Vulnerabilities of autonomous vehicles and possible security solutions

Ferreira, F.
Project Manager
Master's degree in Cybersecurity
University of Aveiro
Aveiro, Portugal
fabio.f@ua.pt

Folgado, J.
Development
Master's degree in Cybersecurity
University of Aveiro
Aveiro, Portugal
joao.folgado@ua.pt

Caramalho, M.
Tester
Master's degree in Cybersecurity
University of Aveiro
Aveiro, Portugal
marcos.caramalho@ua.pt

Although road accidents have been decreasing since 1970 in Portugal, mostly because of improvements in the quality of roads and vehicles, accidents remain an important cause of non-natural death, where human error assumes an important role. The development of artificial intelligence, machine learning and computer vision, used in the construction of autonomous vehicles, allowed to mitigate human error, and substantially increase road security using tools like driving assistance, V2X communications and crash avoidance systems. However, with these technologies and the fact that they are connected to a network system, came some vulnerabilities like security on traffic control, potential GPS spoofing and several types of attacks against V2X technologies that can cause data loss, component failure as well as some damage to the environment/infrastructures. Based on these issues, it is argued that, concerning to networks security issues, it is crucial to search for robust, feasible and reliable solutions like the development of antennas and the encryption of GPS signals to mitigate GPS spoofing, using IDS/IPS technologies trained by artificial intelligence in an on-board V2X system and, also of importance, overall education and empowerment of society, businesses and workers, as well as adaptability mechanisms of the vehicles, all contribute to the increase of traffic security.

Keywords— autonomous vehicles, software security, V2X communications, GPS spoofing, security lifecycle

I. Introduction

Security threats to V2X systems depend on the attacker's capabilities and methods available to access the target and the type of attacks depends on different incentives to destabilize V2X systems; these can be physical damage/vandalism (e.g., denial of service, causing an accident), financial incentives (e.g., steal user's private information) or non-monetary incentive (e.g., improved attacker reputation).[1]
Taking this into account, the solutions for the main problems that we present in this article must attend to some Software Quality Attributes like confidentiality, integrity, non-repudiation, and a few more.
One of the key challenges in V2X communication security is in verifying the authenticity and integrity of exchanged messages, which is evident in GPS spoofing because the GPS signal is an open signal, without encryption, which is what makes it vulnerable. TerraPoiNT©, a company from NextNAV offers encryption in their protected antennas, making its signals extremely difficult to reproduce or falsify, stepping by on the confidentiality problem.[2]
Also, in V2X technologies attacks can compromise the integrity of data, introducing fake data or altering it, e.g., giving wrong data about traffic control provoking accidents, or stealing users' private information. As another example, an impersonation attack consists of a malicious entity that connects with the host vehicle with false identification via spoofing, then it establishes a communication channel, sending the malicious code and receiving the sensitive data capturing it, subsequently logging, and storing this data. For this reason, IDS/IPS (Intrusion Detection System / Intrusion Prevention System) trained by artificial intelligence, can be an effective solution.[1]
V2V communications occur in the form of a continuous broadcast of Basic Safety Messages (BSMs), which contributes to reducing vehicle crashes by 80%, however, the correctness and reliability of BSMs are of prime importance as they directly affect the effectiveness of safety applications based on them. This authenticity problem is solved by the digital signature. The receiver evaluates each message, verifies the signature, and then decides whether a warning needs to be displayed to the driver or just ignore it.[3]
The last example is also a solution for guaranteeing compliance, ensuring that only individuals with the appropriate credentials can access the secure systems and databases. Public-Key Infrastructure (PKI) manages digital certificates that are necessary for building trust among participants and for the proper functioning of the system. The Security Credential Management System (SCMS) implements a PKI that has several novel cryptographic constructs to provide a high level of security and privacy to the users, while keeping the system very efficient.[3]

II. Secure Life Cycle Description

A. *Education and awareness*

The awareness shall be brought by a central entity, preferably part of the country regulation system, which creates new security standards/guidelines for software development on cars and country road signals and road rules. (The condition of signals and road lines shall be maintained regularly). Regulations regarding the management of new vulnerabilities should make it mandatory for the automotive companies to provide the proper information regarding the vulnerability and follow the fix in time (More Awareness and knowledge sharing). Mandatory audits of a Certification Authority to the software that is directly or indirectly supporting the technology stack (More Education through legislation.). Software robustness mindset shall be the basis in every automotive development area. Training is also important, the technical teams shall be trained in developing and operating V2X systems, as well as security topics, to gain the ability to analyze and avoid security issues. The police forces shall also be trained regarding this new use case when approaching an autonomous driving car.

B. *Project inception*

The project inception is the early phase of any project where the software process is defined, requirements are gathered, and the planning is designed. The main scope of this project is to make the V2X technology more secure, the transversality of the vulnerable use cases is shocking, making this project highly valuable and reasonably profitable. To better describe it, it was decided to break it into three different subcategories, the “Definition of the software process”, the “Requirements Engineering” and finally “Planning and Budgeting” accordingly.

1) *Definition of the Software Process*

There are several software development frameworks, the one chosen for this project was the SDLS V-Model.[4] This software development model breaks a project into well-defined stages. Whenever the team completes a stage, tests are performed to assess functionality and validate user and client requirements. The model allows the team to identify issues with each module of the program and fix them proactively, rather than waiting until after every aspect of the project is complete to perform product-wide tests. This allows the team to save time and resources since the final product should be less buggy. There are some benefits in using this V-Model framework, being the most important ones the following: V-Model provides clear guidelines through focusing on one phase at a time, the simplicity and straight forward understanding of the framework strategy for the developers and manager alike; it makes complex projects easy to manage and track; review and test mindset throughout the entire process; and finally includes thorough documentation of each phase, promoting quality gates in each iteration.

2) *Requirements Engineering*

The requirements of the project are collected by analyzing the use case. During this phase it is established what the ideal system is, however, it does not determine how the software will be designed or built. Usually, stakeholders are involved to better understand the real needs and the different scenarios that our product will face during activity. The user requirements will typically describe the system's functional, non-functional, interface, performance, data, and security as expected by the user. The users carefully review this document, as this document would serve as the guideline for the system designers in the design phase. After all requirements are clear, there are several meetings where every requirement is analyzed and reviewed by all the engineering teams which will evaluate each one of the features translated from the requirements provided earlier.

3) *Planning and Budgetting*

In this step, a list of effort estimations shall be done that results in a proper description of how much person/day effort shall all the implementation of the requirements cost to the company. Having this in mind we are in a good position to propose a final budget to the customer and negotiate expectations of the delivery date of our product. It’s not unusual to also negotiate some ‘code drop’ dates where integration and consequently field tests can be performed by the customers with demos. This promotes the quality of the final product and controls the expectation of the customer, providing also vital information for the developer team in case some problem occurs.

4) *Key Performance Indicators*

A Key Performance Indicator, or KPI, is a type of performance measurement that serves the purpose of evaluating the success of a particular activity. In the scope of this document, our KPIs are based on what are the absolute metrics that need to be met when it comes to autonomous vehicles, with large focus on reliability and making sure crash control.

These are the main KPIs that are considered imperative:

- Trajectory alignment of the autonomous vehicle should be 99.99% reliable.
- Sensor reliability should be from 99.00% to 99.99% with latency between 3ms and 10ms.
- Minor crashes rate should be below 5.00%.
- Mortal crashes rate should be 0.00%
- Reaction distance at a speed of 120 km/h should be 1.5 meters, for a braking distance of 75 meters.

C. *Analysys and requirements*

1) *Analysis*

When analyzing the current landscape of AVs (Autonomous Vehicles), there’s still a lot to be desired, mainly in the way software security is approached, and the multiple possible attack vectors these entities entail. A lot of modern cars, and certainly AVs, have what are called over the air updates, meaning these vehicles have a constantly opened channel with the outside, they’re usually exposed to the internet somehow, and although some brands do employ the best security standards for these types of communication, most other brands do not. To ensure a proper standard, a complete threat analysis shall be performed to identify all potential threats and possible solutions. All of these bring us to the first two requirements, software robustness and secure communication channels only.

With AVs, another major concern is the fact that, in the future, inter-connectivity will reign, and these vehicles will not only be constantly communicating with each other, but they’ll also gather information from the environment, e.g.

traffic control (lights, cameras, etc…). These shall be tackled in a major requirement, securing environmental feedback.

2) *Requirements*

Software robustness: All software should follow all robust software rules, without any exceptions. Everything on an autonomous vehicle is accessible, whether it be over the air, or physically (harder for an attacker, but possible, especially when it comes to taxi-like cars). To mitigate the physical part, every kind of communication channel should be accessible with special hardware only, and preferably, out of a normal user's reach.

Secure Communication channels only: Whether we're communicating with the AV via a wireless channel, or physically, all communications need to be encrypted and any commands that trigger physical events (steering, breaking, accelerating, etc...) should be signed and only allowed by authorized parties.

D. Architectural and detailed design

1) *GPS Spoofing*

Autonomous vehicles rely a lot on orientation systems, such as GPS, which is possible to spoof, potentially leading an end-user to a malicious entity pre-defined place. To secure this, all AVs should implement measures to detect spoofing. At the time of writing, there's no perfect solution agreed upon everyone, however, design wise, it's preferable that any mitigation endeavors are done at the hardware level.

2) *Vehicle-To-Everything (V2X)*

Unless a clearly better model arises, V2X should be used, as its main motivations are road safety, traffic efficiency, energy savings and mass surveillance. Newer implementations of autonomous vehicles can use either the WLAN or Cellular variant, as there's some dispute to which one is better, but no clear winner yet.

3) *Hardware Communication channels should be secure against third parties*

Although there's no perfect system and everything's hackable, AVs should follow the best practices that are sometimes disregarded, such as making sure any hardware ports that allow communication with the software are protected and only useable by certified and authorized entities. An example of this is to have something connected to these ports constantly, making sure that when these are disconnected an alarm is triggered, which can only be disarmed by the certified user.

4) *Critical communication parts should be modular*

Any communication part with the exterior that's somehow critical, such as GPS receivers/emitters, or even communication ports and receivers/emitters (Wifi) should be replaceable easily, making sure that even older models of AVs can get security upgrades in the form of improved hardware. Of course, mechanisms should be added that make the cars only work with certified hardware.

5) *AI Models*

The best way to have improved autonomous vehicles is to make sure the models that drive these are constantly improving. This is only achievable with huge amounts of driving experience; therefore, all AVs should monitor as much as possible and constantly feed this to whatever model is used to make sure it improves. Precautions must be taken to never associate this data with any one person/car.

E. Implementation and testing

As evident, before testing even starts, it's imperative that a secure implementation is in place, following the industry's best-practices, mainly when it comes to security. After a strong basis of confidence in the implementation is acquired, thorough testing can start.

Dynamic analysis security testing (DAST) is of the greatest importance to guarantee strong security in our software since these tests can identify application errors. DAST consists of a black box testing technique used to detect security faults by performing a simulation attack, constituted by a team of professionals' penetration testing, that will constantly try to attack and perform all types of tests to the autonomous vehicles (AV) system to identify a different behavior from the expected and guarantee that it cannot be easily attacked by the intruder and to identify all types of vulnerabilities on V2X onboard.

However, it is advisable that tests of rogue data, sent to the AV are tested intensively with many diversified tests to see how the behavior of security agents IDS/IPS are installed on board. It is of the most important that AV receives attacks not previously tested in the AI learning phase to see if the actual model works properly.

Regarding GPS spoofing it is highly important to have a team monitoring the antennas around the cities to verify if they are working properly and respect all requirements of that antenna type.

Finally, it is also important to test every year the behavior of the AV in a close environment as well as create regulatory identity to perform external audits to guarantee maximum security accordingly with requirements.

F. Release, deployment and support

1) *Standard Incident Response*

Every incident should be categorized with a level of severity (low, medium, high, critical) and anything that can be used to improve the overall landscape of AVs should be documented, communicated, and fixed, potentially with the help of other entities. Ideally, there would be an external institution that would take care of properly propagating important information to all parties, making this approach very similar to what happens in aviation already. The main idea here is that accidents should only happen once, at most, with safety as a priority.

2) *Vulnerability disclosure*

Any vulnerability found should be disclosed as soon as possible to all relevant parties somehow connected to the life cycle of an AV. This ensures that the whole of the autonomous vehicles industry improves linearly.

3) *Release, Deployment, Support*

As AVs are in an early stage, support should be given 24/7 to all relevant parties, whether it be end users or, mainly, law enforcement. As we've seen previously, autonomous vehicles that use the same roads as normal vehicles will sometimes get caught in less than perfect maneuvers, potentially being picked up by the police; making it a need to have a communication channel with them, when needed, to explain and educate on the matter. Release and deployment of anything that's not hardware related should follow the best practices of software development cycles, this is possible because most of these vehicles are updateable over the air. Hardware updates, if critical, should be done as soon as possible, recalling any potentially impacted vehicles/parts in circulation.

III. Requirements

Confidentiality			
Code	Description	Comments	Priority
REQ-001 (Testable)	The V2X system shall use end-to-end security for communication to external entities	The V2X system shall use one end-to-end security envelope per message according to TS 103 097 V1.3.1: Security Header and Certificate Formats. This essentially means communications with any entity external to the AV, such as road traffic systems, should be secure	1 (HIGH)
REQ-002 (Testable)	The V2X system shall support services for confidentiality. To ensure this, all the information transmitted from and to the client is sent through SSL tunnels, which require a certificate for authentication and authorization	Certificate requests and responses are encrypted and only the intended recipients may decrypt them.	1 (HIGH)
REQ-003 (Testable)	It shall not be possible for an unauthorized entity to analyze the identification of a person through personally identifiable information (PII) within communication messages such as the location or driving route of a particular person	As a third party, it shouldn't be possible to analyze the identification of a person through PII communication messages.	1 (HIGH)

Availability			
Code	Description	Comments	Priority
REQ-004 (Testable)	The V2X system's Decentralized Congestion Control (DCC) mechanism shall be compliant to C2C-CC White Paper Decentralized Congestion Control (DCC) for Day One	Ideally, these congestion control methodologies should be applied in software, to allow for easier updateability, such as using the DSRC-tailored J2945/1 congestion control algorithm.	2 (MEDIUM)
REQ-005 (Testable)	The V2X system shall send and receive messages at an acceptable latency. As per the V2X specification, acceptable latency for message exchanges is 100ms maximum, except for pre-crash sensing messages, capped at 20ms maximum.	As an example, a forward collision-warning system message should be transmitted to an incoming vehicle before the accident occurs, within 20ms at most. To achieve low-latency communications, the message size should be between 50-300 bytes length.	1 (HIGH)
REQ-006 (Testable)	The V2X shall provide the means for an entity to process exchanged information in	To process information in real-time, encryption algorithms must be lightweight. AES in CCM	2 (MEDIUM)

	real-time, for this, the implementation of low-overhead encryption is mandatory, which is achieved by using AES encryption (CCM mode) on small messages, between 50-300 bytes.	mode is a good algorithm to use, as it is highly secure and performant, with the possibility of implementing it in hardware as well as software	

Integrity			
Code	Description	Comments	Priority
REQ-007 (Testable)	The V2X system shall only forward verified messages	The V2X system shall only forward verified messages in the ITS-G5 network	1 (HIGH)
REQ-008 (Testable)	The signature in the end-to-end security envelope shall be generated by using a private key corresponding to a valid authorization ticket (pseudonym certificate)	The signature in the end-to-end security envelope shall be generated by using a private key corresponding to a valid authorization ticket (pseudonym certificate) according to clause 7.2.1 in TS 103 097 V1.3.1: Security Header and Certificate Formats	1 (HIGH)
REQ-009 (Testable)	The verification of a message shall comprise cryptographic verification of the message’s signature.	When verifying any message, a cryptographic signature must exist and be validated	1 (HIGH)
REQ-010 (Testable)	The V2X system shall only send CAM messages (Cooperative Awareness) with valid position and time	It shall be possible to detect when fake location or data, such as time, is being transferred and in these cases, the messages shall not be sent	2 (MEDIUM)

User			
Code	Description	Comments	Priority
REQ-011 (Testable)	The driver shall be informed in advance about the expiration of the Long-term Certificates, 1 month before they expire, and again 1 week prior	A warning shall be implemented to notify the car owner regarding the expiration of certificates beforehand.	2 (MEDIUM)
REQ-012 (Testable)	The driver shall be informed in advance about the expiration of the Pseudonym Certificates, 1 month before they expire, and again 1 week prior	A warning shall be implemented to notify the car owner regarding the expiration of certificates beforehand.	2 (MEDIUM)

Development Process/Organization			
Code	Description	Comments	Priority
REQ-013 (Analyzable)	The organization shall implement the V-Model development process	The V-Model introduces a test mindset approach, where the implementation of each feature is thought alongside the test that will assure the quality of the requirement developed.	1 (HIGH)
REQ-014 (Analyzable)	Each requirement to be implemented shall be evaluated by a team of Software Experts	To have a clearer and accurate requirements list, each request must go through the expert's quality gate before being considered as a valid requirement	2 (MEDIUM)
REQ-015 (Analyzable)	An evaluation checklist sheet shall be implemented at the end of the process. This list shall be written first by the software experts and then validated by all testing parties involved	An official document shall be delivered and signed by both parties, customer, and software provider. This document shall ensure all the quality measures taken and evaluation process. The quality department shall review this before the official release	1 (HIGH)

IV. Security Test Plan

A. Penetration Testing

Provided the Autonomous Vehicle has proper defenses against physical attacks, e.g., with USB thumb drives containing malicious payloads, most of the vulnerabilities will be within the scope of communication. AVs can communicate with other vehicles (V2V), or with the environment (V2X) and it's important for these communications to be secure against tampering, or even eavesdropping. As such, penetration testing should focus on this, by using man-in-the-middle (MITM) attacks to try and eavesdrop communications or tamper with the payloads that are exchanged, such as changing positional and time data.
The following attacks should be done:

- Eavesdrop messages between vehicles.
- Eavesdrop messages between vehicles and the environment, such as traffic sensors/cameras.
- Tamper with the exchanged data on both the scenarios above, for instance, by changing the positional data, which could be catastrophic under certain situations, i.e., two cars communicating with each other crash because they miscalculate distances between them.
- Intercepting messages and delaying them.
- Intercepting messages and dropping them.

To achieve the tests above, the main tool(s) will be hardware based, since these communications happen wirelessly, a wi-fi pineapple (tool used to perform targeted MITM attacks) can be used. Solving MITM attacks in all instances may be impossible when there's a dependency on external factors, i.e., V2X communications, but everything found should be reported thoroughly, as some mitigations may be achieved by the engineering team.
There is one more vector that is paramount to the security of the whole system, the humans involved. While developing an Autonomous Vehicle, blueprints, protocols, code, credentials, and other sensitive information will be stored in a centralized fashion, accessible by the engineering team and other members. It's important to test them and make sure that even if a mistake is made by humans, an attacker still doesn't get access to critical information. To achieve this, there should be a suite of carefully crafted phishing attacks and poisoned links targeted at all relevant emails and public communication channels, with a focus on the education of this workforce, making them less prone to clicking or saying what they shouldn't. If an attacker were to get, for instance, a partial/full copy of the codebase, it could be possible to craft a malicious payload to hack into the AV, via multiple entry points, some of them discovered by fuzzing the system. Also, an entire network probe should be done to the relevant servers, although AVs can be self-contained, chances are there will be services running on private servers which can be vulnerable. These vulnerabilities can be found and exploited with tools such as Metasploit.

B. Fuzzy Testing

Fuzzy Testing, often known as fuzzing, involves injecting incorrect or random data (FUZZ) into a system to find coding errors and security flaws. The testing phases are determining the system to be targeted, determining the inputs, producing fuzzed data, running the test with ambiguous data, analyzing the system's performance, and keep a defect log. In the context of this document's scope, we will need three essential components: a message generator, a message publisher, and a target monitor.

For that, we want to present two strategies that can use an existing tool like DreamView (referred further), adapted to the context of our autonomous vehicles. Alternatively, it may be helpful to create small, dedicated tools to assist in this.

The first strategy consists in having two different types of message generators. First, one of them originates completely random messages, then a second one that promotes unexpected alterations in valid notifications with the primary objective of testing the behavior of an AI model responsible for the IDS/IPS, using tools like AFL. For this, one must create many attack vectors that simulate values/messages sent by other vehicles, infrastructures (such as traffic lights, crossings, buildings, etc...), and even by the own vehicle, like distance from another vehicle(s), the temperature inside the car, or humidity. All this data should be analyzed simultaneously by the AI model for testing its behavior. In the following subsection, we will announce some attack vectors. Finally, the third message generator should create some valid data that randomly makes alterations in their size and encoding to test the system's behavior in dealing with this data.

The second strategy consists of testing invalid inputs like improbable or not-so-common situations, for example, a plastic bag flying against the car, some fallen stones on the road, or an animal that suddenly appears in front of a vehicle.
The primary purpose of these tests is to analyze the vehicle's behavior towards these improbable situations that often fail to be tested.
Because these tests are dangerous when simulated in real life, software with an integrated graphic interface must test and manufacture all these situations and show the vehicle's behavior and response. An example of this kind of software is DreamView, developed by Apollo which is a web application that visualizes the current output of relevant autonomous driving modules (planning trajectory, car localization, chassis status, etc...), provides a human-machine interface for users to view hardware status, turn on/off modules, and start the autonomous driving car. It also provides debugging tools, such as PnC Monitor to track module issues efficiently. This software can be found on https://github.com/ApolloAuto/apollo/blob/master/docs/specs/dreamview_usage_table.md

If budget allows, the ideal scenario is to build a custom made tool like DreamView, tailor made for the AV's in testing.
After thorough testing in simulations, real-life testing should be done, following the same scenarios as in the simulators.

In the autonomous driving industry, a solution to the test case generation problem is achieved through fuzzy testing, which

can create thousands of scenarios and produce and test different attack vectors.
These scenarios can be simple or complex by adding simulated pedestrians and motorcycles, altering the timing of traffic lights, or the speed of oncoming vehicles to see how that changes the driving. Scenarios thus generated could represent a variety of realistic traffic and driving conditions, including extreme weather driving.
The vector attacks can be by positive testing, which aims to ensure that the system under test performs correctly for the expected input, or it can be negative testing, which attempts to ensure that the vehicle does not misbehave when the input data is invalid or unexpected, ensuring the safety and robustness of autonomous driving, as the figure X illustrates.

Some use cases to test with attack vectors are:

1) Reacting to several static and moving objects in front of the vehicle

The vehicles must detect either static or moving objects in the sense of safety. The car must detect the distance between it and the walls, pedestrian roads, or other moving objects to constantly adjust its trajectory, assuring that its course is centered on the road. In the context of moving objects, it is essential that the vehicle continuously analyzes information about the environment to detect any object's movement that can cause an accident, e.g., people crossing the road outside crosswalks.

As an example of testing behavior via simulation, one can add a human or an animal as an attack vector to the simulation and see if it is detected anywhere along the path. In this case, the vehicle should be capable of following that movement, slowing down its speed, if necessary, until that object, person, or animal stays at a sufficient distance considered safe.

This behavior must be also tested with metamorphic fuzzy testing (MFT)[5] The idea is to generate random obstacles of various types and publish them on the vehicle's path simulation to test situational awareness on events that aren't common.

2) Vehicle's capabilities for monitoring free space in a T-intersection, at a pedestrian crossing, and a bus stop

It's essential that in situations that are more likely to occur in accidents, the vehicle slows down and analyzes the environment carefully. In a T-intersection, for example, the vehicle should be able to receive information about both sides of the car, information about circulating vehicles, and their speed to know when the AV can cross safely, regardless of legal priorities. Pedestrian crossing and bus stops are other situations where the car should travel at a lower speed to ensure they can safely stop if urgent braking is necessary. Sometimes some bus stops are along the road but separated from the principal highway, allowing intersections, thus, the autonomous vehicle should be able to detect these portions of the road that are common for both cars and buses to be aware of a possible collision.

3) Capability to wait for other cars or pedestrians first to pass

Like the waiting time in the T-intersection to know when the AVs can pass carefully, in pass-through zones or pedestrian crossings, it's crucial that the vehicle receives information about movement in front, on the lateral, and behind the car to know when it can carefully advance.

4) Capability to deal with other vehicles infractions like circulating against traffic or dangerous overtaking

The vehicle must know the road code and basic rules of traffic, both to fulfill them and to understand the other vehicle's behavior. When other cars don't respect the rules and circulate on the opposite side, for example, the vehicle should be aware of possible collisions and take a decision to stop or proceed with caution, warning the other driver of his infraction. Autonomous vehicles should not only be aware of his own decisions and behavior on the road but of other vehicles too.

5) The behavior of the car in extreme meteorological conditions like intense fog or intense rain/snowing

AV's will not only be used on days with the ideal meteorological conditions; therefore, they must adjust their conduction to the external conditions and decide based on that. With intense fog, the car must turn on the fog lights, when the floor is frozen, he must adapt speed even when circulating inside the allowed velocity because the conditions of the road could not be ideal if an urgent braking must be done.

6) Capability to park by measuring the margins and distance to other vehicles accurately

When the vehicle must park, it must measure the margins and distance to other vehicles already parking in that spot to avoid crashes. Distance sensors should be installed in various points of the vehicle to facilitate the measurement and free space along the car. An extra feature to include that could be interesting is the possibility of the user deciding if he wants to park only in free parks or in a paid parking space, and the vehicle with information received by parking lots knows where he can park based on user preferences.

7) Behavior and decision-making of a car when the light sets on the panel indicating malfunctioning with the engine or other crucial parts of the vehicle

The autonomous vehicle should be able to receive and integrate information from the panel, engine, and other parts of the car in the decisions when driving. If a light appears in the panel indicating any kind of malfunctioning in the motor or other essential parts of the vehicle, it should analyze what possibilities it has and decide to potentially stop immediately, asking for vehicle assistance, or if the warning sign allows the car to proceed normally without compromising the safety of everyone on the road.

8) Capability to decide on the best road given the information about traffic conditions or information about an accident

Nowadays, some applications exist that allow GPS information to know the traffic conditions and give some alternative roads to avoid traffic and reach their destination faster. Integrating a GPS and software identical to these apps on autonomous vehicles can allow AVs to decide, based on traffic conditions or in an existing accident, the best path that should be taken to reach the destination in the shortest amount of time, not compromising safety.

9) Capability to decide if the AV has to stop or deviate from an obstacle regarding all sensed data

It is vital to test rare situations that require quick decisions. Imagine a situation where a car circulates behind our vehicle at a small distance and suddenly something appears (a person, an animal, or an object) in front of our car. It is imperative to determine and structure how the vehicle will respond, depending on what appears in front of a car doing a rapid calculation about the distance behind us in case of an emergency braking. If it is an animal or a human, the car must stop regardless of the inevitable crash in the back. On the other hand, if it is just a plastic bag or another object, the car can slow down but continue avoiding the crash with the other vehicle.

10) Capability to recognize emergency vehicles on movement

Autonomous vehicles should be aware of all components on the road, like infrastructures, pedestrians, and other vehicles, but should also consider that not all vehicles are equal. Towards an emergency vehicle on duty, the AV should deviate even if that means temporarily leaving the defined route and returning to the normal path after the emergency vehicle passes.

To test this use case, the possibility to add an emergence vehicle on the simulator must be created to allow testing the behavior of the AV in these situations. This can also be tested in real life where the attack vector is a real ambulance in emergency (of course, faking it).

For each example, the system must be guided by a collection of sensors, sophisticated computing systems, and an autonomous driving software stack to interpret the amount of information and decide based on that.

The autonomous vehicle AI Model shall also be tested. For this, the usage of an AFL tool for the input vectors is recommended.

11) Use sensor information such as position, velocity, meteorological data and distance as attack vectors

Program an AFL tool to publish random position, velocity, meteorological data, and distance then save all that data and the output from the AI model, evaluating if it has the right behavior

C. Requirements Tests

In this section, tests are specified, for each of our Requirements.

TST-01: On a controlled scenario, send a message to a test external entity, such as a listener, listening to V2X traffic on trace mode. Analyzing the traces, check if the ASN.1 formats are correct, as per the TS 103 097 V1.3.1 specifications, as well as checking if the data portions of the messages are not in plaintext - *REQ-001*
TST-02: From an untrusted source, send a V2X message to a trusted vehicle and check if the vehicle's communication system discards the message - *REQ-002*
TST-03: From a trusted source, send a V2X message to a trusted vehicle and check if the vehicle's communication system processes the message - *REQ-002*
TST-04: Test sending one of each message type, intercepting it, and seeing if there is no personally identifiable information. Check if it's impossible to identify a person from them - *REQ-003*
TST-05: Send an increasing number of messages, of various types and check if the congestion mitigation systems work, by making sure the system has a 100% uptime even when in load - *REQ-004*
TST-06: Send an increasing number of messages, on a V2V/X scenario and make sure the latency does not exceed 100ms. This test does not include V2V Pre-Crash sensing messages - *REQ-005*
TST-07: Test the latency of V2V Pre-Crash sensing messages and check if it doesn't exceed 20ms - *REQ-05*
TST-08: Test sending all types of messages and check if the max payload length does not exceed 300 bytes. Coupled with AES encryption (CCM mode), it should provide low-overhead encryption and guarantee low-overhead - *REQ-06*
TST-09: - *REQ-08*
TST-10: Send data to the CAM system with the wrong signature and verify that the system interpreted it as invalid. - *REQ-09*
TST-11: Send data to the CAM system with data and the corresponding signature and verify that the system accepts the information. - *REQ-09*
TST-12: Try to send all types of data possible through the CAM system and verify if the system detects that information as wrong. The data type includes speed, distance, velocity, position, time, signature car data, etc. - *REQ-10*
TST-13: Insert a certificate 30 days from the expiration date and see if a notification is dispatched to the user. - *REQ-11*
TST-14: Insert a certificate more than 30 days from the expiration date and check that no notification is sent to the user” - *REQ-11*

References

[1] Hasan, M., Mohan, S., Shimizu, T., & Lu, H. (2020). Securing Vehicle-to-Everything (V2X) Communication Platforms. IEEE Transactions on Intelligent Vehicles, 5(4), 693-713. [9068410]. https://doi.org/10.1109/TIV.2020.2987430

[2] Kumar, A.D., Chebrolu, K.N., Vinayakumar, R., & SomanK., P. (2018). A Brief Survey on Autonomous Vehicle Possible Attacks, Exploits and Vulnerabilities. ArXiv, abs/1810.0414

[3] B. Brecht et al., "A Security Credential Management System for V2X Communications," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 12, pp. 3850-3871, Dec. 2018, doi: 10.1109/TITS.2018.2797529.

[4] Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. International Journal of Information Technology and Business Management, 2(1), 26-30.

[5] Jia Cheng Han and Zhi Quan Zhou. 2020. Metamorphic Fuzz Testing of Autonomous Vehicles. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, 380–385.