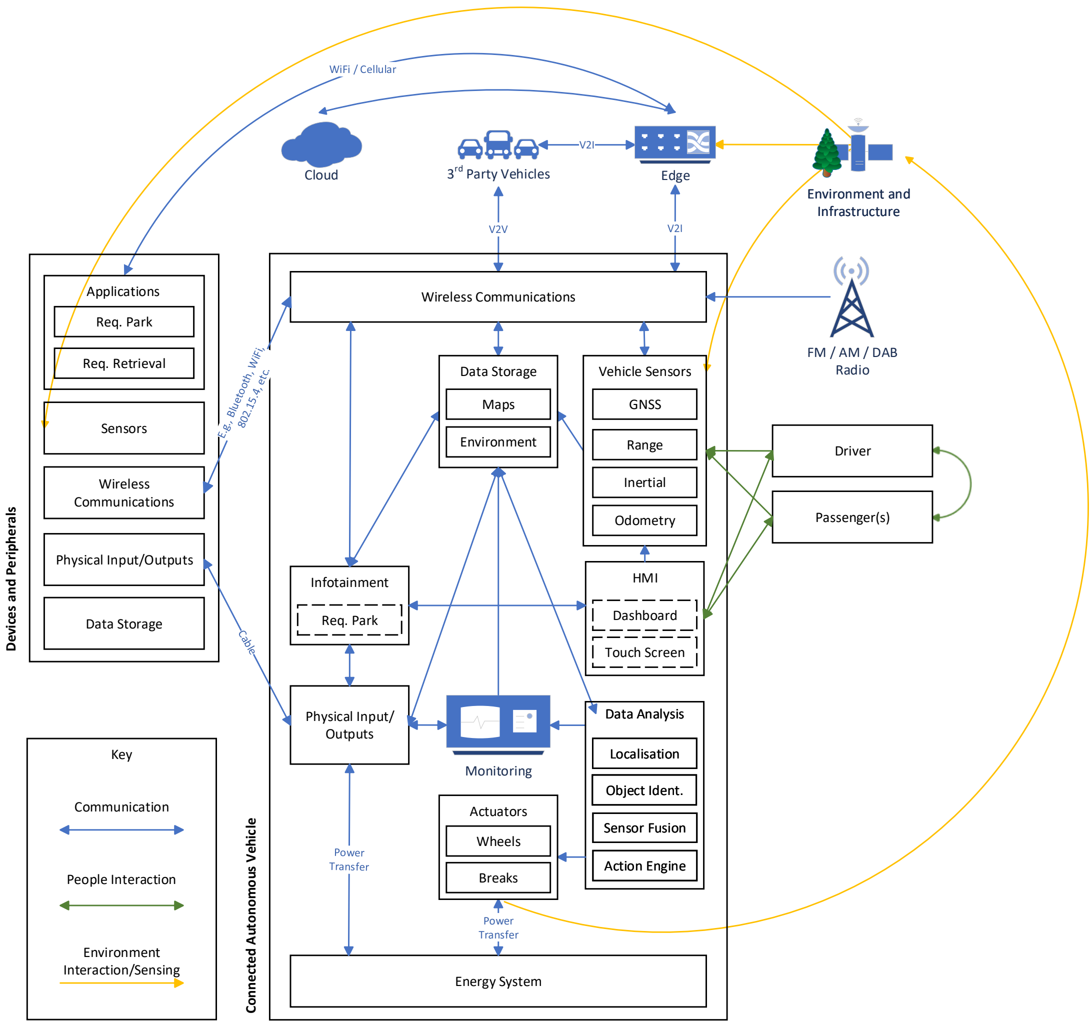


Figure 5. Valet Parking: Vehicle and Devices Instantiation

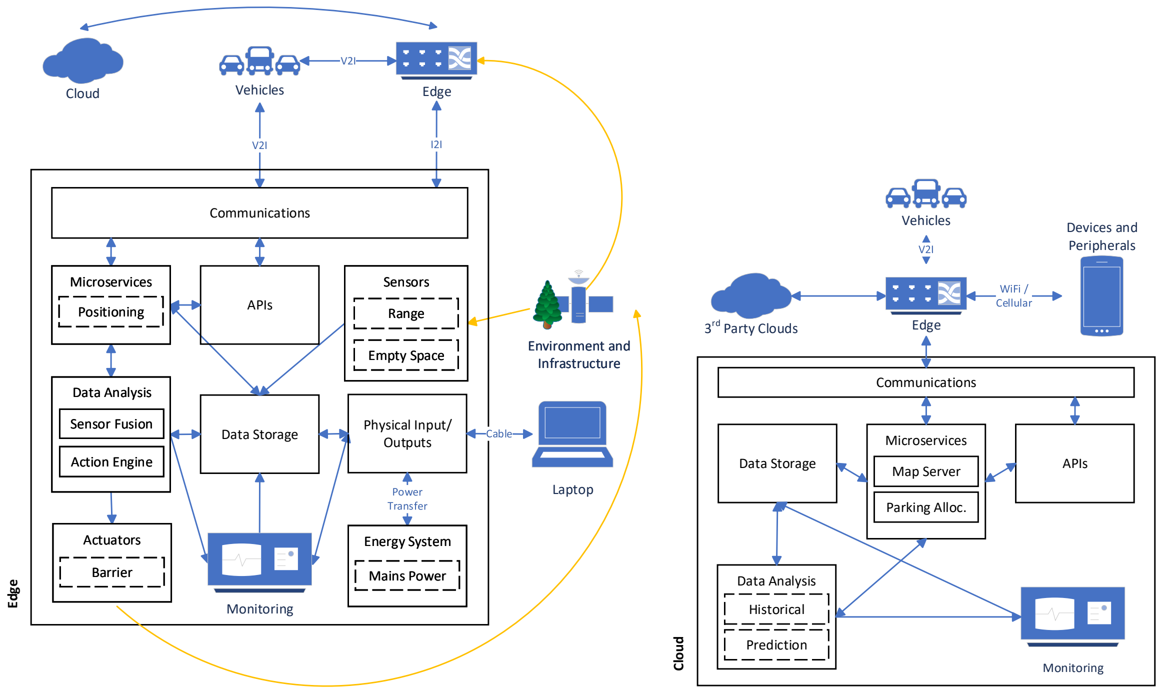
704 phone to request the vehicle autonomously drives back to its owner. Figures 5 and 6 show how certain components were instantiated with concrete components. In this example the Edge resources form the smart parking garage.

As this case study was also examined in [7], by implementing this application we will demonstrate the differences between using our reference architecture and the one proposed in [7]. This comparison will demonstrate that our architecture allows a more detailed analysis of the attack surface due to the consideration of interactions between the vehicle and the devices & peripherals, the Edge, and the Cloud. Some different components are included that are not referenced in the example in [7]. These new components are in boxes with dashed lines. They indicate certain functionality that could be involved with a valet parking system and highlight different ways in which the system could be attacked that are not covered in the previous work.

5.1.1. Threat Identification

A number of threats were identified in the original example in [7] that can also identified using the reference architecture proposed.

- **Spoof GNSS** on **Vehicle**: GNSS signals could be spoofed to assist a thief stealing a vehicle.
- **Modify Map via Update** on **Cloud**: A map update is used to force the vehicle along a route to an arbitrary destination.
- **Replay Retrieval** on **Device**: A thief replays a recorded signal used to retrieve the vehicle.



(a) Edge **(b)** Cloud

Figure 6. Valet Parking: Edge and Cloud Instantiation

- **Blind Range Sensor** on **Vehicle**: An adversary seeking to cause a crash could blind the range sensor to prevent a vehicle from knowing its distance from obstacles.
- **DoS Parking Allocator** on **Cloud**: An adversary seeking to induce a traffic jam or freeze the parking garage could DoS the allocator, preventing vehicles from requesting new spaces.

Using our reference architecture (see Figure 5 and Figure 6), the following additional threats have been identified:

- **DoS Parking Sensor** on **Edge**: Cover sensor that detects a vehicle in a parking space to reduce the availability of the parking garage.
- **Information Disclosure via Vulnerable APIs** on **Cloud**: Vulnerable APIs can potentially execute arbitrary code (such as via SQL injection attacks), allowing an adversary to remotely obtain sensitive data about the parking garage system.
- **MiTM** on **Edge**: A device could be placed in the parking garage that mimics roadside infrastructure. If it has a high signal strength vehicles would prefer connecting to that overt V2I rather than the Edge infrastructure of the parking garage, allowing a MiTM attack between the vehicle and the cloud services. This attack could reveal sensitive information about the user (such as financial details). It could also be used to over allocate vehicles causing a large traffic jam in the parking garage (denying vehicle availability).

This work does not perform the threat modelling in step 3, as it is expected to input this information from one of the many different threat modelling techniques.
Considering the additional components contained within the dashed lines, the following additional threats have been identified:

- **Cut Mains Power** on **Edge**: A vehicle should be able to autonomously exit the parking garage even if power is lost, whether the power loss is malicious or not. If the vehicle is unable to exit the parking garage then availability of the vehicle is denied to its owner.

Table 3. Attack Tree Analysis for Driverless Valet Parking Use Case with example threat actors and their goals.

TA	Goal(s)	Attacked Functions	Attack Surfaces	Detailed attacks on assets
Thief	Steal the CAV	**F1** Stop the CAV at location that is convenient to steal	Sensors that are responsible to stop the CAV in incidents; OR edge (can ask CAV to stop)	A12 or A22
		F2 Mislead the CAV to false location by falsifying the route	Cloud (giving false map); OR Edge (giving false location); OR GNSS sensor (responsible for location sensing)	A21 or A11 or A23 or A24
		F3 Control the CAV: compromise the command to make it go to false location	Edge (giving false command); OR Key (control the CAV directly	A31 or A32
Hactivist	Manipulate the CAV operation	**F1** Stop the CAV	(See Thief analysis)	(See Thief analysis)
		F2 Mislead the CAV	(See Thief analysis)	(See Thief analysis)
		F3 Control the CAV	(See Thief analysis)	(See Thief analysis)
		F4 Track the CAV	Cloud (storing location information of the CAV)	A41
Terrorist	**G1** Manipulate the CAV operation to create accident or damage	**G1-F1** Stop the CAV	(See Thief analysis)	(See Thief analysis)
		G1-F2 Mislead the CAV	(See Thief analysis)	(See Thief analysis)
		G1-F3 Control the CAV	(See Thief analysis)	(See Thief analysis)
	G2 Disrupt the station operation	**G2-F5** Stop parking management services	Cloud; Or Edge	F5: A21 or A25

- **Incorrect Indoors Positioning** on **Edge**: If the Edge assists the vehicle perform indoors positioning, then spoofed and jammed signals could be used to decrease the vehicle's certainty of its position.

By including the additional interactions with the Edge and Cloud, as well as a better structuring of components and their interactions, our reference architecture has allowed more threats to be identified. The identification process does not require specifying a large amount of details compared to more comprehensive reference architectures such as [8].

Attack trees were built from these specified potential threats by first selecting the most relevant threat agents; which are the thief, hacktivist, and terrorist.For each threat agent the most important goals were identified. The attack trees were then analysed for each of these goals, which is summarised in Table 3. Finally, each tree was combined into a single tree (Figure 7) to illustrate the security analysis of the use case with respect to the threats, threat agents and their goals.

5.1.2. Discussion

One of the interactions specified in [7] was a key/remote that is used to initiate the retrieval of a vehicle from the parking garage. This occurs by the key communicating directly with the vehicle and initiating its automated driving to exit the parking garage. An alternate architecture involves a user using an app on a smart phone to contact a cloud service to request the retrieval of a vehicle. This means the parking garage has greater control over vehicle parking allocating and scheduling vehicle retrieval.

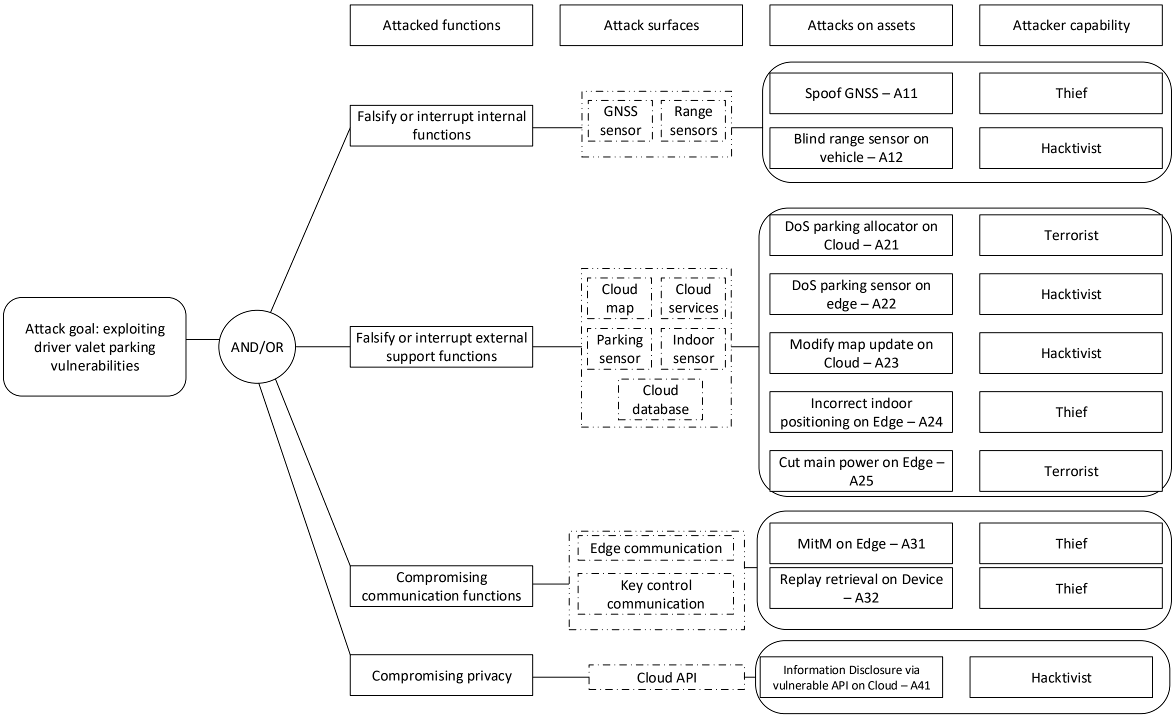


Figure 7. Attack Tree for Valet Parking Example that highlights multiple ways in which an attacker can reach its goal.

We use the alternate model as we believe it more accurately describes the way in which this system will be implemented. This means the steps of vehicle retrieval goes from (a.i) Key communicates with vehicle to request autonomous retrieval, (a.ii) vehicle autonomously drives to owner through parking garage, to (b.i) User requests vehicle retrieval using smart phone, (b.ii) cloud service schedules vehicle retrieval to prevent traffic jams in parking garage, (b.iii) vehicle autonomously drives to owner.

Our proposed reference architecture allows a more comprehensive attack surface to be identified than in previous work. For example, with the valet parking application, we can identify additional threats compared to [7], because we define the previous unspecified Devices & Peripherals and Cloud sub-architectures and more thoroughly define the Edge sub-architecture. A comprehensive attack surface is always important because a missed threat can lead to risk being underestimated, and may create severe consequences if attackers can use it to exploit the system.

5.2. Tesla Exploitation

An example of an attack that needed to compromise multiple components to gain further control of the vehicle was presented in [43]. The default behaviour of the vehicle was to connect to an unprotected WiFi hotspot and opening a website in the infotainment. By setting up an alternate WiFi hotspot with the same SSID but broadcasting at a higher power, the vehicle instead connected to the alternate hotspot, which allowed traffic to be redirected to a custom server. This means an attacker with *semi-local* presence could perform this attack. To perform the initial attack the adversary needed to identify a vulnerability in the web browser running in the infotainment system. By chaining together multiple browser vulnerabilities the adversary could execute arbitrary code through the compromised browser. Privilege escalation was then required to affect the system in substantial ways. Without gaining the privilege escalation the adversary would have little ability to affect the internal systems. However, read access to memory storage was provided through the browser exploit, which revealed debug information that included procedures for upgrading firmware. This allowed a custom firmware to be flashed to certain components, which is capable of performing arbitrary tasks.

Table 4. Attack Tree Analysis for Tesla Use Case with example threat actors and their goals.

TA	Goal(s)	Attacked Functions	Attack Surfaces	Detailed attacks on assets
Hacktivist	HG: to control the CAV components (e.g. IC, Parrot, Gateway) remotely	**HF1** get shell access	**AS-HF1:** IC, Parrot, Gateway	**A-HF1:** A43 or A42 or A41
		HF1.1: overwrite firmware	**AS-HF1.1:** Linux Kernel, Browser	**A-HF1.1:** A22
		HF1.1.1: get firmware address	**AS-HF1.1.1:** Browser	**A-HF1.1.1:** A21
		HF1.1.2: redirect browser to fake domain	**AS-HF1.1.2:** Browser, WIFI	**A-HF1.1.2:** A11 or A12 or A13
		HF2: get root privileges	**AS-HF2:** Linux Kernel	**A-HF2**: A31
		HF2.1: disable the security app	**AS-HF2.1**: Linux Kernel	**A-HF2.1**: A32
Terrorist	**TG:** to create high safety impact attack by autonomous vehicle **TG1:** to control the CAV remotely **TG2:** to monitor the CAV to find environment where it can create high safety impact (e.g. involves many people)	**TG1:** See attacked function analysis for HG	See attack surfaces for HG	See attacks for HG
		TG2: TF-TG2: to track the CAV and its operating environment	**AS-TF2**: See similar analysis in valet driving example	**TG2**: See similar analysis in valet driving example

So the summarised steps are as follows, with FUNCTIONAL components and **implementations** of those components formatted differently (see more details of reference model in Figure 8 and Figure 9):

1. EXTERNAL COMMUNICATIONS connects to popular **WiFi** hotspot
2. A malicious **WiFi** hotspot spoofs the SSID with a greater signal strength
3. Compromise **browser** in INFOTAINMENT
4. Privilege escalation in **Operating System**
5. STORAGE access
6. Flash **firmware** (change STORAGE)
7. Custom **firmware** eavesdrops/transmits/blocks messages on the **CAN bus** (INTERNAL COMMUNICATIONS)

Even with access to the CAN bus safety features limited the adversary's ability to perform certain actions. For example, the authors attempted to open the trunk while the vehicle was in motion, but this was prevented. However, the authors found a way to block certain CAN message which allowed them to open the trunk, or to disable automatic locking of doors when the vehicle was moving.

In response to this vulnerability several controls were added to reinforce the security of Tesla vehicles, including (see Figure 10):

- **C1:** Greater isolation of the Infotainment web browser from being able to interact with other parts of the system
- **C2:** Page Table Isolation (which prevents the kernel from accessing user mode memory and thus preventing the adversary from executing code in user space)
- **C3:** Code Signing (to prevent untrusted code execution)

To the best of our knowledge, this kind of attack has only been reported three times, twice by the Tencent researchers in 2016 and 2017, and once by Checkoway *et al.* [54]. For each attack, the authors reported the step-by-step hacking actions, while the producers quickly provided patches/fixes to the

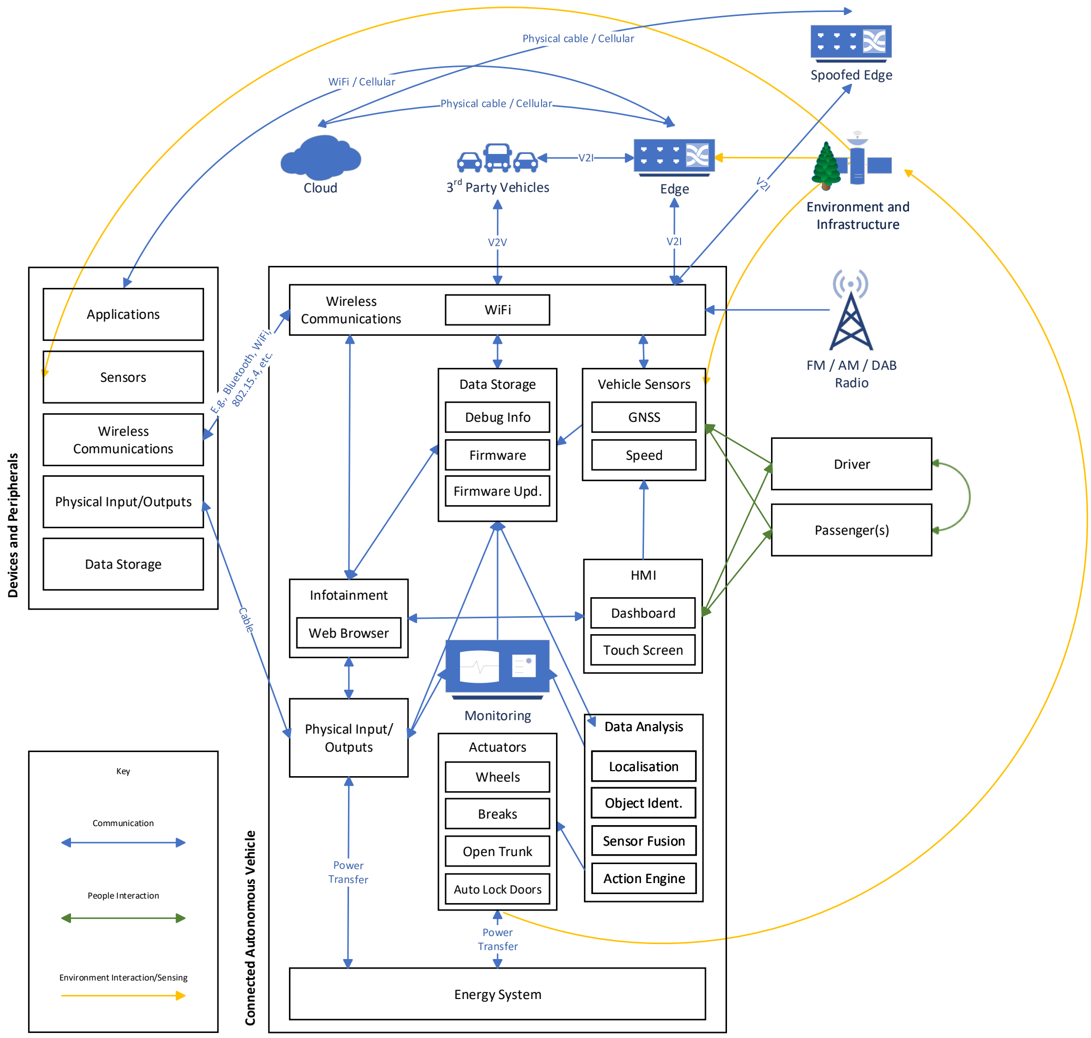
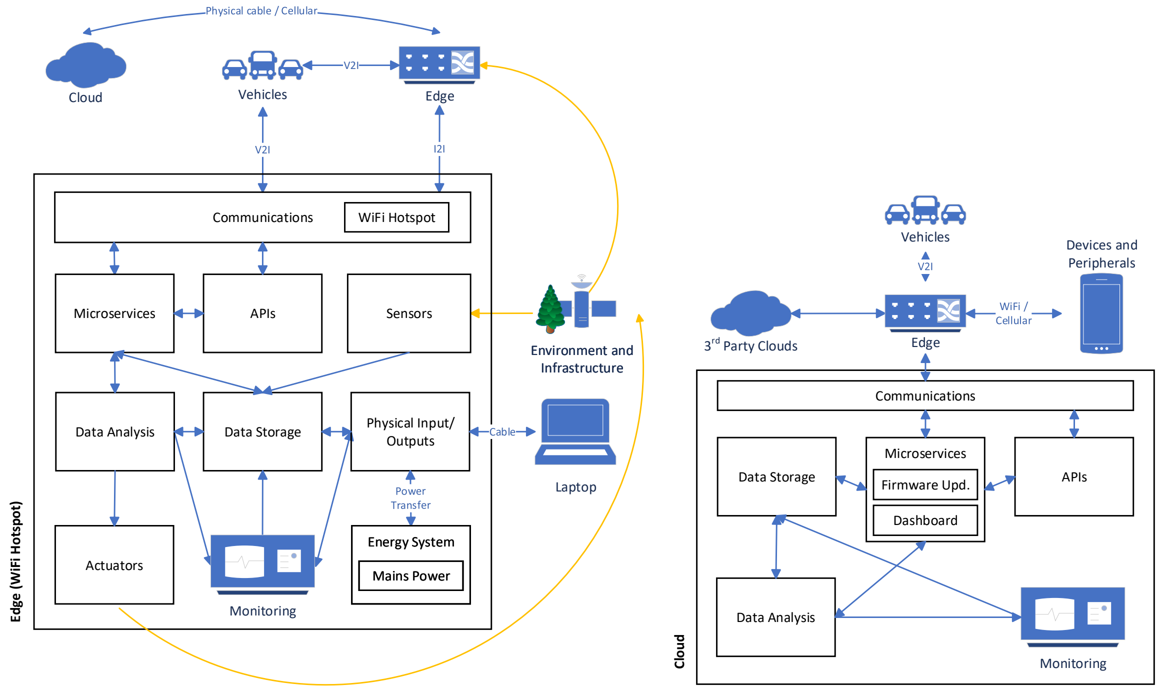


Figure 8. Tesla Exploit: Vehicle and Devices Instantiation

vulnerabilities. Attack tree analysis is an effective framework to address the implications of these attacks and how effective the patches/fixes are in eliminating the risks. Firstly, it can represent and simplify the attack, while highlighting the relevant components and their relations. Secondly, the attack can be understood in greater detail by identifying alternatives in the attack surface that could be used to achieve the similar goals. Thirdly, attack implications can be drawn from considering all the related threat agents and their motivations. Finally, the controls that manufacturers added to the CAV can also be verified and the attack trees can also suggest other effective controls to be considered in mitigating the threats.

The security analysis from [43] is extended by our attack tree analysis in Table 4 and Figure 10, with alternatives to browser attacks being identified as part of the attack surface. The goals of the hacktivists (i.e. Tencent researchers) can be extended to a higher-impact goal of a terrorist, for example, to control the CAV to create accidents when it is operating in a crowded environment. Relations between the threats, goals, and agents are illustrated in Figure 10. When applying the Tesla controls, it can be seen that the detailed attacks are eliminated, the relevant attack surfaces are significantly reduced, while the connections between the surfaces are removed.

Finally, the ability for the manufacturer to develop a fix in a short time period is important, it is also important that the fix can be rapidly deployed to vehicles on the road. In this instance Tesla was able to create and deploy a fix for these issues in two weeks. The infrastructure support to widely



(a) Edge **(b)** Cloud

Figure 9. Tesla Exploit: Edge and Cloud Instantiation

deploy these firmware fixes in a short period of time allows the impact to the vehicle's occupants to be small. If the firmware update needed to be shipped to customers to install themselves, or vehicles needed to be recalled, the risk to drivers will be higher due to the longer time the vehicle's spend unpatched. Having an over-the-air update system could introduce the potential for malicious or buggy firmware to be deployed to vehicles, however, that is a trade-off that needs to be considered with respect to the ability to widely deploy an update in a short period of time.

6. Discussion

Having described the reference architecture and presented two case studies that demonstrate how to apply it, this section will now discuss some of the implications and issues raised.

6.1. Expectation that the CAV Architecture Will Change

We expect that in the next decade and beyond the functionality of a CAV will change in unexpected ways. This reference architecture is designed to reflect the functionality that is expected to be deployed in the near future. For the far future, we expect that changes will need to be made to the reference architecture, which is why it has been designed to be modular. If new components or new interactions between components need to be added, then the reference architecture can be updated to include them. In doing so the attack surface of the system will change and the analysis will need to be re-performed.

6.2. Prioritising Attack Surface Analysis

Given the limited resources, defenders need to prioritise specific attack surfaces to protect, starting with threats that pose a high risk (high likelihood and high impact). Therefore, defenders need to perform a risk assessment which takes into account threat agents' capabilities and motivations as well as the available controls in the system. However, performing a risk assessment is complicated, takes time as it involves a large number of threats, and may contain uncertainties in the calculated

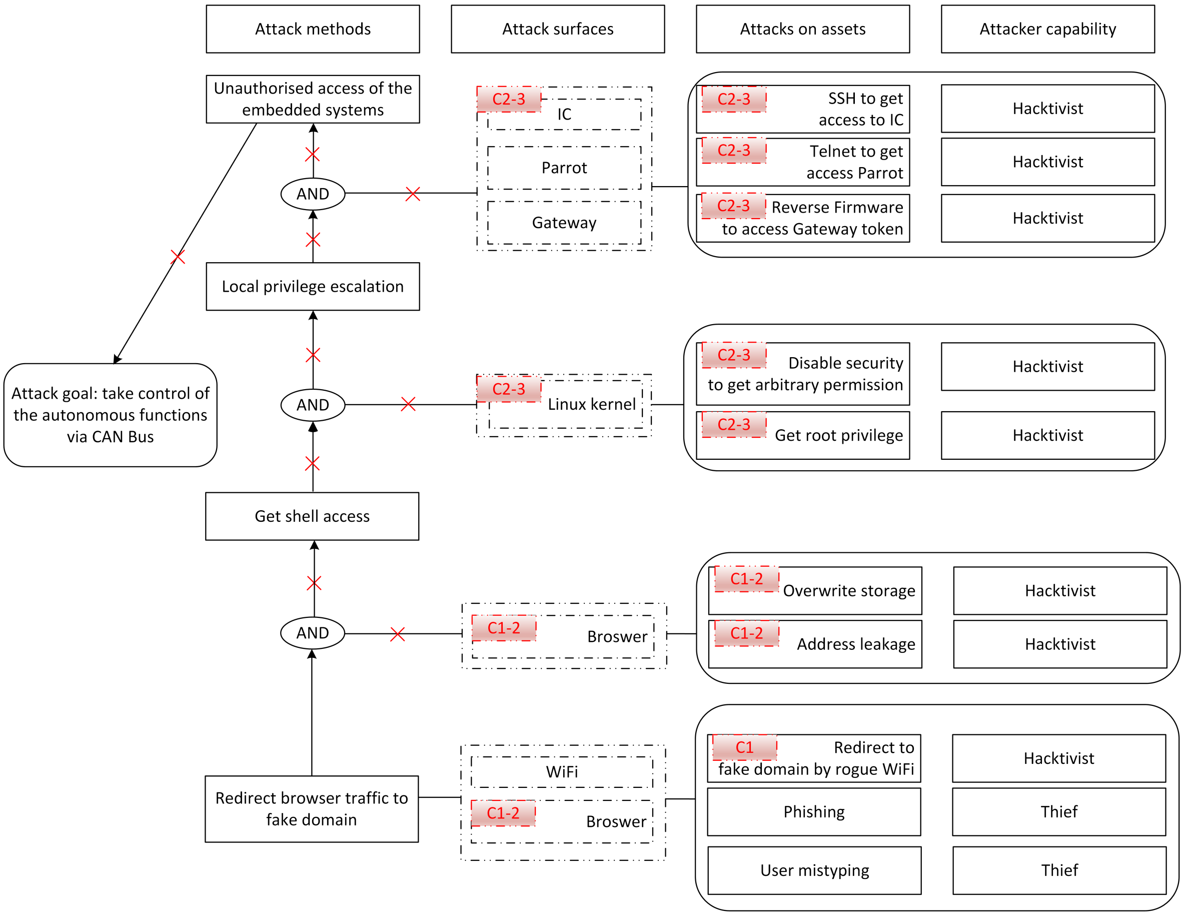


Figure 10. Attack Tree for Tesla Example that highlights a sequential attack for the attacker to reach its goal.

risk. Moreover, CAV risks are not static due to its dynamic operating environments. Consequently, the risk assessment will need to be repeated frequently to adapt to such changes. Therefore, to make the analysis more effective, it is important to shape the focus by prioritising the security resources for several core parts.

We argue that the core parts that should be prioritised are critical functional components exploited frequently by threat agents. When proper controls are applied the corresponding attack surfaces will be reduced, which creates further challenges for attackers. For CAVs these components are: (i) Communication, (ii) Sensors, and (iii) Data Analysis. Wireless or physical communication is the vector through which many cyber-attacks will be perpetrated, as it acts as a gateway between external agents and the internal components. Sensors are important because they provide information about CAV's surrounding environments. If sensor information is unavailable or modified maliciously, CAV may be manipulated to make harmful decisions. Finally, the Data Analysis is important, as it influences CAVs' autonomous functions.

6.3. The Need to Understand Trust-levels in All the Surfaces

Dominic *et al.* [7] recommended that the defenders should not place too much trust in individual CAV components. If any single trusted component exists, it can be the single point of failure that manipulates the whole security of the system once compromised. Consequently, defenders should put redundant security resources in different components to cross-check each other. However, when attackers manipulate more than one component, they may also be able to compromise the cross-check, eliminating a source of redundancy. Therefore, it is also important to understand trust levels properly in each component. When there are inconsistencies between them, understandings of their trust-levels will decide which components are in favour for making security decisions.

6.4. Isolating Critical Subsystems

Depending on applications and stakeholders' interest, some components can be considered more critical than others. For example, safety applications emphasise more on driving functionalities, while privacy applications focus more on data-related components. Putting more security resources on these critical components will not be enough to secure them, given the connections between attack surfaces can bring unknown threats from other vulnerable surface as shown in previous sections. Therefore, it is also important to isolate these critical parts from other vulnerable surfaces, or at least to create a secured shield around them by putting proper controls in their connections.

6.5. Considerations of Hardware and Software Security

In this reference architecture, we chose not to include the physical viewpoint and have only included a virtual implementation viewpoint as full representations of both viewpoints increase the difficulty of performing a high-level security analysis of a CAV. However, it is important to consider the security of the hardware and software of these systems. An issue is that for vehicular systems the software is typically only available as a black box, as manufactures are in general unwilling to supply the source code used for implementation. The same is typically true for the hardware in a CAV. This means that it can be useful to consider the system in terms of its functional components and their interactions. The high-level reference architecture presented in this work will be useful to initially describe the system, but a more detailed modelling language (such as SysML [72]) may be preferable when more details need to be specified. However, a reference architecture will be useful to highlight the attacker's path to achieve its goal and select in which component or interaction to implement mitigations for the attack. Additionally, it can also be referred in security verification when upgrading software or hardware for the system, which can happen frequently in a CAV's life cycle.

6.6. Using Reference Architecture to Mitigate Attacks

The reference architecture can provide an identification of which components and interactions are critical to the attacker research a goal. By analysing the generated attack trees, the component or interaction in which a mitigation is implemented can be justified. A common desire is to apply security controls in all potentially vulnerable components to minimise the attack surface. However, security resources are often limited, therefore, it is necessary to prioritise which countermeasures are implemented. The reference architecture can help to choose which mitigations to prioritise due to the ability to demonstrate the impact the mitigation will have in general. For example, the attack surfaces which lead to critical impacts should have the highest priority; while restricting surfaces which open the chances to attack other surfaces is usually more efficient than restricting isolated attack surfaces.

The reference architecture can also useful at the design phase of a system. For example, if security is critical, the designers should reduce the use of components that have large attack surface (e.g., by replacing them with more secure components) or restricting access to insecure functionality that link to other critical functionality. Finally, in the long term, the reference architecture can help to manage the complexity of systems and attacks. For instance, it can be used to visualise new vulnerabilities at a high level and also identify relevant mitigations when the system design changes.

7. Future Work

In this section two key areas in which the attack surface analysis needs to be developed further are discussed: (i) the automation of the analysis of system, and (ii) how to understand dynamically changing risk in different environments and scenarios.

7.1. Automated Analysis

In this work the reference architecture and the attack surface analysis has been performed manually. The important components, their interactions, and the ways in which they can be attacked have been derived by analysing how CAVs operate and how they can be attacked. Alternatively, the reference architecture and the attack surface analysis could be automated. To achieve this the important components and their interactions would need to be manually identified. These interactions could be specified in terms of the kind of interaction they represent (for example the class of data that is sent from one component to another). With this information the attack surface could be automatically explored using information about how an adversary could attack the system. This would allow attack trees to be automatically generated. However, not all attacks are likely to be interesting or feasible, so some manual pruning would be required.

7.2. Understanding Dynamic Risk

Risk analysis has been becoming compulsory to understand and control the potential system breaches and vulnerabilities [69]. Moreover, this analysis can also be used to rank the threats to help defenders deploy security resource most effectively for a mitigation plan. Although extensive research has been carried out on CAV risk analysis, there is little study which adequately tackle dynamic risks that CAVs are facing. It is not sufficient to assess CAV risks just for a single time because as a moving system, CAV's environment is changing frequently. As a result, risk assessments need updating to reflect new knowledge of environments and systems [73].

In the future, we plan to investigate factors that affect CAV risk assessment by answering when and how new assessments are needed, and the most efficient way to manage dynamic risks. This research would be essential to help CAVs to adapt quickly and more appropriately when operating in dynamic environments.

8. Conclusion

In this paper we have presented a reference architecture from a hybrid functional-communication viewpoint. This combined viewpoint allows easier attack surface analysis as the components and their interactions can be analysed from a single diagram. This reference architecture has been designed with four key sub-architectures for CAVs, the Edge, the Cloud, and Devices & Peripherals. The latter three are key to understanding the attack surface of a CAV, because they present new attack vectors that have previously been hard to specify. Finally, two examples of how to instantiate the reference architecture and analyse that instantiation have been provided showing how new and existing attacks can be analysed using this reference architecture.

Author Contributions: Conceptualization, Carsten Maple; Methodology, Matthew Bradbury, Kevin Ghirardello and Anh Tuan Le; Formal Analysis, Anh Tuan Le; Writing—Original Draft Preparation, Matthew Bradbury, Kevin Ghirardello and Anh Tuan Le; Writing—Review & Editing, Matthew Bradbury, Kevin Ghirardello, Anh Tuan Le, and Carsten Maple; Visualization, Matthew Bradbury and Anh Tuan Le; Supervision, Carsten Maple; Project Administration, Carsten Maple; Funding Acquisition, Carsten Maple

Funding: This work is supported by the Alan Turing Institute under EPSRC grant EP/N510129/1, the UK Hub for Cyber Security of the Internet of Things, PETRAS, under grant (EP/N02334X/1) and the CAPRI project under grant (TS/P012264/1).

Acknowledgments: The authors would like to thank Daniel Fowler for assistance proofreading this work.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the result.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Autonomous Driving
CAN	Controller Area Network
CAV	Connected and Autonomous Vehicles
DoS	Denial of Service
DDoS	Distributed Denial of Service
DSRC	Dedicated Short-Range Communications
ECU	Electronic Control Unit
GNSS	Global Navigation Satellite System (such as GPS, GLONASS, Galileo and BeiDou)
ITS	Intelligent Transport Systems
LIDAR	Light Detection and Ranging (detects object distance using light)
MiTM	Man-in-the-Middle
RCE	Remote Code Execution
RSU	Roadside Unit
TaaS	Theft as a Service
UAV	Unmanned Air Vehicle
USV	Unmanned Sea Vehicle

References

1. Winners of £51 million government competition to develop world-leading self-driving car testing infrastructure unveiled. https://www.gov.uk/government/news/winners-of-51-million-government-competition-to-develop-world-leading-self-driving-car-testing-infrastructure-unveiled, 2017. Accessed: 2018-07-31.
2. Siegel, J.E.; Erb, D.C.; Sarma, S.E. A Survey of the Connected Vehicle Landscape—Architectures, Enabling Technologies, Applications, and Development Areas. *IEEE Transactions on Intelligent Transportation Systems* **2018**, *19*, 2391–2406. doi:10.1109/TITS.2017.2749459.
3. Hussain, R.; Zeadally, S. Autonomous Cars: Research Results, Issues, and Future Challenges. *IEEE Communications Surveys Tutorials* **2019**, *21*, 1275–1313. doi:10.1109/COMST.2018.2869360.
4. Hegde, R.; Mishra, G.; Gurumurthy, K.S. An Insight into the Hardware and Software Complexity of ECUs in Vehicles. Advances in Computing and Information Technology; Wyld, D.C.; Wozniak, M.; Chaki, N.; Meghanathan, N.; Nagamalai, D., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2011; pp. 99–106.
5. The Institution of Engineering and Technology.; the Knowledge Transfer Network. Automotive Cyber Security: An IET/KTN Thought Leadership Review of risk perspectives for connected vehicles. Technical report, The Institution of Engineering and Technology, 2015.
6. SAE. *J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*, 2016. J3061_201601.
7. Dominic, D.; Chhawri, S.; Eustice, R.M.; Ma, D.; Weimerskirch, A. Risk Assessment for Cooperative Automated Driving. Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy; ACM: New York, NY, USA, 2016; CPS-SPC '16, pp. 47–58. doi:10.1145/2994487.2994499.
8. The Architecture Team. Architecture Reference for Cooperative and Intelligent Transportation. https://local.iteris.com/arc-it/index.html, 2018. Version 8.1.
9. Sheehan, B.; Murphy, F.; Mullins, M.; Ryan, C. Connected and autonomous vehicles: A cyber-risk classification framework. *Transportation Research Part A: Policy and Practice* **2019**, *124*, 523 – 536. doi:10.1016/j.tra.2018.06.033.
10. Petit, J.; Stottelaar, B.; Feiri, M. Remote Attacks on Automated Vehicles Sensors : Experiments on Camera and LiDAR. Black Hat Europe, 2015.
11. Miller, C.; Valasek, C. Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat, 2015.
12. Parkinson, S.; Ward, P.; Wilson, K.; Miller, J. Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges. *IEEE Transactions on Intelligent Transportation Systems* **2017**, *18*, 2898–2915. doi:10.1109/TITS.2017.2665968.
13. Macher, G.; Höller, A.; Sporer, H.; Armengaud, E.; Kreiner, C. A Combined Safety-Hazards and Security-Threat Analysis Method for Automotive Systems. Computer Safety, Reliability, and Security; Koornneef, F.; van Gulijk, C., Eds.; Springer International Publishing: Cham, 2015; pp. 237–250.
14. Luettel, T.; Himmelsbach, M.; Wuensche, H. Autonomous Ground Vehicles — Concepts And A Path To The Future. *Proceedings of the IEEE* **2012**, *100*, 1831–1839. doi:10.1109/JPROC.2012.2189803.

15. Schneier, B. Secret and Lies. *Digital Security in a Networked World* **2000**.
16. Alberts, C.J.; Dorofee, A. *Managing Information Security Risks: The Octave Approach*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2002.
17. UcedaVelez, T.; Morana, M.M. *Risk Centric Threat Modeling: process for attack simulation and threat analysis*; John Wiley & Sons, 2015.
18. McCarthy, C.; Harnett, K.; Carter, A. Characterization of potential security threats in modern automobiles: A composite modeling approach. Technical report, Washington, DC: National Highway Traffic Safety Administration, 2014. Report No. DOT HS 812 074.
19. Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems. https://www.sae.org/standards/content/j3016_201806, 2018.
20. Bartels, A.; Eberle, U.; Knapp, A. Deliverable D2.1: System Classication And Glossary. http://www.adaptive-ip.eu/index.php/AdaptIVe-SP2-v12-DL-D2.1-System%20Classification-file=files-adaptive-content-downloads-Deliverables%20&%20papers-AdaptIVe-SP2-v12-DL-D2.1-System%20Classification.pdf, 2015.
21. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S. Experimental Security Analysis of a Modern Automobile. 2010 IEEE Symposium on Security and Privacy, 2010, pp. 447–462. doi:10.1109/SP.2010.34.
22. Foster, I.; Prudhomme, A.; Koscher, K.; Savage, S. Fast and Vulnerable: A Story of Telematic Failures. 9th USENIX Workshop on Offensive Technologies (WOOT 15); USENIX Association: Washington, D.C., 2015.
23. Petit, J.; Shladover, S.E. Potential Cyberattacks On Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems* **2015**, *16*, 546–556. doi:10.1109/TITS.2014.2342271.
24. Cloutier, R.; Muller, G.; Verma, D.; Nilchiani, R.; Hole, E.; Bone, M. The Concept of Reference Architectures. *Systems Engineering* **2010**, *13*, 14–27, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.20129]. doi:10.1002/sys.20129.
25. Reference Architecture For Space Data Systems. https://public.ccsds.org/Pubs/311x0m1.pdf, 2008. CCSDS 311.0-M-1.
26. Lin, S.W.; Miller, B.; Durand, J.; Bleakley, G.; Chigani, A.; Martin, R.; Murphy, B.; Crawford, M. The Industrial Internet of Things Volume G1: Reference Architecture. http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf, 2017.
27. CEN-CENELEC-ETSI Smart Grid Coordination Group. Smart Grid Reference Architecture. https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf, 2012.
28. Behere, S.; Törngren, M. A functional reference architecture for autonomous driving. *Information and Software Technology* **2016**, *73*, 136 – 150. doi:10.1016/j.infsof.2015.12.008.
29. Osório, A.L.; Afsarmanesh, H.; Camarinha-Matos, L.M. Towards A Reference Architecture For A Collaborative Intelligent Transport System Infrastructure. Collaborative Networks For A Sustainable World; Camarinha-Matos, L.M.; Boucher, X.; Afsarmanesh, H., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2010; pp. 469–477.
30. Stevens, A.; Dianati, M.; Katsaros, K.; Han, C.; Fallah, S.; Maple, C.; McCullough, F.; Mouzakitis, A. Cooperative automation through the cloud: The CARMA project. 12th ITS European Congress, 2017.
31. Sheik, A.T.; Maple, C. Key Security Challenges for Cloud-Assisted Connected and Autonomous Vehicles. Living in the Internet of Things: Cybersecurity of the IoT, 2019, pp. 1–10.
32. Passchier, I.; van Sambeek, M. Architecture For C-ITS Applications In The Netherlands. http://smartmobilitycommunity.eu/sites/default/files/AI_CITSArchitectureNL_v1.00.pdf, 2016.
33. Heise, C.D., Architecture Reference Of ITS In The USA. In *Intelligent Transport Systems*; Wiley-Blackwell, 2015; chapter 2, pp. 18–35. doi:10.1002/9781118894774.ch2.
34. Begoña, M.; Sergio, C.; (Iñaki), O.I.; Isabel, T.A., Reference ITS Architectures In Europe. In *Intelligent Transport Systems*; Wiley-Blackwell, 2015; chapter 1, pp. 1–17. doi:10.1002/9781118894774.ch1.
35. Puñal, O.; Aguiar, A.; Gross, J. In Vanets We Trust?: Characterizing Rf Jamming In Vehicular Networks. Proceedings Of The Ninth Acm International Workshop On Vehicular Inter-networking, Systems, And Applications; ACM: New York, NY, USA, 2012; VANET '12, pp. 83–92. doi:10.1145/2307888.2307903.
36. Davis, A. Broadcasting Your Attack: Security Testing DAB Radio In Cars. Black Hat, 2015.

37. Sampigethaya, K.; Li, M.; Huang, L.; Poovendran, R. AMOEBA: Robust Location Privacy Scheme for VANET. *IEEE Journal on Selected Areas in Communications* **2007**, *25*, 1569–1589. doi:10.1109/JSAC.2007.071007.
38. Gu, P.; Khatoun, R.; Begriche, Y.; Serhrouchni, A. Vehicle Driving Pattern Based Sybil Attack Detection. 2016 IEEE 18th International Conference On High Performance Computing And Communications; Ieee 14th International Conference On Smart City; Ieee 2nd International Conference On Data Science And Systems (hpcc/smartcity/dss), 2016, pp. 1282–1288. doi:10.1109/HPCC-SmartCity-DSS.2016.0182.
39. Xu, W.; Wegner, M.; Wolf, L.; Kapitza, R. Byzantine Agreement Service For Cooperative Wireless Embedded Systems. 2017 47th Annual Ieee/ifip International Conference On Dependable Systems And Networks Workshops (dsn-w), 2017, pp. 10–15. doi:10.1109/DSN-W.2017.45.
40. Francillon, A.; Danev, B.; Capkun, S. Relay Attacks On Passive Keyless Entry And Start Systems In Modern Cars. Network And Distributed Systems Security (NDSS) Symposium, 2011.
41. USB Kill. https://usbkill.com. Accessed: 2018-06-21.
42. Currie, R. Hacking The CAN Bus: Basic Manipulation Of A Modern Automobile Through CAN Bus Reverse Engineering. Technical report, 2017.
43. Nie, S.; Liu, L.; Du, Y. Free-fall: Hacking Tesla From Wireless To CAN Bus. Black Hat, 2017.
44. Evenchick, E. Hopping on the CAN Bus. Black Hat Asia, 2015.
45. Munir, A.; Koushanfar, F. Design And Analysis Of Secure And Dependable Automotive Cps: A Steer-by-wire Case Study. *IEEE Transactions on Dependable and Secure Computing* **2018**, pp. 1–1. doi:10.1109/TDSC.2018.2846741.
46. Rouf, I.; Miller, R.; Mustafa, H.; Taylor, T.; Oh, S.; Xu, W.; Gruteser, M.; Trappe, W.; Seskar, I. Security And Privacy Vulnerabilities Of In-car Wireless Networks: A Tire Pressure Monitoring System Case Study. Proceedings Of The 19th Usenix Conference On Security; USENIX Association: Berkeley, CA, USA, 2010; USENIX Security'10, pp. 21–21.
47. Papernot, N.; McDaniel, P.; Sinha, A.; Wellman, M. SoK: Security And Privacy In Machine Learning. 3rd IEEE European Symposium On Security And Privacy, 2018.
48. GPS.gov. GPS Standard Positioning Service (SPS) Performance Standard. Technical Report 4, 2008.
49. Athalye, A.; Engstrom, L.; Ilyas, A.; Kwok, K. Synthesizing Robust Adversarial Examples. 35th International Conference On Machine Learning (icml 2018), 2018. To Appear.
50. Golson, J. Tesla driver killed in crash with Autopilot active, NHTSA investigating. https://www.theverge.com/2016/6/30/12072408/tesla-autopilot-car-crash-death-autonomous-model-s, 2016. Accessed: 2018-07-31.
51. Sasiadek, J.Z.; Hartana, P. Sensor data fusion using Kalman filter. Proceedings of the Third International Conference on Information Fusion, 2000, Vol. 2, pp. WED5/19–WED5/25 vol.2. doi:10.1109/IFIC.2000.859866.
52. Ivanov, R.; Pajic, M.; Lee, I. Attack-resilient Sensor Fusion For Safety-critical Cyber-physical Systems. *ACM Trans. Embed. Comput. Syst.* **2016**, *15*, 21:1–21:24. doi:10.1145/2847418.
53. Davidson, D.; Wu, H.; Jellinek, R.; Singh, V.; Ristenpart, T. Controlling UAVs With Sensor Input Spoofing Attacks. 10th USENIX Workshop On Offensive Technologies (WOOT 16); USENIX Association: Austin, TX, 2016.
54. Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S.; Koscher, K.; Czeskis, A.; Roesner, F.; Kohno, T. Comprehensive Experimental Analyses of Automotive Attack Surfaces. Proceedings of the 20th USENIX Conference on Security; USENIX Association: Berkeley, CA, USA, 2011; SEC'11, pp. 6–6.
55. Woo, S.; Jo, H.J.; Lee, D.H. A Practical Wireless Attack On The Connected Car And Security Protocol For In-vehicle Can. *IEEE Transactions on Intelligent Transportation Systems* **2015**, *16*, 993–1006. doi:10.1109/TITS.2014.2351612.
56. Kikuchi, H.; Yokomizo, T. Location Privacy Vulnerable From Bluetooth Devices. 2013 16th International Conference On Network-based Information Systems, 2013, pp. 534–538. doi:10.1109/NBiS.2013.89.
57. Li, Z.; Pei, Q.; Markwood, I.; Liu, Y.; Pan, M.; Li, H. Location Privacy Violation Via Gps-agnostic Smart Phone Car Tracking. *IEEE Transactions on Vehicular Technology* **2018**, *67*, 5042–5053. doi:10.1109/TVT.2018.2800123.

58. Volkswagen Apps. http://www.volkswagen.co.uk/about-us/innovation/mobile-apps. Accessed: 2018-06-21.
59. Cunche, M. I Know Your Mac Address: Targeted Tracking Of Individual Using Wi-fi. *Journal of Computer Virology and Hacking Techniques* **2014**, *10*, 219–227. doi:10.1007/s11416-013-0196-1.
60. Gañán, C.; Loo, J.; Ghosh, A.; Esparza, O.; Reñé, S.; Muñoz, J.L. Analysis Of Inter-rsu Beaconing Interference In Vanets. Multiple Access Communications; Bellalta, B.; Vinel, A.; Jonsson, M.; Barcelo, J.; Maslennikov, R.; Chatzimisios, P.; Malone, D., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp. 49–59.
61. Fielding, R.T. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, 2000. AAI9980887.
62. Liu, F.; Tong, J.; Mao, J.; Bohn, R.; Messina, J.; Badger, L.; Leaf, D. NIST cloud computing reference architecture. *NIST special publication* **2011**, *500*, 1–28.
63. Juliadotter, N.V.; Choo, K.R. Cloud Attack and Risk Assessment Taxonomy. *IEEE Cloud Computing* **2015**, *2*, 14–20. doi:10.1109/MCC.2015.2.
64. Alhebaishi, N.; Wang, L.; Jajodia, S.; Singhal, A. Threat Modeling For Cloud Data Center Infrastructures. Foundations And Practice Of Security; Cuppens, F.; Wang, L.; Cuppens-Boulahia, N.; Tawbi, N.; Garcia-Alfaro, J., Eds.; Springer International Publishing: Cham, 2017; pp. 302–319.
65. Mansfield-Devine, S. The growth and evolution of DDoS. *Network Security* **2015**, *2015*, 13 – 20. doi:https://doi.org/10.1016/S1353-4858(15)30092-1.
66. Duncan, A.J.; Creese, S.; Goldsmith, M. Insider Attacks In Cloud Computing. 2012 IEEE 11th International Conference On Trust, Security And Privacy In Computing And Communications, 2012, pp. 857–862. doi:10.1109/TrustCom.2012.188.
67. Bertino, E.; Lin, D.; Jiang, W., A Survey of Quantification of Privacy Preserving Data Mining Algorithms. In *Privacy-Preserving Data Mining: Models and Algorithms*; Aggarwal, C.C.; Yu, P.S., Eds.; Springer US: Boston, MA, 2008; pp. 183–205. doi:10.1007/978-0-387-70992-5_8.
68. Henniger, O.; Apvrille, L.; Fuchs, A.; Roudier, Y.; Ruddle, A.; Weyl, B. Security requirements for automotive on-board networks. Intelligent Transport Systems Telecommunications,(ITST), 2009 9th International Conference on. IEEE, 2009, pp. 641–646.
69. Boudguiga, A.; Boulanger, A.; Chiron, P.; Klaudel, W.; Labiod, H.; Seguy, J.C. RACE: Risk analysis for cooperative engines. New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on. IEEE, 2015, pp. 1–5.
70. Monteuuis, J.P.; Boudguiga, A.; Zhang, J.; Labiod, H.; Servel, A.; Urien, P. SARA: Security Automotive Risk Analysis Method. Proceedings of the 4th ACM Workshop on Cyber-Physical System Security. ACM, 2018, pp. 3–14.
71. Rosenquist, M. Prioritizing information security risks with threat agent risk assessment. *Intel Corporation White Paper* **2009**.
72. Object Management Group. *OMG Systems Modeling Language*, 2017. Version 1.5.
73. Zio, E. The future of risk assessment. *Reliability Engineering & System Safety* **2018**, *177*, 176–190. doi:10.1016/j.ress.2018.04.020.