# BRUTE FORCE ATTACK ON AN AUTOMOTIVE CAN BUS

Research Document

Gary Morrissey C00259786

# Table of Contents

# Research

## Goal

The goal of my research is to reverse engineer a cars dashboard through the cars CAN Bus with the use of Brute Force attacks. A cars CAN Bus system is made up of many Electronic Control Units which communicate with different parts off the car. For this project, the objective of your Brute Force Attack is to reverse engineer the Original Equipment Manufacturers (OEM) protocol, in this case on a Mazda 6. I will use an image Comparison tool to compare and contrast a picture before and a picture after each attack to see any changes and create a report based off what attack did what operation.

## Deliverable Users

The purpose behind this project is not to create an application for general use, but to show the weakness in a cars CAN Bus system and expose its vulnerabilities through reverse engineering the OEM protocols. The people using this would be penetration testers to test their cars vulnerabilities but may also be used by a person with no technological background for an educational purpose.

## Deliverable

This project will deliver an interface that can used to connect to a cars CAN Bus through the PEAK software. Once your laptop is connected to the car via USB to OBD2 port, you can use my application to brute force attack the dashboard. The software will interact with a camera pointed at the dashboard which will take a photo of the dashboard before sending a brute force attack, then send the attack and take another photo of the dashboard. The application will use an Open API to compare the images and point out the differences in them. The software will take this comparison of the two images and take note of what attack created the change. These will be added to a table of all attacks that were successful. Once a full collection is made of all changes, a library will be created, and the user will obtain all information needed to make their own changes to the dashboard. Meaning with a click of a button they can turn on and off the switch for the check engine light on the dashboard.

The Peak software is a trusted fully developed application with an API I will use to communicate with the CAN Bus.

## Deliverable Precent

While there are many brute force attack applications available for use, there is not one specific for a cars CAN Bus system. I gained inspiration from seeing different ethical hackers using wireless connections to gain access to locked cars as well as wired connection to change the cars default manufacturers settings. Many of these ethical hackers created their own Arduino unit to connect to the car, but I will be using a PEAK USB to OBD2 connection along with the PEAK software's API. My software will be unique due to its necessary connection through the PEAK software. The PEAK software allows the user to connect to the cars OBD2 port making an ease of access connection.

# Reverse Engineering

Reverse engineering is the process of breaking down a system in order to understand its structure, functions and design.

For this project, I will be using reverse engineering principles a long with brute force attacks to understand what messages affect what part of the car's dashboard.


On this dashboard above, I will be using brute force attacks to repeatedly send messages changing each bit by increasing it by 1 at a time. When a change occurs, the message sent will be stored in a database and a photo will be taken. The photo will be compared to a control photo, which I will then note and save the message under the action I see occur. Once I have a full database for each message and its corresponding action, I will be able to document how the dashboard works and how the different messages in the CAN Bus affect it.

# Method

## CAN Bus

The CAN Bus (Control Area Network) is the communication protocol used in most cars. It is made up of a number of ECU's (Electronic Control Unit) that interact with different parts of the car.

The CAN Bus sends messages to these ECU's that contain the parameters for that part of the car to run. For example, to turn on the full head lights, once the button is clicked it will immediately send a message to the head lights ECU to switch them on.

The CAN Bus is made up of a two-wire system. A high speed bus and a low speed bus. The low speed bus has a transfer rate of 125 kbps. This bus is used for car diagnostics, window power control, dashboard controls and various displays. The high speed bus is transfers data at a higher rate, ranging from 500 kbps to 1 mbps. This bus is used for more immediate actions such as the airbag system, the braking system and other engine control units.

The CAN Bus is a system with a message collision protocol in place. It occurs when more than 1 ECU is trying to send messages at the same time. The messages will contain priority levels when being sent which are compared against each other. Whichever message has a low priority will stop the ECU from sending the message until the message with the higher priority has been sent. It then retries to send the message.

CAN messages are made up of frames. There are four types of frames, a data frame which carries information and remote frames with requests data, error frames which simply report errors and overload frames which report overload conditions from the ECU's. Each frame contains an identifier, which determines the priority of the message and the actual data of the message which has the command or parameter.

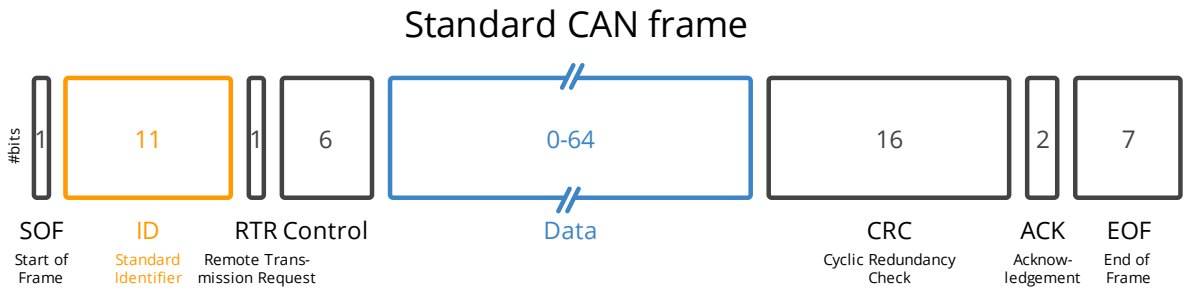
One of the most important features of the CAN Bus system is its standardization. The protocol behind it is standardized which ensures ease of access and communication among different manufacturers and people trying to run diagnostics.

## ECU

ECUs are the electronic control units in the CAN-BUS. They can control various operations in the car such as the engine control unit, the airbags, the indicators, audio system etc. The high and low buses in the CAN system allow the ECUs to communicate to each other. The ECU that wishes to make an operation can send out a broadcast message to all other ECUs through the CAN BUS. The other ECUs then receive the broadcast information and can check the data to decide whether to accept or ignore it.

## CAN Frames

The structure of a CAN frame is built up of 8 components.




These 8 components are:

1. The start of frame (SOF) which tells the other nodes in a CAN Bus that this node requests to communicate.

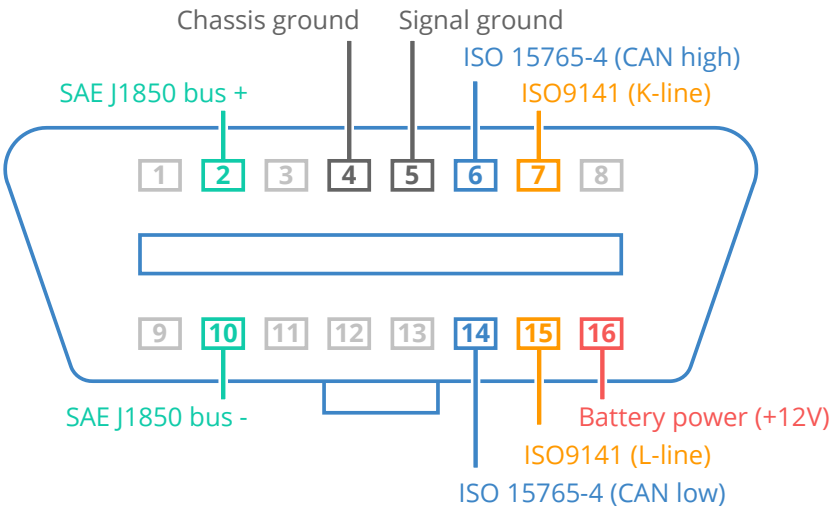
2. The ID is the part that identifies the priority level of the frame, the lower the value of the frame, the higher the priority level.
3. The RTR is the remote transmission request which tells if a node is sending data or requesting data from another node.
4. The Control has the IDE (Identifier Extension Bit). The DLC (Data Length Code) is 4 bits that specifies the length of the data bytes to be transmitted to the node.
5. The Data component contains the data bytes that has the CAN commands to be extracted from the message.
6. The CRC (Cyclic Redundancy Check) is the component used in the frame to ensure the integrity of the data contained.
7. The ACK's (Acknowledgment) purpose is to show the node has received the data correctly.
8. Finally the EOF simply marks the end of the frame.

## OBD2

The OBD2 port (On Board Diagnostics) is used to connect into the cars operating system. Through this port, users can monitor the vehicles performance, detect and report any issues related to emissions, fuel efficiency and any other critical components. If there is a fault with the car, the system will give out DTC's (Diagnostic Trouble Codes) to identify the problem.

### Physical Port

The position of the port itself changes from car to car, but it is mostly somewhere near the steering wheel and may be hidden behind panels. The OBD2 connector contains 16 pins. There are many protocols ran through this port, but the one I will discuss here is the CAN protocol. For every protocol, pin 16 supplies the battery power, but with CAN there is a high and a low bus. The CAN high traffic goes through pin 6 which runs up to 1mgps and the CAN low traffic goes through pin 14 which runs up to 125kbps.




### Protocols

The OBD2 port has a standard set of 5 protocols. These protocols have developed on top of one another over the years and the newest one is always best practice and more than likely required in newer cars. These protocols are:

1. SAE J1850 PWM: This is an older protocol that uses pins 1 and 2. It had a communication speed of 41.6kbps
2. SAE J1850 VPW: This protocol uses pin 1 and its communication speed is 10.4kbps
3. ISO9141-2: Also, an older protocol mostly used in the EU between 2000 and 2004. This protocol uses pin 7 and can use pin 15 but it is not necessary nor required.
4. ISO14230-4 (KWP2000): This protocol uses pin 7 and have two variants. They both have a communication speed of 10.4kbps and the only difference between them is their communication initialization.
5. ISO15765-4 (CAN-BUS): This is the protocol I will be using in the project. It is the most modern protocol today. It uses pin 6 and 14 as stated above. There are 4 alterations of this protocol. They all share the same name, but the difference between them is the identifier length and the bus speed.
    a. ISO15765-4 CAN (11-bit ID, 500 kbaud)
    b. ISO15765-4 CAN (29-bit ID, 500 kbaud)
    c. ISO15765-4 CAN (11-bit ID, 250 kbaud)
    d. ISO15765-4 CAN (29-bit ID, 250 kbaud)

## Security

The standardization of OBD ports in cars have many positives for ease of access to the cars internal systems for diagnostics or services. However, this also means that there

## DTC

The purpose of diagnostic trouble codes is to tell the user the issue with the vehicle if there is one. The on-board diagnostic system sends out these codes. These codes come in the same format, they each start with a letter followed by a series of numbers. The letter at the start identifies the category of the problem and the series of numbers identifies the specific problem in that category.

These categories consist of:

- P codes: These are related to the engine and transmission
- B codes: These are related to the body control modules, airbags and other body related systems
- C codes: These codes are related to the anti-lock brake system, suspension and other chassis-related system
- U codes: These indicate problems with the vehicle's communication system.

Once these codes are read and fixed, the user must use a tool to clear it using a diagnostic tool, as if not cleared the system will continue to log the same code repeatedly.

These codes may also have different severity levels, meaning that a code with a minor severity level will be handles after a code with a critical level.

Am important fact about these codes is that they may not always be standardized, some vehicle manufacturers may use different codes for the same malfunction. So, it is important to refer to the vehicles service manual or an online database to get the correct issue.

## Canon Remote Capture Tool

Canon themselves have an image capture tool that you can download from their website for free. This tool can be used once a canon camera/DSLR is connected to a computer. Once done, the user has access to many functionalities of the camera.

1. Image Capture: This function allows the user to take images remotely. It triggers the camera to take a photo, then uploads said image directly to the computer.
2. Camera Settings: With this function the user can change any settings they deem necessary such as the aperture, the shutter speed and any colour balance needed.
3. File Transfer: Any image captured through the tools can be transferred directly to a dedicated folder on the computer which makes editing, viewing and comparing images easier.
4. Focus Control: Through this function, the user can focus on certain aspects of an image before taking it, which will be useful for this project as the dashboard lights can be very small and tricky to see.
5. Live View: This capability will ensure the user taking the image has the correct frame in position and allows the user to leave the camera in the exact same spot instead of moving the physical camera to check.

## PEAK Software

The PEAK system is an organization founded in 1999 that deals in the software and hardware components of automobiles and the industrial communication sector. This includes the physical connection links and the background software in order to interact with automobiles inner communication systems. They excel particularly in the CAN field.

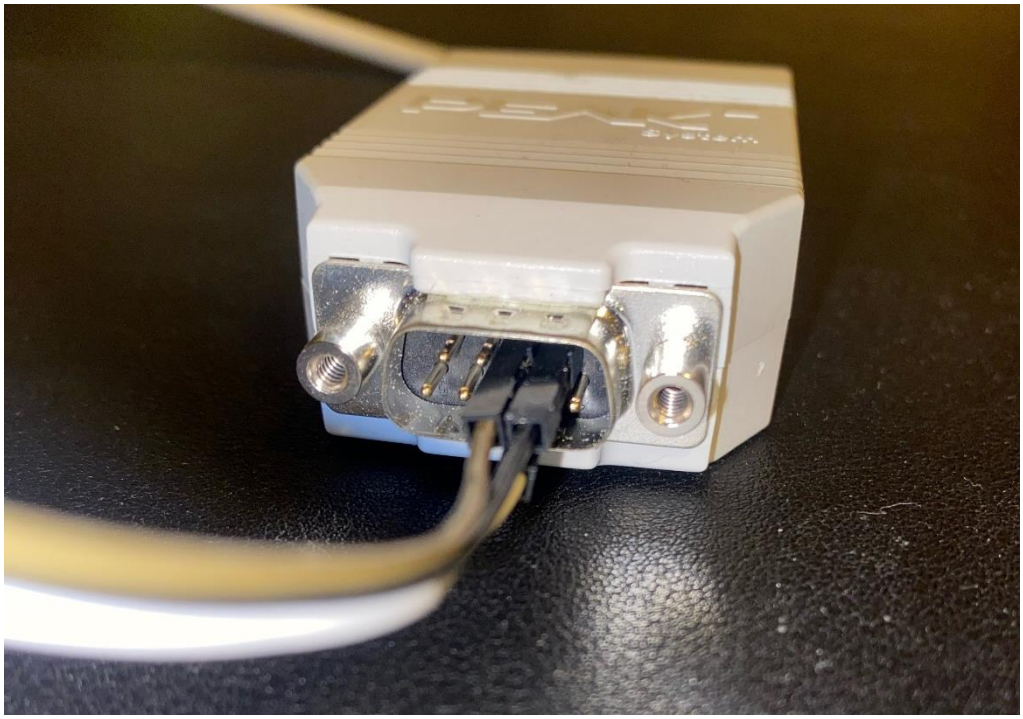
The list of hardware materials varies from CAN connections for High-Speed communication, Input/Output modules with CAN connection for control, Routers and gateways for forwarding message to the CAN bus and receive information back. The different software's available on their website allows windows and Linux users to communicate with CAN buses, provide interfaces for the different CAN protocols and some configuration software for the CAN hardware provided by PEAK System.

The PEAK System website provides users with manuals on how to operate their various services such as user manuals and related documentation.

## PCAN-USB

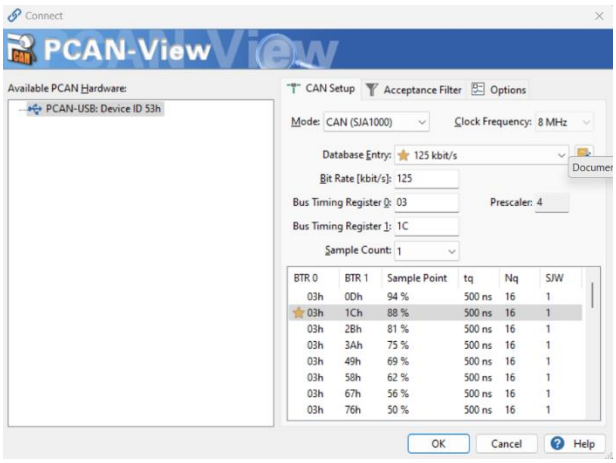
The PCAN-USB dongle is used to create a connection from a computer to a CAN Bus. The dongle can be seen pictured below. This connection will allow the user to monitor the traffic being sent around a CAN Bus. This will be useful to for the purpose of this project as it will allow me to view the dashboards traffic using the PCAN-VIEW. I will expand on this software later in the document.

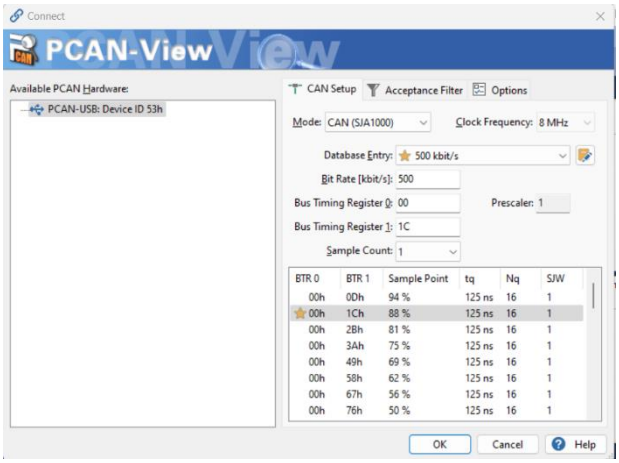

For the purpose of this project, using this cable I will be connecting to the dashboard using pin 2 and pin 7. This can be seen in the image below.



## PCAN View

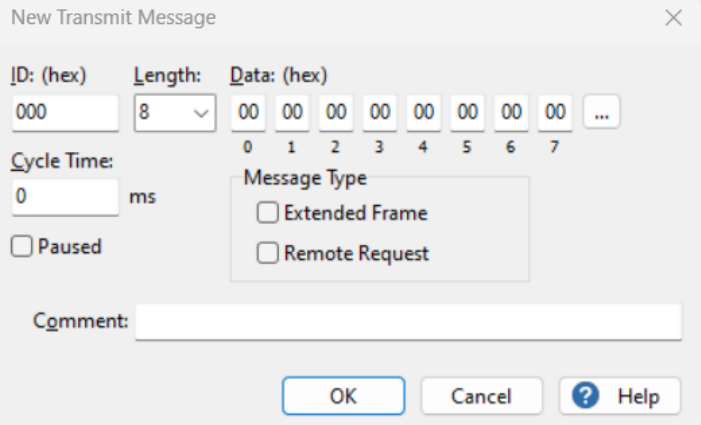
PCAN-View is a PEAK System software for windows that allows the user to view, send, receive and intake CAN data traffic. Once the computer is connected to the CAN bus with the PCAN-USB dongle, the PCAN-View will automatically recognize the dongle driver. From there the user will be met with an interface to select which bit frequency they want to connect to. For this project I will be connecting to the CAN High speed bus and the CAN Low speed bus. When I connect to the Low speed bus, I will select the frequency 125kbps, then for the High speed bus I will select the 500kbps frequency. Below is the selection process for both, first is the low speed bus and the second is the high speed bus.




After selecting these necessary settings, the user can avail to the full software functions on the CAN Bus.

Once the user is connected, they can then send messages to the CAN Bus to make changes. For this project I will be showing this process on a car's dashboard.




These messages are 8 bits long, so in order for me to reverse engineer the car's dashboard I will have to go through each bit and change them one by one for each ID. I will then be able to create a database of messages that worked and through the use of the photos being taken, I can then determine what each message did.

### PCAN Basic

The PCAN Basic API is made available for people to develop the software with CAN. It contains all the functions and necessities to connect to with the PCAN hardware (such as the CPAN-USB dongle above). The download file contains the device driver for the hardware and the DLL for the API's functions. It also includes files with examples of C++, C#, Java and Python to develop your own interface. This will be helpful to me for when I create my own Interface to reverse engineer the dashboard through the CAN Bus.

### IDE

The IDE I will be using to develop my interface will be Visual Studio 2022. I have used this IDE before and I am familiar with how it works. The interface is very user friendly and has a lot of online resources on how to navigate and use it. This IDE is compatible with C++ which is the language I will be using to create my own interface. It is available for free download on the Visual Studio website.

## References


1. https://dewesoft.com/blog/what-is-can-bus
2. https://www.autopi.io/blog/can-bus-explained/
3. https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial
4. https://www.csselectronics.com/pages/obd2-explained-simple-intro
5. https://www.linkedin.com/pulse/understand-can-bus-vs-obd2-arslan-falak/

6. https://www.canon.ie/support/consumer_products/software/eos-utility.html
7. https://www.peak-system.com/PCAN-View.242.0.html?&L=1
8. https://www.peak-system.com/PCAN-Basic.239.0.html?&L=1
9. https://www.peak-system.com/PCAN-USB.199.0.html?&L=1
10. https://www.peak-system.com/Profile.66.0.html?&L=1
11. https://visualstudio.microsoft.com/vs/community/