

Automotive Hacking

Capstone Spring 2019

Andrew Gerlitz, Andrew Swain | ITSC310 | April 21, 2019

Contents

[Why Hack A Car 2](#_Toc6908883)

[CVSS Scoring 2](#_Toc6908884)

[Objectives & Goals 2](#_Toc6908885)

[Required Tools 2](#_Toc6908886)

[Risks 3](#_Toc6908887)

[Selecting A Platform 3](#_Toc6908888)

[2001 BMW 330I 3](#_Toc6908889)

[2006 Chevrolet Cobalt 3](#_Toc6908890)

[2014 Mazda 3 GT 4](#_Toc6908891)

[Getting Setup 4](#_Toc6908892)

[Hardware Setup 4](#_Toc6908893)

[Interface Setup 4](#_Toc6908894)

[Data Output 5](#_Toc6908895)

[Forging Packets 6](#_Toc6908896)

[Manipulating Data 6](#_Toc6908897)

[Proof Cases 6](#_Toc6908898)

[Additional Reading 7](#_Toc6908899)

# Why Hack A Car

## CVSS Scoring

The Common Vulnerability Scoring System is fairly standard in the industry and will assign severity scores for any known issues. These scores are calculated on a number of variables such as attack vector, impacts on CIA, environmental factors and so on. As you will find there is a complete lack of security when it comes to most CANBUS applications. We have scored our findings as a 6.6 base score, 7.6 environmental score, and 6.3 temporal score. These scores are out of a possible 10 ranking 2 medium risk and 1 high risk. This system puts a tangible value to a vulnerability that is easy to understand. For more information on the standard and its meanings see the Additional Reading section.

# Objectives & Goals

The key to this project will be having clear cut goals and objectives. Because of the diverse nature of these exploits, it is important to narrow in on the scope to something manageable. Until you are familiar with the process of modifying and exploiting CAN data it makes the most sense to have smaller, more achievable goals. Once the baseline has been established then you can go back and broaden the scope as you see fit. Our main objective for this project was to move the tachometer without the car running, this allowed a smooth transition from having no knowledge on the subject to being fluent in monitoring and modifying CANBUS data.

# Required Tools

* DB9 – OBD cable
* CANtact controller
* USB A – B cable
* Laptop running Linux (w/ can-utils installed)

Required hardware is listed above and your instructor should have this all on hand at the time you decide to take on this project. If you are interested in ordering your own hardware all of this can be ordered on Amazon for about $100.

Any distribution of Linux will work if you are able to install the **can-utils** package. We opted to use Parrot as the software required comes pre-installed. Kali Linux also has this package natively.

# Risks

There are some very real risks to car hacking especially when not in a controlled lab setting. It is important to remember that these are machines with a lot of moving parts and areas for things to go wrong. ALWAYS perform tests in a safe controlled environment and take the required precautions to protect yourself, the location, the vehicle, and those around you. Due to the nature of the CANBUS this is where all of the critical systems for a car are handled. Be absolutely certain about what you intend to do, and the data involved before sending commands. There is a very real possibility you could speed up the engine, flood the cylinders, or shoot a rod through the engine block if you do not do the proper analysis ahead of time. Every car is different and require their own analysis, packets for RPM on one car will not be the same to a different car, always keep this in mind.

# Selecting A Platform

For a project of this caliber it makes the most sense finding a single vehicle and doing a complete analysis. The difficulty will arise when trying to find a vehicle. There are not many people that are open to letting students mess with critical systems of their vehicle, as there are some inherent risks involved. Always asked for INFOMRED consent before attempting this project.

## 2001 BMW 330I

This car belongs to Andrew G. and was our first pick as it was always at school and we had very in-depth knowledge about the platform. Being a European car the OBD port transmits the ISO 9141-2 protocol, as far as we are aware this is the case with the majority of European vehicles. While it is possible to sniff packets on this protocol, we had already committed to using the CANBUS and ordered hardware accordingly. There is a CANBUS present within the vehicle and it carries the same data found in most vehicles but requires a tie in to the harness elsewhere. If we were to use this car the best place to get a CAN signal is the harness behind the instrument cluster.

## 2006 Chevrolet Cobalt

Belonging to Andrew S. this cobalt uses the CAN protocol via the OBD port making for simple and easy interaction between the laptop and integrated systems. Being that is from the 2006 model year there are inherently less integrated systems making the data far easier to sift through. The basic systems will be on the CANBUS such as the cluster, engine management, and so on. This is an easy vehicle to get started with but provides limited functionality for attack.

## 2014 Mazda 3 GT

Belonging to a friend, the Mazda made sense to test for additional avenues of exploitation. Much like the Cobalt, all of the major systems are managed on the CANBUS with additions including climate features, steering wheel control, and a full infotainment system. We would like to spend more time with this platform to become more familiar with the systems involved and exploring other features such as Bluetooth or infotainment exploitation. Newer vehicles tend to have more data present on the CAN bus.

# Getting Setup

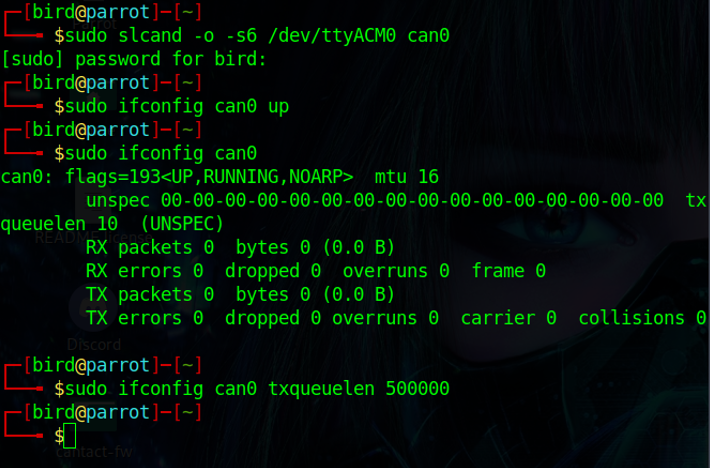
We found that working from a clean restart of the laptop gave us the best results.

## Hardware Setup

* Plug the OBD to DB9 cable into the cars OBD port (usually under the dash by the pedals)
* Plug the DB9 side into the CANtact
* Plug the CANtact into the laptop using a USB A – B cable

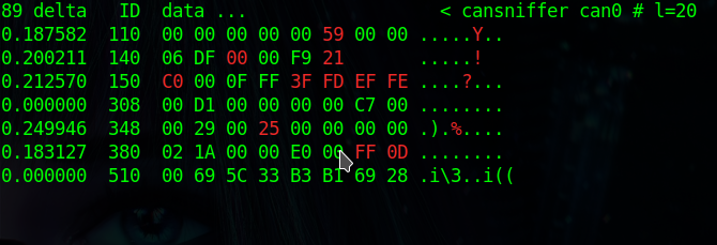
## Interface Setup

1. sudo slcand -o -s6 /dev/ttyACM0 can0
   1. This enables the CANtact device to communicate with the laptop. You might have to check system logs for the actual device path.
   2. -s6 refers to the baud rate of the bus. This speed seems standard across most vehicles and probably won’t need to be changed
   3. Can0 refers to the interface name, you can set this to whatever.
2. sudo ifconfig can0 up
   1. Enables the interface set in the previous command
3. sudo ifconfig can0
   1. Checking to ensure the interface exists
4. sudo ifconfig can0 txqueuelen 500000
   1. Changes the size of the buffer. We found that if this value is too small data will be rejected by the CANtact and by extension the CANBUS.
   2. The length value is arbitrary and can be set to whatever, as long as it’s big enough. Testing will be required to figure out what number works. 500000 worked for us.



## Data Output

Using the command <cansniffer -c can0> outputs all data found on the bus and is filtered only to packets that are changing (these are highlighted in red)



At this point you are ready to start messing around with the systems in the vehicle. Be sure to take plenty of notes and only try one system at a time to reduce clutter, singling out one packet at a time. Use whatever method you find suits you best, we can’t give you all the answers.

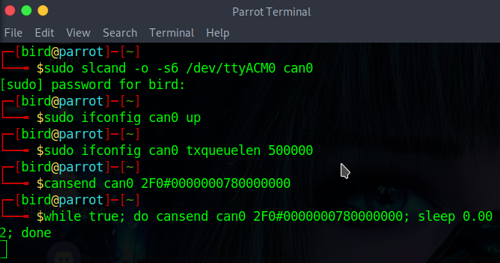
# Forging Packets

There are a couple of methods to send data over the CANBUS, the one we used majority of the time was a simple <cansend> command. You can write scripts to automate the process as well, but we ran into issues when consistently sending information and had to restart the laptop. It is important to note that packets are sent every 0.002 seconds, but we found 0.006 worked the best in a scripting format.

## Manipulating Data

Majority of data manipulation happens when performing one task repeatedly and taking that found information, attempting to replay it to the bus. For example, if you hit the hazard light button in the car do you see a packet that seems to correspond with the on and off function of the lights? If so you need to take note of the ID of the packet (this will correspond with specific functions within the CANBUS) and observe the data that is output. Once you know what that data is you can perform a <cansend ID#DATA> and observe any changes.

The image below has the 2F0 packet from the Cobalt, we know this is the ID required to flash the traction control light on the dash. We were also able to deduce that the ‘07’ byte of data corresponds to the light being on, this would be ‘00’ if the light was off. Therefore, by sending the packet below we were able to observe the light flash on the dash. There are more examples of this in the included notes and final presentation.



# Proof Cases

Have a look at the included videos on this USB, as these will give you a better understanding of the tasks we were able to accomplish. Again, it is important to remember that the ID and data you will see are specific to those vehicles, that data will not perform the same functions across different platforms. We were able to get the RMP to redline, flashed lights on the clusters, unlocked the steering column, enabled the HUD, and steering wheel controls on our two viable platforms.

# Additional Reading

Majority of the work we had done was via trial and error. These links are something you should consider reading. They make mention of other programs you could use as well as other avenues to explore in the realm of automotive hacking. Car hacking is a very rewarding task and it is important to have fun with it.

How to hack a car – A quick Crash-Course

<https://medium.freecodecamp.org/hacking-cars-a-guide-tutorial-on-how-to-hack-a-car-5eafcfbbb7ec>

Charlie Miller and Chris Valasek’s research

<http://illmatics.com/carhacking.html>

Car Hacker’s Handbook

<http://opengarages.org/handbook/>

CVSS Additional Information

<https://www.first.org/cvss/>