Manipulating Car Diagnostics and Mechanics through Cyber Attacks

Sashaank Ganesh[1], Divyarani Sishtla[2]

[1,2]SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu 603203, India

---***---

Abstract - *Interconnected vehicle systems are now growing popular in the automobile industry. These vehicle systems provide remote access for monitoring and controlling the state of the vehicle. With an increasing number of features built to provide the user with utmost comfort and easy usability of the vehicle, it is also important to question the security and safety of these systems. These vehicles are not ready to be fully automated as there are various attacks that can be applied against these systems. These attacks are almost difficult to implement in real life but can be more devastating than expected. Any attempt to obtain unauthorized access to a computing system with the goal to take control of the car is a Cyber Attack. This paper provides a clear step-by-step analysis of creating a car hack workstation and gives to the faculty, the students and researchers the ability to implement car hacking in their own labs and universities. The technique used is CANBUS exploitation. This technique will involve the use of a Controller Area Network or simply referred to as a CAN BUS. The goal is to reverse engineer these CAN packets as safely and securely as possible using open source tools, simulators and a CAN to USB cable or a wireless connector.*

Key Words: CAN BUS, Electronic control units (ECU), On-board Diagnostics (OBD-II), Linux, SDL library

1. INTRODUCTION

These days, an increasing number of objects are being connected to the internet and can be easily controlled via any type of remote these days. With higher speeds and a widespread coverage of networks, one will be able to remotely access and control any gadget that is far from our reach. Automobiles are the latest addition to the Internet of Things (IOT) network of objects and are thus an important factor to evaluate and analyze what this technology has to offer. The rest of the paper is laid out as follows. The first section of the paper discusses Cyber Attacks, how Electronic Control Units are used in different cars. The second section of the paper talks about different people who have worked on Cyber Attacks in cars and their tactics. The third section elucidates how to hack using open-source software like Linux, as well as the setup. The constraints and issues of using CAN are briefly explained in this paper.

1.1 Cyber Attacks and Electronic Control Units

A Cyber Attack uses one computer or device to target single or numerous computers or networks. It is used to intentionally control a device for purposes like stealing data, launching additional attacks, disabling mechanisms and many more. Cyber Attacks are classified into two categories: Web-based Attacks and System-based Attacks.

Web based attacks are the types of attacks that can occur on a website or a web application, whereas system based attacks are designed to compromise a network. Cyber-attacks in cars come under system-based attacks. In cars, Cyber Attack is hacking the electronic systems, software, control algorithms, communication networks, to damage, to manipulate the data and so on.

Electronic Control Units (ECU) is a small device inside the vehicle which is used in automobiles to control the majority of the vehicle's functions. Just like a CPU, ECUs receive messages (or inputs) from different parts of the car. For example, when a person wants to open the window, he/she will press the window button. This signal or input is sent to the ECU of the car, and the ECU responds by opening the window. A few more examples are opening the car door, turning on the radio, the activation of airbags, and etc. A vehicle has many ECUs. Some vehicles have 80, some have 100 while some have more than 150. The ECUs are used for different purposes. Some control the windows, doors and locks, some control the engine and steering, some have crash sensors, etc.

1.2 Controller Area Network (CAN)

[12]When there are several ECUs present in the car, there must be one common network that handles carrying the messages or inputs and connects all the ECUS. Controller Area Network (CAN) connects all the ECUs, and it is one of the most important vehicle networks. Wires are used to connect different electronic devices. All electronic devices in the car were once connected by many wires, but as time passed by, more electronic devices were included for the safety of people which resulted in a bulky wire harness making it heavy and difficult to manage. Therefore, instead

of using wires, manufacturers used in-vehicle networks, and that's where CAN comes in.

CAN cannot be controlled by a host, in simple words, it doesn't have a master, this means there is no one in charge of determining when individual nodes can read and write data on the CAN bus. When an input is sent for some action and when the CAN is ready to transmit the data, it checks whether or not the BUS is available. If the BUS is free, then it writes a CAN frame onto the network. This data from inside CAN, can be easily extracted and analyzed using the On-board Diagnostics (OBD-II) interface. Cybercriminals can easily disrupt the status of the car or control the car because the ECU does not know the unit from which it received the message.

There are many hardware available on the market that provide CAN connectivity. CAN BUS-Y-Splitter, VSCOM Adapter, USB2CAN interface and USRP SDR are a few examples. Though there are many cost effective hardware tools for communicating with the CAN bus, the software needed to interact with these devices might not be present for all such tools, therefore writing your own code might be necessary.

1.3 Reverse Engineering using CANBUS

Reverse engineering the process of trying to understand or deduce how an application, device or a system works or functions with very little insight towards the subject. In automobiles the goal is to reverse engineer the CANBUS or specific packets to gain control of the vehicle. In this technique one would require the help of Linux commands such as candump, cansniffer, canplayer etc. to reverse engineer the CAN ID of the car and to find out specific messages of the target vehicle. Once the CAN ID of the vehicle is known it becomes much easier for a hacker to gain control of the ECU and control every individual command feature with a single set of commands to the CANBUS.

1.4 Software Prerequisites for CANBUS exploitation

To hack into any interconnected automobile one will require a handful of free and open source software which provides the respective tools. Any Linux distribution, CAN utilities and an ICSim (Instrumentation Cluster Simulator) is required. These tools can be downloaded from numerous websites across the internet. The software used in the car hacking module is free and open source. It is available on OpenGarages.org and is mainly composed of the Instrument Cluster Simulator or ICSim package (OpenGarages.org, 2017). ICSim was created by car hacker researcher Craig Smith, the author of The Car Hacker's Handbook (Smith, 2016). Smith's manual was the basis for our initial investigation of the car hacking. ICSim includes a dashboard simulator with speedometer, door lock indicator and turn signal indicator, and a control panel that allows users to interact with the simulated car network and use the throttle and control door locks, doors and turn signals. ICSim is based on other free Linux tools, including CAN utilities or canutils, which can be found in the package installation repository of most Linux distributions.

2. RELATED WORK

There have been several successful attempts made in hacking an automobile in the past decade. The first widely accepted hack was done in 2012 by a group of professors at the University of California, San Diego. By using two different attack vectors, these professors were able to cause a lock up in the car's braking system while driving [8].

Another famous demonstration was that researchers Charlie Miller and Chris Valasek controlled the steering and acceleration of the 2015 Jeep Grand Cherokee [7][8][9]. Finally, they noted that automakers should now consider implementing TLS and SSL encryption algorithms to maintain authenticity and integrity of their cars. At the time, manufacturers announced that they had to find a way to implement TCP in automotive networks before they could move forward with TLS and SSL encryption because they both need to establish a successful TCP session.

[6]Recently, a hacker managed to use a vulnerability in Tesla's servers to control the entire manufacturer's fleet. Elon Musk announced at the National Governors Association of Rhode Island in 2017 that it immediately became a company-wide threat and was recognized as "Fleet Wide Hack, considered one of Tesla's biggest threats."

It is a member conducted by one of Tesla's own people, Jason Hughes. He was one of the first members of the "root access" community and regularly posted software bugs. Just by knowing the VIN number of a Tesla car, Hughes was able to access the "tesladex" database, and thanks to his mothership, he was able to obtain complete information about any car and even be able to send commands to these cars. After reporting the error to Tesla, Hughes received a reward of $50,000 from the automaker, which was several times higher than the maximum reward limit for the error.

3. SYSTEM DESIGN AND IMPLEMENTATION

Every automobile consists of an On Board Diagnostics II (OBD II) port through which one can access the vehicles ECU's for various tasks ranging from emission tests to diagnostics. It is usually found within the reach of the driver underneath the steering wheel, but it depends on one car model to another. In order for us to now interact with the CAN, hardware such as a USB2CAN is required as a computer cannot be connected to the CAN directly. In order to connect to the OBD II port, hardware such as Macchina M2 is required in order to send and receive packets.

At the point when the on-board network controlling data design is compromised, the control messages could be infused into the in-vehicle network through the OBD interface. Before, the information was infused through a hard association among workstation, CAN information analyzer and OBD interface. Administrators need to fabricate extraordinary pass-on wires with many patches between hubs. From one perspective, the contact being bothersome may prompt an infusion of disappointment. On the other hand, as far as possible the scope of assault, which would not joke about this, isn't so natural to hack the vehicle.

The advanced vehicle contains various organizations for correspondences between the different subsystems. These networks incorporate a mix of CAN transports for standard powertrain and body correspondences, a nearby interconnect network for minimal expense applications fundamentally in body hardware, a FlexRay transport for high velocity synchronized information correspondences in cutting edge frameworks like dynamic suspension, media-orientated frameworks transport for sight and sound correspondences, and other restrictive organizations that are producer specific. The main conclusion is that CAN is mostly the primary network in today's modern vehicle

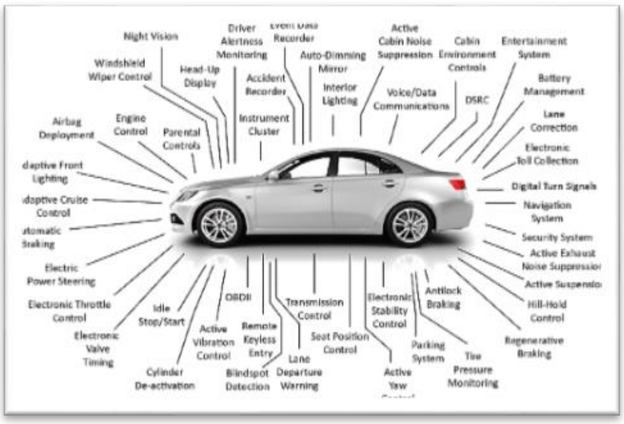


Figure 1 Modern Car Parts and Features

CAN depends on the broadcast correspondence instrument, which depends on a message-arranged transmission convention. It characterizes message substance instead of hubs and hub addresses. Each sent message has one of a kind message identifier, the discretion identifier (ARBid), inside the entire organization, much like a media access control (Macintosh) address, and it characterizes the content and the need for the message. CAN transport discretion authorizes transporter sense various access/impact evasion (CSMA/CA), which forestalls crashes on the transport when a few hubs go after access, otherwise called transport mediation. The different CAN transports work at various bitrates relying upon the cost, execution, wellbeing, and information rates needed on the network. The typical CAN transport information rates are 125 kbps, 250 kbps, 500 kbps, and 1 Mbps, yet different information rates are utilized for producer explicit executions

The process begins by connecting a hardware device or, in technical terms, an OBD device straight into the OBD II port. These OBD devices are as small as the OBD interface itself. These devices can now send, receive, read and write data packets into the vehicle's automotive network.

4. IMPLEMENTATION

Even though deciphering the packets from the virtual CAN network is beyond the scope of this current work, we can demonstrate the effective use of these packets by implementing a replay attack. In a replay attack, a hacker captures packets from a network traffic and attempts to re-inject them into the network later to repeat the same functionality, or to learn more about the packets by observing errors or other abnormal behavior in the affected devices. A replay attack can be one of the first steps a security tester or an attacker might use in attempting to reverse-engineer a device's functionality in a new or unknown network, like a virtual CAN network.

For this, just as the cansniffer software allowed us to view packets on the virtual CAN network interface, there are additional tools in the can-utils library that enable us to capture and replay those packets. For any action to take place in the car, it is important that CAN receives a message, analyses it and reverts back. But, to hack, we need CAN Utils. This enables us to communicate with the CAN network in the car. There are five important CAN Utils to be installed:

1. Cansniffer
2. Cansend
3. Candump
4. Canplayer
5. Cangen

Any packets which are sent into the network will be sniffed and stolen by the Cansiffer Util. Cansend writes down the command that the user gave. Candump allows the hacker to dump all the received data and also to print the data received by the CAN interface. Canplayer is to send CAN frames and to replay CAN packets. Cangen is to generate random CAT packets or frames for testing purposes. All the five CAT utils can be installed using the below command:

```
sudo apt-get install can-utils –y
```

5. SIMULTATION RESULTS

Now the setup is completed, and now one must be able to understand and use the ICSim for controlling the vehicle. Every ICSim consists of at least two components, a controls executable which gives the user the control of the virtual automobile including controls to acceleration, steering, door locks, brakes and turn signals and an icsim Instrument Simulator Program cluster file which simulates the automobile's dashboard and instrument panel. Instrument Cluster Simulator requires SDL libraries. SDL is a cross-platform development library for computer graphics and audio. Since ISCim draws and animates a virtual dashboard, this is required. This can be installed via apt-get.

```
sudo apt-get install libsdl2-dev libsdl2-image-dev -y
```

After the LibSDL libraries are installed, add the CAN utilities. The CAN utilities are included in some Linux distributions, but it's best to ensure the presence of these utilities. If absent, the CAN utilities can be installed using the command:

```
sudo apt-get install can-utils
```

The standard arrangement for running ICSim incorporates something like two segments, the ICSim Instrument Bunch Test system program record, which reenacts an auto's dashboard instrument board, and the controls executable, which gives the client control of the virtual auto, including increase in acceleration, controlling steering, doorway locks, and headlights.

Open three terminal windows. In the first window, open the Instrument Cluster Simulator application, icsim, on the vcan0 virtual CAN network interface we created.

```
: ~/ICSim/icsim vcan0
```

That's vcan0, with a zero, denoting the virtual CAN network we created by running setup_vcan.sh above. The dashboard instrument panel simulator will appear as shown in figure 2

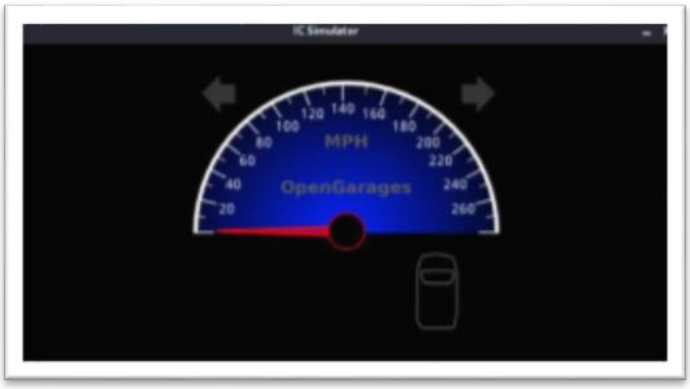
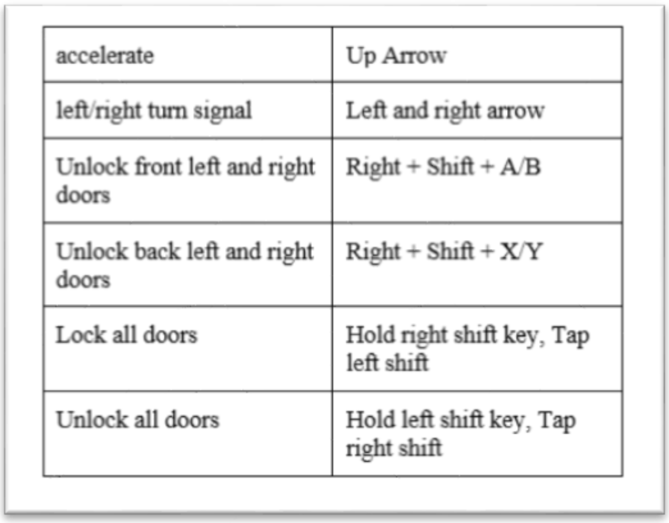


Figure 2 Dashboard Instrument Panel Simulator

At this point there will be no movement of the speedometer, and there will be no response from the dashboard as well. This is because there is no traffic in the vcan0 network. This issue can be easily addressed by working on the ICSim's controls. In a second terminal window, open the controls app:

```
~/ICSim/controls vcan0
```

The CANBus Control Panel application will appear on the screen. These controls will be similar to an Xbox controller through which we can manipulate the vehicle. We can also use a keyboard for this functionality. Changes can be made to the ICSim dashboard using the key combinations mentioned below



accelerate	Up Arrow
left/right turn signal	Left and right arrow
Unlock front left and right doors	Right + Shift + A/B
Unlock back left and right doors	Right + Shift + X/Y
Lock all doors	Hold right shift key, Tap left shift
Unlock all doors	Hold left shift key, Tap right shift

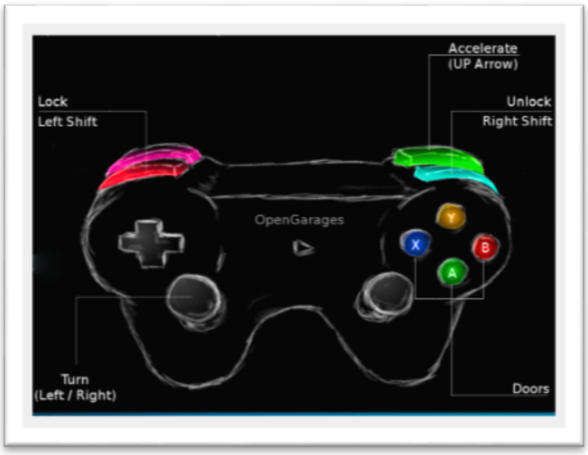


Figure 3 Controls of ICSism using Xbox Controller

For trial purposes, press the upward arrow () and check whether or not there is an increase in speed on the speedometer. After pressing the enter key, you should see the speedometer and other dashboard components do the same thing they did when you captured the packets, as shown in the figure 4. Notice that you should be able to get both the left and right turn signals on at the same time and unlock all the doors. You should also be able to change the indicators of the car even though the car is turned 'off'. In the event that no such change is noticed, check the means executed to see whether or not you have missed anything. Restart the technique if required [12][5]. The controls are the same when it comes to handling a car in a real life scenario. The controls are modifiable according to the user's comfort.

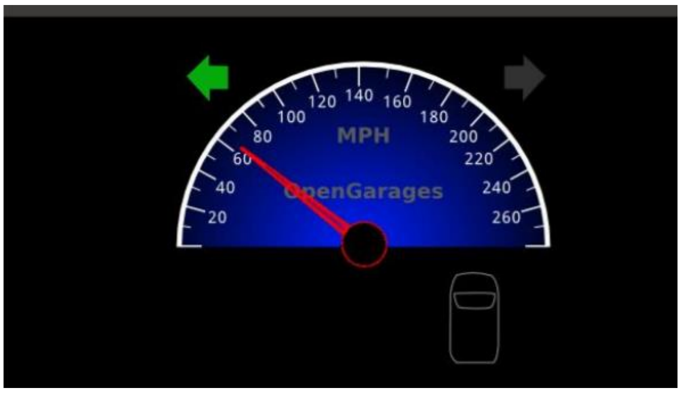


Figure 4 Dashboard Instrument Panel Simulator

6. LEVELS OF SAFETY IN DIFFERENT MODELS

Various experiments have been conducted to identify vulnerability and cyber safety on different models of cars [8]. The ultimate question that needs to be asked is "Is your car hackable?" Different cars have been rated based on three factors, namely, the size of the attack surface, system architecture and what experts like Miller and Valasek would call cyber physical features.

The size of the wireless attack surface refers to basic features like Bluetooth, cellular network connections, keyless entry, WiFi, radio, tire pressure systems, sensor monitoring systems etc. Any of these network connections could be a potential target for a hacker to find security vulnerabilities and gain a foothold into the car's network. Second, the vehicle's network architecture serves as an important factor to assess its cyber safety. Cracking into the system's architecture gives hackers the control of more important features like the steering and the brakes. Lastly, the aforementioned cyber-physical features include the capabilities like automated braking and parking, lane keeping assist, cruise control etc.

After analysis, it is concluded that the Infiniti Q50 in 2014 and the Jeep Cherokee in 2015 are the most hacked cars in the past decade. In particular, the Infiniti Q50 is an insecure architecture model that includes access to functions such as keyless entry, Bluetooth, wireless cellular connectivity, and tire pressure monitoring systems. All these functions are linked to the "Personal Assistant" application on the owner's smartphone. Miller and Valasek discovered that in the Q50 network, there are commands that allow them to access engine, steering, and braking commands. The car also contains computer-controlled functions, such as adaptive lane keeping assist and cruise control, which enable hackers to physically manipulate the car. The same thing happened to the Jeep Cherokee [7].

Another study distinguishes vulnerabilities based on different car price ranges. Low-end cars usually have a linear OBD to CANBUS connection. In other words, CANBUS can be easily located and identified. Another key feature of low-end cars is the low number of ECUs in each car. There are not many messages running on CANBUS, and messages are usually transmitted over a dedicated network. Therefore, it is useless to use data. In short, low-end cars are easy to use, but the attack will not cause serious damage to the vehicle [1].

Due to the intermediate system, the architectures of the OBD and Canbus system are almost the same as of low vehicles. The great difference between these two is the intermediate car generally contains the "quantity" of many ECU. That is, it is easier to access with the messages of the vehicle. Communication between ECU is strongly dependent on the canvas. By using almost in the same way as that of the lower car, the result obtained by the automobile of the intermediate segment is much greater. In fact, the intermediate car is the easiest and most serious damage to the hacker.

Any wise person would think that high-end cars are safer and better than cars in other market segments. However, this is not entirely correct. Although high-end cars have many safety considerations, they also have their own weaknesses. [1][2] Due to the network gateway, any request is restricted. But the main weakness comes from the fact that these cars have too many ECUs, and almost every part of the vehicle can be controlled by ECUs. It is true that the safety system is difficult to destroy, but once a collision occurs, the damage to the vehicle may be serious, and in some cases it may even be life-threatening. At present, if we carry out hacking attacks on high-end cars, the damage may be as serious as what happens on low-end cars. This shows that although taller cars are safer than other cars, they also have serious safety issues and risks.

7. CONSTRAINTS AND ISSUES

Every coin has two sides. Every technology out there has ups and downs. The software most hackers use is Linux. Linux users are increasing day by day, and people are demanding to upgrade it as the day passes. Linux has ups and downs too. Linux cannot be used by a beginner since it has several community-developed editions, so it might get pretty confusing. People who are new to coding and know very little about technology often get confused because of the complexity of Linux. So it is advised to learn more about coding and study the various methods of cyber-attacks that can be implemented. Another huge problem with Linux is its troubleshooting. Not all solutions are available online, so unless and until you're a tech expert who understands Linux extensively, you will not be able to move forward with the attack.

[12] There are a few issues involved in this method which can be avoided if the procedure is followed properly. Locating the OBD port in the car might be difficult. There are many more networks present in the car, so if you can't find what you're looking for, then you're probably in a different network. There are more arbitration identifiers in a real car than in an ICS, so you should carefully deal with them. Data transmitted over CAN depends on the car. For some cars, it is difficult. When there is a lot of traffic, the device can go into a bus off state. When this happens, replaying the message immediately will help with solving it.

Hacking into a car can be a lengthy and illegal process. There is no quick way to gain instant access to any vehicle. It takes from several minutes to even hours to gain control. Even with utmost proficiency a lot of wiring and external hardware are required, which will most likely make the entire process messy. Of course, one could argue that with the development of wireless technology, automobiles can be controlled via Bluetooth equipment. However, in reality, such devices are terrible for hacking as they usually have much slower data rates, and you will end up losing so many packets. Hacking is, most importantly, an illegal task to do. In most countries, hacking is considered a serious crime, and one can be fined or even jailed for committing such activities.

8. CONCLUSION

Modern automobiles have evolved into connected vehicles. In fact, today's automobiles contain a plethora of components that require message exchange via network resources. Given that the safety of drivers and passengers is dependent on the code running in their vehicle, this scenario necessitates mechanisms capable of ensuring the safety of the information exchanged by these network-connected components. This paper has detailed instructions for installing free, open-source car-hacking software for educational or research purposes. In addition, we described how to add a wireless connector capable of connecting to modern production vehicles via the OBD-II on-board diagnostic port. In the near future, we plan on including hands-on testing with faculty and students to experiment and make modifications to customize according to the test cases.

REFERENCES

[1] Yu Zang, Binbin Ge, Xiang Li, Bin Shi and Bo Li, "Controlling a car through OBD injection" in IEEE 3rd international conference on Cyber Security and Cloud Computing, 2016

[2] Sam Abbott-McCune and Lisa A. Shay, "Techniques in hacking and simulating a modern automotive controller area network" in IEEE international Carnhan Conference on Security Technology, 2016

[3] Fabio Martinelli, Francesco Mercaldo, Vittoria Nardone and Antonella Santone, " Car Hacking identification through fuzzy logic algorithm" in IEEE International Conference on Fuzzy Systems, 2017

[4] Sasan Jafarnejad, Lara Codeca, Walter Bronzi, Raphael Frank and Thomas Engel, "A car hacking experiment: when connectivity meets vulnerability" in IEEE Globecom workshops, 2015

[5] Craig Smith, "Car hacking 101: tools of trade" on April 8th 2016

[6] Fred Lambert**, "** The Big Tesla Hack: A hacker gained control over the entire fleet, but fortunately he's a good guy", August 27th, 2020

[7] Andy Greenberg, "How hackable is your car? Consult this Handy chart", June 8th, 2014

[8] Bryson R. Payne, **"**Car Hacking: Accessing and Exploiting the CAN Bus Protocol" Journal of Cybersecurity Education, Research and Practice, June 2019

[9] Charles Q Choi, "Hacking Cars to Trigger Gridlock" in IEEE Spectrum, August 5th, 2019

[10] Sasan Jafarnejad, Lara Codeca, Walter Bronzi, Raphael Frank, "A car hacking experiment: When connectivity meets Vulnerability", in 2015 IEEE Interdisciplinary Center for security, reliability and Trust

[11] Abbott-McCune, Sam Shay, Lisa A, "Techniques in hacking and simulation a modern automotive controller area network" in IEEE 2016 International Carnhan Conference on Security Technology

[12] Yogesh Ojha, "Car Hacking 101: Practical Guide to Exploiting CAN-Bus using Instrument Cluster Simulator", November 24th, 2019