Advanced CAN Bus Fuzzing Framework: Integrating Real-Time Visual and Log-Based Detection for Enhanced Automotive Security

Devangshu Mazumder

A Project Report Submitted to

The University of Birmingham
In Partial Fulfillment for the Degree of MSc Cyber Security

Supervised by

Professor Tom Chothia

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
September 2023-2024

Abstract

Electronic Control Units (ECUs) that are connected using Controller Area Network (CAN) bus are integral in powertrain applications in automotive industry. Yet, factors attributing to lack of security in CAN protocol has exposed these facilitated systems to quite a number of cyber threats. Most capsulation methods that are classified as fuzzing are antagonistic to system crash detection or any other inappropriate behavior among others and thus its insistence on the failure on the visual aspect. This project proposes a novel approach of fuzzing by combining video detection and computer vision into a CAN bus fuzzing environment. The architectural design of the applied fuzzing tool permits the collection of the views from the instrument cluster while the fuzzing experiments are being carried out thus providing an innovative method of interpretation that links the visual perspectives to some particular codes on the CAN bus.

The specifics of the project included obtaining a comprehensive test environment that consisted of a physical instrument cluster, CAN interface and video camera, in order to facilitate observing of how the fuzzing procedures affect this system. Along the way to achieving this goal, certain difficulties were faced, for instance, there was hardly available documentations for the said instrument cluster, and there was also the issue of video detection parameters needing accurate calibration. Nevertheless, the tool made sound use of such a functionality to detect a range of vulnerabilities by determining the correlation between images changes and the CAN messages. This is a great step forward in the automotive cyber security assessment as it introduces a better way of doing CAN bus fuzzing which can also be transferred to other vehicular subsystems.

This research addresses the fact that the deepening of measures utilized for fuzzing the CAN bus around the video detection enabled utilizing the deeper information about how the fuzzing attacks impact the automotive systems. This research opens the way for the further development of more advanced methods of video processing and extending the scope of this direction to other automotive protocols, building more robust and safe vehicle networks in the end.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Tom Chothia, for his invaluable guidance, support, and encouragement throughout this project. His expertise and insights were instrumental in shaping the direction and success of my work.

I also extend my thanks to Dr. Andreea Ina Radu, Pen Test Manager at TCS , Mr. Thompson Martin from ZF Technologies and Mr. John Wong from Jaguar Land Rover for providing expert advice and technical support. Additionally, I am grateful to my colleagues and friends for their continuous encouragement and constructive feedback during the project. Lastly, I thank my family for their unwavering support and patience.

Abbreviations

BCM	Body Control Module
CAN	Controller Area Network
ECU	Electronic Control Unit
FPS	Frames Per Second
GUI	Graphical User Interface
OBD	On-Board Diagnostics
PCAN	Peak Controller Area Network
LIN	Local Iterconnect Network
LTL	Linear Temporal Logic
UDS	Unified Diagnostic Services
USB	Universal Serial Bus
ZF	ZF Friedrichshafen AG
TCS	TATA Consultancsy Services

Contents

List of Figures

List of Tables

CHAPTER 1 Introduction

1.1 Background

As cars have become increasingly complex, Electronic Control Units (ECUs), such as the ones used in the engine, brakes and instruments control panels, have become increasingly common. These ECUs are linked through a communication bus known as the Controller Area Network (CAN) bus designed to facilitate communications among these units. Although integration with the CAN bus has brought in a leap in the sophistication of vehicle features, the technology as it is today offered does not incorporate the security feature.

Over the past decade, it has been noted that the automotive CAN network has serious weaknesses due to the increasing number of pesky cyber attacks. Can these vulnerabilities be exploited? If yes, what would be the consequences? Severe, as control over the vehicle is compromised or safety threats arise. Most of the conventional approaches to testing the security of CAN bus primarily works with traditional black box fuzzers where random inputs or malformed packets are sent to the network after which any resulting crashes or behaviors that are out of normal are noted. Nevertheless, these techniques usually underestimate the importance of the monthly instrumentatie providing feedback if the vehicle's instrument cluster were being fuzzed.

1.2 Problem Statement

The traditional attempts at fuzzing the CAN bus up to this point have been rather shallow and mostly concerned with the occurrence of internal system errors, ignoring the external visual signs of weaknesses. Such absence of visual feedback factor in the fuzzing methodology is a major drawback as it is not clear to the user how the vehicle responds to different CAN messages based on the driver's perspective. The goal of this project is to fill this void by extending video detection to CAN bus fuzzing, thus allowing the investigating of instrument cluster video feedback during tests. This new technique makes it easier to understand the relationship between structural and communicational parasitic loads, thus broadening the investigation scope of automotive security aspects.

1.3 Project Goals

The primary objective of this project is the design and development of a fuzzing framework for CAN bus containing an element of using real time video inspection to examine the results both from the internal logs and external perspective, the vehicle instrument cluster. Target in this case is to identify and record the particular CAN messages which cause demonstrable effects, like illuminated warning lights, movement of gauges, display messages, etc. Fourthly, due to accessories and environmental objects motion and background clutters, optimizing video detection algorithm is of importance to speed and accuracy.

The specific objectives of the have been defined as follow:

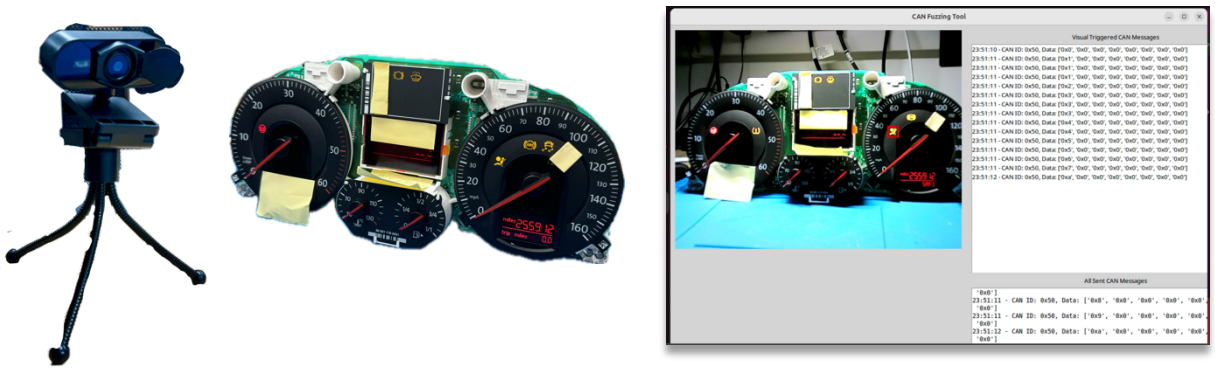


Figure 1.1: Introducing Video Detection in CAN bus Fuzzing

1. **Integration of Video Detection :** A video camera will be integrated into the existing system for practical exercises, which will record the instrument cluster in motion. Computer vision will be used to identify and store the changes.

2. **Development of a Fuzzing Framework**: A fuzzing tool for CAN communications will be created, which will allow sending any or several messages over the CAN bus and observing how the instrument cluster reacts

3. **Calibration of Detection Parameters**: Testing will be conducted to adjust the parameters for video detection such as threshold and min_area of the above components for good results.

4. **Evaluation and Analysis**: The tool will be evaluated and tested on a real physical instrument cluster and the results will be analyzed and systems vulnerabilities will be identified.

1.4 Contributions

Automotive cybersecurity research has been expanded in this project since there was a development of a fuzzing tool that evaluated the internal systems response but also collected and analyzed the feedback of the instrument cluster seen. This dual approach allows for a more holistic assessment of the effects of the CAN messages on the vehicles in a way that can be better than the other methods. The design of the tool is highlighted with the omission of easy to use graphical user interface (GUI) and user administration and multitasking for real time computing purpose, which made it possible for both the fuzzing and the video assessment processes to be executed simultaneously without reduction of effectiveness.

1.5 Report Structure

The report consists of eight interrelated parts explaining different components of the project. **Chapter 1**: Introduction states the background of the project, explains its problem, sets its goals and specifies the originality of this research works contributed towards the project.

Following this, **Chapter 2**: Literature Review examines the CAN protocol, provides the context for the problem of LTL-based fuzing, summarises the state of the art, and explains important theoretical issues such as how LTL is used in the project.

Thirdly, the book presents in more detail the reasons behind the need for hardware and software configuration in **Chapter 3**: Initial Setup and Implementation, as well as, the difficulties that were faced during the wiring of the instrument cluster and the early tests. Fourthly, **Chapter 4**: Design and Methodology then proceeds to delve deeper into the outlines of the fuzzing tool construction. It elaborates on how video detection was included in the design and what approaches were applied to perform in real time while ensuring detection accuracy.

In light of this, the ensuing section, **Chapter 5**: Experimental Setup and Testing, describes the test conditions, outlines the course of action taken, and analyzes the outcome of the effective and aggressive tests performed, as well as, the vulnerabilities and the unusual behaviors identified. This is then followed by **Chapter 6**: Evaluation and Discussion which focuses on the effectiveness of the fuzzing tool developed, its results relative to other work done on related objectives and its constraints if any, realized in the course of the project.

Chapter 7: Project Management and Timeline reviews the timeline management of the project including the milestones achieved in the course of implementation and any obstacles that were encountered and how they were solved. Lastly, **Chapter** 8: Conclusion and Future Work encompasses the major outcomes and contributions of the current work and proposed directions for future endeavors.

2 Literature Review

2.1 CAN Protocol and Fuzzing

The Controller Area Network (CAN) protocol is defined as a robust vehicle bus standard that is specially adapted to communication between various electronic control units in a vehicle system. CAN network was introduced by [1]Bosch's technicians in 1980's to answer the need for automotive networks that would permit the exchange of information in real time, with fewer wires. Even though the protocol facilitates communications among devices, it's operation is susceptible to attacks because it did not take security in its initial design. In the case of CAN bus testing, fuzzing traditionally means the injection of garbled or non-standard data into a system with the aim of finding bugs like malfunctions or any abnormal actions. These methods of testing have been extensively applied for the purpose of finding logical or structural faults and weaknesses in the electronic control modules.

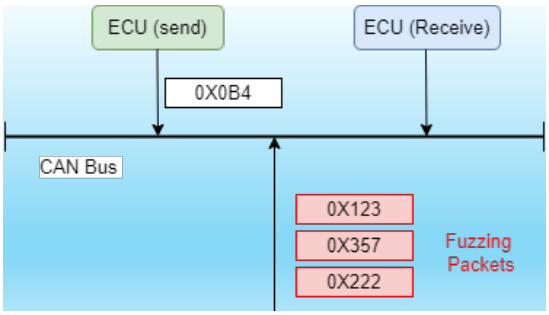


Figure 2.1: Fuzzing in CAN Bus Communication

Setting aside the limitations of conventional fuzzing methods for system crisis resolution, which are rather effective in figuring out the disorder in the system, such methods have several downfalls where they fail to detect any visual problems such as failures of components within the instrument panel of the car. Changes in any visual component of the system such as a warning light or a gauge that signifies normal functioning with such conditions, incidence, security or vulnerability could also take place. However, bulk of concerning fuzzing technique mostly restricts itself to fancier aspects of the system collapse and functional disconnection which usual adverse pictorial feedback may foster. Such a compromise may result to failing to detect possible external threats that could come in the form of the images and other visual indicators which are becoming more common in automotive designs today.

2.2 Related Work

During the last ten years, much effort has been put into the research of automotive cybersecurity, with significant development in the area of CAN bus fuzzing. The initial papers were directed towards the comprehending whether and how ECUs might engage in interactions that could be attacking the system using malformed CAN messages. Moreover, [6]Koscher et al. (2010) has shown the effectiveness of such tremendous vulnerability of the automotive systems against injection attacks of CAN messages where unauthorized messages trigger actions that were not intended in the stand alone ECUs of the automobile. Nevertheless, [2]Miller and Valasek (2015) followed up this research by breaching a vehicle's CAN bus remotely and exploiting control over critical systems of the vehicle which went further towards illustrating the security challenges in automotive systems.

Such synthesis methods enhance the fuzzing techniques and the techniques use different methods of synthesizing video feed into the fuzzing approaches. The vast majority of existing works focused on different types of software based anomalies and essentially neglected the opportunities of remediation of the system based on its visual feedback. Video feed on the other hand is an external factor which improves the fuzzing process because system log analysis is rarely useful when trying to determine the physical state of the internal mechanisms of the systems. Such gap within the literature allows the progress in the area and with it, the advancement of automotive systems which would in turn focus on the synthesis of such two dimensional techniques and tools.

2.3 Theoretical Background

Various logics and models are used in the cyber security domain, in order, to model and validate the behavior of the systems. Linear Temporal Logic (LTL) is one of the formalisms that are especially useful for sequence and timing of events of a system. Various temporal properties that include when a message is sent in response to another message, or in what order they are transmitted are especially useful in LTL which is common in CAN bus.

In this project, instead, it is intended LTL to model the sequence of CAN messages and the accompanying visual detections. For example, within the formulation of an LTL formula a requirement can be expressed that the message will be sent which will necessitate some subsequent visual change i.e. the light will be activated. The LTL formula in its normalized structure can take the following form:

$$\phi = \mathrm{EG}(\text{message} \rightarrow \mathrm{F}(\text{visual}))$$

In this formula, *message* represents the CAN message, and *visual* represents the corresponding visual feedback (e.g., a light or gauge change) that should follow. This formula can also be expressed as *for all possible paths (E), it is always possible to follow a sequence of CAN messages that (F)eventually lead to a visual change*.This formalism

helps in verifying that the system behaves as expected in terms of timing and sequencing, which is critical in ensuring the reliability of the fuzzing framework.

2.4 Reactive Modeling and LTL

The reactive systems such as the ones found in the automotive setting can be characterized by constant engagement with the environment. In the scope of this project, it is possible to consider the interaction between the CAN bus and the instrument cluster as a reactive system which transforms some inputs (CAN messages) into some outputs (changes in the appearance). This interaction over time can also be formally verified using Linear Temporal Logic (LTL) in the reactive model of this application.

LTL is very useful for modeling and verifying the order of events in this fuzzing framework. For instance, it can perform the verification of a certain sequence of CAN messages that is expected to cause a specific movement in the instrument cluster visually. The linear logic based temporal logic (LTL) formula formulated earlier has the ability to be revised to over more than one sequence so as to test the system's reaction to more than one fuzzing behavior.

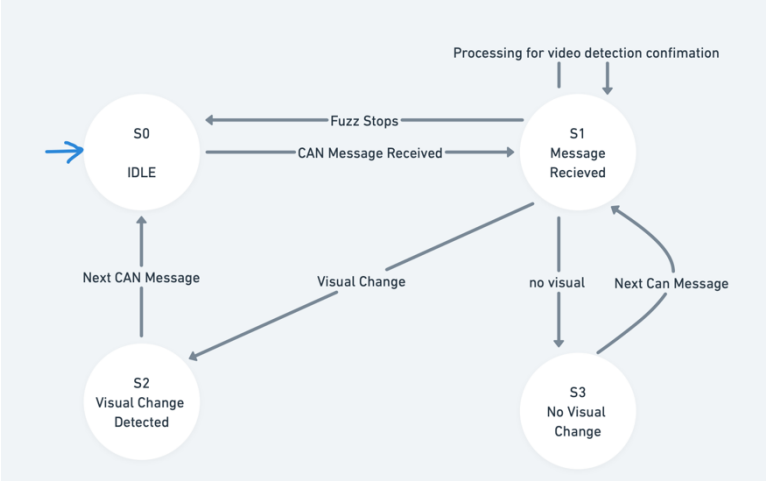


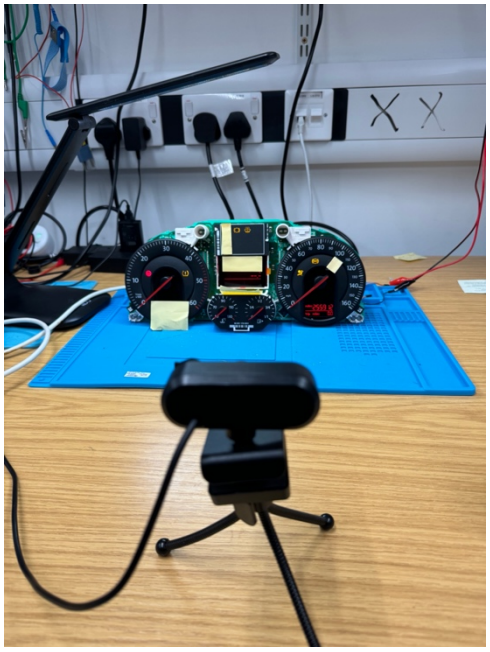
Figure 2.2: Proposed Fuzzing technique as a Reactive Model

Moreover, since LTL can describe whether temporal constraints are satisfied, the use of this property of LTL helps in defining the delays in between when messages are sent and when the visual depiction appears, which is very essential in achieving the goal of the fuzzing tool. With this assurance, the project applies LTL in a way such that it not only incorporates correct behavior of the system but also satisfies all the timing requirements, making it easy to spot anomalies in the operation of the CAN bus as well as the visual portrayal of the outputs.

3 Initial Setup and Implementation

3.1 Hardware and Software Setup

In this section, we will describe the hardware and software that were used in detail. The hardware setup consists of the instrument cluster from Volkswagen Passat which was figured out after studing user manuals, CAN interface (PCAN-USB adapter), a 1080p 30 fps video camera, a 12V power supply, floor lamp and required jumper wires. The instrument cluster was the most targeted device in a fuzzing attack and the camera was used for detecting the anomalies visually in real time.



3. Hardware Placement

1. Power Supply of 12V

2.PEAK PCAN- CAN Bus Adapter

Figure 3.1: Hardware Setup

On the software side, this project was based upon an Ubuntu operating system installed on a Lenovo ThinkPad laptop model. The software stack included python-can library for CAN communication, OpenCV for video analytics, and Tkinter for windows based application for GUI. SocketCan was also used to setup CAN with the help of linux kernel which helps to treat CAN as a network connection.

This project also comprised steps for creating and adapting video detection algorithm in the same environment. Video streams that were recorded using a motion detection camera were used to make a log of any movements, and the logs with fuzzing attacks were correlated to these. This configuration made sure that the system could function optimally even on low-end computers, as the requirements for using realtime video snatching and CAN message fuzzing were well managed.

Structure of a CAN Message

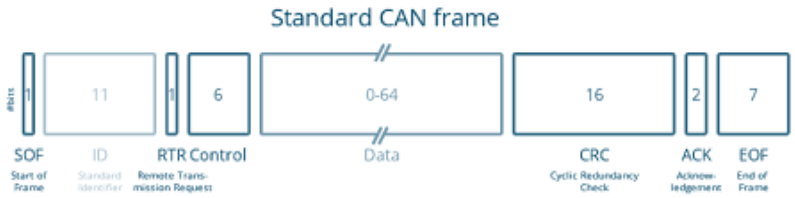


Figure 3.2: CAN Message structure

A conventional CAN message includes the following components:

1. **Identifier (ID):** The ID defines the message priority level. CAN 2.0 A incorporates an 11 bit ID whereas CAN 2.0 B has 29 bits.

2. **Control Bits:** This contains several fields such as the RTR (Remote Transmission Request) bit which designates whether the received frame is a data or a remote frame and the IDE (Identifier Extension) bit which designates the type of the frame as either standard or extended.

3. **Data Field:** Contains the actual data being transmitted, which can be up to 8 bytes (64 bits) long.

4. **Cyclic Redundancy Check (CRC):** This is a 15-bit field used for error checking.

5. **ACK Slot:** A one-bit field where a received message is acknowledged.

Fuzzing Time Estimation:

Given the limits related to the characteristics of the hardware utilized, the craning of the CAN bus was done thoroughly over time even though various aspects of each message were examined. The purpose of the fuzzing tool was primarily to cover the whole spectrum of CAN IDs by dispatching messages and keeping track of the traffic. As a rule of thumb, conducting an entire round of fuzzing which avails itself of every single ID noticeable and available on the device takes around 3-4 hours on average, depending on the alteration of messages as well as looking at the intention and context of the messages. This time frame is important for scheduling the long sucking periods so that no part is left uncovered.

Total Number of Possible CAN Messages

For **Standard CAN (11-bit ID)**:

- **Number of IDs**: 2^{11}=2048 possible IDs.
- **Number of Data Combinations**: For a data field of 0 to 8 bytes, each byte can have 2^8= 256 possible values. The total number of possible data combinations is:

$$\text{Data Combinations} = 1 + 256 + 256^2 + \cdots + 256^8$$

$$\text{Total Standard CAN Messages} = 2048 \times \text{Data Combinations}$$

Therefore, Fuzzing time (T) could be written as

$$\mathbf{T = (Number\ of\ IDs) \times (Data\ Combinations) \times t\ (fixed\ time)}$$

3.2 Wiring and Initial Testing

The wiring of the instrument cluster posed several challenges due to the lack of readily available documentation. After extensive research, including the examination of user manuals, wiring diagrams, and verification with resources from TCS, ZF Group and Jaguar Land Rover, the correct pinouts for the dashboard were identified. These pinouts were crucial for establishing a stable connection without risking damage to the instrument cluster.

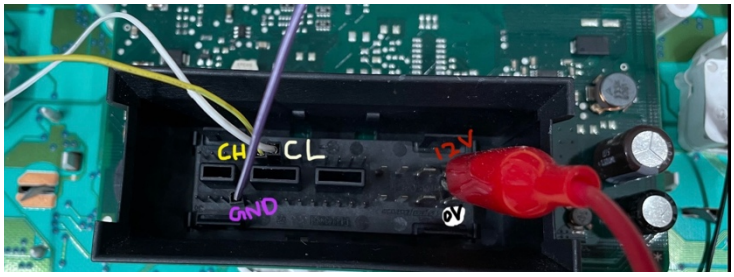


Figure 3.3: Dashboard Wiring

The connections were validated in the following manner:

- **Terminal 15 voltage supply (Pin 1)**: Red wire connected to a 12V power supply.
- **CAN data bus, low (CL) (Pin 12)**: White wire connected to Pin 2 CAN LOW of the PCAN-USB adapter.
- **CAN data bus, high (CH)(Pin 13)**: Yellow wire connected to Pin 7 CAN HIGH of the PCAN-USB adapter.
- **Sensor earth (GND) (Pin 33)**: Purple wire connected to Pin 3/6 of the PCAN-USB adapter.

Apart from this, the pinout of the dashboard, which was discovered after a lot of probing, runs in tandem with the necessary mapping required for additional connections. The 36-pin connector found on the dashboard of this Passat is provided with accessory ports & lamp sockets such as brake fluid, oil pressure & ambient temperature gauges accordingly which help in visual inspection of the system that is being tested for anomalies using fuzzing techniques.

Table 3.1: Dashboard Pinout for the instrument cluster

Pin Number	Description
1	Terminal 15 voltage supply
2	Terminal 30 voltage supply
3	Nil
4	Nil
5	Brake fluid level warning lamp indicator
6	Nil
7	Nil
8	Nil
9	Nil
10	Nil
11	Nil
12	CAN low - data bus
13	CAN high- data bus
14	CAN - screening
15	CAN - wake-up wire
16	Fuel gauge sender when its full and on all-wheel drive indicator
17	Fuel gauge sender when full indicator
18	Fuel gauge sender when empty indicator
19	Terminal 31 grounding
20	Nil
21	Nil
22	Front left brake pad wear sender indicator
23	Fuel gauge sender when empty and only with all-wheel drive
24	Nil
25	Oil pressure warning lamp indicator
26	Oil temprature and oil level sender indicator
27	Nil
28	Sliding sunroof adjustment control unit output,Speed signnal ,control unit with navigation system and display for radio
29	Nil
30	Nil
31	Nil
33	Terminal 31, sensor earth
34	Washer fluid level warning lamp indicator
35	Coolant storage and coolant temprature warning lamp indicator
36	Ambient temperature sensor indicator

Initial testing involved verifying the CAN communication setup by sending basic CAN messages and observing the cluster's response. The challenge here was ensuring that the correct wires were connected to avoid any miscommunication or damage to the instrument cluster. After successfully establishing communication, initial tests revealed a continuous stream of CAN messages, indicating that the setup was functioning as expected.

3.3 Development of the Video Detection Algorithm

The first idea of the project was to use the image detection process to observe the details changed in the instrument cluster during fuzzing. However, it soon turned evident that image detection alone would not work fast enough since it simply could not detect change that was fast or too subtle and that brought delays in detection of the anomaly system.

To overcome this, the project escalated into designing a video detection algorithm. The rationale was that video streams could capture a wider range of subtle alterations in the cluster, such as flashing lights or variations of different gauges, that go unnoticed by methods that employ a single image. The algorithm was aimed at handling streaming videos in which significant change was captured through the real-time adjustment of threshold and min_area parameters, which were then saved in relation to CAN messages.

The flowchart below shows the course of the combination of video detection and CAN bus fuzzing, depicting the times when fuzzing and video analysis are performed at the same time. The process starts with the launching of GUI and CAN bus, then the configuration of video capture follows. A separate thread is utilized to execute the fuzzing process in order to maintain the real time aspect, whereas another thread is devoted to maintaining the video feed in an active state.

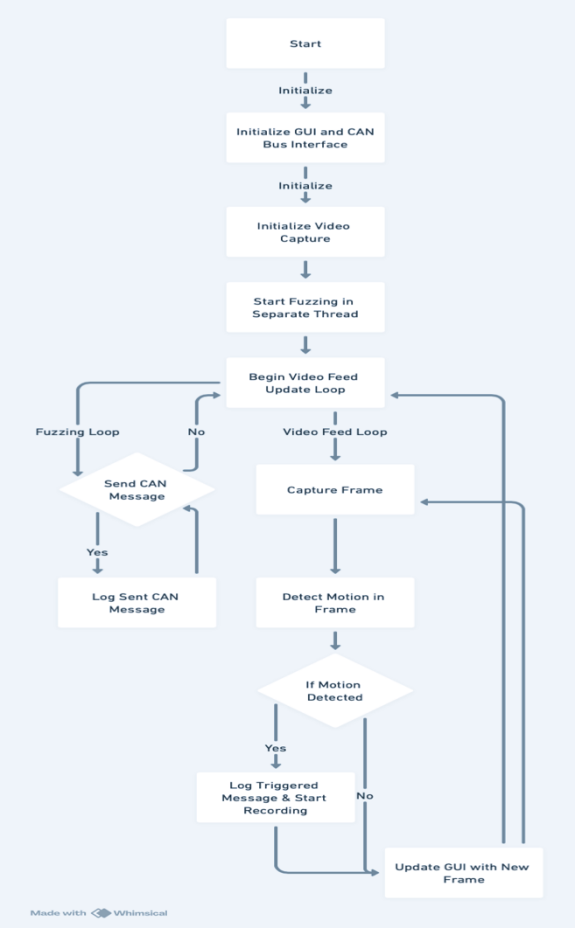


Figure 3.4: Video Detection Fuzzing Algorithm Flowchart

The fuzzing loop is in charge of sending the CAN messages out and receiving them for logging purposes. At the same time there is a video feed loop that captures the images in a live manner. In turn, for each of these images the algorithm performs a motion detection analysis to see whether any change has occurred due to the CAN message that was sent out. In case there is any movement detected in the captured frame, the system will register which CAN message has resulted in that change and will commence video capturing. The graphical user interface is thereafter updated with the canvas where a new frame is added depicting the process of fuzzing and changes induced.

This particular design is such that both the fuzzing and the video assessment can be done concurrently, this makes it possible to locate both internal logs regarding CAN message related activity and external physical changes in the instrument cluster, thus providing a holistic evaluation on the security of the system.

4 Design and Methodology

4.1 Fuzzing Framework Design

The fuzzing framework designed within the scope of this project is a high-level mechanism for discovering CAN bus vulnerabilities by means of many different inputs and responses. The concept which is the main new aspect of the current framework is the incorporation of video detection in the CAN bus fuzzing process to monitor internal log changes as well as detect visual changes outside the instrument cluster.

```
deverror404@deverror404-ThinkPad-T440:~/Desktop/cluster/tests$ python3 gui2.py
Choose fuzzing type: 1) Full Fuzzing 2) Quick Fuzzing 3) Ranged Fuzzing: 3
Enter start CAN ID (hex): 0x050
Enter end CAN ID (hex): 0x051
Enter CAN IDs to ignore (comma-separated, hex, e.g., 0x123, 0x456), or leave emp
ty:
Do you want to record video clips of visual changes? (yes/no): no
```

Figure 4.1: User Accessibility Menu

GUI and Threading Implementation

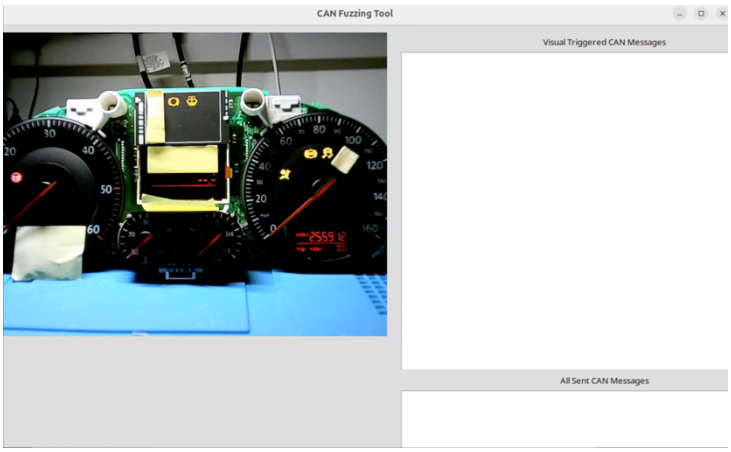


Figure 4.2: GUI structure with live feed

The Graphical User Interface (GUI) of the fuzzing tool was designed to be user-friendly, providing real-time feedback on the fuzzing process. The GUI is built using the Tkinter library in Python, which allows for the creation of windows, buttons, and other interactive elements that make the tool accessible even to users who may not have extensive programming experience.

Threading is a critical component of the tool, ensuring that the fuzzing process can run concurrently with video detection and GUI updates without causing performance bottlenecks. This was implemented within the tool using the Python threading module hence the tool is multi-tasked. For instance, when one thread is engaged in sending CAN messages, another thread is still active in scrutinizing the altered video while a different thread is busy refreshing the GUI with the results obtained. Such design ensures that the operations of the tool remain steady with respect to the rate of speed of rendering and reporting the user on-the-fly about the state of the operations

Full Fuzzing Method

The full fuzzing in the scope of testing is amazing in that it is the most aggressive testing available in the tool. More often than not, a majority or in some instances, the entirety of CAN messages is sent which takes into consideration every possible identifier id and data. The method was implemented n the following manner: The sanitation of every CAN id was carried out for the purpose of sending messages with the various data payload for every id thus its full fuzzing. The tool thereafter observes the system's behavior for any abnormal activities and more so from the instrument cluster which become the basis of further exploration.

The design of the full fuzzing method takes into account the need for thoroughness and accuracy. The tool logs every CAN message sent, the corresponding response (or lack thereof), and any visual changes detected by the video feed. This exhaustive approach helps ensure that no potential vulnerability is overlooked.

Ranged Fuzzing Method

Ranged fuzzing is a much more directed style of fuzzing than full fuzzing. It allows one to enter a few existing CAN ID ranges, which then constrains and closes the testing to the pertinent sections of the CAN bus network. This method proves to be very effective when a user is aware that certain CAN IDs are more susceptible compared to others.

In the creation of ranged fuzzing, the script is made it to enter fields indicating ranges for the ID of the part which will cover the fuzzing from start and to finish. After that, the tool acts within this set parameters, which curtails the amount of work done and speeds the testing process as more time is given to the particular area. In this case the results are displayed just as can be expected during full fuzzing that is even internal records of log files and changes on visually recorded materials.

Quick Fuzzing Method

Quick fuzzing originated as a quick and simple way to test for fuzz quality to characterize basic points of interest. In distinction to the complete fuzzing that entails the whole process, and ranged fuzzing that deals with specific areas, quick fuzzing dispatches unstructured CAN messages to a few IDs. It is aimed at producing quick and

overt effects such as light indicators on the dashboard which can then be followed up by more deep procedures.

When time is limited or needs to be met, the quick fuzzing method can achieve remarkable results in a fast moving environment. It was done by enabling a quick fuzz session directly from the GUI so that the user only needs to click on the button and let the tool select CAN IDs and send junk messages to them. All the significant responses are tagged for further investigation immediately.

4.2 Video Detection Algorithm

The most important aspect of this project is the video detection algorithm, which enhanced the CAN bus fuzzing tool to provide real-time visual monitoring of the changes in the instrument cluster. This was done through the use of the OpenCV library, which is a computer vision library that reaches out into the lower levels of Python.

Implementation and Fine-Tuning

The algorithm works by continuously capturing frames from a video feed of the instrument cluster. These frames are then processed to detect any changes that occur as a result of the fuzzing process. The detection algorithm is designed to be sensitive to small changes, such as the activation of an indicator light or a shift in the position of a gauge.

To improve the accuracy of detection thresholds and area min, two critical parameters were optimized further The threshold parameter specifier provides a measure of how sensitive an algorithm will be to any variance in the image. On the other hand min_area determines how large a detected variation is expected to be so that it would be considered important.

Several tests were done by default including these parameters in the tests and varying their values and measuring the accuracy of the detection. The objective was to evaluate the performance of the algorithm by determining the vicinity within which detecting significant changes is possible without reporting changes that are noise.

Table 4.1: Testing various threshold and min_area values

Threshold Value	**Min_Area Value**	**Detection Accuracy**	**Notes**
20	200	Medium	Some small changes detected, but with noticeable noise
25	300	Medium-High	Detects most changes, moderate noise present
24.5	200	Medium-High	Balanced detection, slight noise detected
30	**230**	**High**	**Sensitive to small changes, minimal noise detected**
25	230	Medium-High	Good detection accuracy, some noise detected
30	200	High	Optimal balance, accurate detection with minimal noise

This table demonstrates how such adjustments are able to improve detection metrics and noise levels, and explains in more detail the process of tuning both threshold and min_area values in the optimization stage. It was determined that the best performance in terms of detection accuracy with the least way of introducing background noise was in the settings of the tool where Threshold = 30 and Min_Area = 230

4.3 Integration with CAN Bus

The integration of video detection with CAN bus fuzzing is what sets this tool apart from traditional fuzzing methods. This integration allows the tool to simultaneously monitor both internal system logs and external visual indicators, providing a more comprehensive understanding of the effects of fuzzing.

Pseudo-code : Integration of Video detection and CAN Bus Fuzzing

```
Initialize CAN bus interface
Initialize camera and start video feed
Set parameters for video detection (threshold, min_area)
while fuzzing_tool_active do
    for each CAN_message in CAN_ID_range do
        Send CAN_message
        Wait for response_time_interval
        Capture current_frame from camera feed
        if video_detection_enabled then
            Process current_frame using video detection algorithm
            if visual_change_detected(current_frame) then
                Log CAN_message and current_frame timestamp
                Highlight change on live video feed
            end if
        end if
        Log CAN_message in internal system log
    end for
    if quick_fuzz_enabled then
        Continue to next CAN_message without processing internal logs
    else
        Compare current CAN traffic with previous logs
        if significant_change_detected then
            Log CAN_message causing change
        end if
    end if
end while
Stop video feed
Close CAN bus interface
Save all logs and video recordings
```

4.4 Data Logging and Live Feed Synchronization

In relation to the topic, so video detection can be appropriately intertwined with CAN messages being captured at that time, the tool supports this scenario by ensuring the video feed is in chronological order with the fuzzing activity. In most cases, a sequence of several frames captured by the camera is recorded with the appropriate CAN message into the log. This synchronization is important since it helps in determining the kind of messages that trigger the given visual alterations.

The live feed is displayed in the GUI, allowing the user to monitor the instrument cluster in real-time. Any detected changes are highlighted on the feed, and the corresponding CAN messages are logged in a text file with visual or internal lablel for further analysis. This real-time feedback loop enhances the user's ability to quickly identify and investigate potential vulnerabilities.

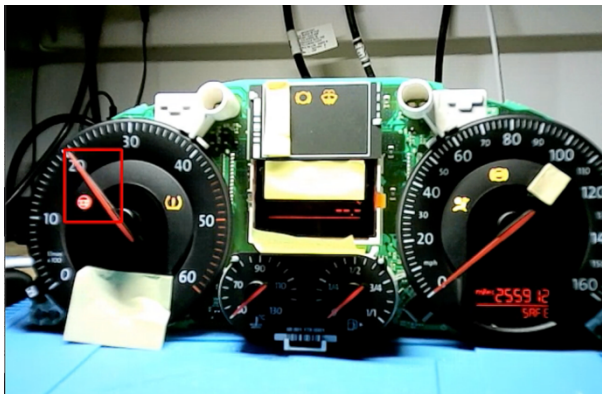


Figure 4.3: Video Detection

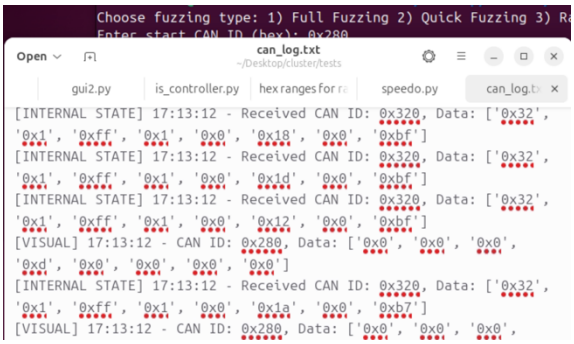


Figure 4.4: Triggred Visual and internal state logs in text file

Figure 4.5: Triggered CAN Message Logs in GUI

5 Experimental Setup And Testing

5.1 Experiment Environment

The experimental setup was designed to replicate a realistic automotive environment where the instrument cluster could be tested under dynamic conditions. The lab environment included a dedicated workspace with controlled lighting to minimize interference with the video detection algorithm. The instrument cluster was securely mounted on a stand to ensure stability during testing, and the CAN bus interface was connected to the instrument cluster through a series of jumper wires.

One of the important aspects that needed consideration while preparing the setup was finding the most appropriate camera position so that every important visual channel located on the instrument cluster is visible in the field of view of the camera. To achieve this, the dashboard camera was mounted about ten inches away from the clustered camera aiming most exceedingly towards the instrument cluster enclosing it in a high resolution video. This was important because it helped in tracking the changes on the visually responsive areas of the dashboard due to the CAN messages. The lab also had variable conditions that incorporated unwanted external and internal light which made modifying the camera features and particular lighting difficult.

The experimental environment was also equipped with a laptop running Ubuntu, connected to the instrument cluster via a PCAN-USB interface. The software environment included Python libraries for CAN bus communication, OpenCV for video detection, and custom scripts for logging and analyzing the results. The total configuration underwent several tests in order to determine whether the camera feed, CAN communication, and the fuzzing tool functioned at the same time.

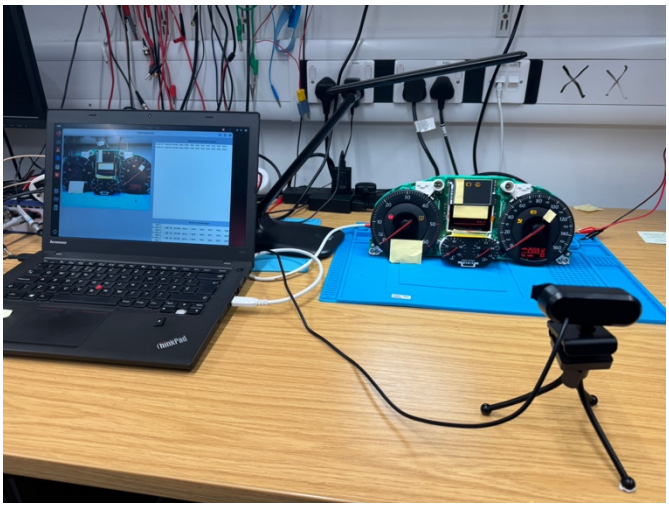


Figure 5.1: Setup while the tool is running

5.2 Testing Methodology

The testing of the fuzzing tool followed a structured methodology to systematically evaluate its performance. More advanced testing was targeting both the video detection algorithm accuracy as well as the fuzzing tool applicability at the vulnerabilities of the instrument cluster away from the devised algorithms.

Step 1: Baseline Establishment: Prior to fuzzing, a baseline of normal CAN traffic and a visual examination of the instrument cluster were determined including normal operations of the system. This was done by great and sophisticated implanting of organizing video segments along with CAN logs of normal working manipulating none of the fuzzing message. Gaining this baseline was critical towards reporting the visible trends during the actual testing**.**

Step 2: Complete Task Based on Full Fuzzing: All other CAN messages were systematically sent through the instrument cluster via the full fuzzing approach so as to assess all possible usage patterns. These messages were all recorded, and the video was also monitored for changes in the feed throughout these. The full CAN message format allowing for various ID's was utilized to send all the available CAN messages, with the tool set to wait for some time before proceeding to the next message so as to capture any long term effects.

Step 3: Ranged and Quick Fuzzing: After the full fuzzing was completed, the tool carried out ranged and quick fuzzing on particular groups of CAN IDs that during the previous phase showed signs of promise 'full fuzzing phase'. With regards to the ranged fuzzing, the completeness of a message is not overly prioritized since focus is on some few ID's and some data bytes. Concerning the quick fuzzing, only coarse analysis of the logs was performed before sending the messages because fine analysis would cause delays in sending out messages. This stage sought to capture more minor changes that could not be observed during full fuzzing**.**

Step 4: Data Collection and Analysis Data in the form of video and log information was collected at every testing phase. Whenever there was an event on the video feed that required no input from the user, it was checked constantly by the tool and tagged. All events that required tagging were looked into the CAN message log in a bid to find particular messages that may have triggered visual changes. The evaluation sought to understand the already foreseen issues, for example, which CAN IDs have been used to turn on the visual indicators or even anomalies in the instrument cluster over time.

Step 5: Validation of Results: This involved focusing on the CAN message of the instrument clusters relevant to the potential weaknesses that had been previously detected. This part confirmed that the weaknesses identified were indeed reproducible and not merely due to some random mistake or background interference.

5.3 Results

It is the experimental testing that has provided worthwhile findings that prove of the improved video detection-fuzzing based method.The tool successfully identified several vulnerabilities within the instrument cluster that would not have been detected through traditional log-based fuzzing alone.

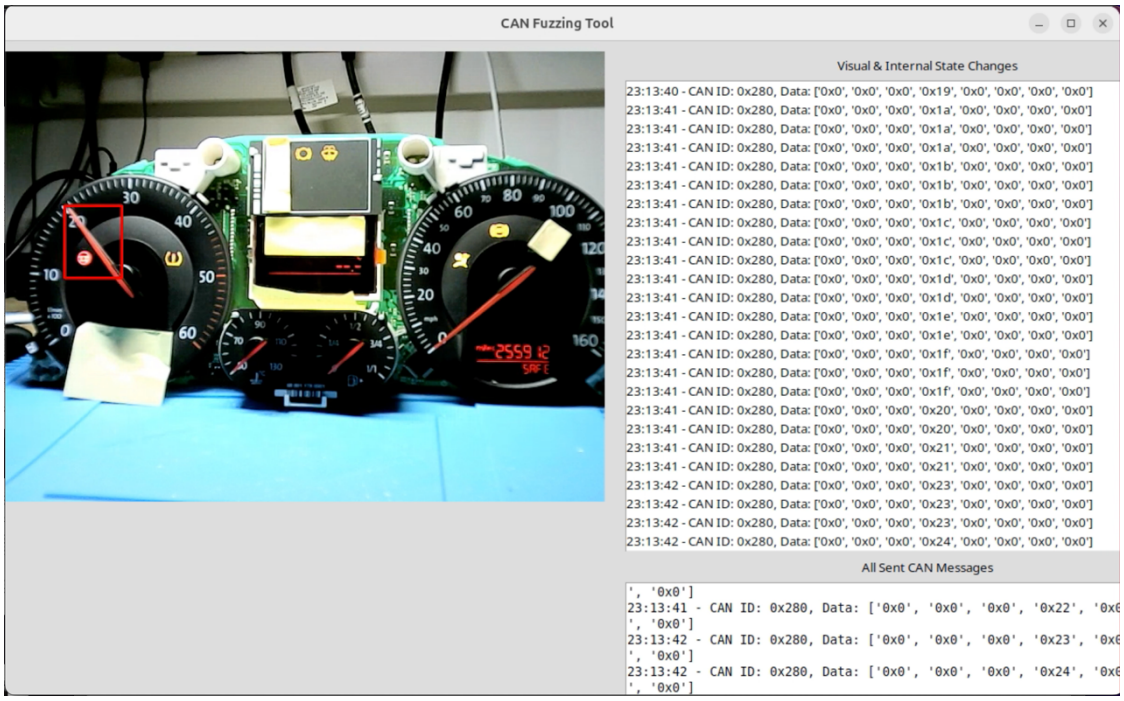


Figure 5.2: Tachometer needle detection

- One of the most significant results was the identification of a vulnerability associated with CAN ID 0x280, which caused the tachometer to move erratically. The video detection algorithm flagged this movement as an anomaly, which was correlated with the fuzzed CAN messages in the logs, the motion detection is shown in the RPM needle with a redbox.

- The CAN IDs 0x50 to 0x51:
  These are associated with triggering the loose seatbelt alert and the power steering warning light, while also generating new traffic from IDs 0x320 and 0x62F. This range of IDs is particularly dense with activity, as shown in the images, where the alerts and warnings appear and disappear rapidly. The traditional methods would struggle to accurately pinpoint which specific CAN message is responsible for each alert, especially given the high frequency of events in such a short interval.

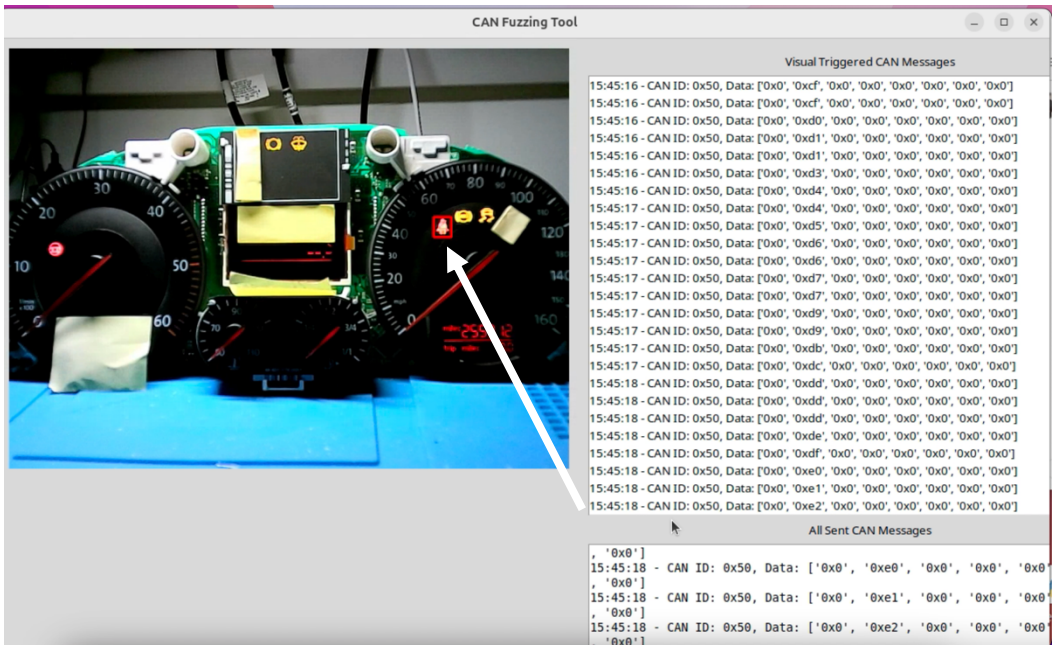


Figure 5.3: Seatbelt warning detection

We can see the results of the fuzzing tool in actual practice, especially concentrating on the motion detection accuracy. The sequence of images shows us a more complex scenario: a series of warnings comes in a very short period of time and it is impossible to focus and discriminate all the particular CAN IDs of the warnings visually using the human eye alone.

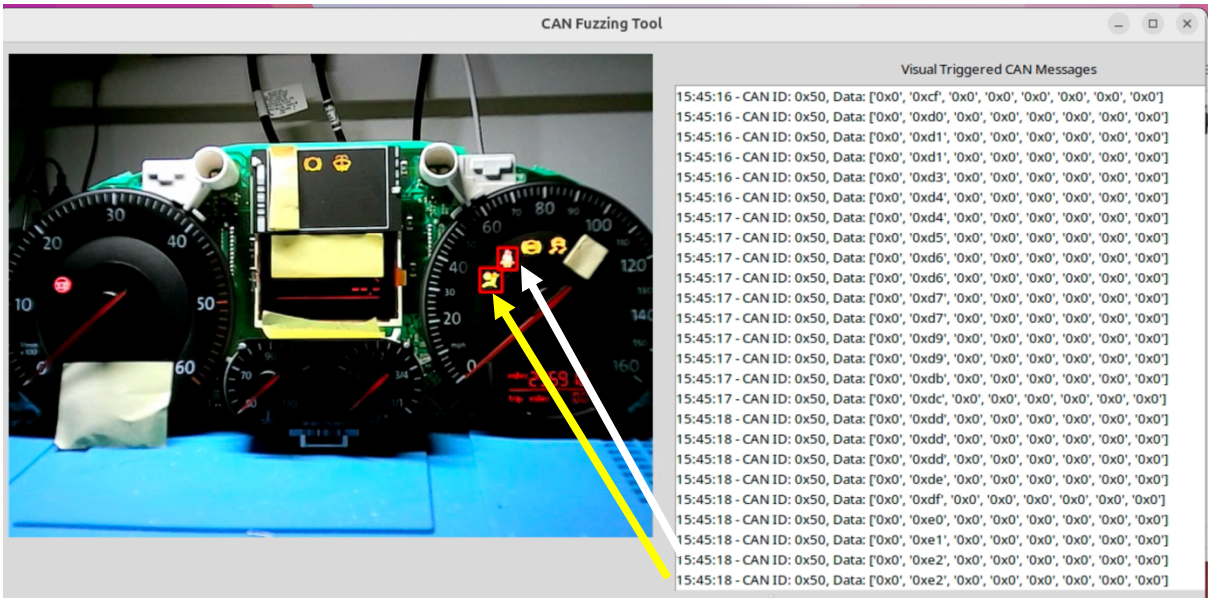


Figure 5.4: Detection of next warning in the same time frame

The fuzzing tool's integration with video detection plays a crucial role here, as it automatically logs and identifies these visual changes in real-time. This capability allows for precise correlation between the visual alerts and the specific CAN messages, enhancing the accuracy and efficiency of the fuzzing process. By doing so, the tool adds the observational depth of how the information would normally be presented without using the tool, thereby ensuring that the important details are captured into memory in scenarios where alerts come one after the other in rapid intervals. This proves the effectiveness of the tool in the operations, situations that are complicated and continue to change, and in which conventional tools fail to function effectively.

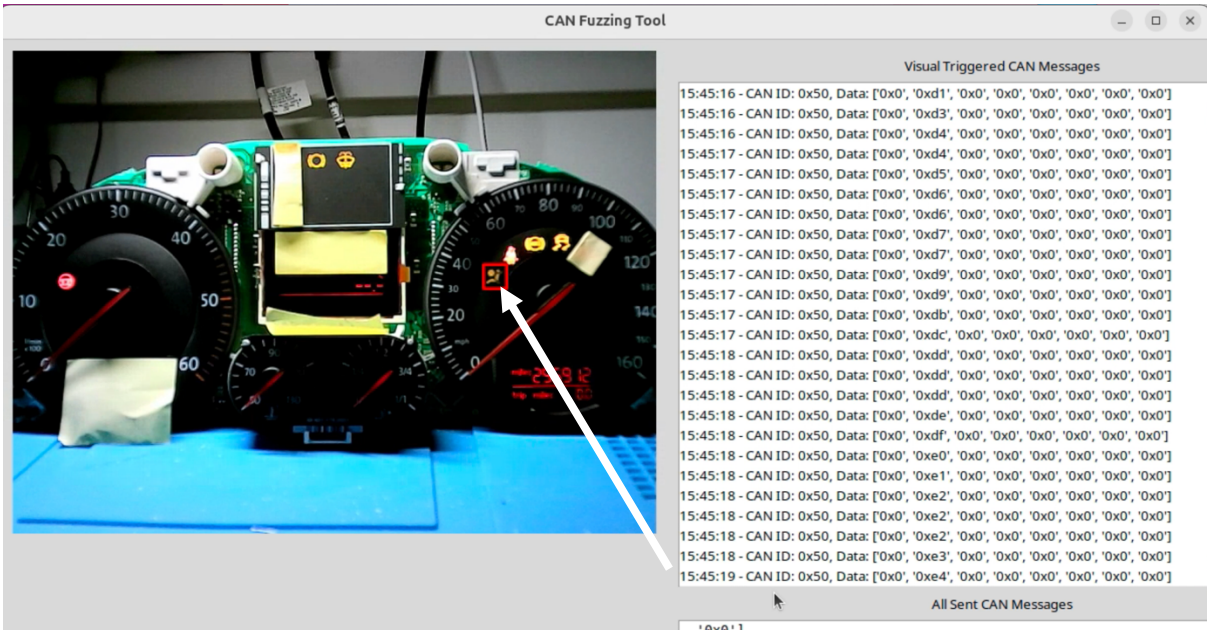


Figure 5.5: Detection of CAN message causing warning to turn off in same time frame

Below is a selection of the most impactful CAN messages identified using the fuzzing tool in the following table, which demonstrates the tool's effectiveness in detecting and correlating visual changes with specific CAN messages.

ID (Data Position)	Effect(s) Identified with Video Detection
0x050	Video detection revealed that setting this byte to FF triggers both the loose seatbelt warning and the power steering warning. Additionally, new CAN traffic from IDs 0x320 and 0x62F was observed.
0x1A0 (Byte 1, Bits 0, 1)	The tool identified that setting bit 1 to 1 activates the Electronic Stability Control warning, while setting it to 0 deactivates it. Spamming bit 0 with a value of 0 disables the ABS warning. New traffic from ID 0x320 was also observed.
0x280 (Byte 3)	The video detection tool precisely identified that a value between 0h and 60h causes the tachometer gauge(RPM) to adjust accordingly.
0x343 (Byte 2)	The tool detected that setting this byte to FF activates the tire pressure monitoring warning.
0x361	Logs identified new CAN traffic from the previously unseen ID 0x625 after sending this message.
0x373	The tool detected that this message activates the power steering warning light and causes unexpected behavior on the display below the mileage indicator.

0x3D0	Video detection showed that this byte lights the power steering in abnormal conditions with a warning sign. Bit 1 handles the amber light while the other part of the byte handles the red light.
0x470	The tool showed that Byte 0 contains the warning about the car battery; Byte 1 most 0 wakes up the open door alarm; Bit 5 brings up the warning for the boot being opened and Byte 4 turns on the exterior light alarm.
0x531	Video detection tool assumed that Byte 0 Bit 2 is for main beam caution lamp; Bit 4 is for rear fog caution lamp; while up to two other indicators of Byte 2 Bit 0 and 1.
0x540	The tool found that this message makes the screen showing the “P R N D 4 3 2" sequence to be shown on the centre screen most likely the gear box indication.
0x5C0	Video detection demonstrated that Byte 5 was effective in controlling the brake pedal illumination whilst Byte 6 was useful in controlling the electronic parking brake illumination.
0x5D0	The tool found unexpected abnormality in the mileage reading which was as a result of the message captured.

TABLE 5.1: Reverse Engineered CAN IDs after using the tool

GUI INTEGRATION: The fundamental reason why the logs are performed along with the video feed is that it aids in comprehending the fuzzing procedure better.

Motion Detection: Red box around the CAN messages triggers motion detection to record the live video and whenever there is a CAN message sent/received, the box gives motion detection.

Reproducibility of Results: All the vulnerabilities highlighted were sanity checked to all be reproducible which therefore implies that the detection algorithm applied is very robust. All the moving anomalies were activated by similar CAN messages which further buttressed the efficacy of the tool in the resolution of real security concerns.

6 Evaluation and Discussion

6.1 Analysis of Results

The integration of video detection with CAN bus fuzzing proved to be a highly effective approach in identifying vulnerabilities that traditional fuzzing methods might overlook. Regular fuzzing is mostly concerned with identifying system's crashes or any anomalous logs, thus, subtle but vital visual shifts in the instrument cluster graphics are overlooked. Doing so allows the fuzzing tool to reconstruct the temporal sequence as well as present instantaneous video feed of the instrument cluster affected by the fuzzing, which would help in appreciating the effect of fuzzing on the automotive systems..

The vulnerabilities discovered using this method underscore the importance of visual feedback in fuzzing. For instance, certain CAN messages caused the RPM gauge to fluctuate or triggered warning lights that would not have been detected through log analysis alone. These visual anomalies indicate potential vulnerabilities that could be exploited in a cyberattack, affecting the safety and functionality of the vehicle.

Not only does the establishment of linkage between particular CAN frames and visual changes in the instrument cluster increases the effectiveness of fuzzing but also the efficiency of this method as a whole. This correlation makes it possible to concentrate resources for expanding the scope of analysis: the emphasis can be placed on further subtleties of fuzzing, such as determining the factors and conditions that will lead to anomalous events. This method not only expands fuzzing but also enhances its effectiveness, thereby making it a more efficient method of conducting fuzzing attacks for auto industry cybersecurity testing.

Furthermore, the application of video detection is expected to reveal another type of vulnerabilty that has indirect or cascading impacts. An example is where an ordinary CAN command message does not directly induce catastrophic failures in a system, but evocation of that command results in some unexpected visual defects that, if done for a long time, may suggest a serious defect. I would say these slight features would be missed by conventional methods, but these would be seen through the video detection methods making them easy to anticipate potential security weaknesses long before they happen.

6.2 Comparison with Related Work

Based on the comparison with other studies, it can be concluded that the use of video detection in the fuzzing process is a promising development for the automotive cyber

security domain. Earlier investigations have generally concerned with log-based ‘crash’ fuzzing approaches, where the predominant aim is simply the detection of any system failure or other event that disrupts the communications bus. These methods are quite useful for debugging purposes, however, they have limitations of undetectable non-disruptive anomalies that are no less dangerous.

Works like [6]Koscher, M., et al. (2010) and [2]Miller and Valasek (2015) have shown weaknesses in car systems but supported this with a lot of manual work on testing and reverse engineering. These methods although quite revolutionary are not easy to employ because it entailed a lot of work and skills so they aren’t ideal for mass use. Also, these methods were rather crash-level vulnerability oriented and thus did not regard other implications of compromise of a system such as visual cues.

This project builds on the foundation of previous research in the field of automotive cybersecurity, particularly the efforts undertaken within our own academic community. Earlier works have made significant strides in exploring the vulnerabilities of CAN bus systems, with notable contributions from researchers such as [3] Dr. Andreea-Ina Radu and her team. Their investigations into automotive security have provided a robust framework that this project has leveraged and expanded upon.

While previous studies primarily focused on traditional fuzzing methods to uncover vulnerabilities within CAN bus systems, this project introduces a novel approach by integrating real-time video detection into the fuzzing process. This addition enhances the ability to detect not only log-based anomalies but also visual feedback from the instrument cluster, which has traditionally been overlooked.

The collaborative environment and the foundational research within our university have been instrumental in guiding the development of this project. By incorporating video detection, this work contributes a new dimension to the existing body of research, offering a more comprehensive method for identifying security flaws in automotive systems. The findings underscore the potential for further innovation in this area, particularly through the integration of multi-modal detection techniques.

6.3 Limitations

The search for detailed and accurate pinout information for the instrument cluster’s functionalities was one of the main challenges encountered in this project. Information of that nature about technical details was classified and so research was done on several user’s manuals and wiring diagrams to get the information. Some of these resources were very difficult to acquire through official bases and a lot of effort and time was spent in searching through unorthodox sources. The assistance from the experts at TCS, ZF, and Jaguar Land Rover after collecting the sought-after information was of massive

help. They were pivotal in wiring configuration of the instrument cluster and therefore set up was as intended by the user.

However, although working with these industry professionals substantially reduced the chances of wiring errors, the absence of readily available documentation at first illustrated challenges associated with using proprietary car systems. This problem highlights the very necessity to provide adequate and clear technical documentation to ensure that automotive cybersecurity research and development processes may be more seamless and productive. Not with standing these challenges, the effective validation and configuration of the cluster unit depict the level of determination and creativity required to work around these difficulties in practice.

Figure 6.1: Lamp used to mitigate dynamic lab lighting

Another limitation concerns the nature of the lab itself because it involves movement in the laboratory. Offline lab practices can vary in terms of how people and equipment move around in the lab, whether the lights in the lab are off and on depending on the users' presence, whenever there is much noise and many factors of the environment and so on can compromise video detection accuracy. While some of these problems, for instance, incorporating one more lamp to provide even illumination, were corrected these problems still caused some level of fluctuation in the outcomes obtained.

Finally, the emphasis on visual anomalies can lead to non-visual problems that are not detected, yet are of utmost importance. For instance, such anomalies in sound or vibration could be treated as weaknesses as well but were not part of this effort. The coupling of more than one form of detection, such as sound and vibration, can give more results in the next rounds of this tool development.

Nonetheless, this is a first step in the evolution of video detection and CAN bus fuzzing in the context of car cyber security testing. The limitations identified are such as to provide future directions of growth and improvement to the tool, so that it can be readily used for more extensive and efficient operations in the field.

7 Project Management and Timeline

7.1 Project Planning (June 2 to September 2)

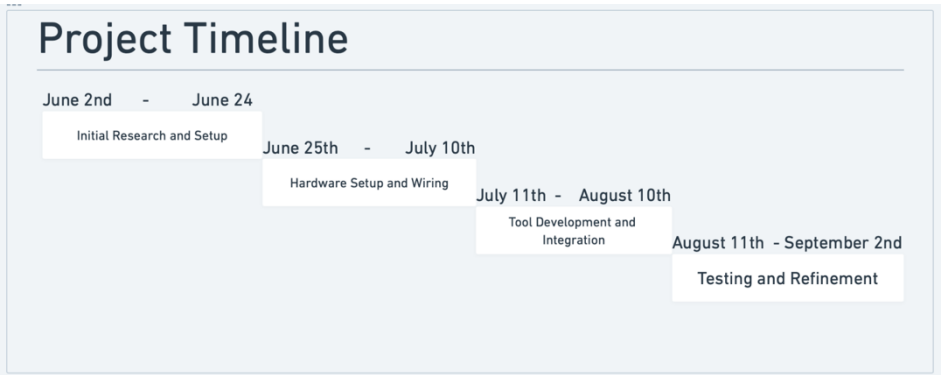


Figure 7.1: Project Timeline

To ensure that the schedule for the project was adhered to as well as the quality of the work complied with the requirements, the stages of the project were carefully drawn up and executed under the approved design. The duration covered three months, from the 2nd of June to the 2nd of September, during which the lab work started on 25th of June. The project planning comprised a set of achievements that were necessary both for the design of the video detection algorithm and for its incorporation in the CAN bus fuzzing system.

The first stage (June 2 to June 24) was focused on extensive background analysis and development environment configuration. In this I should setup all tools available such as operating systems - Ubuntu, program libraries - Python, graphic libraries - OpenCV, and buy hardware instruments like instrument cluster, CAN interface, video camera. At this stage, I moreover practically performed a detailed analysis of the literature to find out which issues are not covered by the existing studies and where my project would fit in.

The next stage (June 25 to July 10) was devoted to hardware installation and primary wiring of the instrument cluster. It is worth mentioning that this stage was quite tricky since it involved a considerable amount of pinouts which needed to be correct and the connections needed to be firm. Further, the scarcity of literature on the instrument cluster meant that a lot of time had to be spent on exploring the literature and interviewing experts. I spent a lot of time going through user guides, electrical

schematics, and the internet. Professionals employed at TCS, ZF, and JLR were essential to me concerning the connection of devices and correctness of the network.

From July 11 to August 10, the focus was mostly on the implementation of the fuzzing tool and the completion of the video detection algorithm integration. This stage was of great importance as it was necessary to work out the structure of the tool, apply the fuzzing techniques (full fuzzing, ranged fuzzing and quick fuzzing) and incorporate the video detection algorithm. One of the major achievements during this period was the incorporation of threading in the GUI in order to provide real time functioning. The video detection algorithm was being optimized through testing various threshold and min_area parameters.

The last stage (August 11 to September 2) saw the tool undergoing extensive tests and refinement. Preparations for this phase involved defining the experimental setting, preparing for and executing the various test runs, followed by examining the telemetry and video data captured in the course of the log and video detection.

In the performance assessment of the tool, the necessary corrections were done towards obtaining better accuracy and efficiency. As a follow up, a report to document the project was prepared with emphasis on the results and their interpretation and suggestions for further work.

7.2 Functional and Non-Functional Requirements

The project was guided by a set of functional and non-functional requirements that ensured its alignment with the overall objectives and usability standards.

Functional Requirements Met:

1. **CAN Bus Fuzzing Capability:**
   The tool is able to do full fuzzing, ranged fuzzing, and quick fuzzing on the CAN bus and be able to detect log and image anomalies.

2. **Video Detection Integration:**
   The tool includes a video detection algorithm that tracks visual changes in the instrument cluster during the fuzzing.

3. **Real-Time Data Logging:**
   The tool can perform data logging of the CAN bus messages, and the concurrent visual CAN bus data messages during a CAN bus fuzzing session for the purpose of temporal message and visual correlation analysis.

4. **User Interface (GUI):**
   The application in consideration has a graphical user interface which is considerably simple and allows users to begin fuzzing tests, track live and logged feeds, and even watches the video data and logs.

5. **Threshold and Min_Area Configuration:**
   The tool provides forecasts by allowing such parameters as threshold and min_area to be set for the video detection algorithm in class discrimination by different environments.

Non-Functional Requirements met:

1. **Performance:**
   It should be feasible to provide real-time data collection and analysis and ensure that the graphical user interface (gUI) is non-suspended throughout selfish fuzzing undertakings.

2. **Scalability:**
   The architecture of the tool should be oriented to future improvements such as availability of more detection algorithms or potential to go beyond the automotive protocol.

3. **Reliability:**
   The tool should carry out its core function of detection and logging of anomalies constantly without malfunctioning or losses of data even with massive amounts of CAN messages.

4. **Usability:**
   This should not present any difficulties to any given person irrespective of their technical abilities as the user interface controls must be simple to use with explicit instructions and interactive designs and easy navigation.

5. **Security:**
   This is a very delicate part of the tool that should be always considered, and it involves the protection of the data during its processing to make sure that no sensitive information that may have been collected during the tests is accessed by other individuals.

6. **Portability:**
   The tool must be versatile enough to work on various hardware configurations, including low-end machines, in order to keep the system usable in numerous settings along the usage lines.

By adhering to these functional and non-functional requirements, the project was able to deliver a robust and versatile tool that meets the objectives of enhancing automotive security through advanced CAN bus fuzzing and video detection integration.

8 Conclusion And Future Work

8.1 Conclusion

This project successfully developed and implemented a novel approach to CAN bus fuzzing by integrating real-time video detection into the fuzzing process. Traditional CAN bus fuzzing techniques have focused on detecting crashes, errors, or unexpected system behaviors based solely on log data. However, these methods often overlook visual anomalies that can be critical indicators of underlying vulnerabilities. The integration of video detection fills this gap by enabling a more comprehensive analysis of how fuzzed CAN messages impact the instrument cluster's display and other visual elements. This advancement represents a significant contribution to the field of automotive cybersecurity, offering a new dimension of analysis that can reveal vulnerabilities that would otherwise remain undetected.

The project involved the creation of a robust fuzzing framework, complete with a user-friendly GUI and real-time performance capabilities achieved through effective threading. The tool was thoroughly tested in a dynamic lab environment, and the results demonstrated its ability to detect vulnerabilities both in the CAN bus communication and through visual cues from the instrument cluster. The successful identification of several CAN messages that cause specific visual changes highlights the effectiveness of the video-enhanced fuzzing method. This approach not only improves the accuracy of vulnerability detection but also provides a more holistic understanding of how fuzzed messages affect vehicle systems.

The outlines of this work have practical dimensions in the automotive domain. With a growing trend for complexity in vehicle electronic systems, the aspect of cybersecurity also gets some room for further improvement. The video approach of deficiency detection in the cutting communication system is a great potential device for the security experts as well as the car manufacturers in the better identification and prevention of dangers to a vehicle. This project forms the foundation of further work and studies in this area and is of great promise in optimizing the elements of enhancing the level of security provision in vehicle antitheft systems and maintenance of the safety and reliability of modern vehicles.

8.2 Future Work

While the integration of video detection into CAN bus fuzzing has proven to be a valuable innovation, there are several avenues for future work that could further enhance the effectiveness and applicability of this approach. One potential extension is the

application of this methodology to other automotive protocols, such as UDS(Unified Diagnostic Service), LIN (Local Interconnect Network) or FlexRay. These protocols, like CAN, play critical roles in vehicle communication networks, and applying the video detection-enhanced fuzzing technique to them could uncover additional vulnerabilities and improve overall vehicle security.

A further improvement in the system lies in the video detection algorithm. The current implementation works well, but there is still some way to go especially with regards to enhancing the level of accuracy of detection as well as the speed of processing. This may include enlarging their ability to deal with complicated video images taken inside moving cars. This may include the optimization of present parameters like boundary and min_area, as well as the incorporation of other strategies aimed at improving motion and anomaly detection.

8.2.1 Incorporating Machine Learning

A promising direction for future development is the incorporation of machine learning (ML) techniques to improve the accuracy and efficiency of the video detection algorithm. Today, deep learning based machine learning techniques are penetrating into various applications including video and image contents processing, specifically in object recognition, tracking, and detection of unusual events. The use of an ML model trained on a set of CAN-initiated visual oddities could enhance the video detection system because it could learn how to recognize patterns that normal algorithms would not detect.

Machine learning could also improve the adjustability of the tool across various vehicles and instrument clusters.. In addition, video detection by means of ML could probably develop the ability of recognizing the effect that some CAN messages might have on a system and help avoid the visual representation of vulnerabilities which can trigger targeted attacks.

The advantages that may range from improving the detection rates to lowering the rates of false positives make this an appealing area for further research. It is suggested that this integration could not only enhance the power of the tool but make it more multifunctional, which could be used in many more automobile systems with less effort to adapt them.

Therefore, the next stage of this project depends on the extension of its use to other automotive protocols, the improvement of the algorithm for video detection, and the addition of more sophisticated approaches to machine learning. These improvements would be able to further strengthen the position of the tool as a real-state-of-the-art equipment in automotive cyber security rather than simply counteracting the current races of vehicles threat.

9 Appendix

- The Git project repository is available in this URL: https://git.cs.bham.ac.uk/projects-2023-24/dxm344.git

- There is a markdown formatted file README.md in the repository that provides steps to run the python scripts used in the project along with a requirements.txt file.

- The repository contains also the video clips with some results of fuzzing.

- Here is the guide to the warning lights on the tested Volkswagen instrument cluster:
- https://www.volkswagen.co.uk/en/owners-and-services/my-car/warning-light.html

REFERENCES

- [1]**Bosch, Robert et al. (1991).** "CAN Specification Version 2.0." *Robert Bosch GmbH*.

- [2]**Miller, Charlie, and Chris Valasek (2015).** "Remote Exploitation of an Unaltered Passenger Vehicle." *Black Hat USA 2015*. Available at: Link [Accessed: 2024-08-31].

- [3]**Radu, Andreea-Ina and Garcia, Flavio D. (2020).** "Grey-box Analysis and Fuzzing of Automotive Electronic Components via Control-Flow Graph Extraction." *Proceedings of the 4th ACM Computer Science in Cars Symposium*. ACM, pp. 1-11. DOI: 10.1145/3385958.3385967.

- [4]**Fowler, Daniel S. et al. (2018).** "Fuzz Testing for Automotive Cyber-Security." *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, pp. 239-246. DOI: 10.1109/DSN-W.2018.00056.

- [5]**Kim, Hyunghoon et al. (2022).** "Efficient ECU Analysis Technology Through Structure-Aware CAN Fuzzing." *IEEE Access, 10*, pp. 23259-23271. DOI: 10.1109/ACCESS.2022.3152168.

- [6]**Koscher, Karl et al. (2010).** "Experimental Security Analysis of a Modern Automobile." *2010 IEEE Symposium on Security and Privacy*. IEEE, pp. 447-462. DOI: 10.1109/SP.2010.34.

- [7]**Lee, Hyeryun et al. (2015).** "Fuzzing CAN Packets into Automobiles." *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 817-821. DOI: 10.1109/AINA.2015.274.

- **OWASP (2022).** "Fuzzing." *OWASP*. Available at: https://owasp.org/www-community/Fuzzing

- [8]**Jo, Hyo Jin and Wonsuk Choi (2021).** "A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures." *IEEE Transactions on Intelligent Transportation Systems*, *23*(7), pp. 6123-6141. DOI: 10.1109/TITS.2021.3059141.

- **SocketCAN.** "An Implementation of CAN Protocol on Linux." Available at: https://www.kernel.org/doc/Documentation/networking/can.txt