POLYTECHNIC UNIVERSITY OF TURIN

Degree course in Computer Engineering

Master of Science Thesis

Exploiting virtual networks for CPS security analysis

Advisor
prof. Antonio Lioy
eng. Andrea Atzeni

Candidate
Santo Orlando

Academic year 2021-2022

Summary

CPS, or *Cyber-Physical System*, is the subject of numerous studies nowadays, with continuous updates from different perspectives. It represents an intelligent system capable of letting the real world interact with the non-real one, the so called "cyber world". There are different types of CPS and, among them, one deserves attention more than the others due to the fact that it is growing rapidly: the world of **VANET**. This acronym stands for "*Vehicular Ad hoc NETwork*" and represents a vehicular network composed of cars that can communicate with each other, with pedestrians and other infrastructures. In the best case, they provide autonomous driving by making the driver simply become a passenger. The vehicles are known as **CAV**, which corresponds to Connected Autonomous Vehicle, and they are able to exchange messages with each other or with the other elements that are part of this described subcategory of CPS.
It is important, however, to focus on an essential aspect concerning this network, namely, the Cybersecurity that involves these systems. It is certainly of primary importance to make sure that the C.I.A. (Confidentiality, Integrity and Availability) triad is secured within a system. In this way it will be possible to make sure that the systems will be protected and there will be no possibility of attack. Unfortunately, this is not always the case: as will be seen in specific sections, some attack that has occurred in real life has affected well-known car manufacturers. So, it is necessary that optimal security measures are implemented within the system.
The **simulation** of a large vehicular network is certainly time and money consuming in many ways. This is precisely why simulators come into play, that is, software designed to run reasonably real scenarios that allow their analysis in order to make the right improvements. Through appropriate configurations and appropriate software is possible to simulate situations and types of attacks such that it is possible to observe what could happen in reality. In the course of the discussion, some attacks will be simulated and it will be possible to observe their consequences and possibly the damage it could cause in real life. For this reason, a real life scenario will be simulated, which is part of the city of Turin.
Of all the simulators, a focus will be given on Veins, the main simulator known nowadays for analysis on VANET systems. As will be seen, however, some attacks cannot always be implemented due to a number of internal limitations of the

software, such as lack of cooperation with other tools or because a hypothetical implementation of a solution cannot take place in reasonable time. For those attacks for which an analysis and implementation is possible, mitigations and countermeasures will be defined. Furthermore, what is important and what is a focus to be taken into consideration is the fact that many simulators are not open source; on the other hand, these are commercial and, as a consequence, the simulations are not possible because they are not accessible to everyone.
Hence, the objective of this thesis is to analyse CPS and then describe VANET in detail. Once the main security concepts are defined, simulators will be introduced and a scenario of the city of Turin will be simulated, implementing those feasible attacks. An essential point, which is the core of this thesis, can be observed in the limitations that the simulator demonstrates. In fact, it is observed how some attack cannot be totally implemented due to software limitations on the chosen simulator. Finally, possible strategies to overcome these limitations will be proposed, and the detection and mitigation of attacks that can be conducted will be outlined.

Acknowledgements

Master of Science thesis represents the final point of lots of years spent among books and moments at the university. For this reason, I would like to thank who has been next to me in these years.

Firstly, I would like to thank Professor Antonio Lioy and the engineer Andrea Atzeni for having given to me the opportunity to cover a really interesting topic.

Thank you mum and dad, your encouragement at all times has been a source of continuity for me and has given to me the strength to accomplish my goals.

Thank you Luana, my sister, for always understanding every circumstance during this university period and for providing me with the right advice.

Thank you Fabiola, for having shared five university years in which we have experienced moments of difficulty and happiness, always facing them with the determination that distinguishes us.Thank you also for having believed in me, always.

Thank you Antonio and Claudia, childhood friends and university colleagues with whom I shared good times especially in Turin.

Thank you Alessandro, Francesco, Matteo, Francesco and Marco, friends since high school with whom I have spent good moments and for having always understood me.

Contents

Chapter 1

Introduction

Nowadays it is right to say that every kind of human operation has become automated. Starting from the simpler things to the more complex, is possible to observe a different level of automation.
Industries, health care, medical devices, agriculture, national security, traffic flow management and many others need to have systems that are capable of solving problems by computing results, communicating them to other sub-systems in real-time, monitoring and controlling physical entities, in complete or partial autonomy.
For example, many companies want to improve their cooperation among different departments, in order to give the right information to the right people, possibly in real-time.
In health care, it is important to observe and monitor patients' data, in order to notify medical doctors if there is something strange, so as to improve the response time and maybe save a life.
In agriculture, systems that monitor plants can be used to collect data in order to help farmers.
Under national security's point of view, is crucial to have real-time systems that monitor devices to give a correct countermeasure in case of attacks. In other words, the aim is the simplification of decisional process, augmenting the efficiency and the productivity.
Furthermore, taking into consideration internet, it is relevant to say that it has changed the way in which we manage our things and solve our problems.
As a result, there is a sort of gap between the cyber world and the physical one. Indeed, to be more specific, the former manages transmission, modification and receipt of information. On the other side, the latter represents the generation of information for which a solution can be found. The idea that has been introduced to describe these types of scenario can be found in the concept of **Cyber-Physical System**.
CPSs are slowly spreading like wildfire, granting and improving that level of binding between human-being and machine, so as to make humans' life easier.
Among all the fields covered by the CPS, the **automotive** one is constantly growing.
To be more specific, one specialization of CPSs in that field is represented by **MANET**(Mobile Ad hoc NETwork) which, on its turn, includes **VANET** (Vehicular Ad hoc NETwork). These are able to permit a driving experience partial or completely autonomous, through communication between vehicles and devices

placed on road side. In other words, developers are trying to make this kind of vehicle "smart", so as to offer services of different nature, in order to improve the quality of driving.
Thanks to VANET is possible to create a system made of vehicles with a constant evolving topology of the net, due to the fact that the position of communicating vehicles is taken into consideration as fundamental parameter. The quality of driving is referred to those kinds of vehicles that are able to drive autonomously. In fact, there are **six** levels of automation; starting from level zero, in which the car is driven by humans without help by the car to the level six, in which the car is without pedals, steering wheel and it is self-driven. Up to now, the level three is particularly used , where the car is partially autonomous but the driver has to be ready to take the control of the car in case of problems.
Many car manufacturers are investing in that field, in order to develop cars able to drive themselves with levels of automation greater than three. Audi, Mercedes-Benz, Tesla, Toyota, BMW, Ford, Volvo, Uber, Hyundai, Nissan-Renault-Mitsubishi alliance and Google too are an example. For instance, the 2009 has been the year in which Google has launched their driverless car, the pioneer of that way of driving. Each of them started studies under the cybersecurity's point of view even if, as it will be seen, lots of them have been subject to hacker attacks.
According to [1], it has been observed that:

- if the autopilot is enabled, there has been 1 incident every 7,92 millions km;
- if not, there has been 1 incident every 2,51 millions of km.

Audi, on its turn, has made a report [2] in 2022 in which describes the myths regarding autonomous driving. One of them focuses on the concept of the security, describing the fact that next to the developing of driverless car (which are more secure than the normal ones):
"manufacturers are constantly developing protective measures against cyberattacks and improving the protective mechanisms, both inside the vehicle and outside in the back end".
In addition, there are many cybersecurity attacks that affect this category of cyber-physical systems.
Furthermore, to mention one of the cybersecurity aspect in this field, the communication one must be able to avoid disruption of systems, generation of misinformation or tracking of vehicles to retrieve network topology and steal sensible information about drivers such as places mostly visited.
In other words, what is worrying a lot manufacturers is the fact that many cyberattacks can be conducted on this type of cars, and these attacks can be very dangerous because, in the worst case, the control of the car (under software and hardware point of view) is lost. For this reason, given the difficulty of analysis of these systems, also under cybersecurity aspect, and to avoid the possibility of setting up a complex, real and expensive system (i.e. a vehicular one made of roads, cars, etc.) to perform simulations, a solution can be found in the **simulators**. They are able to make a detailed and meticulous analysis, generating data to be studied in order to have a vehicular system correctly designed under different points of view, especially under the security one.

1.1 Thesis structure

For that reason, this thesis will be structured in this way:

- chapter 2 starts with the definition of the concept of the CPS for later examine on its principal specialization: the world of the VANET. So, it will defined the general architecture of this type of network, analysing each single component that is devoted to the realisation of it.
  Finally, the internal communication protocol will be described, highlighting the standards implemented for VANET;

- chapter 3 focuses its attention on the concept of security which covers the world of VANET. Specifically, cybersecurity concept is divided into its three main aspect: *cyber*, *cyber-physical* and *physical*.
  Once defined the security requirements, it will follow a detailed analysis of which can be the real attacks that affects the VANET system with its associated countermeasures, through the listing of a series of attacks concretely happened in famous car manufacturers;

- chapter 4 firstly introduces simulators, describing the different types and showing the principal challenges. Then there will be the analysis of two important simulators, **Veins** and **Eclipse MOSAIC**, since they are the principal open-source framework.
  Finally this chapter will be concluded by listing the cybersecurity aspects that they offer and discussing, under a critical point of view, how some functionalities have not been implemented yet, resulting in a difficult realization of simulations;

- chapter 5 contains the simulations of different types of attack. In particular, lots of them are implemented according to the classes of attack defined in Chapter 3. It is configured the simulation by defining the map that will be used. In particular, SUMO commands are used to generate the map obtained from the city of Turin. As a result, some attack will be simulated inside the previously generated map. As it will be seen, in some situations the mobility of nodes inside the network is important and plays a role in the analysis of the outcome. On others situations there are attacks that, for instance, focuses primarily on the exchanging of messages or on the wireless communication, so it is relevant to take into consideration other parameters.
  The totality of simulations have been conducted on **Veins**. Despite being the latest open-source framework with continuous update and several releases, some attack is still not implementable, due to the limitation of the simulator;

- chapter 6 analyses the simulations previously realized and observes the behaviour of the network under attack and how CAVs and RSUs behave. All implemented attacks are analysed and mitigations are proposed. When feasible, they are practically implemented and proposed as a solution. In some case it is not possible to do it due to the limitations that will be discussed in Chapter 7, and solutions are proposed according to the literature. Furthermore, it is interesting to observe the values that have been observed during the attacks, such as the number of packets that the systems do not have been

able to manage and the number of packets per second lost, which give a concrete idea of what could happen in real life. Again, different limitations are raised from the analysis of these cybersecurity attacks;

- chapter 7 concludes the thesis work. It focuses on what has been explained, analysed, simulated and observed. As a result of it, it is explained how has not been possible to implement all cybersecurity attacks and some attack that has really happened due to the limitation of Veins simulator. In particular, different limitations are explained and, where possible, improvements for future works are proposed. The message that has been given is that nowadays is still difficult to create a scenario to run a simulation because some pieces are still missing. These pieces can be identified in different modules that involves the cars and their functionalities, such as the management of different parts of the car (horn, car doors etc.) and functional modules (i.e. integrated Veins security functionalities);

- appendix contains the end user guide to install the software that has been used in the simulations and a manual on how to make a simulation by using Veins tool. These are Appendix A and Appendix B, respectively.

Chapter 2

CPS overview and its specialization: VANET

2.1 CPS overview

According to **NIST** (National Institute of Standards and Technology):

"Cyber-physical systems (CPSs) are smart systems that include engineered interacting networks of physical and computational components." [3]

In line with this definition, it is obvious to understand that CPSs are not the traditional systems or the real-time ones. On the contrary, they have a set of features that makes them unique.
In fact, these interconnected systems are able to provide new characteristics to improve the quality of life. Depending on what is the sector in which this type of systems is applied, it can be particularly useful. As indicated previously, a CPS

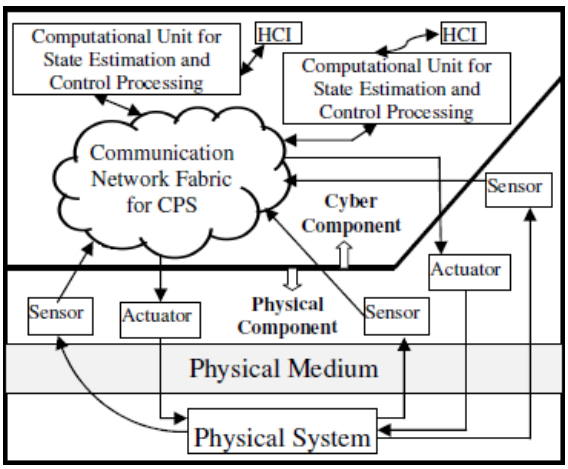

Figure 2.1. View of a Cyber-Physical System [4]

consists of a cyber component and a physical part. The figure 2.1 is the sum of its formal definition. The core of the cyber component is made by the communication network, composed by sensor networks and computational units. The former are used for communication between sensors. The latter is linked to the actuators (for

controlling the physical system) through internet with wires or wirelessly and performs computation, generation of results and control signals for other parts of the system. On the other side, the human-computer interfaces (HCIs) allow humans to take control decisions based on results that come from other parts of the CPS.
The physical part of a CPS includes the entities that are closer to the particular kind of physical system under observation. For example, in smart cars, sensors that take measurements related to the speed of cars and inform drivers about it.
Even though this figure depicts a sort of logical separation, in real life there is a continuous communication between these components.

2.2 CPS scope

Scopes of a Cyber Physical System are wide. Its segmentation involves many application sectors. Among them, it is relevant to take into consideration the one that includes the **smart transportation**. It is one of the most studied and, over time, it is evolving a lot. Smart transportation is composed by different elements, such as autonomous vehicles and traffic system controls. In other words, these are a clear example of CPSs.
To be more precise the formers, as it will be seen in section 2.3, are capable of communicating with other autonomous vehicles or with road side devices; the latters, on the other hand, receive information from vehicles and are able to make measurements in order to manage traffic and to inform other vehicles in different cases, such as during an emergency.
As it can be inferred, CPS can communicate with each other to produce more complex systems. These are called **SoS**: "Systems of Systems". In this way it will be possible to have a continuous communication in the evolving and just created cyber space, setting up a sort of community of host that communicates and informs who comes inside this cyber space, from time to time, so as to make a decision, depending on what information comes up, such as lane changes to avoid incidents, vehicles in proximity, deceleration, stop, etc.
Furthermore, vehicles collect data about the on location traffic in order to send them to traffic monitoring systems which, on their turn, are able to change the route of vehicles to avoid traffic. CPS can communicate with another CPS regarding weather, which collects data and inform drivers early.
The next section will describe in detail the type of Cyber Physical System previously mentioned. As it will be seen it is called VANET, Vehicular Ad hoc NETwork.

2.3 VANET: characteristics

VANET is a type of network that wires an enormous field of mobile circulated applications which runs in vehicle. Vehicular Ad-hoc Network (VANET) are an exact specialization of the Mobile Ad-hoc Network (MANET) where the vehicle acts as the mobile nodes; the node should communicate with each other through single hop or multi hop. Their main characteristics are :

- **High mobility**. In VANET, the node inside the network moves in a fast way

and, as a result, it is important to calculate and to have the right position of the vehicle;

- **Rapid change of net topology**. The node in VANET is high mobile and their speed is also very random. As a result, the node position will change with high frequency. For this reason, the topology of the net is dynamic and it depends on the route of the vehicles;
- **Frequent exchange of information**. Normally VANET collects various information from their neighbour vehicles and other infrastructures. So that, the nodes exchange their information periodically;
- **Time critical**. Within the time period, the information in VANET should send to the accurate node. The node will make a decision and execute action correspondingly;
- **Infrastructureless**. VANET is an infrastructureless network, if it is considered that the network is formed by vehicles. For that, there is no need of any physical medium between vehicles for communication and there is no need of any centralized controlling authority. In VANET is possible a type of communication known as "hop-to-hop", and for that there is no need of hardware devices like switches or hubs. "Even if we consider other units such as Road Side Units in the network, these are very basic resources which are deployed along the road side."; [5]
- **Self-organized**. "Nodes in VANET takes their own decisions for forwarding messages. Nodes itself act as a switch for transferring data. Hop-to-hop communication is possible. These features make VANET as a self-organized network."; [5]
- **Critical latency requirement**. "Latency is nothing but the time interval between sending messages by a source node and receiving messages by a receiver node is different from zero". In a VANET, due to highly mobile nodes, i.e., vehicles, they may be running in opposite directions. The nodes remain in the vicinity of each other for very short time periods. It is important to receive the message by the destination vehicle in a given time period. To achieve this network, a critical latency requirement is needed. Communication in the VANET network should be made with low latency. [5]

2.4 VANET components

The VANET, as it will be seen, is made of several components, each of them dedicated to the execution of specific tasks. They allow different kinds of communication. Figure 2.2 is an example of it. Two of them are particularly relevant to be analysed:

- **V2V**: "Vehicle to Vehicle", which represents a direct communication between CAVs (Connected Autonomous Vehicles). In this type of communication a vehicle can accept transmission and exchange helpful traffic information such as information regarding distances, speed, incidents and so on;

- **V2I**: "Vehicle to Infrastructure", that represents a kind of communication between a CAV and an infrastructure such as the RSU (Road Side Unit). To be more specific, in this transmission type, the details will become transmitted between the nodes (i.e auto mobile) and the infrastructure, to talk about beneficial information such as street safety and conditions events. An auto mobile (node) launches a connection between RSU and get in touch with exterior systems, which is usually internet.

In general, the communication between CAV and other entities is marked as **V2X**, which stands for "Vehicle to everything", since it can also communicate with other components, apart from infrastructures or vehicles, such as pedestrians' devices or network (**V2P** and **V2N** respectively).
In conclusion, the opposite of V2I is **I2V**, in which there is a communication from the infrastructure to the vehicle, generally when there is a broadcast message to be sent to all vehicles from RSUs.

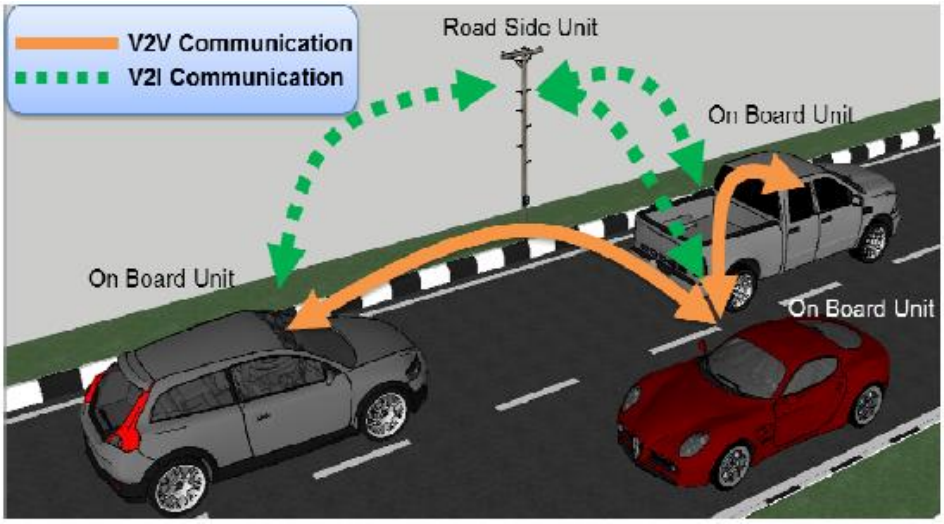

Figure 2.2. VANET Architecture [6].

After having discussed the way in which CAV can communicate with other devices, it follows a description of each element.

2.4.1 RSU: Road Side Unit

This component, the RSU, is generally described as a vehicular communication system. According to the *FCC* (Federal Communications Commission), it states:

*"[..] A Roadside Unit is a **DSRC** (Dedicated Short Range Communications) transceiver that is mounted along a road or pedestrian passageway. An RSU may also be mounted on a vehicle or is hand-carried, but it may only operate when the vehicle or hand-carried unit is stationary. Furthermore, an RSU operating under this part is restricted to the location where it is licensed to operate. However, portable or hand-held RSUs are permitted to operate where they do not interfere with a site-licensed operation. An RSU broadcasts data to OBUs (On Board Units)*

or exchanges data with OBUs in its communications zone. An RSU also provides channel assignments and operating instructions to OBUs in its communications zone, when required."

RSU includes a processor that runs the applications, data storage such as memory, and communications capabilities such as 4G/LTE or 5G and GPS receiver that support secure communications with passing vehicles, other field equipment, and centres. Each RSU consists of a large electrical cabinet that houses all the different modules of the RSU. These include modules for wireless communication, modules for local processing on the RSU, and modules for remote management (reduce field service operating costs.). The hardware and communication modules inside the RSU provides support for V2X radio links, such as short-range based on ITS-G5 and C-V2X with PC5 interface working on 5.9 GHz. Vehicles equipped with the V2X will be able to receive information from the RSU. The long-range communication is based on 4G. The modular design approach of the RSU has increased the serviceability as well as making it easy to add new capabilities to the box.

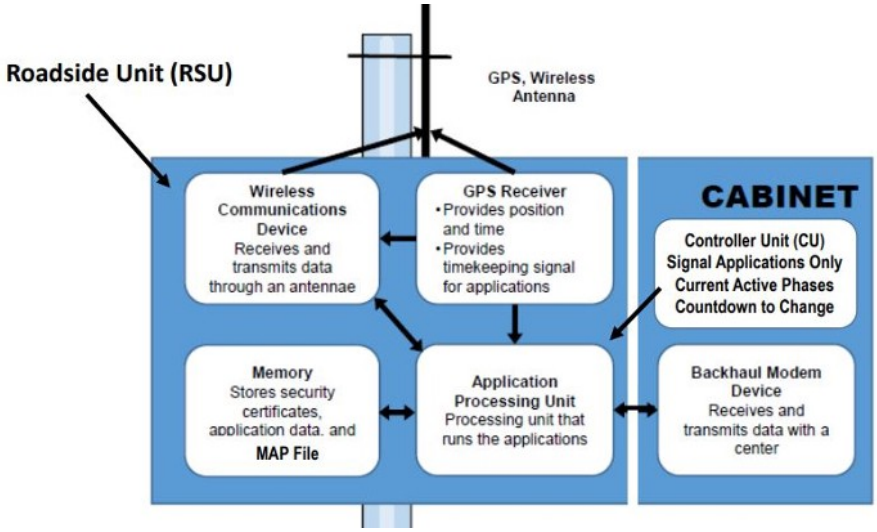

Figure 2.3. RSU Architecture [7].

2.4.2 OBU: On Board Unit

This kind of unit is the electronic device placed inside the vehicles, which is made of several parts. To be more specific:

- **Processor**: it is the computational part, in which all the operations are executed;
- **Global Positioning System (GPS)** : it is used for identifying the physical location acceleration and direction of movement of vehicle at specific interval of time;
- **Read/write memory**: since each OBU records messages, through this memory is possible to perform reading and writing operations;

- **Sensor nodes**: radars and sensors are used to detect obstacles that appears during movement of vehicle;
- **Event Data Recorder module (EDR)** : EDR is an electronic device and part of OBU. It stores all the transmitted and received messages to the nearby OBUs and RSUs. It also records all activities that happened in vehicle environment during the trip.

In general, OBUs are mounted on-board and exchange messages with others CAV (which mount OBUs) and RSUs. Furthermore, OBUs controls ad-hoc connection, routing, IP-based mobility management, data security issues and network congestion. A special purpose-computing device is attached with OBU. It is responsible for taking necessary action corresponding to messages received from other OBUs or RSUs. To identify a vehicle uniquely, an Electronic License Plate (ELP) is also associated with every vehicle.

2.4.3 AU: Application Unit

The AU is a device mounted inside the vehicle that is used with the application given by the provider in order to communicate with the OBU. This type of communication can be practically implemented through wired or wireless communication, so as to have a way for sending and receiving data. The distinction between AU and OBU is purely logical. Physically they are part of a single unit. Furthermore, the Application Unit is most of the time considered as the Graphical Interface used between the user and the OBU. The AU can be a dedicated device for safety applications or a normal device such as a personal digital assistant (PDA) to run the Internet.

2.4.4 TA: Trusted Authority

As stated by [8], this component is responsible for the trust and security of the overall VANET. It includes also the verifying of the authenticity of nodes and the revoking of vehicles in case of malicious behaviour. As a result, TA must have high computational resource and high storage capacity.

2.5 VANET communication

As seen in section 2.4, nodes inside VANET can communicate or one another or with fixed location device such as RSU. The exchanging of messages contains different type of information, from the temporary to the emergency ones.
Going into details, the communication inside this type of network is defined through the concept of **DSRC**. It is a standard and it stands for "Dedicated Short Range Communication". According to its definition, it is stated that:

Dedicated short-range communications (DSRC) are one-way or two-way short-range to medium-range wireless communication channels specifically designed for automotive use and a corresponding set of protocols and standards .[9]

From October 1999 up to now, it has been discussed the band to be used by this systems. After several years, it has been decided to allocate 75 MHz, divided into 45 MHz to the ISM (portions reserved for Industrial,Scientific and Medical purposes), and the remaining used by VANET. Both work in the band of frequencies that goes from 5.850 to 5.925 GHz. Furthermore, the DSRC band is also regulated by ETSI (European Telecommunications Standard Institute), using only the channels 180 of CCH (Control CHannel) and 172,174,176,178 of SCH (Synchronisation CHannel). [10]

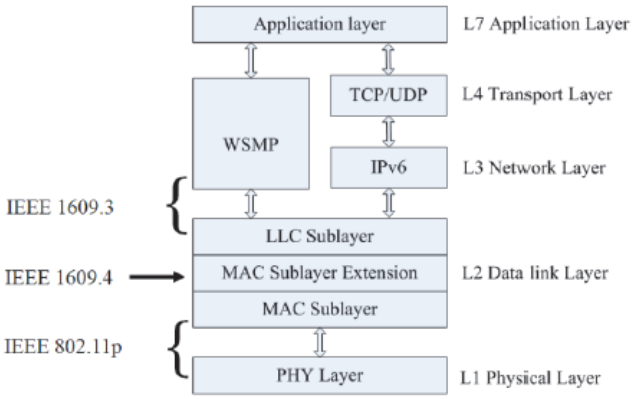

Figure 2.4. Protocols used in ISO/OSI stack [11] .

Taking into consideration the figure 2.4, it depicts the layered architecture for DSRC standard used for VANET. Comparing it with the OSI model, DSRC has an architecture made of five levels: physical, data link, network, transport, application. Focusing on the lowest, DSRC uses 802.11p for physical layer and data link, so as to manage better the medium access and the physical layer management . It specifies also the communication frequency, radiant power, data rate.
IEEE 802.11p is based on IEEE 802.11. It is a specialization of the latter, that has been designed for intelligent vehicles (CAV) so as to guarantee and facilitate the provision of fast and reliable wireless access in the VANET. It is obvious that 802.11p works in the band between 5.8 and 5.9 GHz.
In particular it is based on an orthogonal frequency-division multiplexing (OFDM) PHY layer but uses 10-MHz channels. As a result, data rates ranges from 3 to 27 Mb/s for each channel, where lower rates are often preferred in order to obtain robust communication.
In this way, IEEE 802.11p technology is targeted as the common technology used for traffic applications.
The upper levels use the IEEE 1609 protocol. It is used in part of the layer two and three. The union of IEEE 802.11p and IEEE 1609 protocol is denoted as **WAVE** (Wireless Access in Vehicular Environments). It is considered the standard for managing data that routes from one vehicle to another, or from one vehicle to a fixed unit (such as RSU). In other words, it is seen as the most promising technology for vehicular networks.

2.6 VANET architecture

After having defined the individual components and the various communication protocols, it is reasonable to define a reference architecture to better understand how they communicate one another and through what communication mechanisms.

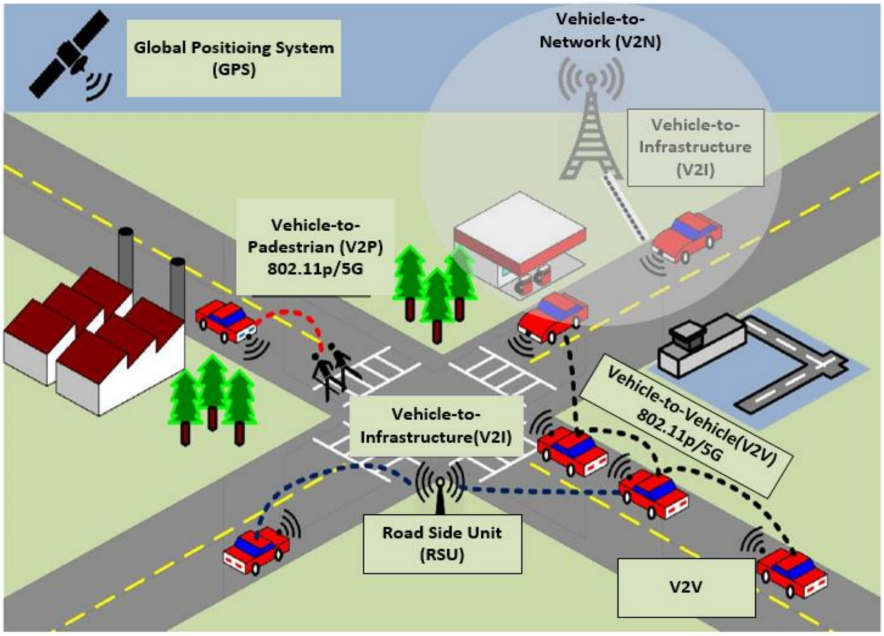

Figure 2.5. A general VANET architecture [12].

According to the figure 2.5, it follows a representation of how all the components previously mentioned are linked one another.
The principal actors are represented by CAVs and RSU. A general CAV is made of an OBU, AU and other components that contain sensors and control units. Internal components of a driverless car communicate through **CAN** (Controller Area Network) protocol, which is defined as a serial communication digital bus of "broadcast" type. It allows a real-time and distributed control with a high level of security. In other words, it is a way of letting all the CAV components talk.
Furthermore, it is a standard ISO (ISO. 11898). [13]
Each CAV talks with another device (pedestrian, fixed location unit, or other vehicle) through its OBU mounted inside the vehicle. The OBU notifies the user through the AU installed on the vehicle, in order to give to the user information of different nature. Meanwhile, RSUs collect data coming from other OBUs and in emergency case they broadcast emergency message to nearby nodes.
Each OBU, on their turn, communicate each other to be self-managed through the exchanging of information such as velocity, speed limits, traffic lights, stop signals, order of precedence, lane changing, weather information etc. All of it is possible through a 5G communication, that exploits WAVE protocol.

Chapter 3

Cybersecurity and VANET

The chapter 2 has described the CPS and its specialization, the VANET. These two concepts have many aspects through which is possible to perform analysis. One of them, without any doubt, is the security one. In general, and according to [14], since VANETs are a particularization of a CPS, its security aspects can be organized in two macro arguments: security involving CPS and the one that refers on VANET. Will follow a further description.

3.1 Security in CPS

Under this point of view, it is possible to describe the cybersecurity concept under three aspects. These are:

- **Cyber**: cyber components do not interact directly with the physical ones. They are intended to be the software part of the CPS. As a result, it includes computations, monitoring activities, communication processes;
- **Cyber-physical**: it is considered a sort of bridge between cyber and physical world. It involves all these cyber modules that interact with the physical modules;
- **Physical**: this view includes all the components that interact in a physical way with the physical world. An example of them can be sensors and actuators.

It is nowadays obvious and, in some situations, predictable to buy a new car equipped with all kinds of comfort. All these optional functionalities are wide and lead to significant improvements about the driving experience.
It is important, on the other hand, to understand if these are secure and if the cybersecurity milestones are respected. It must be considered if the overall aspects of cybersecurity are taken into consideration too.
These are only partial doubts that is possible to ask about the usage of this kind of components inside these systems.
It is reasonable, under this terms, to make a notable distinction between what is safe and what is secure. In fact, according to [15] :

- **Safety**: it is the set of measure and tools used to prevent or reduce accidental events that could injure people or could damage things;

- **Security**: it is the set of actions and tools in response to a running threat, coming from malicious action, organized with the aim to cause damage.

[14] states that is true that cars are safe by design, but most of the time the security is not considered during all design stages at system development time. Consequently it will be obvious to understand the fact that there will be a number of issues, denoting vulnerabilities of the system.
Having pointed out these two concepts, it is now time to describe something which is more important, the "security threat".
Specifically, this type of threat is defined as
"A potential for violation of security, which exists when there is an entity, circumstance, capability, action, or event that could cause harm." [16].
A part from all the components that make a CPS, one of the most critical asset to be taken into consideration and on which to focus more, without any doubt, is the people.
In fact, since the CPS components can be analyzed through security software and malware detection systems, the behaviour of people is, in many cases, unpredictable. As a result, they are cheated and some attacker can take the control of their device.
Before going on with the explanation of security factors that affect a CPS (and in general a VANET since the latter is a subset of the former), is necessary to define key concepts under security point of view, whatever they are. In details, these are:

- **Weakness**: result from poor coding practices and have the potential to result in software vulnerabilities; [17]
- **Vulnerability**: A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy; [16]
- **Exploit**: it is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug or vulnerability to cause unintended or unanticipated behaviour to occur on computer software, hardware, or something electronic (usually computerized).[18]

Each CPS component that could have a threat can be defined under five factors. These are: source, target, motive, attack vector, potential consequences.

1. **Source**: the threat source is effectively the initiator of the attack, and it can be divided into:
   (a) Accidental threat: threats that have been caused accidentally or through legitimate CPS components;
   (b) Adversary threat: which pose malicious intentions from individuals, groups organizations, or states/nations;
   (c) Environmental threat: which include natural disasters (floods, earthquakes), human-caused disasters (fires and explosions), and failures of supporting infrastructure (power outage or telecommunications loss).
2. **Target**: targets are CPS applications and their components or users on which an attack is conducted;

3. **Motive**: CPS attackers usually have one or more reasons to launch an attack: criminal, spying, terroristic, political, or cyberwar;

4. **Attack vector**: a threat might perform one type or more of four mechanisms for a successful attack. Formally it is known as "[...] a method or pathway used by a hacker to access or penetrate the target system."; [19]

5. **Potential consequences**: Compromising the CPS's C.I.A., privacy, or safety. It is known that the acronym CIA stands for **Confidentiality, Integrity and Availability** and it represents the milestone on which the cybersecurity is built. [14]

3.2 Security in VANET

Being VANETs subset of CPSs, the analysis performed at previous paragraph can be reflected on them, defining under three aspects vulnerabilities to which VANET could be subject to.

1. **Cyber vulnerabilities**: since smart cars allow a binding with a mobile phone in order to have more functionalities inside the vehicles, it has been observed that many security threats can be identified in attacks on mobile devices, by tracking GPS and user's microphone. As a result, it is simple to understand that the privacy of the user will decrease. This connection reveals a target's whereabouts, or can become a spying tool via eavesdropping on the in-car conversations by exploiting the microphone. Another problem can be referred to the usage of Bluetooth. In fact, when the user connects his device to the smart car through the Bluetooth, if there was Bluetooth vulnerabilities, it would be possible to exploit them to run attacks and to brute force the PIN used for the binding so as to have the complete control of the system. As a result, Bluetooth connections could expose the car to traceability attacks if an attacker successfully extracts the Bluetooth's media access control address, which is unique and traceable;

2. **Cyber-physical vulnerabilities**: under this aspect and according to literature, smart cars are vulnerable due to the lack of security considerations in their design. CAN protocol, has lack in encryption, authentication and authorization mechanism. As a result, these vulnerabilities contribute to most of the attacks on CAVs. Being CAN protocol a bridge between all CAV's components, it has been observed how it would be possible to perform eavesdropping of the communication through the TPMS(Tyre Pressure Monitoring System) in order to obtain its ID and trace the car. Furthermore, CAN protocol can be subject to DoS attack, because it does not manage well the errors. As a result, it is also missing the concept of non repudiation because, as previously said, it is a consequence of a DoS attack and it is not possible to identify the source of the message;

3. **Physical vulnerabilities**: it represents that kind of vulnerability in which, sometimes, it is not requested to have cybersecurity skills. Exposing the smart car to all type of physical access is another kind of vulnerability that

can cause critical attacks. For instance, a mechanic could exploit some part of the car to get access to the components of the CAN network.

3.2.1 VANET security requirements

Going into details, it is appropriate that VANET observes a series of security requirements, since several attacks can be conducted on it. Next to the CIA triad (Confidentiality, Integrity, Availability), there are other requirements that this kind of network must respect. To be more specific and referring to [20]:

- **Authentication**: it is a key requirement in the exchanging of messages. It ensure that sent messages are possible from legitimate nodes; so, unauthenticated nodes or adversary ones are not capable of sending the message. Among all types of authentication, the one referred to the Connected Autonomous Vehicles is the entity authentication, that is used in those situations in which the sender of the message is part of the network, granting the fact that the message has been sent in a short period of time;

- **Availability**: it ensures the the information is available to the users when it is required. Also in this case the dynamicity of the nodes foresee that the response time is very fast, resulting in the idea that every type of delay in the delivering of the message makes it worthless;

- **Message Integrity**: it states that the message has not been altered during the transmission and that it has been generated by the legitimate node;

- **Message Non-Repudiation**: it prohibits a sender from denying that he or she has not sent the message. However, everyone else cannot identify the sender rather than only authorities should be allowed to identify who has sent the message;

- **Access control**: authorization specifies what a node can do. It is required to enforced the access control which state that all nodes will function according to the roles and privileges assigned. For instance, a node is allowed to perform a function only if a node is authorized for that. In this way it is granted a level of security in the topology of the network;

- **Message Confidentiality**: the message transmitted must be confidential i.e. free from alteration by intruders. In order to grant confidentiality, ciphering and deciphering is used. Some nodes want to communicate secretly. However there are borderline cases, that is those situations in which no one other than the law enforcement authority cannot do that. An example could be the situation in which is requested to find the location of a criminal or a terrorist;

- **Privacy**: privacy ensures about unauthorized access of the private information of the driver. Location privacy assures that the past or future locations of vehicles cannot be determined in any case. Though, the various law enforcement authorities can trace user identities to determine criminal responsibilities;

- **Real time guarantees**: the message transmitted should have the hard deadline of delivery which is essentially required in safety related applications.

On the other hand, if it is taken into consideration what it is stated by [21], it is listed in general terms a series of possible vulnerability that affect the VANET. In fact, in the situation in which some of the requirements listed above were lacking, it would be observed a vulnerability of the following type:

- **Jamming**: an attack that attempts to interfere with the reception of broadcast communications. [16] It is achieved by interjecting electromagnetic waves on the same frequency; [22]

- **Forgery**: it affects the correctness, validity and reception of transmitted data. The compromising of one of these fields can cause chaos in the zone in which the VANET is located;

- **Impersonation**: any vehicle owner deliberately and hideously taking on the identity of another vehicle and attributing it to his own vehicle or vice-versa is known as impersonation. It also involves fake message fabrication, message alteration and message replay. For instance, an attacker appearing falsely as an emergency vehicle and misleading other vehicles to useless or harmful consequences is an impersonation attack;

- **Privacy**: the illegal monitoring of driver's personal data could violate their privacy. Attacks on driver privacy are a severe vulnerability in VANET due to the periodic and frequent nature of vehicular traffic. Driver's personal data can be retrieved by means of illegal in-transit traffic tampering of safety and traffic related messages sent by the driver, management messages, or even from transaction based communications such as automated payments. Such frauds and deceptive scams have been on an increase especially among networked devices as cyber criminals get an opportunity to send spurious messages to any device on the network.

A part from these type of requirements that are necessary in order to have a secure and well designed VANET, there could be situations in which tampering with the road side infrastructure, removing, dislocating or destroying them is another security issue in VANET. OBUs are tampered in a manner similar to that of modifying an odometer in earlier vehicles. Use of magnets, electric fields and malicious software to damage OBUs is a source of concern that needs to be addressed for safer and secure VANET communication. Although the OBUs could be subject to periodic examinations and inspections for any signs of tampering, limitations exist in relation to the frequency of inspection and the honesty of technicians performing the inspections. To ensure reliable and secure V2I communication, it is required that the roadside equipment is not damaged o purpose.

3.3 Real attack mechanisms on VANET

Previous paragraphs have described the concept of cybersecurity at higher levels. Now it is time to go into details about the real attacks that can be conducted.

Firstly it can be said that VANET, unfortunately, are susceptible to several attacks such as unauthorized access, the sending of fake messages, leaking of private information, eavesdropping and so on. In the security of VANET different entities are involved. In particular, and as stated by [23], these are:

- **The vehicles**: in VANET world they are considered as Connected Autonomous Vehicle and they represent all types of vehicles such as cars, buses etc;
- **The infrastructures**: it includes all the infrastructures involved in a VANET, i.e. RSU, traffic lights and so on;
- **The drivers**: since in future CAV will not have a driver, it is intended as the human that is inside the CAV and can be confused by the attackers;
- **Third parties**: it includes traffic police, transport regulators and so on;
- **The attackers**: the thing that conducts the attack, which can be of different types.

In particular, and in order to give a better and clear description about the last item, it is reasonable to identify the various types of attackers. According to [20], they can be categorized in:

- **Insider v/s Outsider** attackers: the authenticated members of network, known as Insiders, have full knowledge about the network and hence are very dangerous because this knowledge will be used for understanding the design and configuration of network. On the other hand, outsiders are the intruders and they have the limited competence to attack;
- **Active v/s Passive** attackers: active attackers are those who attempt to modify the network resources or disturb their normal operation. They either generate signals or packet with an intension of some alteration of the original data or the creation of a false stream. Passive attackers, on the other hand, only listen the network traffic to identify the information/pattern that is being transmitted. It is very difficult to detect passive attacks as compared to active attacks;
- **Malicious v/s rational** attackers: Malicious attackers just check the security mechanisms of the network and don't have any personal benefit; whereas rational attackers may have the personal profit that's why they attack on the network.

It is also important to point out the **coverage area**, which is the main area in which the attack is conducted. Generally, and according to [24], it is about 1000 meters that can be improved by exploiting the DSRC channels better. After having introduced the actors of a general attack, it is now time to describe their classes.

Attack class	Description	Attacks involved
Class 1	Network attack	Denial of Service Attack; Node impersonation attack; Black Hole attack; Sybil attack; Masquerading attack; Brute force attack; Distributed Denial of Service; GPS spoofing attack; Worm Hole attack;
Class 2	Application Attack	Bogus information attack; Safety application attack; Non safety application attack; Broadcast tampering attack; Illusion attack; Message alteration attack;
Class 3	Timing Attack	Peer to peer timing attack; Timing attack for authentication; Extended level timing attack;
Class 4	Social Attack	Social engineering attack;
Class 5	Monitoring Attack	Man in the middle attack; Traffic analysis attack;

Table 3.1. Table of attacks [23].

3.4 Classes of attacks

Attacks, as stated by [23] can be divided into five classes. The table 3.1 describes properly them. In particular:

1. **Network attack**:

   (a) **Denial of Service attack**: it is a kind of attack in which a malicious user sends a huge number of packets to the other nodes such as RSU or other OBUs. The aim for performing this attack is to stop the user in the usage of the network services. A concrete example of this type of attack is the *SYN flooding attack*.[25]
   In fact, a huge number of half-opened TCP connections are created between two vehicles in a VANET but never closed. Since TCP connection depends on three way handshake, firstly one connection is established and then the attacker floods the victim with a huge number of SYN messages to a specific remote station or a vehicle. The more received SYN messages by a vehicle (victim node), the more size required in its buffer to register these messages at its tables. Hence, a lot of resources' consumption has happened; its system may be out of service for a period of time;

   (b) **Distributed Denial of Service attack**: it works same as the DoS attack but it is done through a huge number of nodes;

(c) **Node impersonation attack**: this type of attack is the one in which the attacker impersonates another node, since each CAV has a specific ID for being identified. With it, the malicious node is able to impersonate another node and to act as a real user and showing that the message is originated by it. As a result, the attacker is able to alter the message;

(d) **Black Hole attack**: it is an insidious attack because the attacker claims itself to be the node that has the shortest path to the destination node. In this way, the victim will discard the other nodes. As a result, the attacker will receive all the messages and all of them will be lost;

(e) **Sybil attack**: this attack is performed in those situations in which the attacker wants to control multiple nodes. For this reason, the attacker creates fake identities that represent different nodes that will control;

(f) **Masquerading attack**: a node is able to masquerade its identity and it is able to act as another node such as the police vehicle. The result of it is the lost of trust from VANET;

(g) **Brute force attack**: it can be considered as a cryptographic attack. In fact, since the exchanging of messages is encrypted, a malicious node tries to intercept a message and brute forces it in order to read the plaintext;

(h) **GPS spoofing attack**: according to [26], in this type of attack attacker tries to change current geographic location identity and produce false information from GPS system by using such technique user is hiding his current position from the network and show the wrong position to others. This attack can be done by single vehicle or group of vehicles;

(i) **Worm Hole attack**: according to [27] this type of attack is done in a way in which two or more malicious nodes are placed in strategic places inside the network. One of them record part of its communication and it creates a direct tunnel with the other attacker. In this way, the latter will broadcast the messages the former attacker received. The result of this attack can mutate into a DoS attack;

(j) **Byzantine attack**: analysing what is stated by [25], this attack can be launched by a single node or by a group of nodes. The aim of it is to degrades the routing performance and to disrupt the routing service by creating routing loops by intermediate compromised nodes in order to forward routing packets in a long route instead of the optimal one, so as to drop packets in the worst scenario.

2. **Application attack**: it is a malicious attack since there is an alteration of the message in V2V communication. Suppose that CAV 'A' wants to say to CAV 'C' to change road due to traffic but there is another CAV 'B' and the message goes from 'A' to 'B' and from 'B' to 'C'. If 'B' is the attacker, the bad node is able to change the message into "the road is ok there is no traffic" and to forward it to 'C';

3. **Timing attack**: it represents those attacks in which there is a delay in the communication. In particular, supposing that the sender sends a warning

message at time t and the messages are sent and received each second, it will not be received at t+1 but probably at a time greater than one. The aim of this attack is to create delay, dangerous situations by increasing the transmission time of messages;

4. **Social attack**: the principal scope of this attack is to disturb and distract the user. The attacker can send messages to another node such as "you are an idiot" in order to distract it through ugly messages. The result of it can be identified in the behaviour of the receiver, who will be angry and not happy of receiving such messages;

5. **Monitoring attack**: these are types of attack in which there is a sort of control in the overall network. In particular, vehicles are tracked inside the VANET. Attackers will monitor the network through **man in the middle** and **traffic analysis attack**.

Since section 3.2 describes the vulnerabilities under three aspects (cyber, cyber-physical and physical), it is now reasonable to join that separation with the five classes of attacks, just mentioned.
In particular network attacks can be mostly included in the cyber-physical set, a part from Brute force and GPS spoofing attack, which are part of the cyber set.
On the other side, Application, Timing and Social attacks are completely part of the cyber set, because each attack works with the software part of the system.
In conclusion, Monitoring attacks are cyber-physical because there is an involvement of the cyber part and the physical ones.
As a result, this distinction will be useful at simulation time, because it will help in the choosing of the simulator to be used to conduct analysis.

3.5 Countermeasures on VANET attacks

Previous section has given an overview about the attacks that can be conducted on VANET. Them can / cannot have mitigations. The fact that there is not a countermeasure yet is the sign of the fact that is necessary to continue studying VANET world and to perform several simulations. It, associated with the difficulty of performing simulations (as will be explained in further chapters), can now give an idea of how much is difficult to analyse in practice the security aspect of VANET. However, and referring to an updated literature, it will follow a list of possible countermeasure.

- **DoS**: in Denial of Service attack the security property that is violated is, without any doubt, the availability. A possible preventive measure could be the identification of the IP information. In particular, and according to [28], they propose a solution in which they create an array X in which they store incoming and outgoing IP devices. In this way, this algorithm is able to detect duplicated IP addresses and to detect possible situations that can be similar to DoS attack;

- **DDoS**: in Distributed Denial of Service the situation is much more dangerous since its nature. For this reason, a possible countermeasure can deal with

statistical approach and the usage of IPS (Intrusion Prevention System). As described by [29], the sending and receiving of messages among CAV and RSU can be observed in order to define a normal distribution. As a result, is possible to compute the entropy of the system. If the DDoS is performed, the entropy and the normal distribution will not be the same as the normal situation. Thus, the IPS will notice this changing and will notify the network by dropping packets that are sent inside it. This implementation has pros and cons because it depends on where the IPS is mounted. In fact, supposing to have two IPSs (IPS1 and IPS2) in two different areas (A1 and A2), there could be situations in which the IPS1 is mounted in A1 with a huge exchanging of messages and IPS2 in A2 in which there is less exchanging of them. If they have the same entropy threshold, IPS1 could generate false positives since there are only more messages exchanged. So, it is necessary to have different thresholds since the number of packets sent and received is different;

- **Node Impersonation / Masquerading attack**: this attack affects the authenticity of a CAV. Possible preventive measures deals with the usage of a trust authority and PKI. In fact, through the usage of a Public Key Infrastructure, it provides unique digital identities for Connected Autonomous Vehicles;

- **Black/Worm Hole attack**: a way for counteracting this attack is, and according to [30], to connect the SIN (So-called Intelligent Nodes) to the network. They are special nodes that have access to the database. It is known that all nodes have public and private identifier (keys). Periodically, SINs send queries to all nodes inside the VANET, asking for their ID. Among the CAVs, a special RREQ (Route REQuest) message is broadcasted. Each RREQ keeps an ordered list of all nodes the message passed through. After receiving a response, the received data are compared with the sent one. When a RREQ message arrives at destination, a routing REPly (REP) is passed back to the origin, indicating that a route to the destination was found. If any of the nodes is not recognized during the backtrace of the REP, it is considered malicious by the SIN and it is removed from the communication. If the rest of the answers received are correct, the SIN does not take any further actions;

- **Sybil attack**: as proposed by [31], a possible solution can be found in the usage of keys. In fact, since each OBU has its public key, we can identify if the node is genuine or not, but attack can happen at any time. For this reason key management technique to identify attacked node at the time of routing using (Public Key Infrastructure) PKI is used. So, if the node has its unique public and private keys it is considered as genuine node or else attacked node and it is rejected. Since its OBU has a PKC (Public Key Certificate), an iterative check on the CA that signed the Certificate can be done, up to the root CA, which is trusted for definition. If one of these certificate is untrusted, the node is not considered genuine;

- **Brute force attack**: this attack can be prevented by using strong cipher algorithm. In this way, messages exchanged between nodes can be brute

forced in amount of times that overcome the years;

- **GPS spoofing attack**: different countermeasures have been developed but lots of them result in an increasing cost for the development of the devices to be installed on CAV and, for this reason, they have never been taken into consideration;

- **Man in the Middle attack**: literature offers a solution through the usage of the ANN, which is the Artificial Neural Network. [32] describes better the concept;

- **Application / Social attack**: since application attack deals with the exchanging and the manipulation of messages, a countermeasure can be found in the usage of strong hash and asymmetric encryption mechanism;

- **Attack on Authentication**: according to what is said by [33] among different authentication mechanisms, one of them is represented by the "Broadcast authentication". This type of technique is used in these circumstances in which is present a net with a huge number of nodes, such as inside VANET. This type of authentication is achieved by a public key signature. To ensure that a public key belongs to a node, reference is made to the PKI (public key infrastructure). In a PKI, the CAs sign the certificates, which is a kind of document that certifies that the public key belongs to the respective node. Referring to the IEEE 1609.2 protocol (used in VANET as explained in section 2.5), messages are authenticated by using ECDSA (Elliptic Curve Digital Signature Algorithm). Each message, by the way, has a certificate. It has been observed that using this type of mechanism is particularly expensive due to the hardware cost for verifying the digital signature of each incoming message.
  For that reason, it has been proposed a mechanism called TESLA (Timed Efficient Stream Loss-tolerant Authentication). With this type of mechanism, the CAV sender sends the message using a symmetric signature algorithm, and then broadcasts it with the signature. After a short period of time, the sender sends the key and it is used only for that message, and it cannot be used in future. Receivers cache the original message until the key is received and then verify the signature. Furthermore it has been demonstrated that, by using symmetric primitive, it requires approximately computational resources 1000 times less than the ECDSA.

Up to now, literature offers a huge number of countermeasure that can be implemented in the world of VANET. Unfortunately, lots of them are only developed theoretically and never implemented since VANET is an emerging network. Furthermore, many countermeasures lead to the advantage of making the system more secure but, on the other hand, there are many problems in the fact that security implementation makes systems heavier. As a result, lots of study must be conducted to avoid this type of problem too.

3.6 Real use-case attacks on cars

This section describes the attacks that have been conducted in real life in order to hack several cars. Different car manufacturers, unfortunately, are listed; it shows how cybersecurity is a pillar in development of cars.

3.6.1 Jeep Cherokee

In 2015 has been announced a vulnerability present on Jeep Cherokees, manufactured in 2014. The attack has been possible by exploiting a breach present in the **infotainment Uconnect system**.
This module is the equivalent of the AU (Application Unit) previously described, which let the user interact with the car. In fact, the word "*infotainment*" is the union of words "information" and "entertainment". To further analyse what has been done by attackers, they have been able to exploit a weakness in the cellular network in order to control:

- Jeep's communication;
- breaks and throttle.

All of it has been possible by enabling or disabling the cellular network remotely. Furthermore, it has been shown the possibility to attack the steering wheel when in reverse. They have been able to keep track of GPS coordinates and of the speed of the car, by indicating their position on the map too.
Through the IP address of the car, it has been possible to realize it. FCA (Fiat Chrysler Automobiles) diffused an ad hoc note regarding the update of the car's software, by specifying that the fix could also be installed through a USB pen.[34]

3.6.2 Volkswagen and Audi

Even in this case, as in the previous one, in 2017 it has been announced the founding of vulnerabilities on the infotainment system of cars. In particular, the car manufacturer which is involved is the Volkswagen (and Audi, which is a brand of it); the models are Golf GTE and Audi A3 Sportback e-tron.
It has been demonstrated that these cars are vulnerable to **remote hacking**.
The attack allows the access to the IVI (In-Vehicle Infotainment) system of the car, manufactured by electronic vendor Harman. Furthermore it has been stated that it was possible to go inside the system by gaining the **root's access** and, as a result, to have the capability to manipulate all data.
So, it has been possible to execute every type of operation, such as enabling or disabling the microphone or to have access to the contacts list and to the conversation history. Again, hackers have been able to know the GPS position of the car and to follow it in real-time, so as to have the control of the acceleration and brake system.
It is interesting to point out that in August 2016, Volkswagen fixed another major security flaw in its key fob system that affected almost all models sold in the past 20 years.[35]

3.6.3 Toyota Prius

Toyota car manufacturer has been involved in cybersecurity attacks in 2013 too. Attackers have taken the control of some important functions of vehicles. In details: malware is installed by inserting a "device" under the steering wheel. Once gained control, it is possible to control brake, horn and headlamps system. Differently from the previous attacks, in this case it is possible to take the control of the Toyota only by installing physically the device.
To be more specific, the "*device*" is a laptop scan tool mounted to the OBD (On Board Diagnostic) port and it is used for overriding certain functions.
Obviously all of it has been possible when the device is connected, otherwise the car would have worked normally.[36]

3.6.4 Mercedes-Benz

Security researchers from the sky-go team found more than a dozen vulnerabilities in the E-class Mercedes Benz. This attack allowed them to open **remotely** doors and control the engine. Hooking up a car on the internet puts it at risk of remote attacks because they are visible in the network and therefore recognizable by all. Moreover, this model of car is modern, also because it provides navigation infotainment and GPS, as well as several radio stations.
The final result was a series of vulnerabilities that formed an **attack chain** that could remotely control the vehicle. The sky-go team thoroughly analysed the TCU (Telematic Control Unit), one of the most important components of the car that allows the device to communicate with the internet. Tampering with the TCU's file system, the researchers had access to the **root shell**, so as to perform several operations. The TCU file system stored car's secret, such as passwords and certificates. It's been shown that these were easily removable and therefore having access to these, it was possible to manipulate cars.
The attack was possible by tearing down the embedded SIM card of the vehicle and by modifying the router to falsify the car. With the vehicle's firmware dumped and other parts of the system cracked, researchers said they could remotely control an affected vehicle. [37]

Chapter 4

Analysis of the state of the art of VANET simulators

4.1 Introduction to simulators

Chapter 2 and 3 have given a theoretical description of the overall elements useful to the usage of a simulator. In fact, simulators are considered valid tools in order to execute testing operations on VANETs at low cost and without risking the users. Obviously, simulators must represent as much as possible the reality, thanks to continuous update and to the adding of new elements. For this thesis' purpose, these will be used taking into consideration the fundamental aspect of cybersecurity, so as to be more useful and convey credible results.
On the other hand, it must be said that simulators do not evolve hand in hand with respect to the reality. To summarize:

1. many simulators are **outdated**, so not maintained anymore;
2. many simulators are **commercial**. Few of them are the open-source ones. As a consequence, simulations have a cost, because the simulator must be bought. Luckily, among all of them, Veins represents the simulator with the best support for novel technologies, being open-source too;
3. simulators **lack of support for** realistic models, for instance the ones for modelling faulty nodes (i.e. unreliable RSU).

For this reason, analysis on that type of simulators is not as easy as it could be.
Following paragraphs will give a further and detailed description of the simulators that will be used in this thesis.

4.2 Simulators

Developing and testing VANETs require an intensive labor since simulations deal with complex scenarios, each of them requires huge resources to be used at the same time. For this reason everything is substituted by simulation, which are useful and less expensive. So, it is reasonable to generate accurate models, so as to obtain good result to analyse.
In details, simulators are divided into two categories:

1. **Mobility simulator**: these simulators allow the definition of a model which reflects the real behaviour of vehicles in the traffic. They are used for defining the pattern in which CAVs move in the traffic, under proper conditions and following specific route. Mobility simulator, on its turn, are divided into:

   - Macro-mobility: it considers all these aspects at high level, such as the topology of the net, movement of cars, speed limits, traffic signs that govern the crossing rules at intersections;
   - Micro-mobility: it considers individual drivers behaviours in those situations in which they interact with the others such as: acceleration and deceleration, overtaking criteria, driving attitude.

   It is obvious that VANET simulation considers both micro and macro mobility. Examples of these type are SUMO, VISSIM, SimMobility, PARAMICS, and CORSIM.

2. **Network simulator**: in this case, it is intended that type of simulation in which there is an exchanging of messages between nodes. In VANET's case it concerns the exchanging of messages between vehicles and devices such as RSU, through wireless communication. In these simulations are identified main components and the events that occur among them. The latter incorporate those circumstances in which there is the transmission of data and packet errors.
   The output of this simulation deals with network level metrics, link metrics, and device metrics. It is also analysed cases in which "pending events" are considered, i.e. the possibility in which an event generate the sending of another message. Examples of network simulators available (some of them widely used in VANETs) include OMNeT++, OPNET, JiST/SWANS, NS3, and NS2.

4.2.1 Mobility simulator: SUMO

SUMO stands for Simulator of Urban MObility. It is one of the most known mobility simulator. It has been developed by the German Aerospace Center. It is free and available since 2001 and since 2017 it is part of Eclipse Project foundation. It is divided in different modules, in order to cover a specific area for the network to be simulated. According to [38]:
"*Traffic simulation within SUMO uses software tools for simulation and analysis of road traffic and traffic management systems. New traffic strategies can be implemented via a simulation for analysis before they are used in real-world situations. SUMO has also been proposed as a toolchain component for the development and validation of automated driving functions via various X-in-the-Loop and digital twin approaches.*"
What is essential for understanding its usage is to have the network topology file. It is the starting point. From that, the routes can be defined in another file. In other words, a SUMO project will have:

- file.net.xml: it will contain the network topology;

- file.rou.xml: it will contain the routes involved in the network;
- file.add.xml: inside this file there will be additional information about the network.

4.2.2 Network simulator: OMNeT++

On the other side, OMNeT++ (or omnetpp) is the most known network simulator. According to [39], it is stated that:
"OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. "Network" is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queueing networks, and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modelling, photonic networks, etc., is provided by model frameworks, developed as independent projects. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools."
The components of it are:

- simulation kernel library (C++);
- the NED (NEtwork Description) topology description language;
- simulation IDE based on the Eclipse platform;
- interactive simulation runtime GUI (Qtenv);
- command-line interface for simulation execution (Cmdenv);
- utilities (makefile creation tool, etc.);
- documentation.

4.3 VANET simulators

Going into details, VANET simulators are the union of mobility and network simulators. The latter are responsible for modelling communication protocols and the exchange of messages between nodes; on the other hand, the former, controls the movement of each node.
As can be inferred from the table 4.1, each simulator has its own licence, which can be "proprietary" or "open-source". In other words, the usage of the first category foresees that the software has been bought by someone. On the contrary the latter can be used without any purchase. For the purpose of this thesis, it has been chosen to focus on Veins and Eclipse MOSAIC, which have the latest release in the open-source category.

Simulator	Last release	License	Network Simulator	Mobility Simulator
NetSim	2021	proprietary	own	SUMO
Veins	2020	open-source	OMNET++	SUMO
Eclipse MOSAIC	2020	open-source	NS-3, OMNET++, SND and Eclipse MOSAIC cell	SUMO and VISSIM
EstiNet	2020	proprietary	own	own
exCar2X	2020	proprietary	NS-3	SUMO
VENTOS	2018	open-source	OMNET++	SUMO
VANETsim	2017	open-source	own	own
GrooveNet	2013	open-source	NS-2	own
VNS	2012	open-source	NS-3, OMNET++	own
iTETRIS	2010	open-source	NS-3	SUMO
NCTUns	2010	proprietary	NS-2	own
CityMob	2009	open-source	NS-2	own
TraNS	2009	open-source	NS-2	SUMO
FreeSim	2008	open-source	NS-3	own
STRAW	2007	open-source	JiST/SWAN	own
VanetMobiSim	2007	open-source	NS-2	CanuMobiSim

Table 4.1. VANET simulators. [40]

4.3.1 Veins

Veins, among all the simulators, is an open source framework for simulating vehicular network. It is based on SUMO and OMNeT++. According to the figure 4.1, all the modules that make the Veins architecture are shown. Firstly, it is instantiated an OMNeT++ node for each CAV present in the simulation and then it is paired with its movement in the road traffic simulator (SUMO). As can be inferred, both network and mobility simulations can run in parallel. This is possible because of a bidirectional coupling achieved by a standardized connection protocol, the Traffic Control Interface (TraCI).
It allows the exchanging of messaged between OMNeT++ and SUMO while the simulation runs, as part of the TCP connections.
Veins has different extensions (more than 17) that allow the usage of different protocol stacks (such as IEEE 802.11p).
In summary, Veins is designed to serve as an execution environment for user-written programs, which facilitates modelling new environments and applications. As a disadvantage, it needs both SUMO and OMNeT++ to run correctly in order to obtain precise results. Any bug in one of those can cause Veins to give unreliable results. Veins can run on Linux,Windows, and Mac OS. The figure represents the overall architecture used by Veins, with its module and how they communicate.

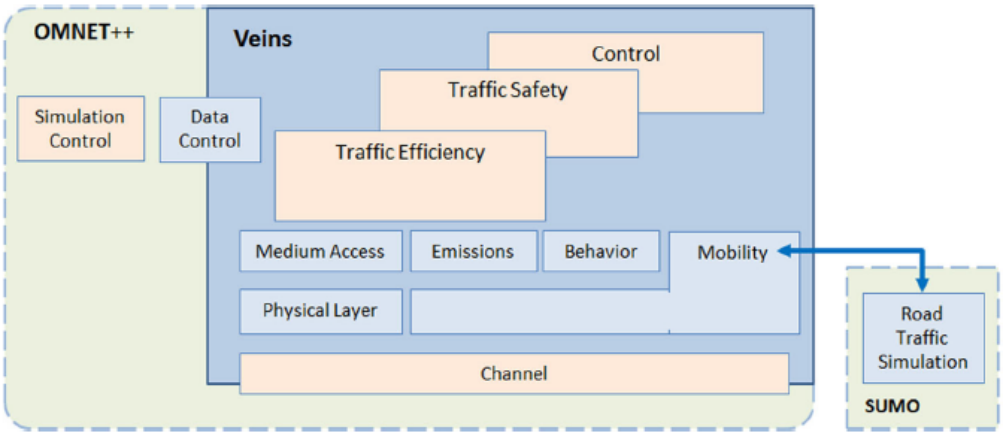

Figure 4.1. Veins architecture [40].

4.3.2 Eclipse MOSAIC

Eclipse MOSAIC is another simulation framework, open-source, known as a V2X Simulation Runtime Infrastructure. Its aim is to give to the users the flexibility to perform different simulations through different simulators. In fact, in order to guarantee that, Eclipse MOSAIC uses different simulation framework.
For example, for traffic simulation is supported the usage of SUMO and PHABMACS; for communication simulation OMNeT++ and SNS; and Eclipse MOSAIC Application for application simulation.
Since Veins works with SUMO and OMNeT++, it has been decided to use them for MOSAIC too. As a result, it can be deduced that it is a multi-scale and multi-domain simulator framework. According to the figure, the Eclipse MOSAIC

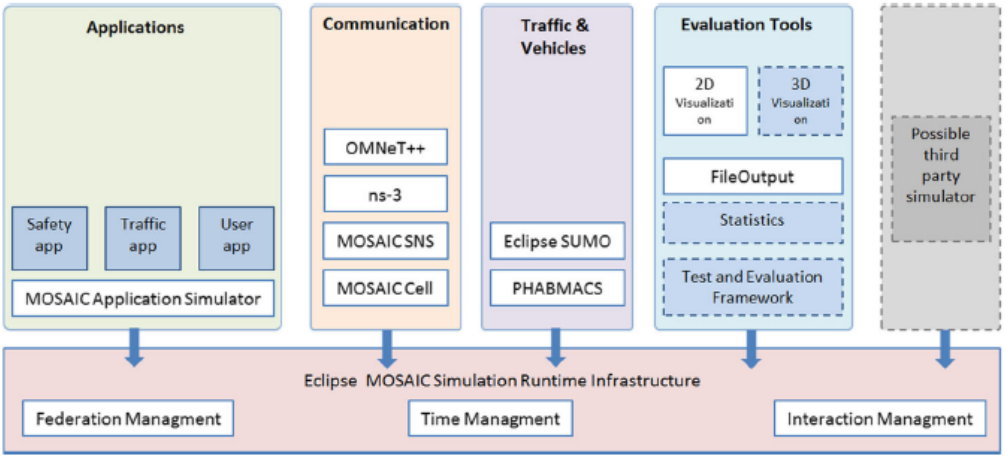

Figure 4.2. MOSAIC architecture [40].

Simulation Runtime Infrastructure is made of three core elements.
The Federation Management has the duty to couple each simulator with the runtime infrastructure. To give a further description, a federate consists of a simulator and two connectors, one to receive data from the runtime infrastructure and the other one to send data to it.
The Time Management coordinates the simulation and synchronyzes the federates that participate to the simulation by processing information in a correct order.
The Interaction Management enables the exchange of data among federates through the publish-subscribe pattern.
Since Eclipse MOSAIC is multi-domain and multi-scale, data can be visualized in several ways. In other words, the same scenario can be evaluated in different visualization tools, which are connected to a running simulation. Some of them are WebSocket Visualizer, Integrated Test and Evaluation Framework (IETF), and PHABMap for 3D visualization.

4.4 Simulators' functionalities

Technology	**Veins**	**Eclipse MOSAIC**
Software Defined Networking (SDN)	X	
Edge computing	X	
5G	X	X
Self-driving cars	X	X
Unmanned aerial vehicles(UAV)		

Table 4.2. Simulators' functionalities. [40]

The table 4.2 represents a general overview about the functionalities of different simulators. As can be seen, not all of them are capable of having all the functionalities required. Taking into consideration the ones chosen for this thesis, Veins supports all of them whereas MOSAIC does not. As a result, this is a proof of what stated before, that is the difficulty to perform simulations since many simulators lack these functionalities.
For example, the SDN (Software Defined Networking) is a technology for dealing with a lot of nodes. It is the case for vehicular networks. In fact, in this case, SDN has become SDVN (Software Defined Vehicular Networking). As can be seen, SDN is present in Veins but not in MOSAIC: it means that is difficult to perform analysis in particular situations such as traffic with the latter simulator. On the other hand, self-driving cars and 5G are present in both simulators. Among Veins and Eclipse MOSAIC, the focus must be given on the former, because it is the tool that will be used to simulate attacks on chapter 5. Despite all these functionalities, several limitations will be pointed.
Next paragraph will focus on the security aspects regarding simulators.

4.5 Security functionalities

Focusing on this aspect, VANET simulators must provide support for testing safety, security and other issues such as privacy (with associated countermeasures, where possible). The following subsections will describe these concepts.

4.5.1 Safety

As can be inferred, VANET development and its continuous study is made to improve road traffic safety. It can be increased if nodes communicate correctly, sharing information about position, speed, direction, aquaplaning, accidents, etc. However, collisions and accidents can occur when safety messages are not correctly broadcasted through the network. Analyzing what is said by [41], the reliability of a VANET depends on the reliability of each end nodes and of their communications. Failure of nodes can be caused by many reasons, from damaged sensors to general malfunctioning at hardware or software level. They are also subject to Byzantine faults (3.4). Considering the table, it summarizes the current support of VANET simulators to different types of faults. As can be seen, simulators do not

Vanet component	**Fault Type**	**Example**	**Veins**	**Eclipse MOSAIC**
Node (OBU,RSU)	Software	Malformed packet		
		OS bug		
	Hardware	Power outage		
		Malfunctioning antenna		
Link	Byzantine interference	False messages	X	
		Congestion	X	
		Ground reflection		
	Attenuation	Building shadowing		
		Vehicle moving away	X	

Table 4.3. Simulators safety. [40]

provide any features for simulating faults, but some functionalities can be adapted to recognize safety issues. Unluckily, since there is not enough support for safety, it requires the generation of large amount of code.
Observing the functionalities proposed by Veins, it can be inferred that it is able to simulate Byzantine faults through the F2MD, which is the Framework for Misbehaviour Detection. It provides a solution for simulating malfunctioning nodes that result in the generation of wrong information, such as velocity, acceleration etc. It has also two modules, used for capturing the effect of buildings and vehicles on the quality of data transmissions. They are Obstacle Shadowing and Vehicle Obstacle Shadowing.
On the other hand, Eclipse MOSAIC does not have support for fault injection. As a result, this is a proof of what has been stated previously, that is the difficulty of testing all security properties of a VANET system.

4.5.2 Security

This section includes the security aspects described in the previous chapter. VANET security depends on the security of messages exchanged. As a result, there must not be modification on the messages delivered and received.
Although the usage of simulators cannot solve these issues, they could be used to analyse security aspects in order to find possible solutions. Among the set of simulators, security properties and privacy will be considered on Veins and Eclipse MOSAIC. Table 4.4 shows different security mechanisms and their relationship to

Attribute	**Example mechanism**	**Veins**	**Eclipse MOSAIC**
Confidentiality	Symmetric and asymmetric cryptography	X	X
Integrity	Message Authentication code		
	Digital signature	X	X
Availability	Watchdog		
	Redundancy	X	X
Authentication	Position verification		
Analyze signal strength			
Dual authentication		X	X
Non-repudiation	ID-Based cryptosystem	X	
Privacy	Silent periods		
Mix zones		X	
Periodical pseudonym change			

Table 4.4. Security and privacy in simulators [40].

VANET simulators that can be *included* when running a simulation. Again, it is shown that Veins has some mechanism that MOSAIC does not have, highlighting the difficulty of simulations. A part from the difference in the presence of mechanism, it is important to emphasize a fundamental point. Considering Veins, security mechanisms that have been labeled "X" in the table are not to be considered as implicitly present within the tool; instead, these are features that Veins allows the user to employ (e.g., through third-party libraries or through coding). It is essential to emphasize this aspect because this thesis will demonstrate the fact that these features will not be present inside the tool and will make difficult the implementation and simulation of attack, even in the latest version of Veins.

1. **Confidentiality**. Support for confidentiality can be included by both Veins and Eclipse MOSAIC. It is not directly implemented but through the usage of third cryptographic libraries such as Crypto++, installed and used by the network simulator part, OMNeT++. Unfortunately, several bugs are present due to some incompatibility according to their different versions;

2. **Integrity**. Similarly to confidentiality, Veins and Eclipse MOSAIC do not

have native support for integrity services, but allow them to be included as third-party libraries or by manually coding, wasting lot of time;

3. **Availability**. Veins and Eclipse MOSAIC let the programmer implement a watchdog on the network simulator. In details, the watchdog define a set of nodes as monitor ones and they monitor the neighbourhood;

4. **Authentication**. Among Veins and Eclipse MOSAIC, the former implements this functionality through the ID-Based mechanism. The ID can be the email address, network address, username or another combination;

5. **Non-repudiation**. In order to provide this functionality, digital signature can be used. For this reason, this mechanism is present in Veins and not in Eclipse MOSAIC. To be more specific, it is implemented internally in the network simulator, OMNeT++;

6. **Privacy**. Privacy is one of the most important requirements for VANET security. In particular, Veins has a privacy extension called PREXT, which has a set of privacy schemes that can be attached to the development of simulation. It supports the situation in which an adversary want to track vehicles by eavesdropping messages. In literature, it has been shown that the simulation is 30% slower when running this extension.
   Eclipse MOSAIC, on the other side, does not support privacy services and, as a result, makes it difficult to study this fundamental property.

Chapter 5

Simulation framework: implementation of attacks

This chapter defines the implementation part of the cybersecurity attacks that can be conducted on VANET. These simulations will be performed through the usage of the tool that has been cited in chapter 4: Veins.
To be more specific and according to what stated previously, it is relevant to observe that the entire set of cybersecurity attacks can be **partially** simulated and subsequently analysed. In fact, simulators such as Veins do implicitly have settings that do not allow the implementation of some of them. As a result, the behaviour of these simulators represents a limitation and a challenge in the analysis of cybersecurity in VANET. Chapter 6 will discuss in details these limitations and how and when is possible to avoid them in order to have a "complete" simulation.

5.1 Simulation configuration

The environment used for the execution of simulations consisted of:

1. **Veins 6.0 OMNeT++ IDE simulator**. It represents the Integrated Development Environment used for developing and running the simulations. It is based on Eclipse IDE [42], which has been specifically designed and extended for VANET;

2. **OMNeT++ 5.7**. It represents the framework used internally to Veins 6.0 and it consists of a library used for simulating networks (in particular Vehicular Ad hoc NETwork);

3. **Inet framework 4.4**. This framework has been manually inserted inside the IDE and it is an open-source OMNeT++ model suite for wired, wireless and mobile networks.
   With it, it is possible to define models that are able to communicate wirelessly so as to simulate moving nodes (i.e., CAVs). The framework library has been built in order to be used;

4. **SimuLTE 1.2.0**. According to what stated by [43], it is a simulation tool for OMNeT++ projects, written in C++ that models the data plane of the LTE Radio Access Network;

5. **Veins_INET framework**. It is included in the IDE. Since Inet framework allows the modelling of nodes that communicate at wireless level, Veins_Inet has been designed with the aim to join this type of communication with the VANET world, so as to recreate the type of exchange of messages peculiar to this network.
   Consequently, it has been implicitly defined the protocols used by this architecture (explained in section 2.5). Also in this case, the library of this framework has been built before its usage;

6. **SUMO 1.14.1**. As mentioned before (see section 4.2.1), it allows the user to show at graphical level the designed simulation;

7. **Oracle VM VirtualBox 6.1.36**. It is the software used for the execution of virtual machines. It creates a virtual environment and give to the user the possibility to deploy an operating system. In this case, it has been loaded **Ubuntu 20.04** since all software run better under this OS.

5.2 Veins settings

The code developed inside Veins IDE is in a set of file, which follow the predefined path for this type of project. In detail, its configuration consists of:

- **Omnetpp.ini** : it is the initialization file which contains all references to files, such as the map, the route, the file containing codes with extension ".cc". It contains also general information about the duration of the simulation and the configuration of the IEEE 802.11p standard;

- **Package.ned**: it lists the package used for the entire simulation;

- **Scenario.ned**: it represents the scenario in which are present all components and it is made of a series of submodules, such as the CAVs of types "VeinsInetCar" and the Road Side Units of type "RSU". Depending on what is the current simulation, the scenario could be slightly different;

- ***.xml**: they are a set of files in which there are the descriptions about route, network and obstacles, created with SUMO tools;

- **VeinsApplication.cc**: it is the core of the simulation, in which the C++ code models the behaviour of cars and infrastructures. It is also the place in which the attacks are implemented. Will follow further description in the subsequent sections.

5.3 Network configuration

When a simulation is started, it is necessary and obviously important to observe graphically what is going on. Veins satisfies it through its internal tool called **Qtenv**. At runtime, it executes what written inside the Veins project and shows the result to the user. Unfortunately this tool does not offer the typical features of a network simulator. Because of that, it is used SUMO which, thanks to the

binding through a port, it receives all the parameters written inside the project and allows a correct visualization.
Practically, the binding is performed by typing the following line of code from the terminal, which is

```
./sumo-launchd.py -vv -c /usr/bin/sumo-gui
```

and, in this way, there will be a listener on port 9999 which will receive the simulation parameters from Veins and will prompt it on SUMO.

5.3.1 Map generation

The chosen environment for some attack recalls a piece of the map of Turin city. It has been precisely chosen this type of network in order to observe how a VANET behaves in the context of a real urban environment, so as to analyse how the VANET is damaged in case of cybersecurity attacks from malicious nodes.
Starting from the map in Figure 5.1, through the tool **OpenStreetMap**[44] a file called "map.osm" has been generated. The extension ".osm" stands for OpenStreetMap, and this tool let the programmer download a piece of map in order to work with it.

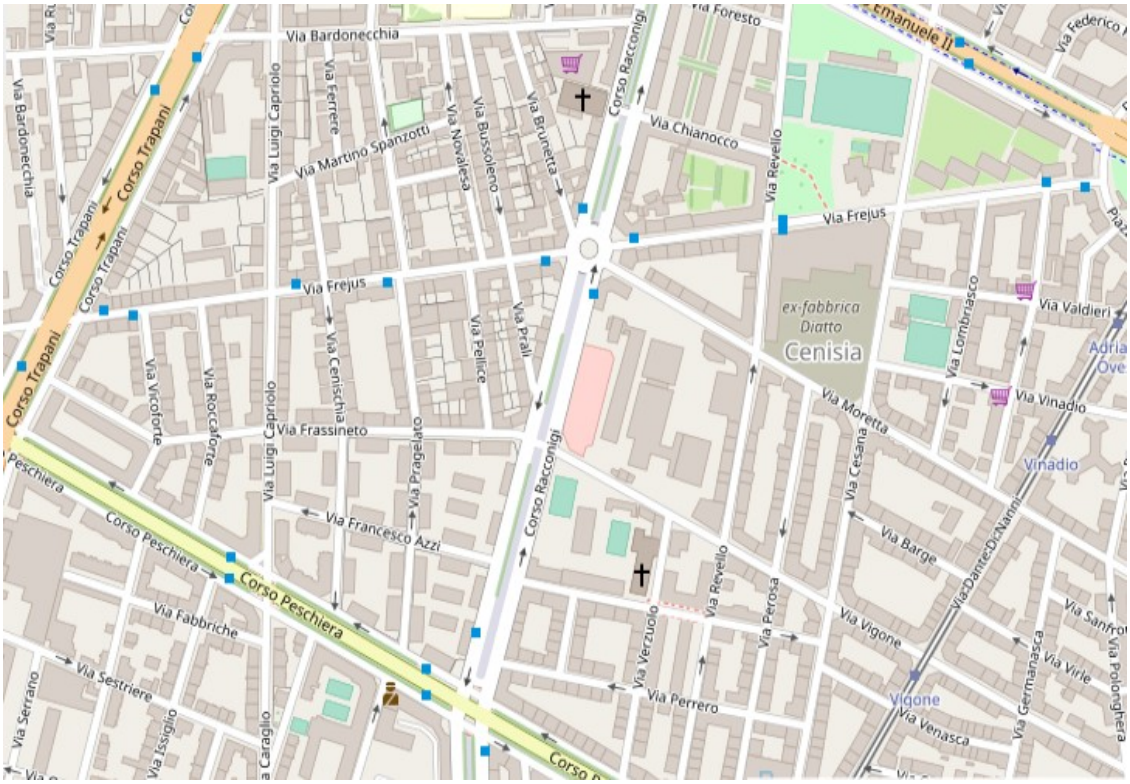

Figure 5.1. Real map chosen for the simulation. Snapshot of Turin city.

After the generation of this file, it has been executed a command from terminal in order to obtain the equivalent .xml file, by specifying a set of parameters so as to simplify the visualization. The command is

```
netconvert --osm-files map.osm --output-file map.net.xml
```

```
– –geometry.remove – –roundabouts.guess – –ramps.guess
– –junctions.join – –tls.guess.signals – –tls.discard–simple
– –tls.join
```

where the input file is the file called "map.osm" and the output file is "map.net.xml". Once obtained the network file, it has been run the program called "**randomTrips.py**". It is a SUMO tools used for the generation of possible trips inside the network, which is exploited by vehicles inside the map. Again, the command given from terminal is

```
/usr/share/sumo/tools/randomTrips.py –n map.net.xml –e
250 map.trips.xml
```

and in this case, the input file is "map.net.xml" and the output one is "map.trips.xml". It is specified the number of possible trips, which has been chosen to be equivalent to 250. This number depends on how bigger is the network to be simulated. In fact, the bigger the network, the greater the number written inside the command so as to generate more trips and to study the network better.
At the end, it has been run the program **DUAROUTER** (another SUMO tool) for the routes generation inside the map. In this way each vehicle will be automatically generated and will have its route to follow. The command is the following:

```
duarouter –n map.net.xml – –route.file map.trips.xml –o
map.rou.xml – –ignore–errors
```

In conclusion, the figure below represents the resulting map that will be used for the execution of the simulation.

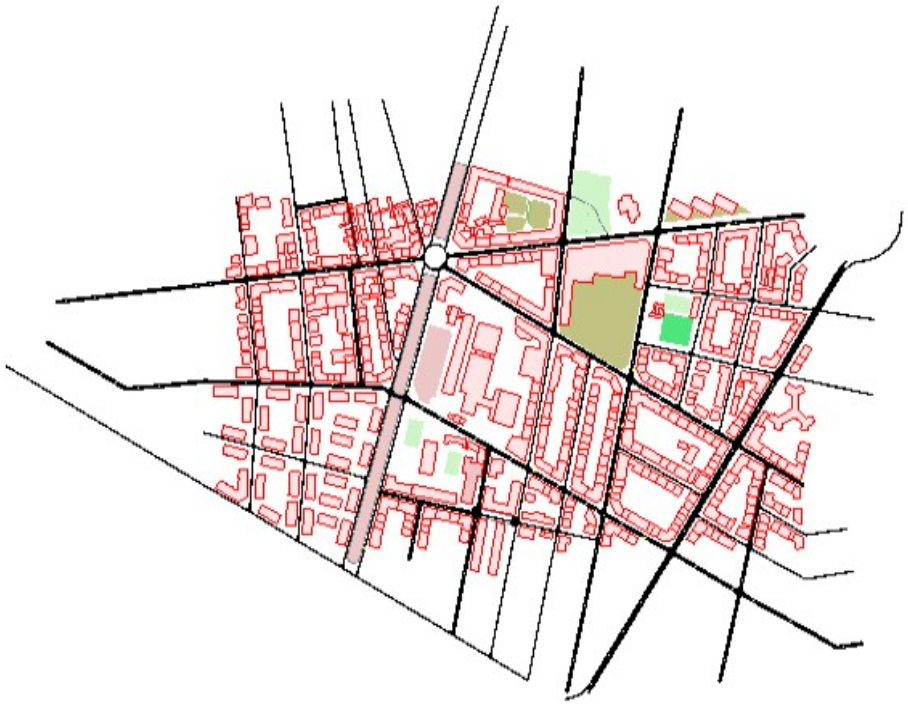

Figure 5.2. Map obtained for the simulation.

5.4 Implementation of attacks

The following section implements attack simulations in a realistic environment, which is the city of Turin, to outline how a hacker attack on a self-driving vehicle could cause disruption and inconvenience to the urban network and to the vehicles, and specifically to a VANET. Different attack scenarios will be implemented, modulated by the writing of code in C++.
Starting from the scenario described above (figure 5.1), these simulations have been conducted on a network consisting of 67 vehicles. In some situation, since the logic of the attack is different, the same scenario is considered but with a zoom on the communication with vehicles.
The following table will contain a reference to the attacks that have been implemented.

Type	**Description**
Message Alteration attack	The attacker is able to send false information and to confuse CAVs inside the network.
Denial of Service attack	The attacker breaks the Availability of the targeted victim by sending lots of packets infew seconds.
Distributed Denial of Service attack	The attacker simultaneously send thousands of packets to the victim.
Timing attack	The attacker is able to delay the sending of the message.
Social Engineering attack	The attacker sends useless message to alter the behaviour of the victim.
Black Hole attack	The attacker claims to have the shortest path to the destination and drops the packet instead of forwarding it to the receiver.
Jeep Cherokee attack	The attacker is able to control the car and to execute several operations.

Table 5.1. Implemented attacks.

5.4.1 Message Alteration attack

In this specific case,a particular type of attack has been implemented that falls into the five classes defined at section 3.4. Specifically, it is known as "application attack", that is a type of attack in which false information is sent to the receiving node, thus creating some sort of confusion within the CAV and the network itself. The scenario consists of 67 vehicles and one of them is the attacker ("**node[11]**").

Simulation

This attack very often is carried out to destabilize the *continuity* of the nodes inside the network and, at the same time, the *entropy* of it. In fact, the victim is forced

to recalculate its route to the destination because of false information received (most of the time) from the node in front of it, which almost always represents the attacker.
There are two actors involved in this attack, "**node[11]**" and "**node[3]**" which represent respectively the threat actor and the victim. For simplicity, they are recognized by colour inside the simulation. The attacker has the car icon red whereas the victim has the same icon but in green. The attack has followed three steps. In particular:

1. **Phase 1**: at t=30s, the malicious node spreads the fake message inside its neighbour network. Figure 5.3 captures the moment of this action. Specifically, it is sent the following kind of message: "Incident on road *[ID]*. Change road." In this particular case, the ID is equivalent to "38027668#4";

2. **Phase 2**: once spread the information by the attacker to the connected autonomous vehicles inside its neighbour network range, they can discard or accept the message. It is assumed that it is discarded if the path from the source to the destination does not contain, at that moment, that road ID. Conversely, it is considered and managed in the opposite case. For this reason, the message has been considered only by the victim, the "node[3]" which, unfortunately, had in its trip the transition in that road.
   In this way, once the victim has read the message, at t=30,000,505 micro seconds it replies by informing its neighbour node that it will change the road. Again, figure 5.4 shows the spreading of victim's packet, containing inside the payload the message "Changing route.";

3. **Phase 3**: as a result of this successful attack, the attacker will proceed its normal trip whereas the victim will, unfortunately, change its route by inverting its sense of travel. In details, by focusing on the victim and as can be inferred by figure 5.5, it can be observed that the victim car will be in the part of the road corresponding to the opposite running direction, with respect to the situation in the picture at phase 1.

Implementation

The implementation of the attack has included the writing of code referring to the behaviour of the attacker and the victim. With respect to the attacker, its behaviour has been modelled through the usage of a callback, executed 30 seconds after the start of the simulation. In fact, Veins considers the usage of lambda functions a best practice to manipulate the various nodes in the network in order to better simulate what is happening inside the VANET.
In details, what takes place within the callback is shown below:

```
auto payload = makeShared<VeinsInetSampleMessage>();
traciVehicle->setSpeed(traciVehicle->getSpeed()/2);
payload->setChunkLength(B(100));
payload->setRoadId(traciVehicle->getRoadId().c_str());
auto packetFake = createPacket(``Incident on road id
38027668#4. Change road'');
```

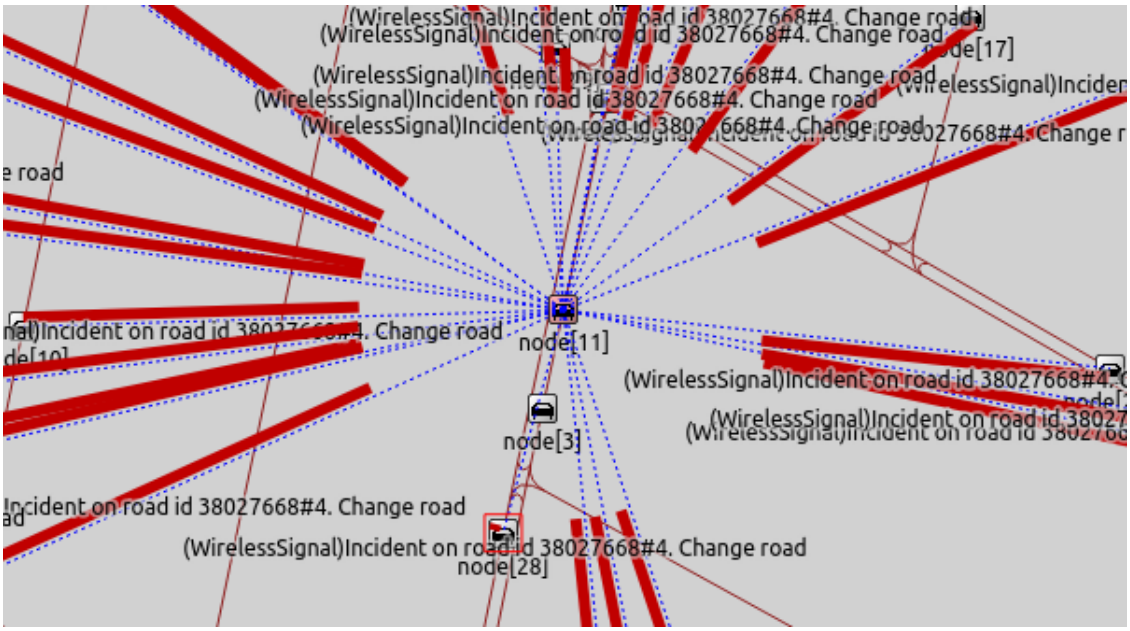

Figure 5.3. Phase 1: attacker spreading the false message.

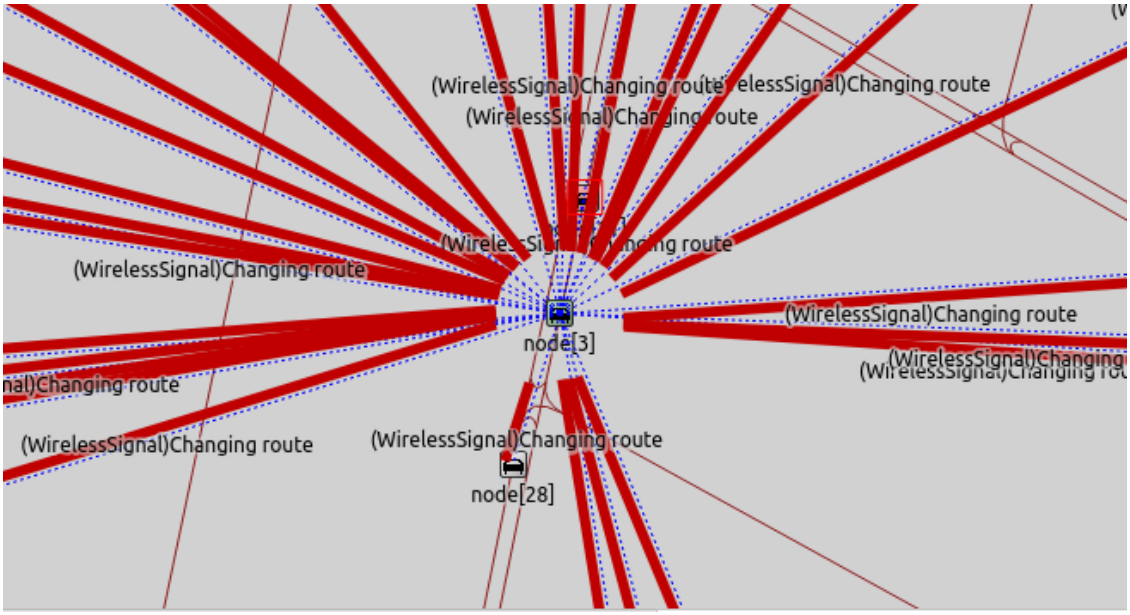

Figure 5.4. Phase 2: victim replies to attacker's message.

```
packetFake->insertAtBack(payload);
sendPacket(std::move(packetFake));
traciVehicle->setSpeed(-1);
```

Analyzing the code, it can be observed the generation of the payload packet of type *VeinsInetSampleMessage*, filled with information useful for the proper execution of the attack. To make the attacker's behaviour even more malicious and tricky, the reduction of speed by half has been simulated, as if to simulate the case in which the CAV realizes of this accident and then preemptively decelerates. This action will

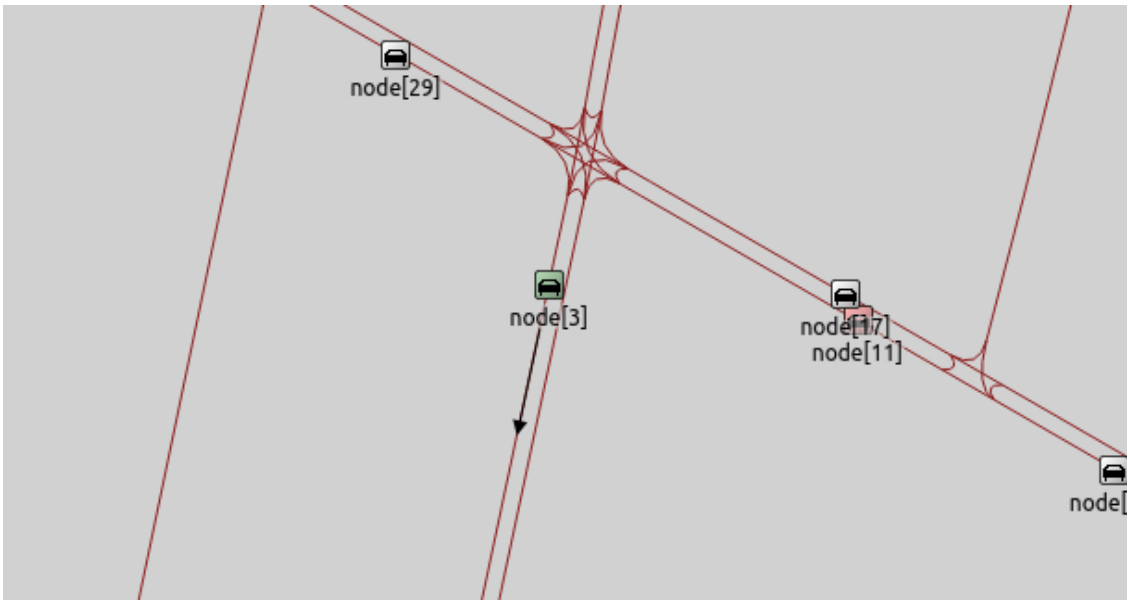

Figure 5.5. Phase 3: victim change its road uselessly.

be propagated to following nodes: in this specific case to node[3], which represents the victim. Once the message has been sent, the malicious node accelerates again. On the other hand, analysing the victim's code

```
void VeinsInetApplication::processPacket
(std::shared_ptr<inet::Packet> pk){
auto payload = pk->peekAtFront<VeinsInetSampleMessage>();
if(strcmp(pk->getPayload()->getRoadId().c_str(),
traciVehicle->getRoadId().c_str())==0){
        auto packet = createPacket(``Changing route'');
        packet->insertAtBack(payload);
        sendPacket(std::move(packet));
        }
}
```

it can be observed how is implemented the logic of accepting or rejecting a packet, thanks to the if-statement. If the node is the one inside the same road of the attacker (as in the case of node[3]), it will be generated the response packet.

5.4.2 Denial of Service attack

In this specific case has been simulated a well-known attack, the Denial of Service. Since it is very frequent nowadays, it has been considered as an attack to be analysed and simulated. As stated at section 3.4, this attack is known to disrupt the **availability** of a targeted system, by creating problems and difficulties. The aim of this particular simulation is to understand and to collect data about the attack, and also to see what is the behaviour of the involved nodes during the execution of it. In this way, it will be possible in the next chapter to analyse the obtained result and to propose practical countermeasures.

Simulation

The actors involved in this scenario are the *RSU* (RoadSide Unit), the *threat actor* and a limited number of *CAVs* (precisely 5). Specifically, the victim is the RSU. It has been chosen not to include the entire network, but to focus only on a piece of the network, and in particular the communication between the attacker and the RSU. As a result of it, it will be possible to concentrate with more details and more granularity on what is happening, so as to observe how this type of attack is able to damage a node. Figure 5.6 depicts the starting scenario, in which is possible to recognize the parties previously defined. The positions of the cars have been

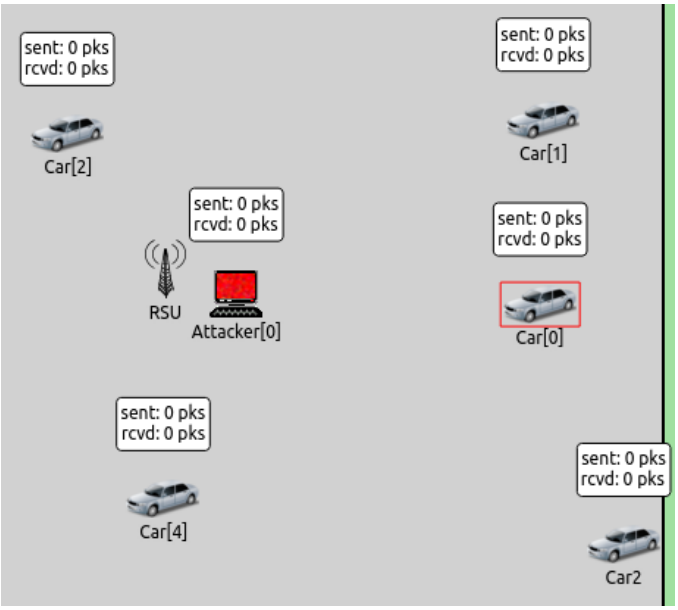

Figure 5.6. Starting scenario for DoS attack.

chosen randomly, as the ones of the RSU and the threat actor. For the purposes of the simulation, each node contains the information about the number of sent and received packets.
The simulation of this scenario begins with the exchange of messages between nodes and RSUs. At time t=3s, the attacker starts flooding the RoadSide Unit with a huge number of packets. In a few moments, the RoadSide Unit will find itself in a situation in which it will be unable to process other packets because it will be flooded with those received by the attacker. As a result, the RSU will have its system resources completely exhausted, and therefore it will no longer be able to provide its service to other nodes. It is interesting to observe how the actions prior to the attack revealed continuous and normal communication between the nodes and the infrastructure, through the periodic sending of messages from the RSU to the CAVs and vice versa, which can be inferred from the fact that some CAV has sent some message to the RSU. From the attack onward, it is evident that the infrastructure appears to be completely **dependent** on the attacker, since it is engaged in attempting to process incoming packets.
According to what stated previously, the attack has started at t=3s. Only after 3,192ms the situation is the one depicted in figure 5.7: the attacker has sent a number of packets equal to 1824, all of them addressed to the RSU. At each message,

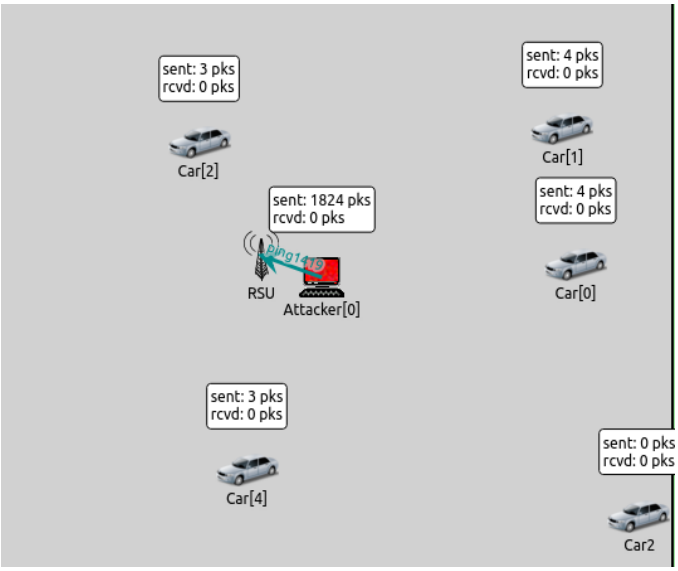

Figure 5.7. Execution of attack.

a ping message of size equal to 43B is sent, so the victim is hit with ICMP packets sent rapidly via ping without waiting for replies from the infrastructure. As a result, the attacker will not wait for some ack and will restart the attack. At the end, when the simulation stops at t=20s, the attacker has flooded the RSU with a number of packets corresponding to **169.528**. In other words, 8.476 packets per second have been sent.

Implementation

With respect to what is the implementation, the nodes within the network were implemented following the same network configuration. A spotlight on what is the attacking node is required, in which it has been decided to model the sending time of packets, by making it short enough and, specifically, equivalent to 0.0001 second. In this way, it has been made an attempt to simulate this particular attack as closely as possible to the real world.
In order to do so, the "Omnetpp.ini" has been modified to contain the following line of code:

```
*.Attacker[*].app[0].sendInterval=0.0001s
```

5.4.3 Distributed Denial of Service attack

Just as with DoS, the Distributed Denial of Service (DDoS) follows a very similar logic at simulation level, with the major difference in the involvement of the number of malicious node: in this particular case and due to the nature of the attack, the number of them is greater than the previous attack.
In this way a "**botnet**" is created, which represents a network of nodes that are affected by malware and then controlled by the attacker with the purpose of harming the victim. Since it is considered the VANET world, the victims are Connected Autonomous Vehicles which are considered "**zombies**". In fact, they act passively

in the sense that they carry out what the attacker orders. The simulation of DDoS is particularly relevant, since most of the attacks in real life are of this type. Furthermore and according to what is stated by [45], a 300 second attack can have a cost of about 5$.

Simulation

In this particular case, the botnet consists of CAVs affected by malware, identified graphically by red cars. Specifically, the botnet consists of 15 elements. The total duration of the attack is 300 seconds, just for simulating a real timeline. The network has been slightly modified from the data alteration attack precisely to give a greater focus on what the packet exchange is about. If this type of simulation had been done through the libraries of that attack, it would not have been possible to do the proper analysis and any observations precisely because two different implementation logics are defined. The network then represents a part of the VANET on which the attack is implemented.
Figure 5.8 shows that at time t=100s a number equal to 970.000 packets have been sent for each node. Consequently, the total number of packets sent in that time, stating that the number of attacker is 15, is 11.640.000. At the end of the simulation, the number corresponding to the amount of packet sent is 34.920.000 , with an average of 116.400 packets per second.

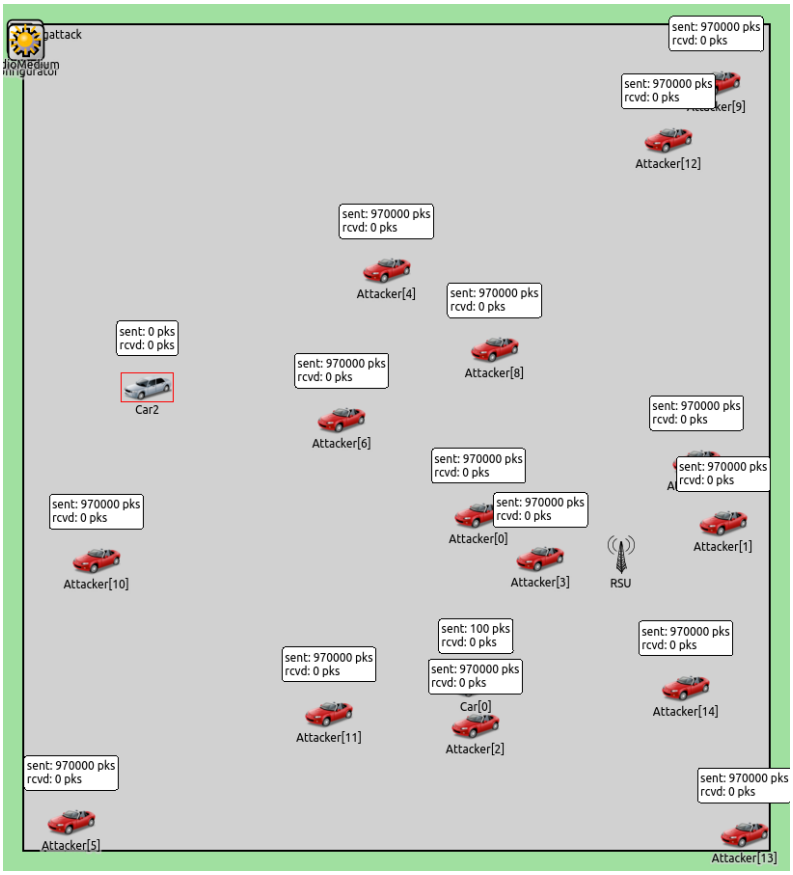

Figure 5.8. Distributed Denial of Service at t=100s.

Implementation

The implementation of this type of attack involved a behaviour similar to what has been done for Denial of Service with the difference that a distinct number of nodes have been instantiated, equal to 15. Again, a number corresponding to 0.0001 seconds between packets has been chosen as the sending time. The execution time of the simulation has been set equal to 300 seconds.
In order to simulate some mobility in the nodes, a linear type of movement has been applied to them, so as to simulate movement on a road network. The network protocol employed is the Address Resolution Protocol (ARP) type, so as to implement a type of DDoS attack by means of ICMP echo-requests (pings).
Since this type of attack is implemented through the use of pings, it is very often referred to as **PoD**, or "Ping of Death". It can be considered as a particular use case for implementing a DDoS-type attack. Finally, since this type of attack focuses more on the communication between nodes at the wireless level, it has not been necessary to employ the SUMO tool, because it has been sufficient only to use Veins' internal environment, called Qtenv.

5.4.4 Timing attack

Even in this case, and taking into consideration the table 3.1 that describes the classes of attack, among these there is one which describes a particular attack known as "timing attack". The aim of it is to add delay to the communication after an event (generally a crash or some emergency message). As a result, there will be interferences inside the network. Furthermore, the threat actor is able to manipulate the communication of the network. Due to this delay, the nodes inside the VANET will be notified with a delay that is the sum of the effective delay plus possible delays caused by the propagations of the message and the reception of it. Generally the delay is something that should not be verified because, since VANET are considered communication in real time, they could not afford to have some delays, otherwise there will be a sort of chain reaction in which the delay will cause other problems in the communication.

Simulation

The simulation has used the topology of the network that has been described in section 5.3.1. In this case, the scenario is the one in which the CAV called "node[9]" for some reason is forced to stop, at t=20s. The autonomous car that is behind it, is the "node[15]" which represents the malicious node. The attacker, instead of starting the notification immediately, starts the process of notification at t=22s. This time period, which is equivalent to 2 seconds, represents a very dangerous situations if it would happen in the reality. In fact, if it verified for real, there would be serious consequence to the other nodes. The following figure represents the moment in which the malicious node notifies with delay of the car stopped in front of it.

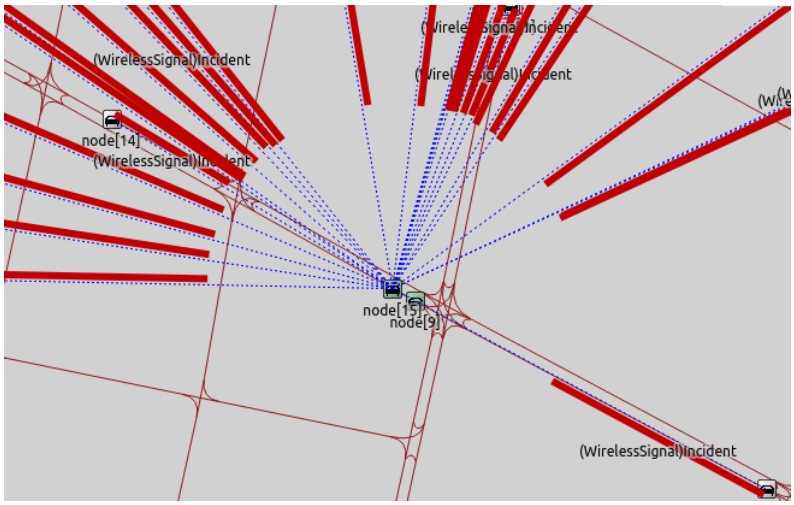

Figure 5.9. Timing attack.

Implementation

With respect to the implementation, the Connected Autonomous Vehicles called "node[9]" and "node[15]" have been modelled through the usage of callback. In particular:

- as far as it concerns the actions to be performed by "node[9]", it has been designed the behaviour in case of stopping of the car, thanks to the following command

  ```
  traciVehicle->setSpeed(0);
  ```

- taking into consideration the "node[15]" the callback contains lines of code referred to the operation of sending of the message. Hence, an object of type VeinsInetSampleMessage is created, its chunk's size is set, it is put inside the packet and it is sent through the usage of

  ```
  sendPacket(std::move(packetIncidentDelay));
  ```

5.4.5 Social engineering attack

This type of attack, unlike the others, aims to flood the network with messages that make little sense and deviate significantly from the communication nature of CAVs. In fact, what occurs is the reception of messages that are useless for the purposes of the network. For example, a Connected Autonomous Vehicle may receive several attack messages containing the text "you are stupid", or "slow down" followed by "accelerate now". Furthermore is important to stress that this type of attack is possible if the OBU has been manipulated in order to send this messages. In this way, the person inside it has no idea that the victim receives these packets containing this kind of message.
As a result, what occurs is a lack of trust in the network.

Simulation

The network is made of 67 vehicles and one of them is the attacker, which is known to be the "**node[3]**". With a period of 10 seconds, the attacker sends a message containing the text "You are an idiot!". The entire network is flooded with these packets.

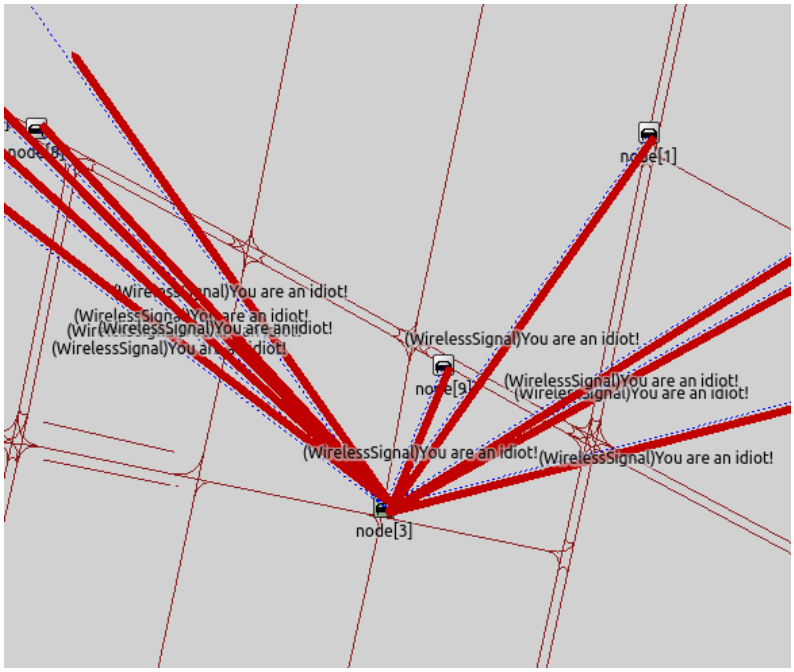

Figure 5.10. Social engineering attack.

Implementation

The implementation of this attack consists in the usage of a callback that is executed each 10 seconds. As previously said, the callback is a key concept in the writing of code for modelling the behaviour of CAVs and RSUs.
Even in this case, a packet of type "*VeinsInetSampleMessage*" has been created and populated with different information, such as the length and the payload.

5.4.6 Black Hole attack

Focusing on this attack, which falls under the network attack class, it is important to observe its primary purpose. In fact, what is attempted in this particular case is the fact that the attacker identifies itself in the network as a node containing the shortest path to the target node. Consequently, what the attacker node will try to do is precisely to receive packets and subsequently manipulate and/or drop them. In this way, the recipient node will never receive its message.
In order to better understand the attack, the structure of the network can be transformed into a graph. In fact, considering the following figure and giving each edge a unit weight, it is easily observed that the source node is at distance 2 from the destination node while at distance 1 from the RSU. The next section will better explain what will happen during the simulation.

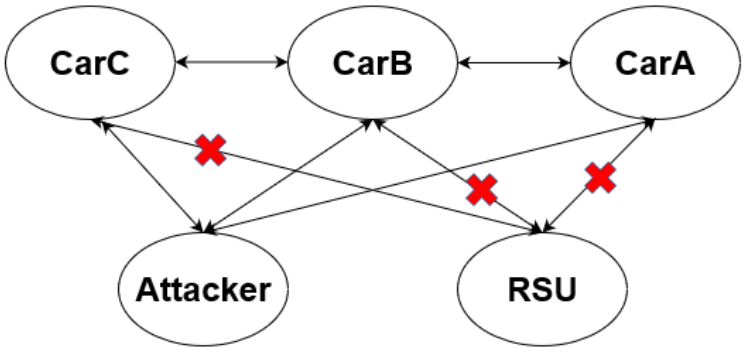

Figure 5.11. Graph of the network.

Simulation

The simulation can, again, be considered as a focus in a part of the previously created network in figure 5.2. In detail, three vehicles are moving along the road and the first one, carA, notifies the presence of an accident in 200m. In this way, having to notify the RSU, the source node sends the message to this infrastructure. And that is where the attacker comes in, who is able to break the links with the RSU and declare himself as the node that has the shortest path to the destination node. Hence, the attacker receives the packet and will never forward it to the recipient, precisely the RSU. Therefore, the infrastructure will not be informed and will not be able to forward the information to other nodes. Figure 5.12 depicts the execution of the attack, in which the message is sent to the attacker instead of being received from the RSU.

Implementation

At implementation level, the behaviour of cars and the RSU has been modelled as objects of type Car and RSU, respectively. The following figure observes the fact that the attacker's range is able to include communication with the RSU although this will never happen, just for the purposes of the attack. It is also important to note that the attacker's location could have been even as far away from the infrastructure, precisely because the attacker claims to have the shortest path to the target node.

5.4.7 Jeep Cherokee attack

This attack simulates the one happened in real life, as stated in section 3.6.1. According to what happened, the attackers have been able to take the control of the car by connecting to its internal Wi-Fi. After that, it has been possible to control the CAN bus and to manipulate throttle, break, steering wheel and the opening and closing of car doors. For doing that, the password has been discovered through brute force and the number of attempt has been reduced because of the fact that the password is the combination of the first time and the date of the first

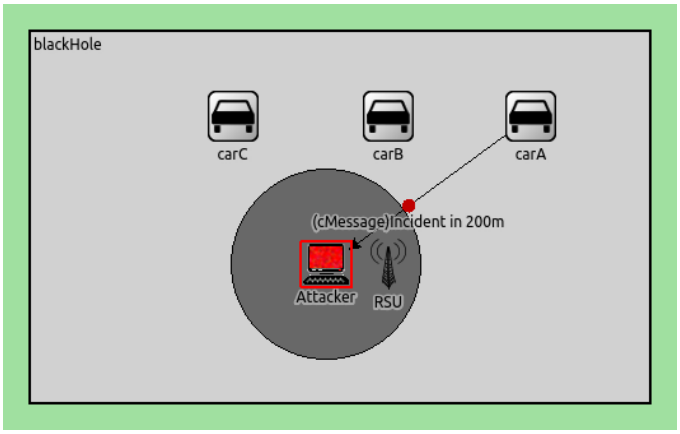

Figure 5.12. Black Hole attack execution.

usage of the car, to which has been added some seconds that corresponds to the car to start up.

Simulation

In this particular implementation, the attacker, once he takes control of the vehicle, is able to give all possible commands.
Specifically, the Jeep is represented by "**node[5]**" and at time t=10s sees its speed halved; at time t=20s it is stopped while at time t=30s the car restarts. In this way, the car behind the Jeep ("node[8]") is obliged to decelerate, to stop, and to restart.

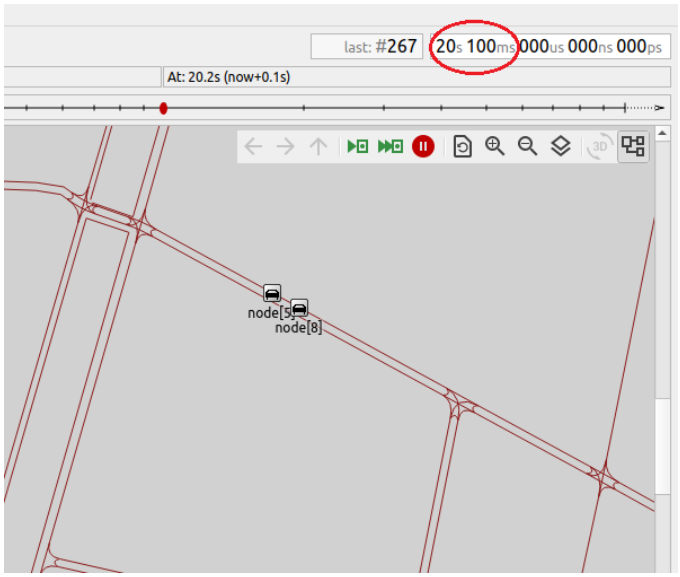

Figure 5.13. Jeep Cherokee stopped by the attacker.

Implementation

Again, speed manipulation is practically implemented through the use of several callbacks that are triggered at time t=10,20,30s. Each callback contains respectively

```
traciVehicle->setSpeed(getSpeed()/2);
traciVehicle->setSpeed(0);
traciVehicle->setSpeed(-1);
```

and, the first line halves the speed of the car; the second stops it and the third restart the car.

Chapter 6

Analysis of results

After this set of simulations, it is necessary to analyse and observe what happened. For this reason, the following chapter aims to analyse the simulations and the data collected in order to produce a series of observations and to propose mitigations to the types of attacks. Some of these have been realised using simulators; others, however, could not be carried out due to the limitation of the latter, and will therefore be proposed countermeasures at a theoretical level. Once again, it is necessary to point out that it is not always possible to simulate any scenario with these tools present, as there are internal limitations to the tools themselves.
The following sections will analyse each attack previously implemented so as to produce observations and mitigations.

6.1 Message alteration attack

As seen at simulation time in section 5.4.1, in this attack an alteration of the message is observed. The mitigation that can be considered when this type of attack occurs involves the idea of not having some sort of "**direct trust**" towards the node within the VANET that sent the message. This is precisely why an *intermediate* step is necessary, so as to make sure that what the attacker has notified is really happening. One kind of solution, as cited by [46], provides the usage of *particular* nodes inside the network which are able to reply to the message received by the malicious node. It is highlighted the concept of "particular" because they are considered **intermediate** nodes, in the sense that they verify what is said by the attacker. In fact, since they are part of the road which, according to the attacker, contains the incident, their role is to verify if there is a real one.
Exactly for this reason, it has been defined the function "*checkMessageIsOk()*" that checks the payload contained inside the message and verifies if it is true what said by the threat actor.
In this particular case, the attacker states that an incident occurred inside the road with id "38027668#4". For this reason, the function queries the presence of CAVs inside that specific road, in order to verify if it is really present an incident or, more generally, if it has happened something for which is necessary to alter the trip of a node. And in fact, two nodes inside that road with that specific id ("**node[24]**" and "**node[26]**") analyse the packet received by the attacker and observe that no incident is present. As a result they forward a message containing a payload with

an attached message which states that the situation is different from what was said by the attacker (see figure 6.1). So the victim, which is recognized by being equivalent to "node[3]", will not consider the message received at the beginning of the attack and will proceed with its real route, avoiding the alteration of the entire trip (figure 6.2).

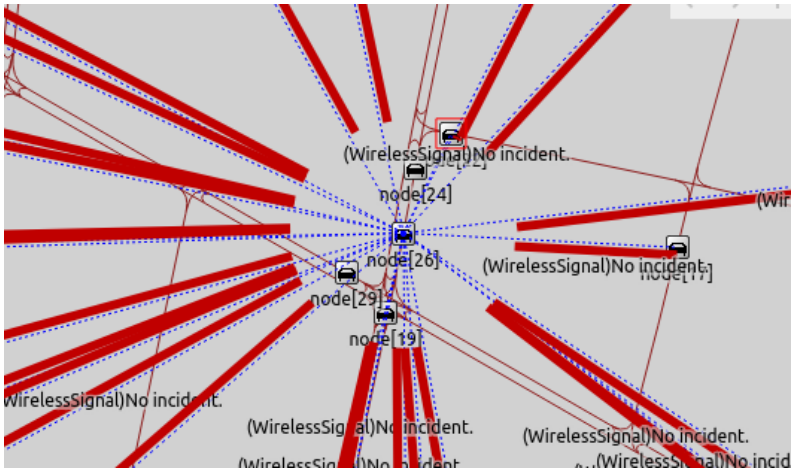

Figure 6.1. Notification of no incident.

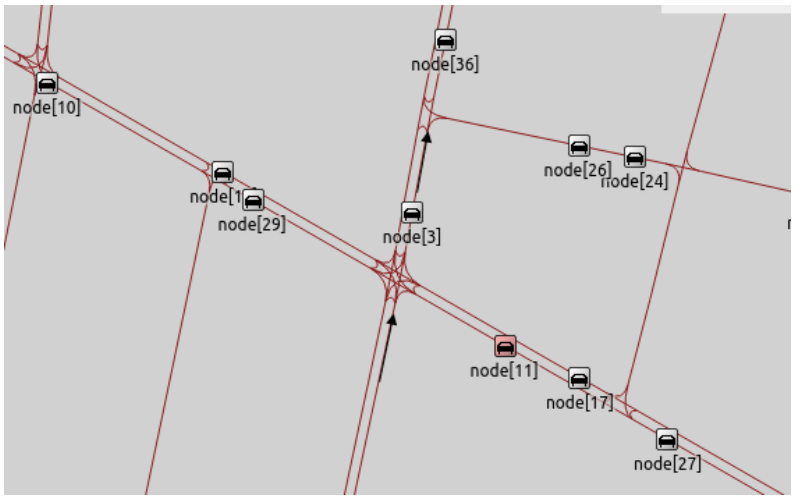

Figure 6.2. Victim following its real trip.

6.2 Denial of Service

The purpose of the simulation has been the demonstration of how capable this attack is of causing disruption within the network. It is appropriate to state that a DDoS-type attack is preferred to a DoS-type attack as it is more effective. However, analysing how much a simple node is capable of bringing problems within the network shows how much there is a need to take appropriate preventive measures. The Veins tool, used for the simulations, has allowed the creation of graphs, so as

to better study what happened during the simulations. The chart in 6.3 shows the linear growth in the number of packets sent by the threat actor from the time when the attack begins, i.e., from time t=3s. The abscissae represent seconds while the ordinates represent the number of packets sent. The attack starts after 3 seconds

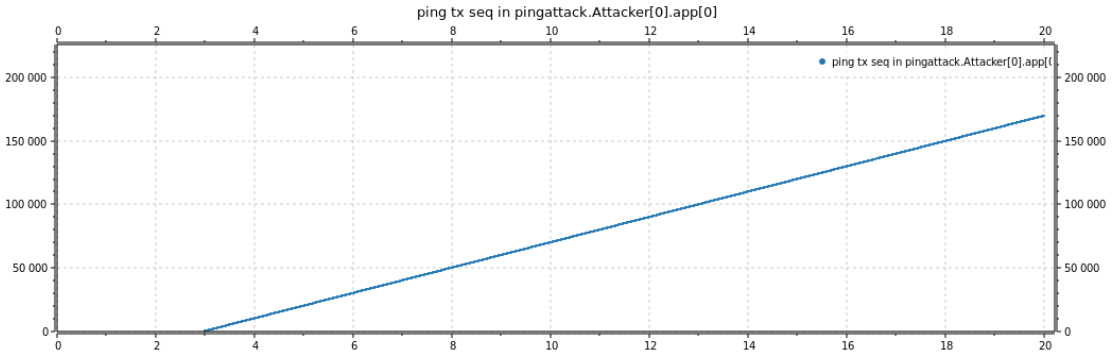

Figure 6.3. Line chart DOS.

and at the end, so at t=20s the number of packets sent is approximately equal to 170.000. Given the nature of the attack, it is obvious to expect that a certain number of packets will be lost. In fact, cases arise where the network appears to be congested given the large number of packets sent to the single node (victim node), with a higher rate of sending than handling individual packets. Figure 6.4 represents this analysis, where the RSU (the victim) has its packets dropped due to the attack. For convenience, values are reported within the table 6.1.

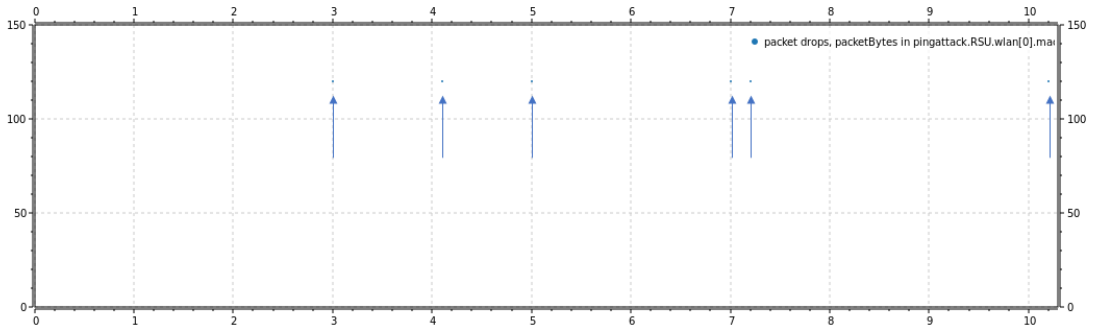

Figure 6.4. Packet drop DOS.

One possible solution involves the possibility of implementing an algorithm that is able to sense the number of packets that come to be exchanged, so as to understand whether it is a DoS attack or a simple communication, according to what has been proposed by [28]. In particular, it is firstly created an array X in which incoming and outgoing IP are stored. In this way it is created a sort of threshold for the condition of the network. In this way, every time that a CAV generates a RREQ and does not get RREP, the hop count is checked and then the prevention or detection is respectively applied by taking into consideration the case in which the packet is ok or not.

	number of packets dropped
t=3s	120
t=4s	120
t=5s	120
t=7s	120
t=7.12s	120
t=10.12s	120

Table 6.1. Values of packet dropped.

On the other hand, taking into consideration what stated by [47], the use of an **Intrusion Detection System** (IDS) allows detecting if non-legitimate conditions arise, so it is possible to detect any anomalies, it is possible to record logs, make detection of this type of attack. Its function is to detect unauthorized access to computers or local networks at an early stage, identifying cyber threats-caused by experienced hackers, automated tools, or inexperienced users using semi automated programs-before they can cause damage to systems. The only consequence is that there is more delay and weight in the network.

6.3 Distributed Denial of Service

According to what has been analysed and observed in section 6.2, in this one is possible to follow a similar study on the values obtained from the simulation. Firstly, it is peculiar to observe the fact that each attacker, after only about 132 seconds has sent a number of packets very close to 1.300.000 . Figure 6.5 outlines what has just been said. Continuing the simulation and reaching the time limit, that is the time limit of 300s (a value chosen to simulate a real time frame), each threat actor has sent a number of packets equal to 2.910.000 . Multiplying what has been obtained by the number of attackers, which is 15, the result gives a number equal to 43.650.000. Considering the fact that each packet has size 56B, the total is equivalent to 2.27GB. Again, it has been possible to take advantage of the graphs

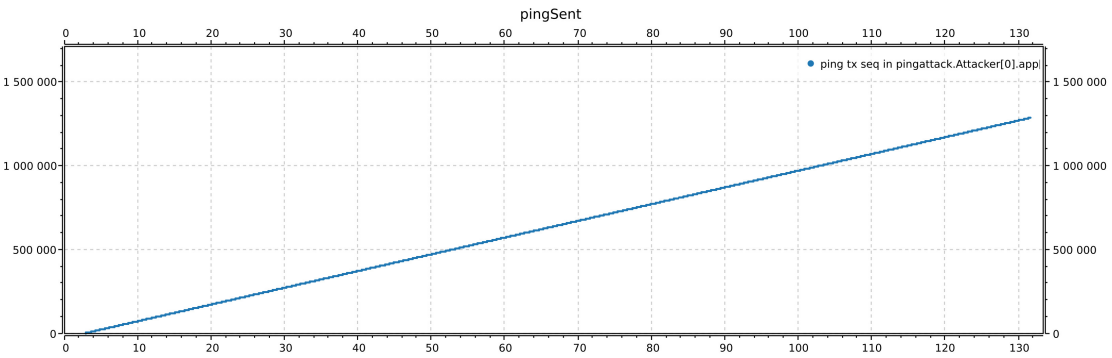

Figure 6.5. Line chart DDoS.

proposed by the Veins tool, which can collect all the parameters useful for analysis. 2.27GB in a total time of 5 minutes turns out to be an amount that can be tolerated

by a system, in this case by an RSU. But if we take into consideration the fact that the Mirai malware was able to create a botnet of 380.000 bots, then the value turns out to be

```
dimPacket x botnet x packetsIn300sFromOneNode =
= 56 x 380000 x 2910000 = 56,3TB
```

Willing to propose a possible solution, the literature has proposed a number of countermeasures and mitigations in order to succeed in combating and avoiding this type of attack. Considering what has been proposed by [29], in order to prevent this type of attack, an algorithm could be implemented and placed inside a controller, which is able to calculate the entropy of the system and then evaluate, based on a series of statistical indices such as the frequency distribution matrix and covariance, whether this value exceeds the threshold. If it does not exceed the threshold, access is granted; if so, a reCAPTCHA mechanism is implemented and, if properly passed, access is given to the service.
Another solution, which deviates slightly from what has just been proposed, binds DDoS attack prevention through the use of an IDS via Machine Learning, as explained by [48]. Specifically, the Intrusion Detection System that comes to be placed inside a VANET network first is subject to the training phase and then to the testing phase. In these two phases the model comes to be trained, able to perceive instances of DDoS attack. Once this is done, at the time an attack is being detected, it comes to be located and all packets from that area are ignored, later informing other vehicles about the attack. It is important to point out the fact that this solution deals only with the DDoS coming from objects that are inside the VANET, i.e. the botnet is made of CAVs.

6.4 Timing attack

This attack, unlike the others, has as its sole purpose to delay communication and thus cause the network to experience delays in messaging. It is obvious to expect that this type of communication involves messages being exchanged with appropriate timing, thus almost in real-time. Therefore, it is considered imperative that everything is sent and received with proper timing. What an attacker therefore seeks to do in implementing this attack is to delay the communication. The consequence of this is that the receiver may find before him an event for which he should have received a message before observing what happened.
As a result, according to what is stated by[49], one possible countermeasure about this type of attack is to define an algorithm that is able to insert within the packet a timestamp of when it was generated and sent, and then the packet will be encrypted. In this way, the receiver will be able to control the actual sending of the message and will not suffer this type of compromise. What is more, the fact that the packet is encrypted provides an additional layer of security.

6.5 Social Engineering attack

According to the simulation that previously took place, this attack aims to flood the network with messages that have no well-defined meaning and/or purpose. In

this way, it is only loaded the VANET with useless messages and changed the mood of the users inside the CAVs.
A possible countermeasure for this type of attack could deal with the implementation of a software device capable of filtering incoming packets to other nodes, so as to avoid the consequences of a social engineering type of attack towards the network and the person reading the message. In this way, messages can be automatically discarded and precisely this type of message exchanging will be avoided. What is more, only packets containing valid information related to driving and/or emergency circumstances will travel within the VANET network.

6.6 Black Hole attack

This category of attack has been mitigated, also in accordance with the present literature, such as [30], using a system known as SIN, which stands for "So-called Intelligent Node". These are nodes installed within the network that periodically send **special** RREQ messages to make sure that all nodes are legitimately part of the VANET and that the path from a source node to a destination node is actually the one present in the real world. These messages have been defined as "special" because they can be considered as an alternative version of the RREQ in that each message internally maintains an ordered list of previously routed nodes. In this way any node receiving this type of message can register the route back to the source node. Each node as a consequence will respond with an RREP to the previous node.The SIN then at the end of everything compares what it has received with the database present in the TA and checks if there is any unrecognized node pretending to be a node that has the shortest path to a destination node. If it is, it would be excluded from future communications.
Analysing what has been implemented under a critical point of view, however, it could be problematic that the SIN periodically sends messages into the VANET, because it could overload it.

6.7 Jeep Cherokee attack

The execution of this attack has demonstrated how it is possible for an attacker to manipulate the car in the moment there is access to its internal Wi-Fi network. In this way, malicious software is injected so as to manipulate the CAN bus and have full control of operations. It has been possible to implement behaviours such as changing the speed and using the brake but it was not, on the other hand, possible to manipulate parameters such as opening or closing the doors or activating the windshield wipers. Furthermore, the execution of the brute force attack has not been implemented and it has been assumed that the simulation is started in the moment in which the attacker has discovered the password of the Wi-Fi network and the malware has been injected. In fact, for doing so, the attacker must follow the car because the malicious node need to be extremely near to the car in order not to lost the signal for brute forcing the Jeep.
One important mitigation could deal with the generation of password for protecting the Wi-Fi connection inside the car: it must be safe and complex enough not to

avoid this type of cracking. However, Jeep spread a patch to fix these vulnerabilities.

Chapter 7

Conclusion

The aim of this chapter is, at the end of the thesis work, to conclude what has been discussed in previous chapters and to define different aspects that have been come up from simulations in order to propose possible improvements for future developments.
In the course of the thesis work, recent real-life attacks affecting cars that are part of different car manufacturers, such as Mercedes-Benz, Audi, Toyota and others have been listed. For simplicity, these attacks can be divided into two categories: those internal to the vehicle and those external to the vehicle. This thesis work has focused more, even from the type of simulations conducted, on those outside the vehicle; it means that have been simulated those situations where an attacker remotely targeted a victim and then carried out an attack. In this way, those types of attacks in which the internal software of a vehicle was manipulated have been excluded. For example, the attack on Toyota Prius, where a hardware device has been mounted inside the car in order to manipulate the software, has not been taken into consideration.
By aligning these attacks with the literature, several attack classes have been listed to which the different attacks that actually occurred can be labelled. In this way, it has been defined a reference between the attack simulations and the classes that have been defined in the previous chapter. For each implementation, where possible, a mitigation has been proposed and in some cases it has also been implemented. Different cyber attacks have been implemented such as Denial of Service, Distributed Denial of Service, Data alteration etc and also the ones happened in real life, such as the one to Jeep Cherokee. During the simulation on this specific car it has been realized a scenario which is similar enough to the real life, where the attacker has been able to manipulate the speed of the victim's car remotely, by reducing it or decreasing it up to zero km/h. The result of this has certainly been a series of disruptions within the network, resulting in queues in the network inside the VANET for no real reason, simply because of an attacker exploiting the vulnerabilities in this car.
For this reason, it is of primary importance to consider cybersecurity as a fundamental and essential pillar during the development of these types of devices.
It is right to say that the Veins simulator has allowed the development of this type of simulation in order to perform analysis, to analyse case that occurred in reality, and to implement different attacks to observe their consequences on the network topology. But what is important to be underlined is the fact that, up to

now, it is not completely possible to simulate the total behaviour of the car. In fact, and for giving a detailed analysis, different attacks cannot be simulated due to Veins' software limitation. For instance, it has not been feasible to simulate an attack happened in real life, in which the attacker has been able to manipulate a car remotely and to disable horn, headlights and the opening/closing of car doors. Nevertheless, this turns out to be of primary importance, especially the last case, because at high speeds a malicious hacker could simply manipulate in a remote way the behaviour of the car to make the doors open and to cause serious damage to the people inside, theoretically being able to produce considerable damage and chain reactions with other cars. What is more, disabling the car's headlights could lead to serious consequences especially during night hours where light is scarce, causing in the worst case accidents due to poor visibility. In fact, it is not possible to control some functionality of the car during the simulation, such as the headlight or horn management, which are of primary importance. As a future development it could be possible to think about the realization of an extension for the simulation software, but still it would be necessary to write ad hoc code for the specific simulator version and especially in a short time or with reasonable efforts, just because going down to the specifics the management of the functionalities of a car could vary from car to car and therefore it would be not too easy to implement.
Another aspect to be analysed deals with the fact that when the attack on Jeep Cherokee has been simulated, it has been assumed that the attacker has been able to discover the password of the Wi-Fi of the car. This feature is something that is not implemented in Veins simulator, because it does not have the tool to manipulate this particular aspect of the car. By the way, it is particularly important as the other that unfortunately cannot be implemented and simulated nowadays.
It is also important to point out the fact that the Veins simulator is one of the few open-source ones to which updates are periodically released, as explained in table 4.1. Despite this rapid evolution in driving, which will take even more importance in the coming years, precisely autonomous driving, it should be relevant that simulation software also keep pace with the changes occurring in reality.
Another negative aspect is to be identified in one of the fundamental aspects of computer security: confidentiality. In fact, the program does not have internal mechanisms that are capable of providing encryption and decryption implementation. A reference could be made to external libraries, which are either out of date or not compatible with current versions of Veins. Among them NETA(NETwork Attack framework) could be taken into consideration, which allows the usage of a library capable of giving to the user mechanisms of encryption of packets. Unfortunately this framework appears to be particularly out of date, as its manual appears to have been defined in 2013, and the framework versions are not compatible with current Veins version. It might therefore be considered to use this framework with an older version of Veins. However this has the cost that improvements and updates to the simulator would not be taken into account. Software would then be used but it would result to be deprecated in some parts: this approach therefore does not match with a development that is constantly updated and in step with continuous technological development. What is more, since the simulations are written in C++, it is necessary for the external library to be linked to the project created for the purpose of the simulation, to be build, and subsequently included in the code. As can be seen, this solution turns out to be particularly time-consuming. As a

result, simulations may lack encryption and decryption mechanisms. One possible solution would be to use flags, to specify that the message has been encrypted or decrypted. Specifically, to the packet sent from one node to another during the simulation, it could be associated a Boolean variable that says whether the message is encrypted (e.g., 1) or decrypted (e.g., 0). It comes naturally, though, to think that this approach does not simulate what is really happening in reality; consequently, the simulation could provide results that deviate, even if slightly, from reality and thus not produce reasonably exact results because, for example, one would lose those (although small) message encryption and decryption timings.
Because of this mismatch between reality and simulators, these issues are still present.
As a result, many scenarios are not totally feasible according to what could happen in reality, thus leading to limitations that do not allow for proper simulation and thus it will not be obtained the correct values of results, observations, analysis, damage to people or things. Despite these limitations, where attacks with real-life scenarios have not been fully implemented, it has been decided to simulate a generic attack (DoS, DDoS etc.) to try to prevent what could be a large-scale attack in real life and thus be able to implement the right moves in advance, as if they were considered as **precursors** in a way that the user can use these simulations to understand the behaviour of the network and of the other actors involved in the attack, despite the limitations of the simulator. What is more, these simulations have been implemented on a map that recalls the city of Turin, precisely to simulate an urban life situation, an "everyday" situation, in which every person could be part of. Also for this reason, the number of vehicles has been chosen to simulate a busy area, in order to emphasize what might cause an attack.
Looking critically at the performance of the simulations, some of these have shown that it is possible for packets to be lost and therefore that optimal communication is not possible, as explained in section 6.2 and 6.3; others have observed how the delay in sending messages is highly problematic and causes inconvenience; still others have outlined the fact that the sending of "altered" messages can lead to situations in which the path from the point of departure to the destination is altered and so the driver/passenger finds himself in the condition of seeing his car take completely different routes from those chosen at the beginning just because of a message that is capable of creating problems to the network path. This is exactly why it is necessary for all these systems to have internally modules that have been designed with all security aspects in mind, from the first to the last, in order to avoid the situations listed in section 3.6, where real attacks have been described.
It is also true the fact that now drivers do not have a level of autonomous driving equal to 6 (which means total autonomy of the vehicle), so the driver is able to regain control of the car and consequently be able to switch to a lower level. For this reason, knowing that these attacks are present nowadays, from a certain point of view it does not help in the future for the following reason: if nowadays drivers are under attack, they could try (not in every situation) to regain the control of the car, even if it does not work in every situation. But in those situations in which the level of automation is equal to 6, it is much more worrying to think about an attack and to the consequence of it, because the "driver" could not get the control of the CAV.
For this reason, it is necessary that a high level of security be achieved. In this

way, it is necessary that all strategies/implementations/design choices are put in place for proper operation from a security standpoint. Consequently, it is of primary importance that all the limitations previously outlined be overcome through the implementation of extensions, new tools, libraries, and frameworks that are capable of aiding research in the realization of simulations that are as close, if not identical, to what might occur in reality.
Hence, being aware of all these limitations, it is necessary (if not appropriate) that these limitations are overcome in the shortest possible time by implementing what is necessary and being able to include a simulation between vehicles that is the closest possible to reality. This will avoid implementing real scenarios in reality with the risk of generating unexpected and therefore particularly costly situations. And this is precisely why simulators are optimal candidates for performing this type of operation. In this way, as future developments it is surely necessary to define some sort of extension even to the simulations themselves that has been done in the previous chapter, to be considered as a stepping stone for defining more detailed operations and thus be present within the simulation framework.

Appendix A

User Manual

The following appendix has been prepared for the purpose of assisting the end user in the installation of the components necessary to run the simulations previously demonstrated.
These can be run on a Windows or Linux Operating System. About the former, it will be necessary to download and install a virtual machine, install a Linux operating system in order to run the SUMO and Veins programs within it. In the latter, on the other hand, it will be sufficient to simply install the programs just mentioned.

A.1 VirtualBox installation

VirtualBox represents a software widely used for the creation of virtual machines within an operating system. What is attempted is precisely the possibility of recreating an operating system that is effectively "virtual", just as if it were completely installed on the device hosting the VirtualBox program, when in fact everything is done in a virtual manner, dedicating a portion of RAM to it.
To install the software, it is necessary to visit the site https://www.virtualbox.org/, to click the "**Download**" button and to proceed with the installation. Simply the user can follow the guided installation process ("**Wizard**"), in order to properly and quickly install it. Once finished, the following steps must be followed:

1. download a version of Linux operating system in ".iso" format. For convenience, it is recommend Linux Mint or Ubuntu. This format is useful for image-type files, so files that can contain the entire contents of a DVD, as if it were an installation disk but in the form of software;

2. open Virtualbox and click on "new" button;

3. give a name to the virtual machine, specify the type and operating system. In this case, assuming we want to use Ubuntu, we will specify "**Linux**" and "**Ubuntu(64-bit)**" respectively;

4. now the amount of RAM to be allocated for the virtual machine will be specified, which by default is set to 2048MB, that is equivalent to 2GB, which turns out to be a valid size. Hence, click on "next";

5. select “Do not add a virtual hard disk”. This setting will cause the user to manually add an image file, then the one previously downloaded. Then click on ”create.”

Once you have created this virtual machine you need to specify the “.iso” file to proceed to boot. Specifically:

1. selecting the virtual machine you just created, click on “settings”;
2. click on “storage”;
3. click on the icon containing a CD icon with a text field that says “empty”;
4. click on the CD icon on the right and select “choose a disk file”;
5. navigate among folders to find the “.iso” file downloaded in step 1 of the numbered list above;
6. click on “ok” button and then on “run” button.

A.2 Ubuntu installation

When the virtual machine is booted, the operating system that has just been placed inside, is started. In this case Ubuntu has been chosen. Below are the steps for installing this operating system:

1. at the first window that is shown click on “install Ubuntu”;
2. give “continue” until the button that says ”Install now” appears;
3. once the installation is finished, proceed with setting the timezone, a username and password, as well as the keyboard type;
4. when everything is finished, restart the virtual machine and log in to the operating system with the previously created credentials.

A.3 SUMO installation

Once you have installed the environment for running a virtual machine and have installed and started Ubuntu, the following section is aimed for defining the steps for installing this program.
First you must open a terminal window, either by selecting it from Ubuntu’s program menu or by pressing the *ctrl+alt+t* buttons combination clicked simultaneously. When starting a terminal window give the following commands in the order shown below:

```
sudo add-apt-repository ppa:sumo/stable
sudo apt-get update
sudo apt-get install sumo sumo-tools sumo-doc
```

Once installed you will be able to run commands from the terminal or select the program from the list of applications.

A.4 Veins installation

Fortunately, developers of this simulator have made available reference documentation for each type of operation to be performed. In this specific case, its related operation are available at the following link.
On the other hand, these are the steps that must be followed in order to install and run it correctly.
First of all download OMNeT++ from omnetpp.org and specify the Linux archive, called "omnetpp-6.0-linux-x86_64.tgz". Then copy the downloaded file into a folder. After that open the terminal, move to the folder and extract its content through

```
tar xvfz omnetpp-6.0-linux-x86\_64.tgz
```

command. After the extraction a folder called "omnetpp6.0" is created. Move into it and type

```
source setenv
./configure
make
```

These operations set the environmental variables and then start the configure script which detects installed software and configuration of your system. At the end it is possible to run the compile process through the last command.
If everything is performed correctly, is possible to type

```
./omnetpp
```

in order to run the program.

Appendix B

Simulation execution Manual

This appendix has been defined with the aim to follow the user into the execution of the simulation of attacks. In particular, will be considered the usage of the tool named Veins.

B.1 Map creation

There are situations in which is necessary to build a simulation with a specific environment, so by defining particular scenario. In this way, it is the turn of OpenStreetMap. It lets the user download a selected piece of map. After that, is important to transform it in something that is useful for the simulation in Veins.
In order to download a specific map:

1. visit OpenStreetMap and then select the part of the map to be downloaded;
2. click on "Download" button. A file ".osm" will be downloaded.

Once downloaded the map, the user has a file with ".osm" extension. Suppose that it is called "map.osm". If the user wants to include it into the simulation, firstly the terminal must be opened. After that, the following command must be typed:

```
netconvert --osm-files map.osm --output-file map.net.xml

/usr/share/sumo/tools/randomTrips.py -n map.net.xml -e 250
-o map.trips.xml

duarouter -n map.net.xml --route-files map.trips.xml -o
map.rou.xml --ignore-errors

polyconvert --net-file map.net.xml --osm-files map.osm -o
map.poly.xml
```

The first command transforms the file previously downloaded into a ".xml" file.
The second one defines trips inside the net just created.
The third command receives in input the trips and outputs the routes that the CAVs will follow.
The latter will put the buildings inside the network.

B.2 Veins attack simulation

Once the map has been downloaded and all the necessary files have been created with the proper commands, Veins can be launched. Once started, the following figure represents the home of the IDE:

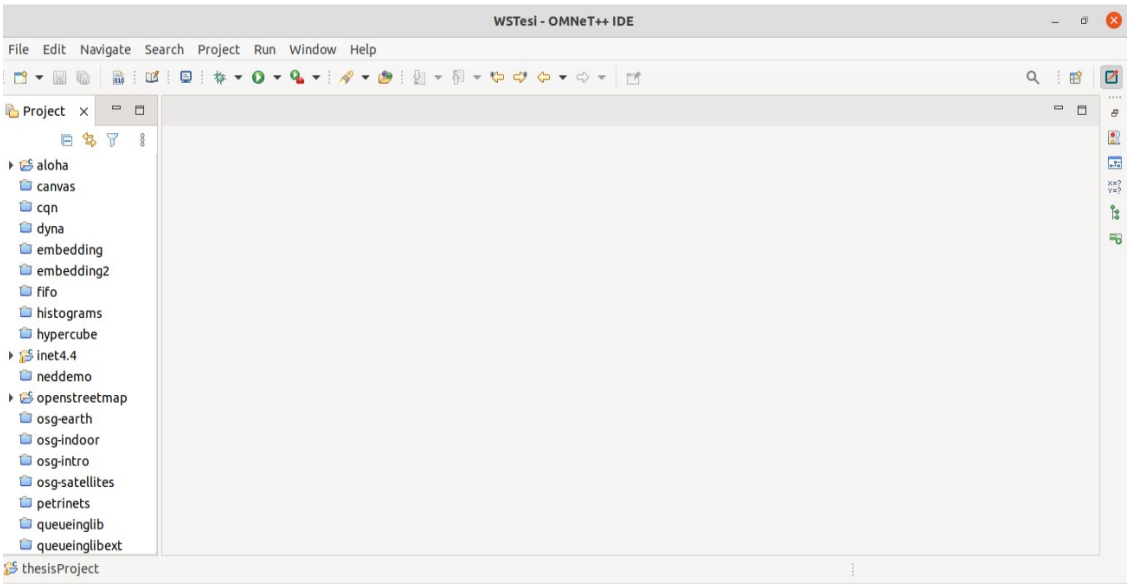

Figure B.1. Veins IDE.

Creation of a project

As can be seen, the left bar contains a number of projects included with this program's library. In case, however, the user would like to create a project from scratch, just follow these simple steps:

1. click on "File";
2. click on "new";
3. click on "New OMNeT++ project";
4. follow the wizard and click on "Finish" button.

In this way the folder will contain the following files:

- file **.ini** which contains the initialization of the simulation, with all the files to refer to.
- file **.ned** which includes the scenario to be run during the simulation.

Running a simulation

The files created in the previous section will then be added in case it is considered necessary to use a particular map. Once this is done, in case you want to run the simulation you need to follow the steps shown in the following figure.

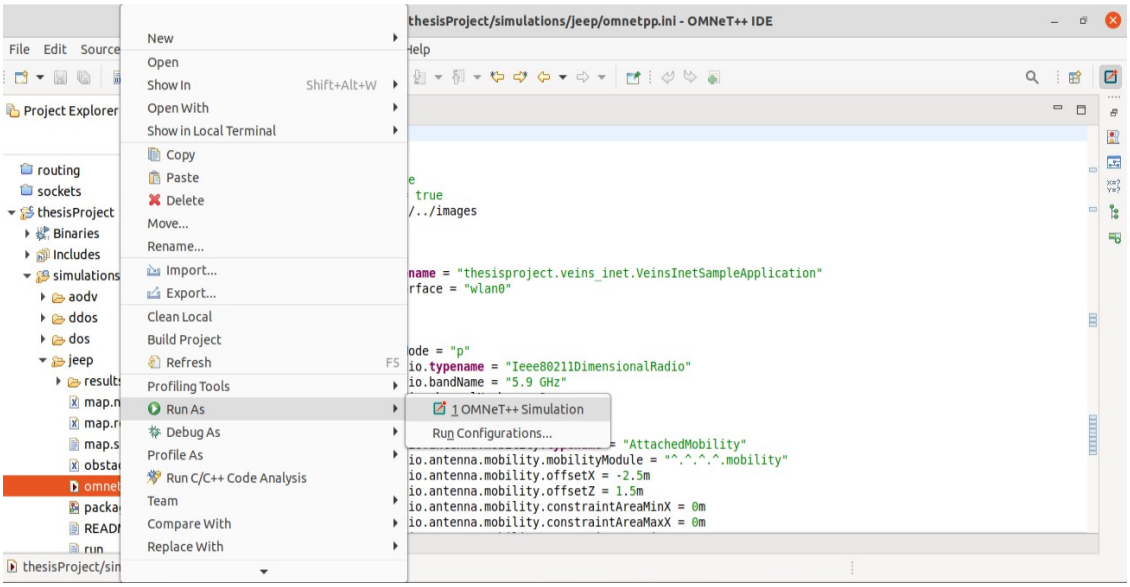

Figure B.2. Launching the simulation.

SUMO and Qtenv simulation windows

Once launched, just choose which scenario to run (if more than one was created in the same .ini file) and then reach this window.

Then clicking on the run button the simulation will start and it will be visible on

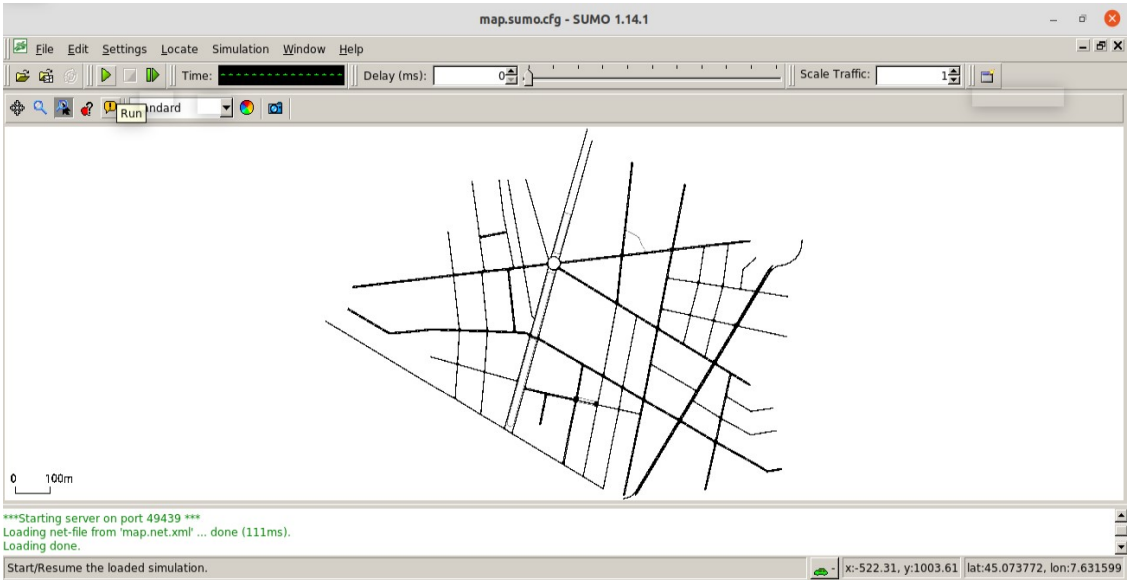

Figure B.3. SUMO window.

both SUMO and Qtenv.

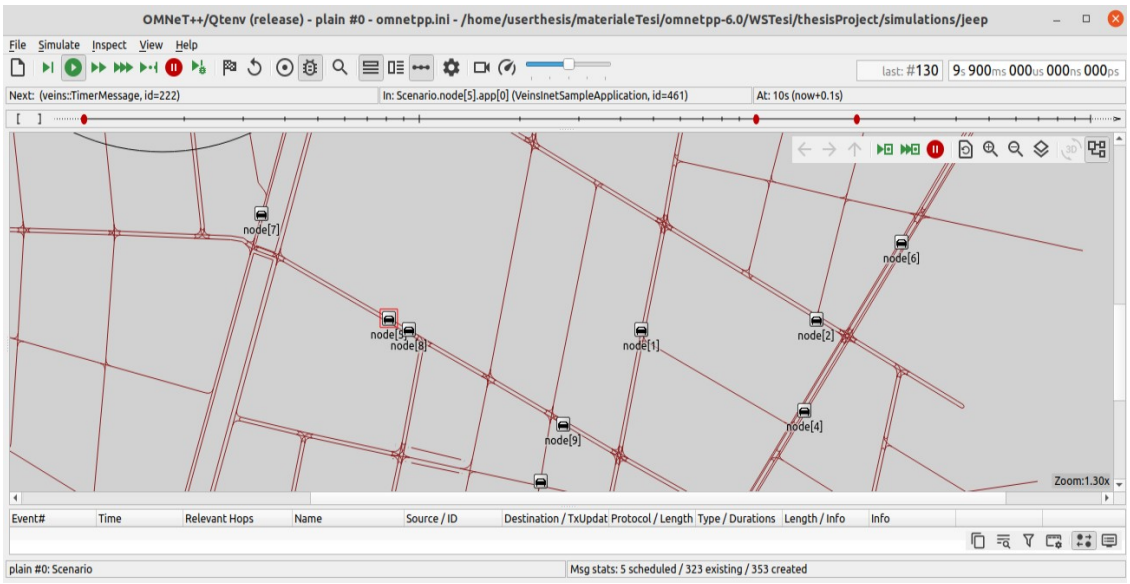

Figure B.4. Running of the simulation.

Bibliography

[1] Article on autonomous driving, https://www.repubblica.it/tecnologia/2022/06/16/news/fake_news_la_guida_autonoma_di_tesla_la_piu_pericolosa-354179281/

[2] Myths about autonomous driving by Audi, https://www.audi-mediacenter.com/en/press-releases/myths-about-autonomous-driving-14729

[3] NIST Special Publication 1500-201, *"Framework for Cyber-Physical Systems: Volume 1, Overview"*, https://doi.org/10.6028/NIST.SP.1500-201

[4] Guturu(1), Bharat Bhargava(2), *"Cyber-Physical Systems: A Confluence of Cutting Edge Technological Streams Parthasarathy"*, (1)University of North Texas, Department of Electrical Engineering, 1155 Union Circle 310440, Denton, Texas 76203-5017, USA guturu@unt.edu (2)Purdue University, Department of Computer Sciences, 305 N. University Street, West

[5] S. Al-Sultan, M.M. Al-Doori, et al., *"A comprehensive survey on vehicular Ad Hoc network"*, Journal of Network and Computer Applications, vol. 37, pp. 380-392. Elsevier, February 2013

[6] Prabhakar D. Dorge, Dr. Sanjay S. Dorle, Megha B. Chakole, *"Implementation of MIMO and AMC Techniques in WiMAX Network based VANET System"*, I.J. Information Technology and Computer Science, 2016, 2, 60-68, DOI: 10.5815/ijitcs.2016.02.08

[7] Article on Road Side Unit, https://medium.com/predict/edge-computing-is-so-much-more-fun-ac2a8a23e696

[8] Zhaojun Lu, Gang Qu, Senior Member, IEEE, and Zhenglin Liu, *"A Survey on Recent Advances in Vehicular Network Security, Trust, and Privacy"*, IEEE transactions on intelligent transportation systems, vol. 20, no. 2, February 2019

[9] DSRC definition, https://en.wikipedia.org/wiki/Dedicated_short-range_communications

[10] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil, *"Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions"*, IEEE Communications surveys & tutorials, vol. 13, no. 4, fourth quarter 2011

[11] Qiong Yang, Lin Wang, Weiwei Xia, Yi Wu, Lianfeng Shen, *"Development of On-Board Unit in Vehicular Ad-Hoc Network for Highways"*, 2014 International Conference on Connected Vehicles and Expo (ICCVE) National Mobile Communications Research Laboratory Southeast University Nanjing, China

[12] Muhammad Naeem Tahir , Pekka Leviäkangas, Marcos Katz, *"Connected Vehicles: V2V and V2I Road Weather and Traffic Communication Using Cellular Technologies"*, Centre for Wireless Communications (CWC)-Networks and Systems, University of Oulu, FI-90014 Oulu, Finland

[13] CAN definition, https://it.wikipedia.org/wiki/wikipediaController_Area_Network

[14] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo, *"Cyber-Physical Systems Security—A Survey"*, IEEE Internet of Things journal, vol. 4, NO. 6, December 2017

[15] Distinction between safety and security, https://poliziamoderna.poliziadistato.it/articolo/3535e185b1e70134099281328

[16] RFC 4949, https://datatracker.ietf.org/doc/html/rfc4949

[17] Weakness definition, https://csrc.nist.gov/glossary/term/weakness

[18] Exploit definition, https://en.wikipedia.org/wiki/Exploit_computer_security

[19] Attack vector definition, https://www.sumologic.com/glossary/attack-vector/

[20] Amit Kumar Goyal, Arun Kumar Tripathi, Gaurav Agarwal, *"Security Attacks, Requirements and Authentication Schemes in VANET"*, 2019 2nd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)

[21] Parul Tyagi, Dr. Deepak Dembla, *"Investigating the Security Threats in Vehicular ad hoc Networks (VANETs): Towards Security Engineering for Safer on-road Transportation"*, JECRC University Jaipur, India

[22] Guidelines for Securing Radio Frequency Identification (RFID) Systems, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-98.pdf

[23] Rajdeep Kaur, Tejinder Pal Singh, Vinayak Khajuria, *"Security Issues in Vehicular Ad-hoc Network(VANET)"*, proceedings of the 2nd International Conference on Trends in Electronics and Informatics (ICOEI 2018) IEEE Conference Record: # 42666; IEEE Xplore ISBN:978-1-5386-3570-4

[24] Irshad Ahmed Sumra,Iftikhar Ahmad, Halabi Hasbullah, Jamalul-lail bin Ab Manan, *"Classes of Attacks in VANET"*, Computer and Information Sciences Department Universiti Teknologi PETRONAS Bandar Seri Iskandar 31750, Tronoh, Perak, Malaysia

[25] Bassem Mokhtar - Alexandria University, Mohamed Azab - Virginia Military Institute, *"Survey on Security Issues in Vehicular Ad Hoc Networks*, Article in AEJ - Alexandria Engineering Journal - August 2015

[26] Prof. Ajay N. Upadhyaya, Dr. J.S. Shah, *"Attacks on VANET security"*, Computer Engineering Department, Faculty of Technology, RK University, Rajkot, India

[27] Wormhole attack definition, https://www.sciencedirect.com/topics/computer-science/wormhole-attack

[28] Mithun Sahay Shrivastava, Ravi Khatri, Anand Singh Bisen, *"Hybrid Approach for detecting and preventing DOS attack in VANET"*, International conference on Signal Processing, Communication, Power and Embedded System (SCOPES)- 2016

[29] M. Poongodi, V. Vijayakumar, Fadi al-Turjman, Mounir Hamdi (Fellow, IEEE), and Maode Ma (Senior Member, IEEE), *"Intrusion Prevention System for DDoS Attack on VANET With reCAPTCHA Controller Using Information Based Metrics"*, special section on secure communication for the next generation 5G and IoT networks

[30] Krzysztof Stepie´n, Aneta Poniszewska-Mara´nda, *"Security methods against Black Hole attacks in Vehicular Ad-Hoc Network"*, Institute of Information Technology Lodz University of Technology Lodz, Poland

[31] Salman Ali Syed, B.V.V.S Prasad, *"Merged technique to prevent SYBIL Attacks in VANETs"*, Department of Computer Science, College of Sciences and Arts Jouf University, Tabarjal , KSA

[32] Robert A. Sowah, Kwadwo B. Ofori-Amanfo, Godfrey A. Mills, Koudjo M. Koumadi, *"Detection and Prevention of Man-in-the-Middle Spoofing Attacks in MANETs Using Predictive Techniques in Artificial Neural Networks"*, https://www.hindawi.com/journals/jcnc/2019/4683982, Journal of Computer Networks and Communications, 2019

[33] Hannes Hartenstein, Kenneth P. Laberteaux, *"A Tutorial Survey on Vehicular Ad Hoc Networks"*, IEEE Communications Magazine, June 2008

[34] Jeep Cherokee hacked, https://www.quattroruote.it/news/nuove_tecnologie/2015/07/22/cybersecurity_valasek_e_miller_hackerano_una_jeep_cherokee.html

[35] Volkswagen Audi hacked, https://www.bleepingcomputer.com/news/security/volkswagen-and-audi-cars-vulnerable-to-remote-hacking/

[36] Prius hacked, https://news.sky.com/story/car-hackers-take-control-of-toyota-prius-10439089#

[37] Mercedes-Benz hacked, https://techcrunch.com/2020/08/06/security-bugs-mercedes-benz-hack/

[38] SUMO definition, https://en.wikipedia.org/wiki/Simulation_of_Urban_MObility

[39] OMNeT++ definition, https://omnetpp.org/intro/

[40] Julia Silva Weber, Miguel Neves and Tiago Ferreto, *"VANET simulators: an updated review"*, Journal of the Brazilian Computer Society (2021) 27:8, https://doi.org/10.1186/s13173-021-00113-x

[41] Dharmaraja S, Vinayak R, Trivedi KS, *"Reliability and survivability of vehicular ad hoc networks: An analytical approach."*, Reliab Eng Syst Saf 153:28–38

[42] Eclipse IDE website, https://www.eclipse.org/ide/

[43] SimuLTE framework, https://simulte.com/

[44] OpenStreetMap site, https://www.openstreetmap.org/

[45] Distributed Denial of Service cost, https://www.infoced.net/?p=1355

[46] Abdullahi Chowdhury, Gour Karmakar,Joarder Kamruzzaman,Alireza Jolfaei and Rajkumar Das *"Attacks on Self-Driving Cars and Their Countermeasures: A Survey"*, School of Engineering, IT and Physical Sciences, Federation University Australia, Ballarat, VIC 3350, Australia publication November 16, 2020

[47] Shivam Kumawat, Harneet Kaur, Omdev Dahiya *"An Analytical Study on Intrusion Detection System in Integrated Vehicular Ad-Hoc Network Attacks"*, 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)

[48] Ammar Haydari and Yasin Yilmaz, *"RSU-Based Online Intrusion Detection and Mitigation for VANET"*, Sensors 2022, 22, 7612, https://doi.org/10.3390/s22197612

[49] Ahmad Arsalan, Rana Asif Rehman *"Prevention of Timing Attack in Software Defined Named Data Network with VANETs"*, 2018 International Conference on Frontiers of Information Technology (FIT)

List of Figures

List of Tables