Hacking Automotive Systems

Workshop by warpnet

Agenda_

- Intro
- Icebreaker
- Background & Context
- Exercise 1
- Exercise 2
- Regroup
- Next Steps

About me_

Roald Nefs

Chief Technology Officer at **Warpnet**

Board Member at **Stichting Friendly Cyber**

Organizer at **BSides Groningen**, **BSides Amsterdam**, and **Cloud Native Groningen**

Solution Architect Security at **CJIB**

Hardware Hacket at home

Disclaimer_

This workshop is for **educational purposes only**.

Hacking cars:

🧱 **Brick your car** (permanently damage ECUs or components)

⚠️ **Modify safety-critical systems** (e.g. brakes, steering, airbags)

🚫 **Break the law** (e.g. tampering with odometer or VINs)

Proceed at your own risk. Only work on vehicles you own or have explicit permission to test.

Disclaimer_

This workshop is for **educational purposes only**.

Hacking cars:

🧱 **Brick your car** (permanently damage ECUs or components)

⚠️ **Modify safety-critical systems** (e.g. brakes, steering, airbags)

🚫 **Break the law** (e.g. tampering with odometer or VINs)

Proceed at your own risk. Only work on vehicles you own or have explicit permission to test.

Threat Modeling modern connected cars_

EXERCISE

Threat Modeling_

🛠️ **Scenario:**
Imagine you are an attacker targeting a modern connected car. List as many attack surfaces or entry points as possible in 5 minutes.

Modern cars_

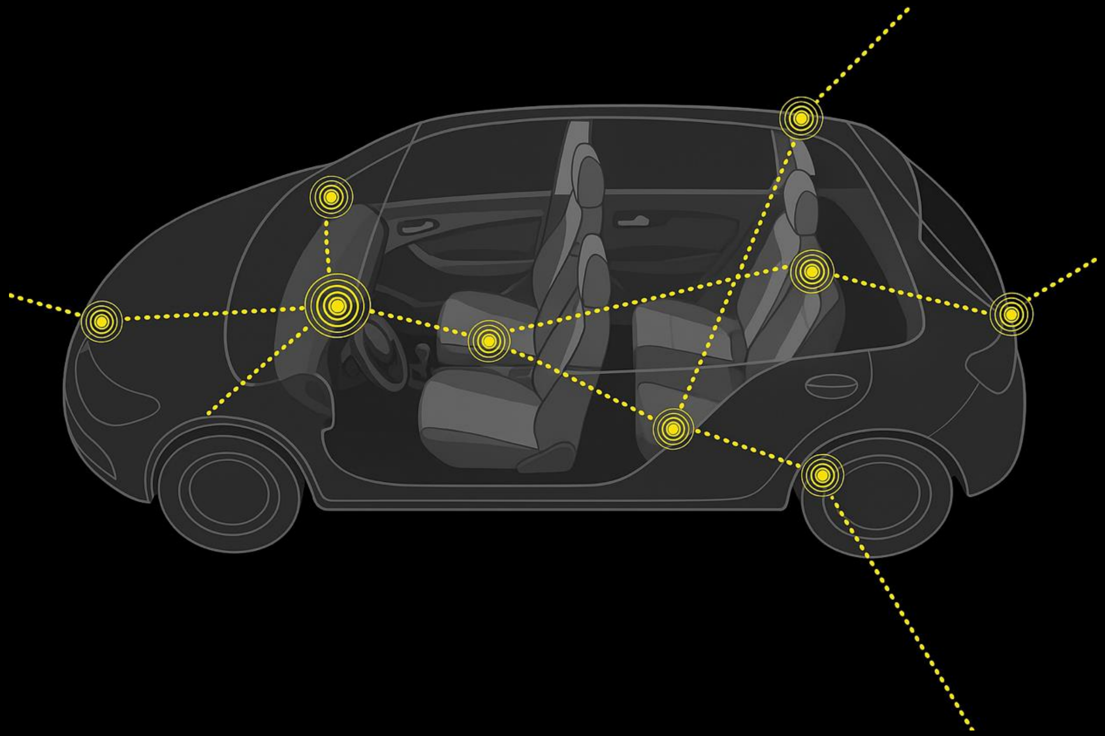
Modern cars are computers on wheels:

- **Driving assistance:** anti-collision, line detection, ESP/ABS, road sign reconnaissance
- **Connectivity:** GPS, LTE, WiFi, Bluetooth
- **Onboard services:** remote control, localization, auto-diagnostic
- **Emissions:** fuel efficiency, hybrid systems, battery optimization
- **Autonomous driving:** Honda reached level 3 autonomous driving in 2021

ECUs_

ECU: Electronic **C**ontrol **U**nit

- Acts as the brain of a car subsystem.
- Reads data from sensors, manage actuators (e.g. throttle, brakes, steering).
- Communicates with other ECUs, external devices, and backend servers of wired or wireless networks (e.g. CAN, Ethernet, Bluetooth, cellular)
- May be built using microcontrollers (MCUs) or more powerful systems-on-chip (SOC), depending on complexity.



Common ECUs_

ECU are commonly labeled using abbreviations, which may vary between manufacturers.

Common ECU names

- **ECU:** Engine Control Unit
- **TCM:** Transmission Control Module
- **PCM:** Powertrain Control Module
- **BCM:** Body Controle Module
- **ICM:** Instrument Cluster Module
- **ABS:** Anti-lock Braking System
- **ADAS:** Advanced Driver Assistance System
- **IVI:** In-Vehicle Infotainment
- **BMS:** Battery Management System

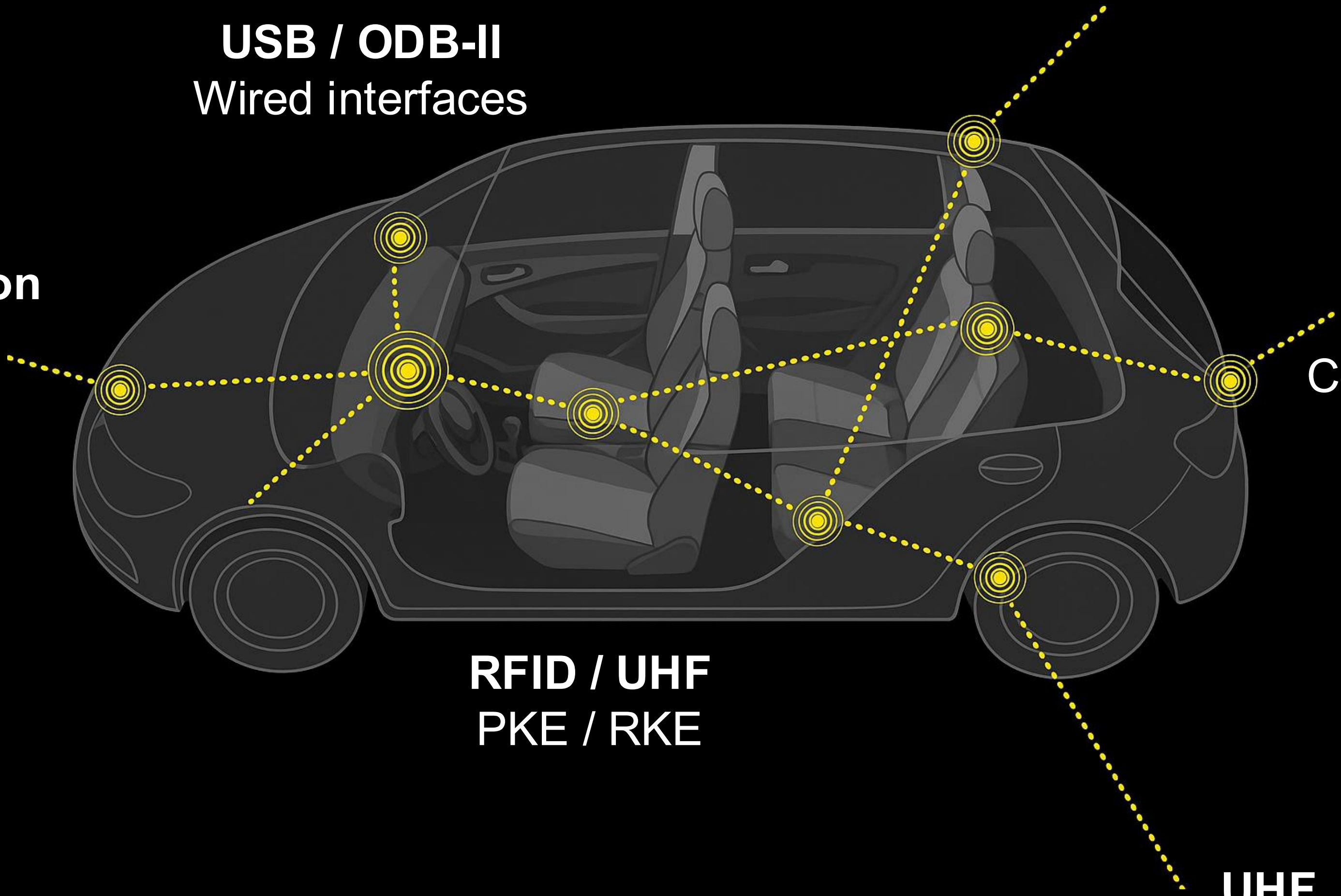
Connected ECUs_

ECU networks in modern cars. Different networks balance cost, speed, robustness, and safety depending on ECU needs. Some ECUs connect to multiple networks.

Key network types

- **CAN:** Controller Area Network, robust and widely used e.g. engine, brakes, airbags
- **LIN:** Local Interconnect Network, low-cost, used for window motors, mirrors
- **Automotive Ethernet:** high-speed, used for ADAS, cameras, infotainment
- **FlexRay:** high-speed, fault-tolerant, used advanced braking, steer-by-wire
- **MOST:** Media Oriented Systems Transport, designed for multimedia and infotainment data

Attack surfaces_



V2V / V2G / V2L / V2X
Short / medium range communication

USB / ODB-II
Wired interfaces

Computer vision
ADAS

PLC
Communication

RFID / UHF
PKE / RKE

UHF
Direct TPMS

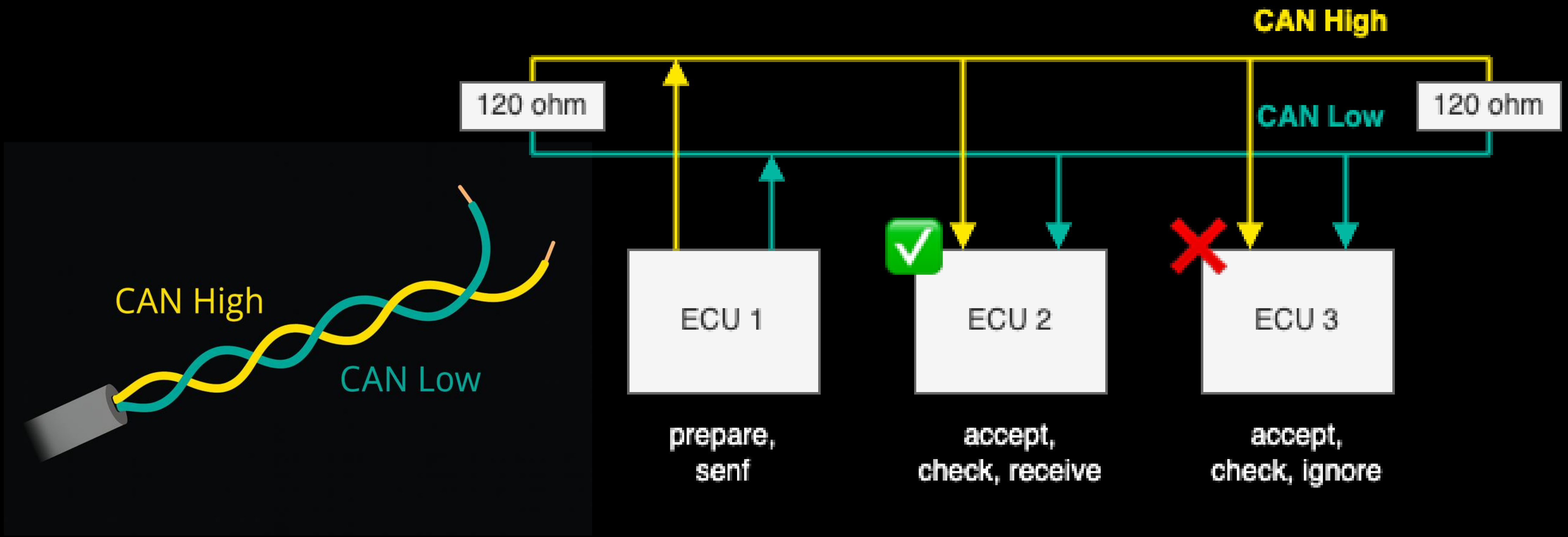
Automotive security_

In **2015** security researchers shown **major vulnerabilities** in a Jeep, causing the recall of **1.5M** of cars in the US.

Hackers Remotely Kill a Jeep on a Highway by WIRED
https://www.youtube.com/watch?v=MK0SrxBC1xs

The CAN bus_

Controller Area Network (CAN) is a robust, real-time communication protocol designed for vehicles and industrial automation.



CAN High

120 ohm

CAN Low

120 ohm

ECU 1

prepare,
senf

✓ ECU 2

accept,
check, receive

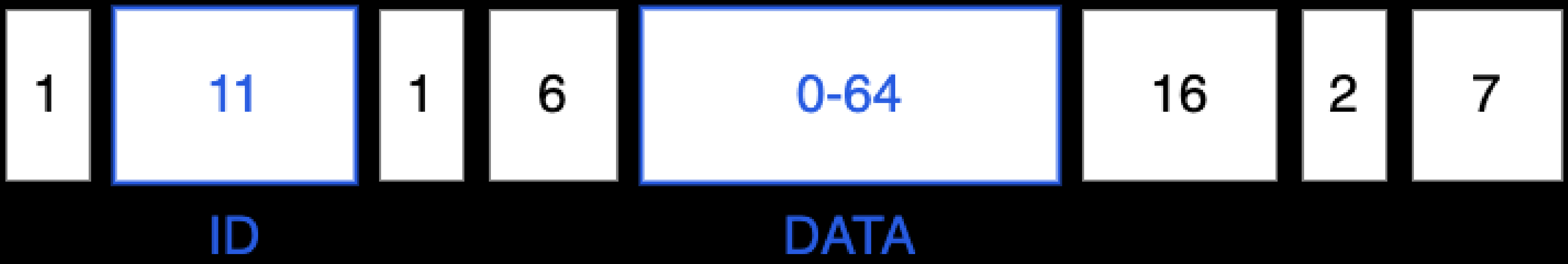
✗ ECU 3

accept,
check, ignore

The CAN frame_

Contains 8 fields including:

- **ID:** the arbitration identifier, lower values have higher priority
- **Data:** data bytes aka payload
- …



1	11	1	6	0-64	16	2	7
	ID			DATA			

CAN bus security flaws_

🔁 **No native protection against replay attacks**
Messages can be captured and resent without detection, allowing an attacker to repeat valid commands.

👤 **Lack of sender authentication**
The arbitration ID system does not verify the origin of a message, making it easy for an attacker to spoof a legitimate ECU.

💣 **Susceptibility to Denial-of-Service (DoS) attacks**
By injecting error frames deliberately, an attacker can force ECUs into bus-off state, effectively disabling them.

🔓 **No built-in encryption**
All data transmitted on the CAN bus is in plain text. Sensitive information (e.g., diagnostic data, text messages) can be easily intercepted and manipulated.

📢 **Broadcast nature of the bus**
Every ECU receives all messages, increasing the attack surface and making eavesdropping trivial.

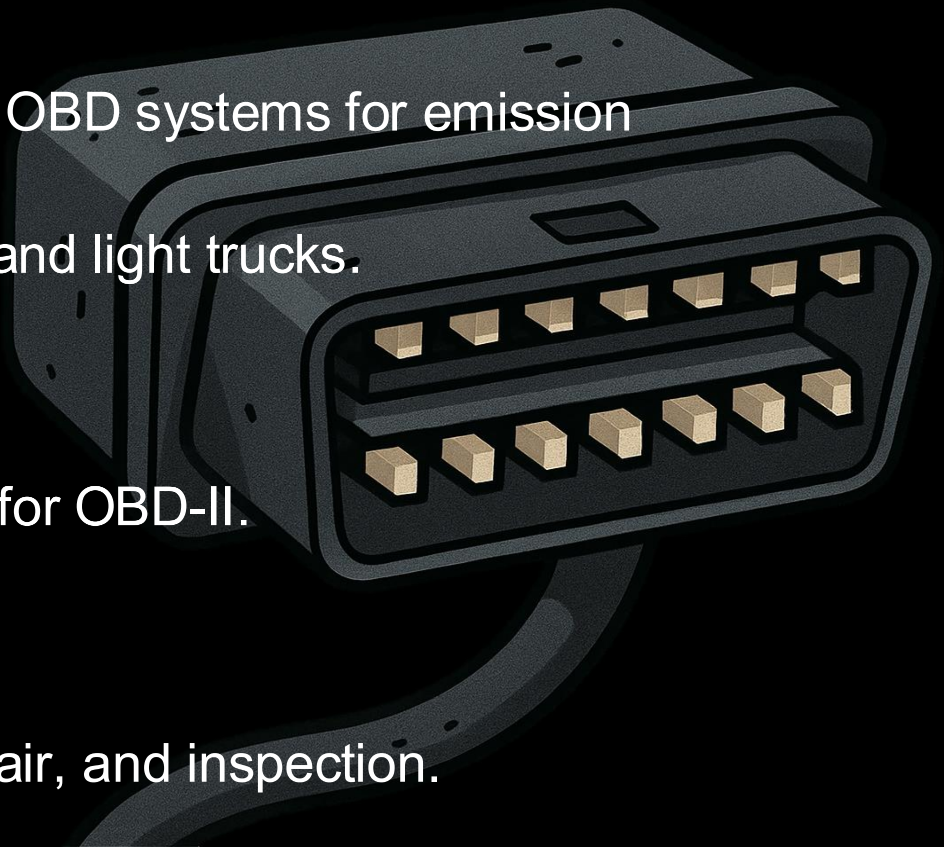
ODB-II_

🔌 Connector

Standard 16-pin connector, typically located near the steering wheel.

🕰️ History & milestones

- **1991:** CARB (California Air Resources Board) required OBD systems for emission control.
- **1996**: OBD-II became mandatory in the US for all cars and light trucks.
- **2001:** Mandatory in the EU for gasoline vehicles.
- **2003**: Mandatory in the EU for diesel vehicles.
- **2008**: US vehicles must use CAN as the protocol base for OBD-II.

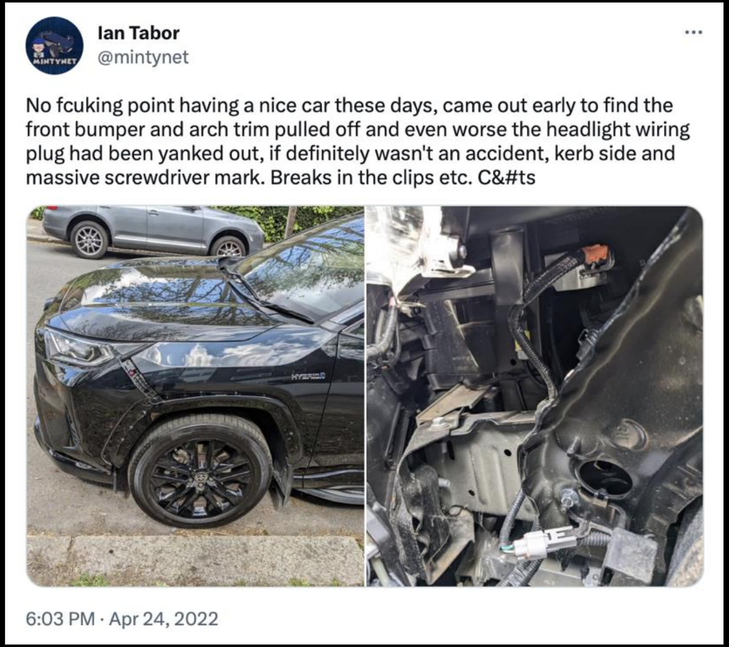


💡 Why important?

Enables emission monitoring and diagnostics.
Provides standardized access to car data for service, repair, and inspection.

Automotive security_

In April 2022 an automotive security researcher Ian Tabor tweeted that vandals had been at his car, pulling apart the headlight and unplugging cables.



Ian Tabor
@mintynet

No fcuking point having a nice car these days, came out early to find the front bumper and arch trim pulled off and even worse the headlight wiring plug had been yanked out, if definitely wasn't an accident, kerb side and massive screwdriver mark. Breaks in the clips etc. C&#ts

6:03 PM · Apr 24, 2022

CAN Injection: keyless car theft by Canis Automotive Labs
https://kentindell.github.io/2023/04/03/can-injection/

Discovering the CAN bus_

EXERCISE

Setting up a virtual CAN bus_

On Linux, you can create a virtual CAN bus using the ip command.

The commands to create and activate such a bus are shown below:

💻 Command

```
$ sudo ip link add dev vcan0 type vcan
$ sudo ip link set up vcan0
```

📌 Tip

Buses are commonly named based on their type:

- **vcan#** for virtual CAN interfaces (e.g., vcan0)
- **can#** for standard physical CAN buses (e.g., can0)
- **slcan#** for devices using the serial line CAN protocol (e.g., USB-to-CAN adapters running in SLCAN mode)

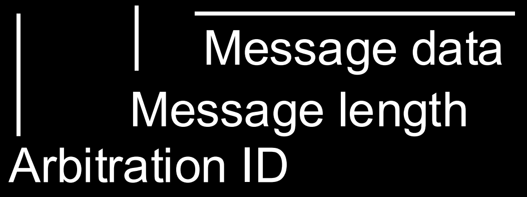
The can-utils library_

The **can-utils** package provides a collection of powerful tools for working with the CAN bus.

For example, the **candump** command can be used to capture and display all CAN messages on the vcan0 interface:

💻 **Command**

```
$ candump vcan0
 vcan0  123   [4]  DE AD BE EF
```



Message data

Message length

Arbitration ID

Use the **cansend** command to transmit a single, custom CAN message:

💻 **Command**

```
$ cansend vcan0 123#DEADBEEF
```

Use the **cangen** command to generate and send random CAN messages for testing purposes:

💻 **Command**

```
$ cangen vcan0
```

The virtual car_

We’ll use the **Instrument Cluster Simulator (ICSim)** by zombieCraig to simulate a car on our virtual CAN bus.

To download and compile ICSim, run the following commands:

💻 **Command**

```
$ sudo apt install libsdl2-dev libsdl2-image-dev meson can-utils
$ git clone https://github.com/zombieCraig/ICSim.git
$ cd ICSim
$ meson setup builddir && cd builddir
$ meson compile
```

The builddir folder should now contain two executables: **controls** and **icsim**.

The virtual car_

⚠ Troubleshooting: lib.o not linking

If you encounter the error:

/usr/bin/ld: lib.o: error adding symbols: file in wrong format

This is due to an architecture mismatch (for example, using an x86 object file on an ARM system). To fix it, follow these steps:

```
$ rm –r builddir
$ rm lib.o
$ gcc -c lib.c -o lib.o
$ meson setup builddir && cd builddir
```

The virtual car_

The **icsim** simulates the instrument cluster of our virtual car.

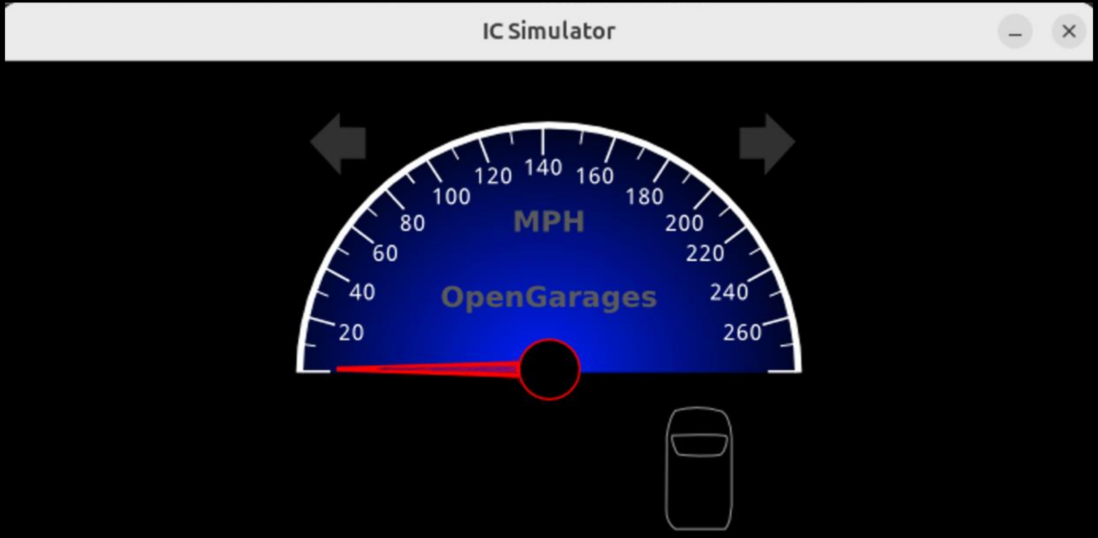
💻 **Command**

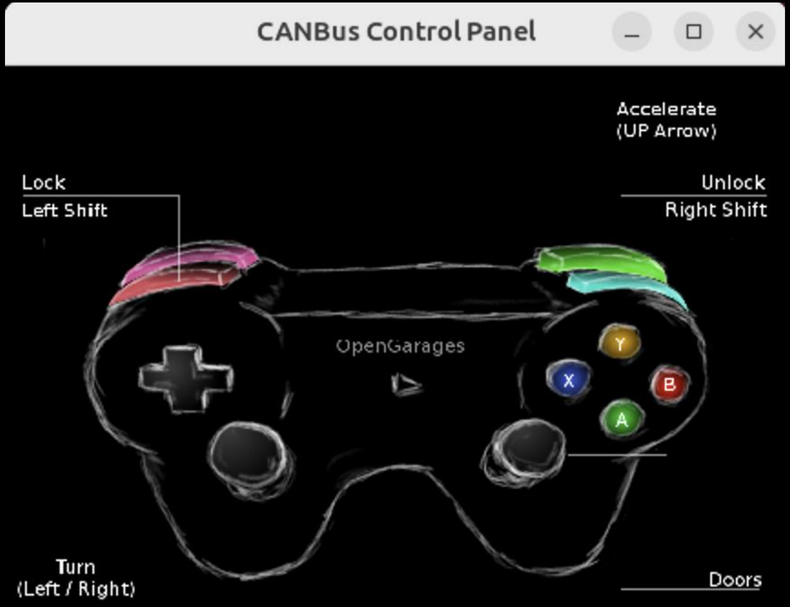
```
$ ./icsim vcan0
```

The **controls** is used to operate (drive) our virtual car.

💻 **Command**

```
$ ./controls vcan0
```





Find specific CAN message_

🎯 Goals

- Use the **controls** executable to activate the turn signals. Then, run **candump** to monitor messages on the **vcan0** bus. Take a few minutes to identify the **arbitration ID** associated with the turn signal messages.

Answer:

📌 Tip

Use the following **candump** command to filter on a single arbitration ID:

```
$ candump -x -e -a vcan0,<Arbitration ID>:7FF
```

ID e.g. 123

Filter mask (only the specified ID, see Wikipedia)

Enable ASCII output

Dump CAN error frames in human-readable format

Print extra message information, e.g. RX/TX

Using cansniffer_

You might have noticed that it’s quite difficult to find specific CAN messages using **candump** alone. Fortunately, other tools like **cansniffer** exist to make this process much easier and more efficient.

💻 Command

```
$ cansniffer vcan0 -c
```

|
Color any changing bytes in the message

In **cansniffer**, any recurring messages that remain unchanged are automatically hidden after a few seconds. When a message does change, its data is updated on the same line, making it much easier to track active signals.

📌 Tip

Make sure your terminal window is large enough to display all message lines clearly.

Using cansniffer_

🎯 Goals

- Use the **controls** executable to activate the turn signals. Then, run **cansniffer** to monitor messages on the **vcan0** bus. Take a few minutes to identify the **arbitration ID** associated with the turn signal messages.

  Answer:

- Now, identify the message value used to activate the left turn signal.

  Answer:

- Then, repeat the process to find the value for the right turn signal.

  Answer:

📌 Tip

Use the **filtering** options in **cansniffer** to hide or display specific messages as needed.

Send CAN messages_

🎯 **Goals**

- Now, use **cansend** to transmit a turn signal message to the virtual instrument cluster. Write down the exact command you used.

  Answer:

- Use **cansend** to activate the hazard lights (both turn signals flashing together) on the virtual instrument cluster.

  Answer:

- Write a small shell script to make the hazard lights blink on and off repeatedly, simulating a real blinking sequence.

  Answer:

Send periodic CAN messages_

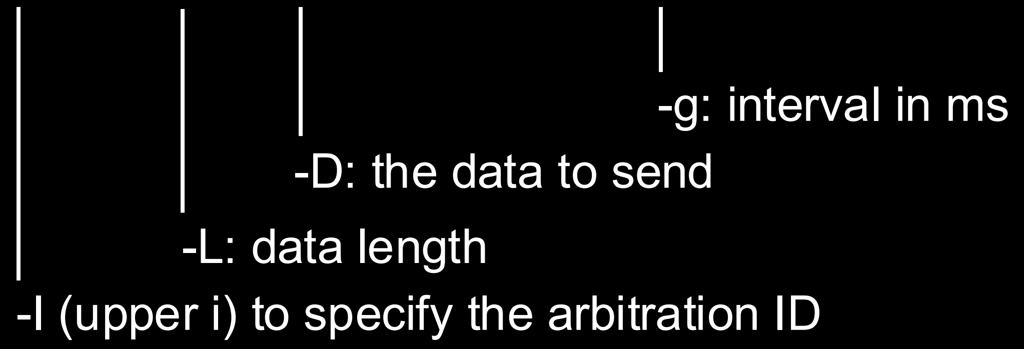
You can use **cangen** to generate and send periodic CAN messages automatically.

This is useful for testing how an ECU or instrument cluster handles continuous traffic, stress-testing the bus, or simulating real vehicle activity.

cangen allows you to customize options such as the message ID range, data length, and interval (gap) between messages.

💻 Command

```
$ cangen vcan0 -I 123 –L 8 –D DEADBEEF -g 100
```



-g: interval in ms

-D: the data to send

-L: data length

-I (upper i) to specify the arbitration ID

Send CAN messages_

🎯 Goals

- Now, use **cangen** to make the hazard lights blink on and off repeatedly, simulating a real blinking sequence.

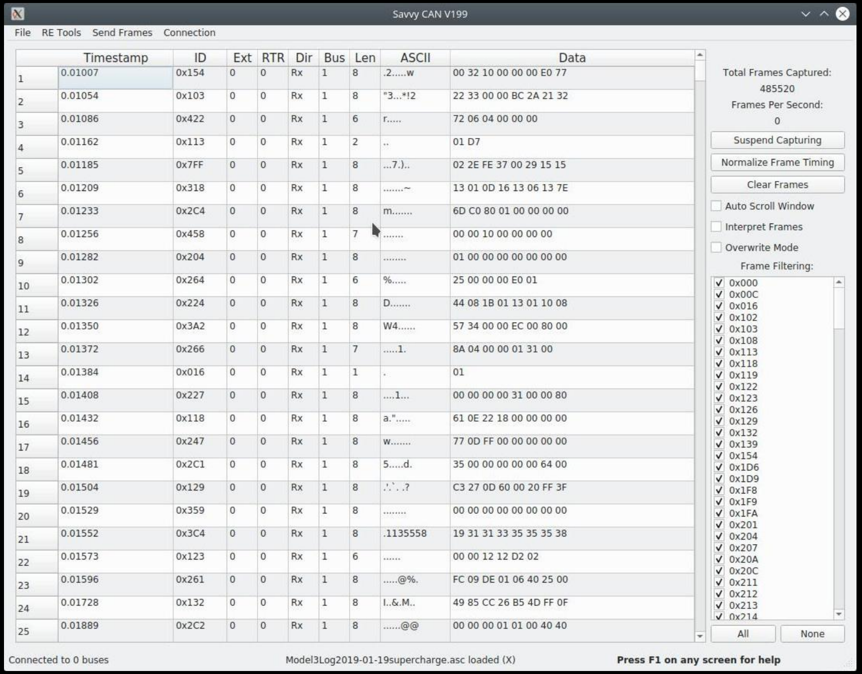
Answer:

📌 Tip

This is particularly helpful for sending wake-up frames to keep the CAN bus and its connected ECUs active, preventing them from entering sleep mode.

Using a GUI_

While we often use command-line tools like candump, cansend, and cansniffer, there are also several graphical tools that make it easier to monitor, analyze, and interact with CAN networks.



	Timestamp	ID	Ext	RTR	Dir	Bus	Len	ASCII	Data
1	0.01007	0x154	0	0	Rx	1	8	.2.....w	00 32 10 00 00 00 E0 77
2	0.01054	0x103	0	0	Rx	1	8	"3...*!2	22 33 00 00 BC 2A 21 32
3	0.01086	0x422	0	0	Rx	1	6	r.....	72 06 04 00 00 00
4	0.01162	0x113	0	0	Rx	1	2	..	01 D7
5	0.01185	0x7FF	0	0	Rx	1	8	...7.)..	02 2E FE 37 00 29 15 15
6	0.01209	0x318	0	0	Rx	1	8	.......~	13 01 0D 16 13 06 13 7E
7	0.01233	0x2C4	0	0	Rx	1	8	m.......	6D C0 80 01 00 00 00 00
8	0.01256	0x458	0	0	Rx	1	7	....	00 00 10 00 00 00 00
9	0.01282	0x204	0	0	Rx	1	8	....	01 00 00 00 00 00 00 00
10	0.01302	0x264	0	0	Rx	1	6	%.....	25 00 00 00 E0 01
11	0.01326	0x224	0	0	Rx	1	8	D.......	44 08 1B 01 13 01 10 08
12	0.01350	0x3A2	0	0	Rx	1	8	W4......	57 34 00 00 EC 00 80 00
13	0.01372	0x266	0	0	Rx	1	7	.....1.	8A 04 00 00 01 31 00
14	0.01384	0x016	0	0	Rx	1	1	.	01
15	0.01408	0x227	0	0	Rx	1	8	....1...	00 00 00 00 31 00 00 80
16	0.01432	0x118	0	0	Rx	1	8	a.".....	61 0E 22 18 00 00 00 00
17	0.01456	0x247	0	0	Rx	1	8	w.......	77 0D FF 00 00 00 00 00
18	0.01481	0x2C1	0	0	Rx	1	8	5.....d.	35 00 00 00 00 00 64 00
19	0.01504	0x129	0	0	Rx	1	8	.'.`. .?	C3 27 0D 60 00 20 FF 3F
20	0.01529	0x359	0	0	Rx	1	8	....	00 00 00 00 00 00 00 00
21	0.01552	0x3C4	0	0	Rx	1	8	.1135558	19 31 31 33 35 35 35 38
22	0.01573	0x123	0	0	Rx	1	6	......	00 00 12 12 D2 02
23	0.01596	0x261	0	0	Rx	1	8	.....@%.	FC 09 DE 01 06 40 25 00
24	0.01728	0x132	0	0	Rx	1	8	I..&.M..	49 85 CC 26 B5 4D FF 0F
25	0.01889	0x2C2	0	0	Rx	1	8	......@@	00 00 00 01 01 00 40 40

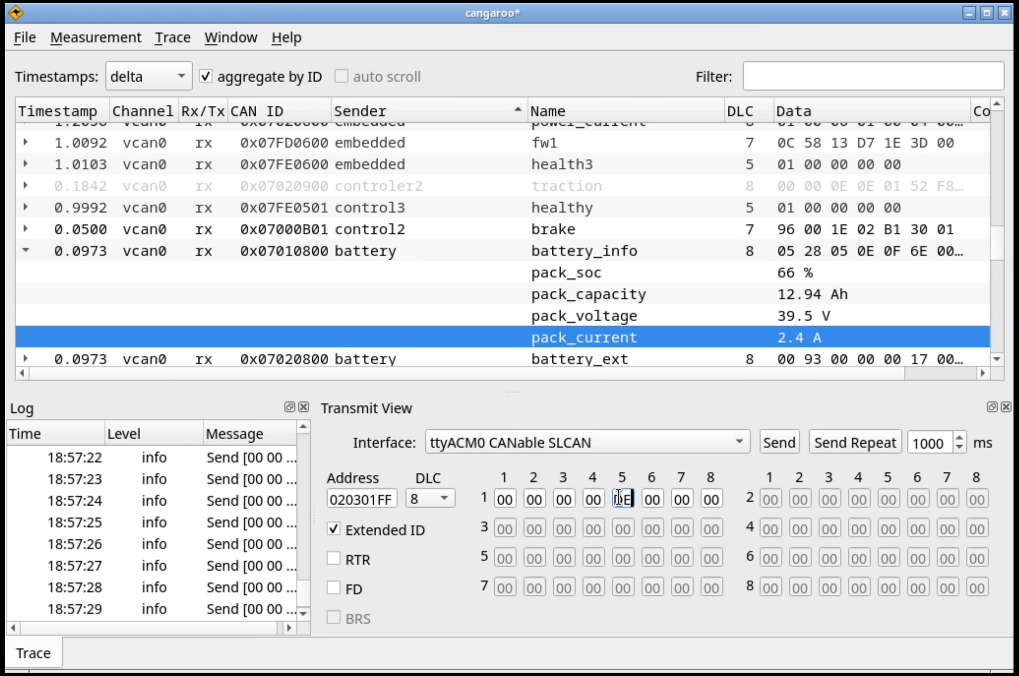
Total Frames Captured: 485520
Frames Per Second: 0

□ Auto Scroll Window
□ Interpret Frames
□ Overwrite Mode

Frame Filtering:
☑ 0x000
☑ 0x00C
☑ 0x016
☑ 0x102
☑ 0x103
☑ 0x108
☑ 0x113
☑ 0x118
☑ 0x119
☑ 0x122
☑ 0x123
☑ 0x126
☑ 0x129
☑ 0x132
☑ 0x139
☑ 0x154
☑ 0x1D6
☑ 0x1D9
☑ 0x1F8
☑ 0x1F9
☑ 0x1FA
☑ 0x201
☑ 0x204
☑ 0x207
☑ 0x20A
☑ 0x20C
☑ 0x211
☑ 0x212
☑ 0x213
☑ 0x214

Connected to 0 buses
Model3Log2019-01-19supercharge.asc loaded (X)
Press F1 on any screen for help

SavvyCan (Linux, Windows, and OSX)



Timestamps: delta ☑ aggregate by ID □ auto scroll Filter:

Timestamp	Channel	Rx/Tx	CAN ID	Sender	Name	DLC	Data
1.0092	vcan0	rx	0x07FD0600	embedded	fw1	7	0C 58 13 D7 1E 3D 00
1.0103	vcan0	rx	0x07FE0600	embedded	health3	5	01 00 00 00 00
0.1842	vcan0	rx	0x07020900	controler2	traction	8	00 00 0E 0E 01 52 F8…
0.9992	vcan0	rx	0x07FE0501	control3	healthy	5	01 00 00 00 00
0.0500	vcan0	rx	0x07000B01	control2	brake	7	96 00 1E 02 B1 30 01
0.0973	vcan0	rx	0x07010800	battery	battery_info	8	05 28 05 0E 0F 6E 00…
					pack_soc		66 %
					pack_capacity		12.94 Ah
					pack_voltage		39.5 V
					pack_current		2.4 A
0.0973	vcan0	rx	0x07020800	battery	battery_ext	8	00 93 00 00 00 17 00…

Log

Time	Level	Message
18:57:22	info	Send [00 00 ...
18:57:23	info	Send [00 00 ...
18:57:24	info	Send [00 00 ...
18:57:25	info	Send [00 00 ...
18:57:26	info	Send [00 00 ...
18:57:27	info	Send [00 00 ...
18:57:28	info	Send [00 00 ...
18:57:29	info	Send [00 00 ...

Transmit View

Interface: ttyACM0 CANable SLCAN 1000 ms

Address 020301FF DLC 8
☑ Extended ID
□ RTR
□ FD
□ BRS

	1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	00	00	00	00	0E	00	00	00	2	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	4	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	6	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	8	00	00	00	00	00	00	00	00

Cangaroo (Linux, and Windows)

Fuzzing_

You can use **cangen** as a simple ‘poor man's fuzzer’.

Instead of specifying exact data values, you can use the **-i** (incremental) or **-r** (random) options to automatically generate changing payloads.

💻 Command

```
$ cangen -I 123 –L 8 –D i -g 100 vcan0
```

🎯 Goals

- Use **cangen** to send random signals to arbitration ID **0x19B**. Observe the virtual instrument cluster and describe what happens. What changes do you notice, and why do you think they occur?

  Answer:

Fuzzing_

⚠ **Warning**

Be extremely careful when fuzzing a real vehicle!

- Triggering critical safety functions unexpectedly (e.g., airbags, ABS).
- Putting the vehicle into a fault or "limp" mode.
- Bricking ECUs, especially if they receive a so-called "crash frame" or unexpected diagnostic command, which can require dealer-level tools to recover.

Always perform fuzzing only in a **controlled lab environment**, on a test bench setup, or on a fully isolated vehicle system that is safe to reset.

Record and replay signals (optional)_

Waiting for the other groups to finish? Try these extra challenges in the meantime.

🎯 **Goals**

- Use **candump** with the **-l** option to save the captured CAN data to a file. pecify the exact command you ran, and indicate where the log file is stored.

  Answer:

- Use **canplayer** to replay the recorded CAN signals in smaller parts. This can help you reverse-engineer and analyze real CAN logs more effectively. Specify the exact command you used.

  Answer:

Manipulate an instrument cluster_

EXERCISE

Manipulate an instrument cluster_

👥 **Small Groups:**
Split into small groups. There is a limited number of instrument clusters, so make sure to give everyone the opportunity to take the primary role in each of the tasks.

🛠 **Scenario:**
Imagine you are an attacker who has gained access to a car’s **CAN bus** (for example, through an exposed diagnostic port or vulnerable ECU). Your goal is to send crafted CAN messages to manipulate the instrument cluster behavior.

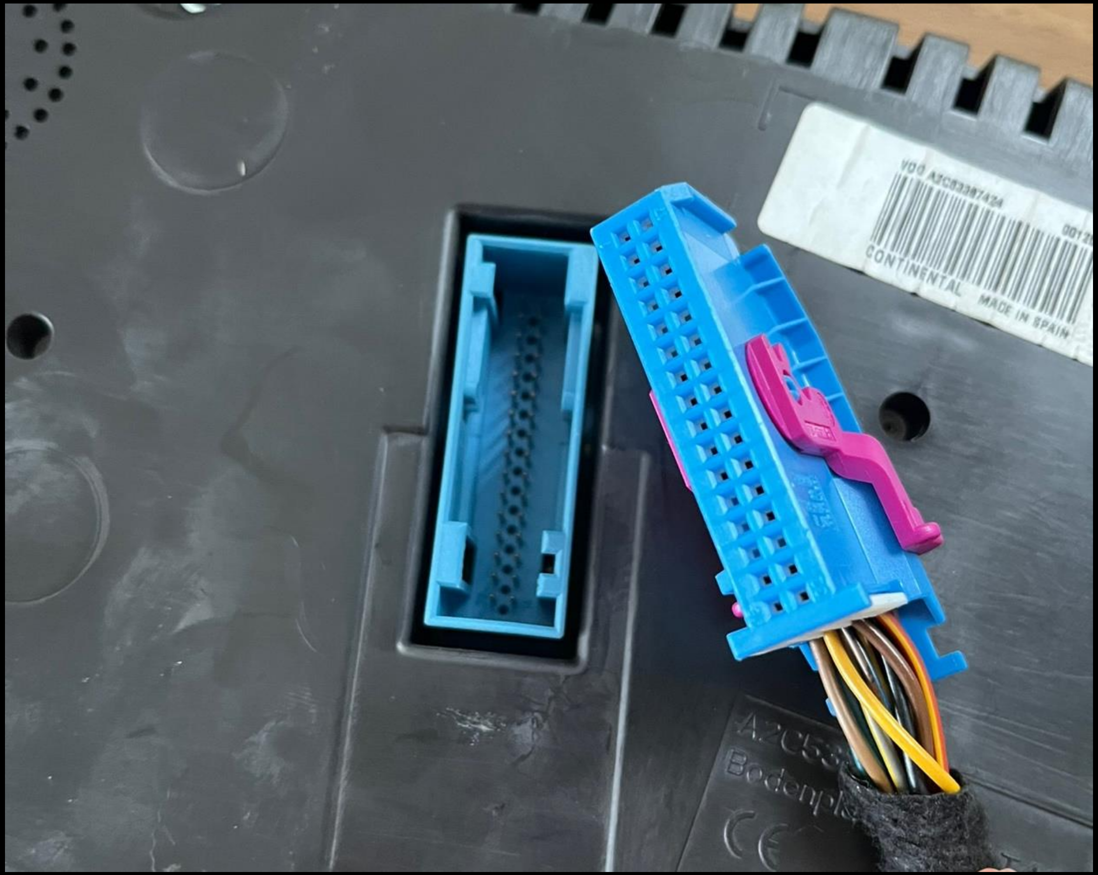
🎯 **Objectives:**
Understand how the instrument cluster receives and displays information from the CAN bus.
Practice crafting and injecting CAN frames.

Instrument cluster_

For this workshop, we will be using instrument clusters from various Volkswagen Group cars produced around 2010. These include:

- Škoda Fabia
- SEAT Ibiza
- SEAT Leon

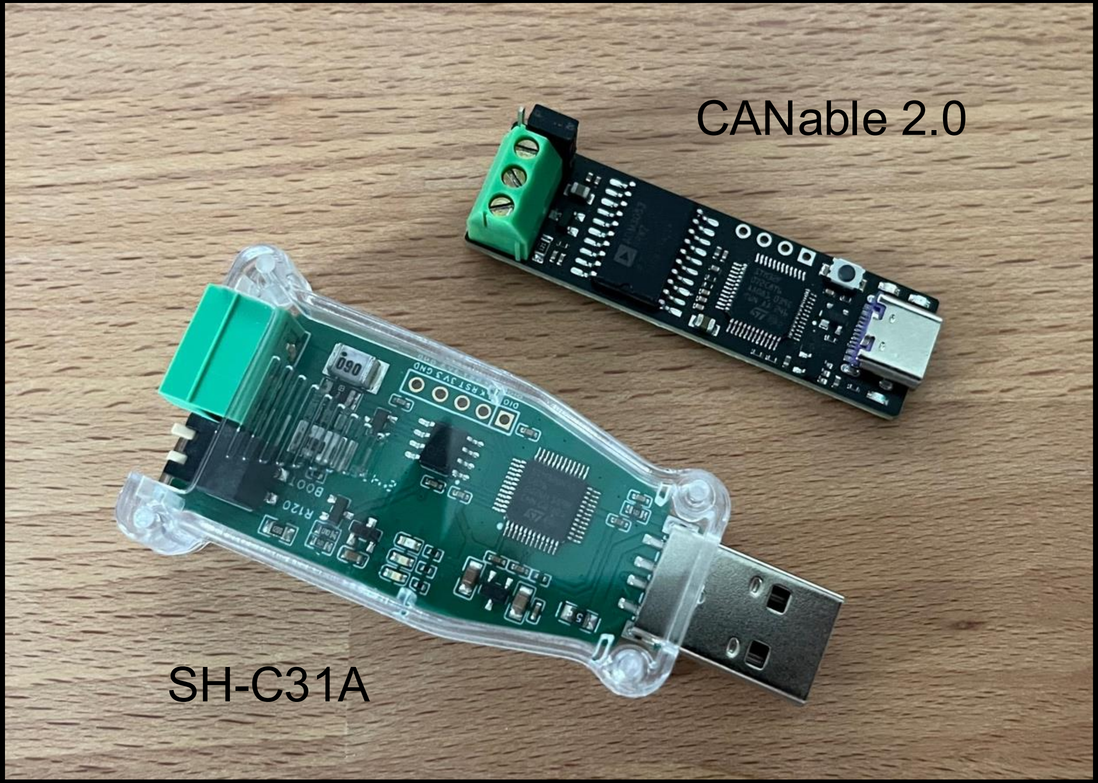
All of these clusters use a **single blue 32 pin connector**, making them easy to interface with for testing and security research.



USB to CAN adapter_

During this workshop, we'll use the **CANable 2.0** and **SH-C31A** adapters to connect to the CAN bus.

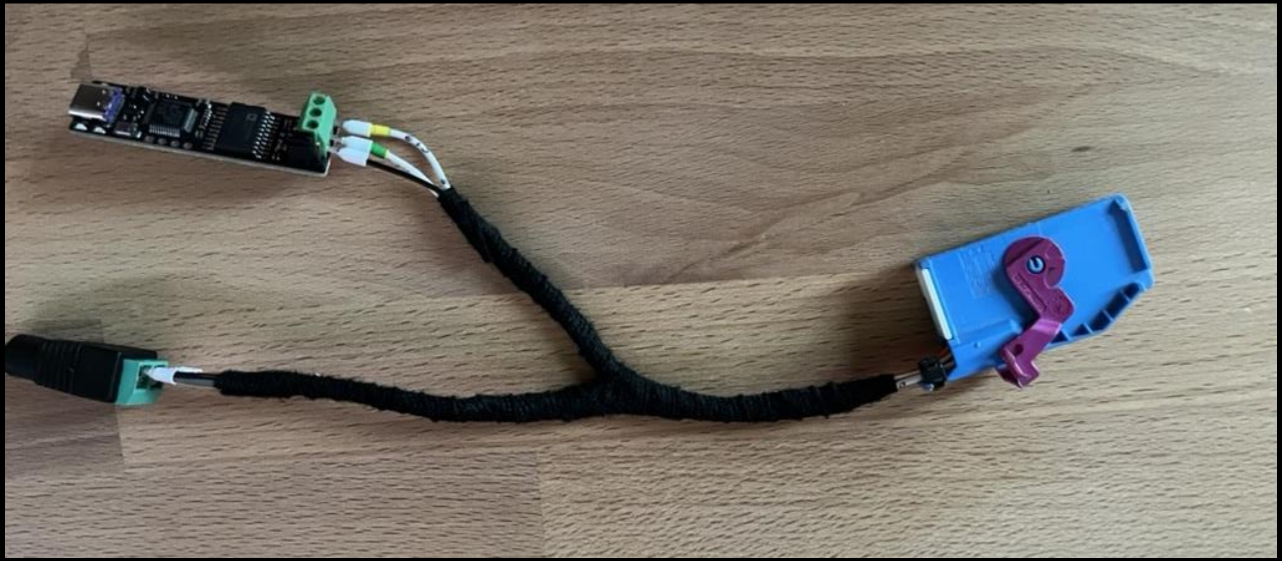
- Compact and affordable USB-to-CAN adapters
- Open-source, widely used in automotive security research
- Support **CANtact**, **SocketCAN**, and **candlelight** firmware
- Compatible with Linux, Windows, and macOS
- Support high-speed CAN communication (up to 1 Mbps)



Connector_

It’s relatively easy to build a **custom connector** or **harness** to interface with an instrument cluster for testing and security research.

During this workshop, we will keep it simple and connect directly to the cluster using jump wires.



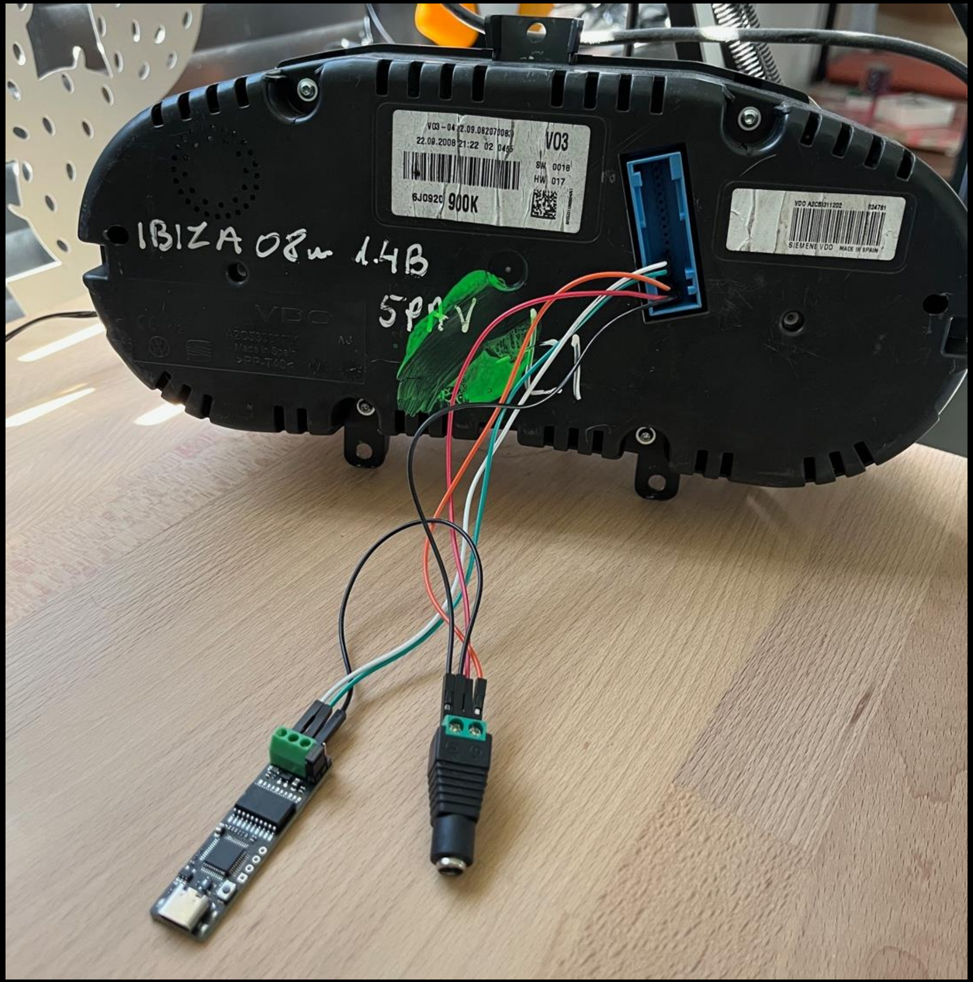
Connect to the cluster_

To connect to the instrument cluster, you'll need to:

1. Connect the USB-to-CAN adapter to the cluster's CAN bus.
2. Connect the USB-to-CAN adapter to your laptop.
3. Ensure power is correctly supplied to the cluster.

⚠ **Warning**

Make sure to double-check all your connections before plugging in the computer and power supply. Incorrect wiring can damage the cluster.



Connect to the cluster_

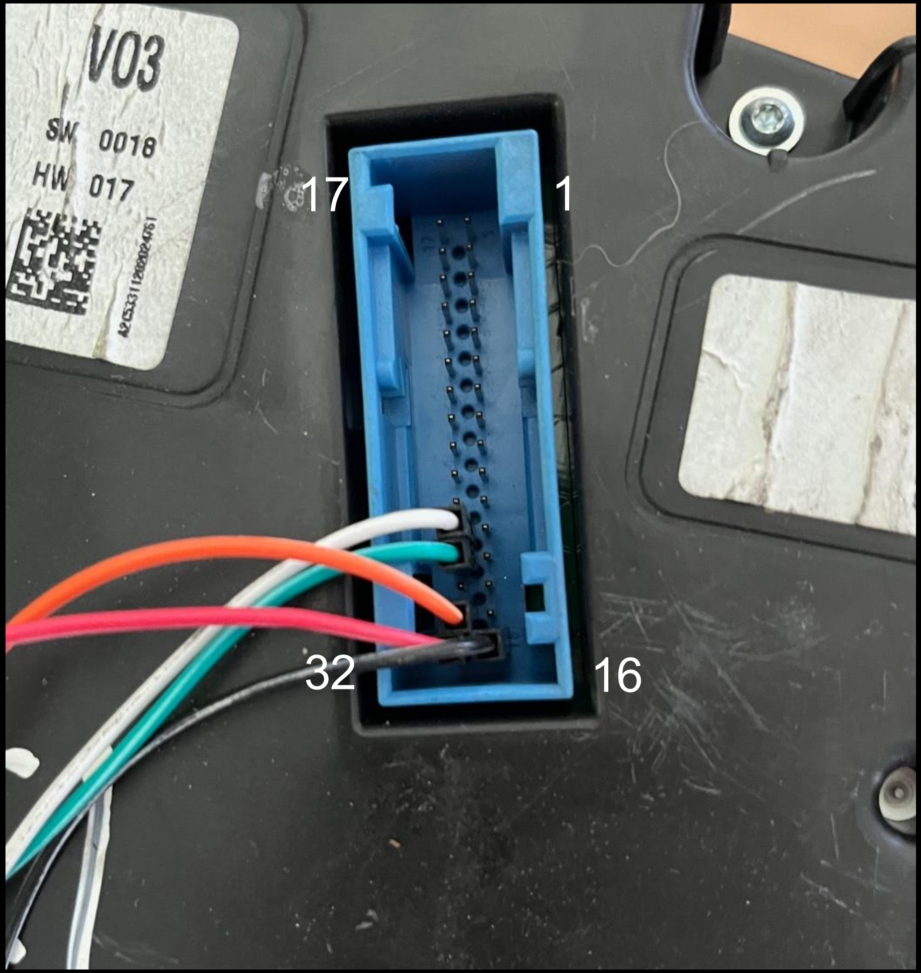
Pin layout

Pin 16: Ground
Pin 28: CAN high
Pin 29: CAN low
Pin 31: Switched ignition (12v)
Pin 32: Constant power (12v)

Source: https://mhhauto.com/Thread-need-pinout-cluster-MK6-golf

Tip

The pin numbers are labeled directly on the connector. Using a light source can help you clearly see and identify the digits.



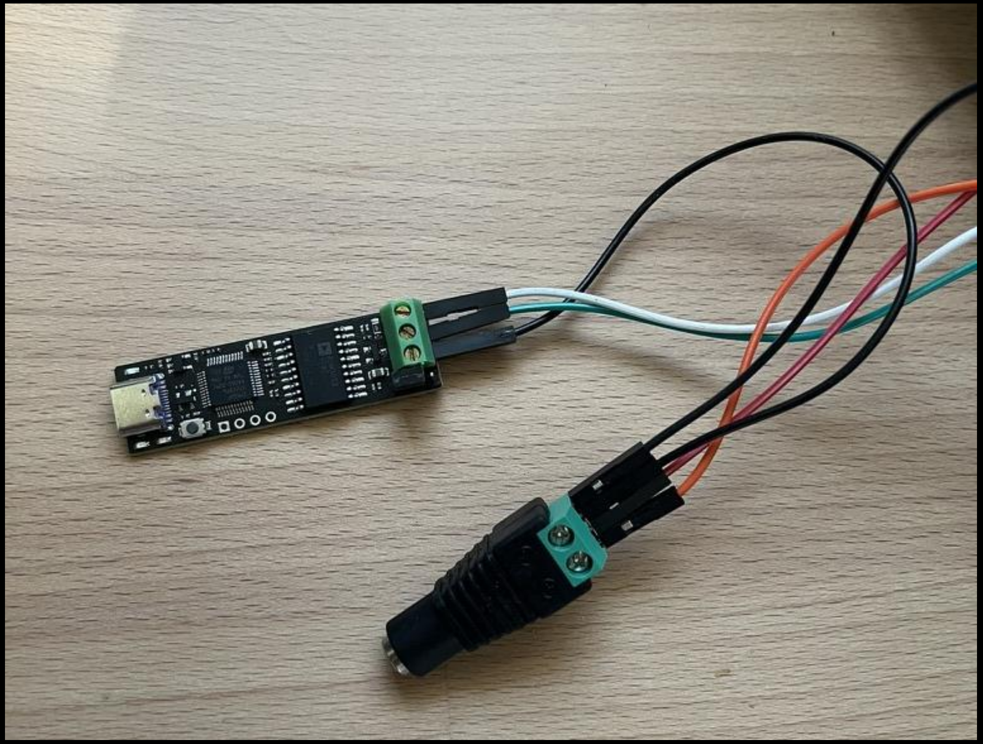
Connect to the cluster_

⚠ **Warning**

Make sure the **ground connection** of the USB to CAN adapter is connected to the ground connection of the power connector.

📌 **Tip**

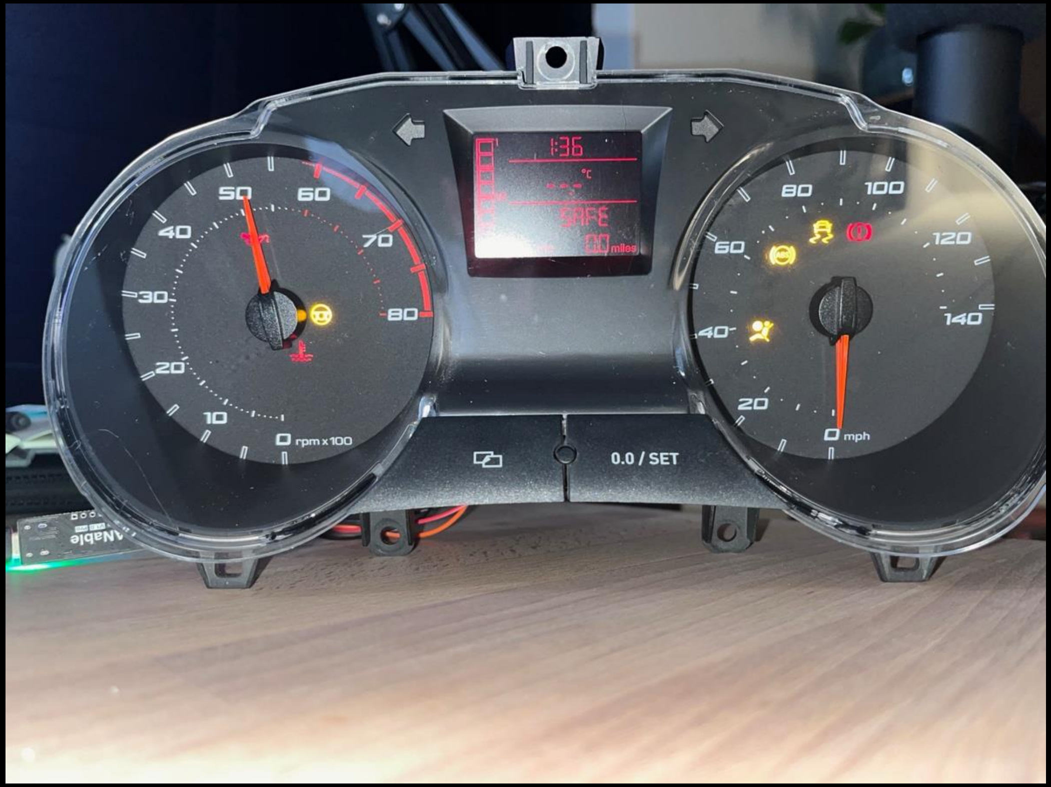
The power connector should have **two ground connections** and **two power connections**. This requires **multiple wires in a single socket**.



Connect to the cluster_

If the connection is correct the instrument cluster will show multiple warning lights and starts beeping.

On the USB-to-CAN adapter you should see multiple LEDs blinking to indicate the instrument cluster is sending data.



💭 Question

Any ideas why the cluster might display multiple warnings and start beeping?

Connect to the cluster_

Configure the interface to interact with the USB-to-CAN adapter on can0:

💻 Command

```
$ ip link set can0 up type can bitrate 500000
```

🎯 Goals

- Now, use **candump** to listen for CAN messages from the instrument cluster. Make sure to record the exact command line you entered.

  Answer:

- Use **cansniffer** with color highlighting enabled to capture and analyze CAN messages from the instrument cluster. Record the exact command line you entered.

  Answer:

Spoof immobilizer_

An **immobilizer** is an electronic security device in cars designed to prevent the engine from starting unless the correct, authorized key or key fob is present.

On some vehicles, it is theoretically possible to **bypass or fake the immobilizer signal** by sending a crafted CAN message with arbitration ID **0x3D0** to the instrument cluster.

💻 Command

```
$ cangen -I 3D0 -L 8 -D 0000000000 -g 100 can0
```

🎯 Goals

- Use the above command and adjust the timing interval. At what interval (in milliseconds) must you repeatedly send the immobilizer message to keep the warning light off?

  Answer:

Research_

📌 **Tip**

There are some differences between instrument clusters. The SEAT Leon uses different arbitration IDs and data values to achieve the same signals as the SEAT Ibiza and Skoda Fabia.

🎯 **Goals**

- Research online to find information about CAN messages used in the **PQ24**, **PQ34**, and **PQ35** platforms. Which platform does your instrument cluster belong to?

  Answer:

- What are the **arbitration IDs** of the CAN messages used to control the turn signals and display the RPM speed?

  Answer:

Send turn and RPM signals_

Hopefully, you've found some information about the correct arbitration IDs to use. In practice, you would typically analyze a **CAN log from a real car** to reverse-engineer these IDs and their corresponding data values. If that information isn't available, you can always fall back on **fuzzing the instrument cluster** to discover the correct IDs and signals.

🎯 Goals

- Identify the correct arbitration ID and data value needed to activate the **turn signals**. Then, use **cangen** to send this command to the instrument cluster. Which exact command did you use?

  Answer:

- Repeat the process for the **RPM signal**. Can you make the instrument cluster display the redline? What exact command did you use?

  Answer:

Connect ICSim to cluster_

Let’s connect the ICSim controller to the actual instrument cluster.

🎯 **Goals**

- Connect the **ICSim controller** to the same CAN bus as the instrument cluster. Are you able to successfully send turn signal commands? If not, why do you think they might not work?

  Answer:

- Begin by carefully analyzing (reverse-engineering) the **ICSim controller source code**. Then, update or adjust the required sections to ensure the code functions as intended.

  Answer:

Send other signals (optional)_

Waiting for the other groups to finish? Try these extra challenges in the meantime.

🎯 **Goals**

- Can you turn off the other warning lights? Which exact arbitration IDs and data values do you need to send to achieve this? Write them down.

  Answer:

- Are you also able to send the **car speed signal**? You should be able to find additional information about the required arbitration IDs and data values online.

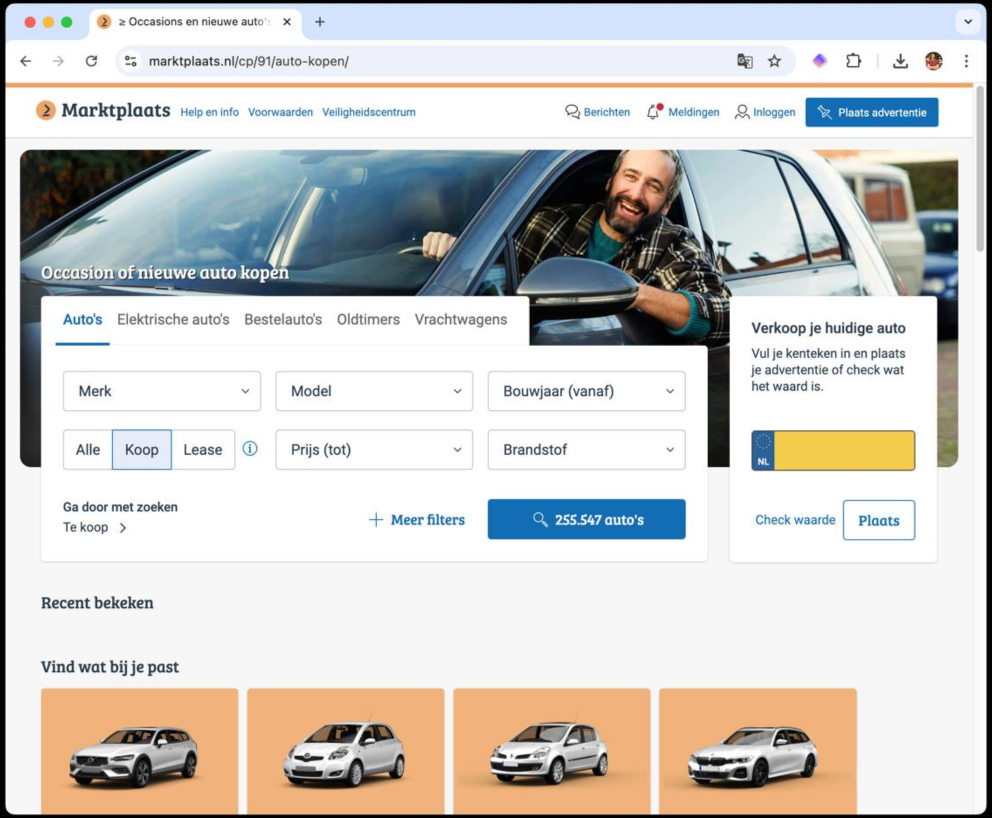
  Answer:

- Implement your solution using the python-can library to both send messages to the instrument cluster and receive feedback from it.

  Answer:

Regroup_

Car in a box_



≥ Occasions en nieuwe auto's

marktplaats.nl/cp/91/auto-kopen/

Marktplaats Help en info Voorwaarden Veiligheidscentrum

Berichten Meldingen Inloggen Plaats advertentie

Occasion of nieuwe auto kopen

Auto's Elektrische auto's Bestelauto's Oldtimers Vrachtwagens

Merk | Model | Bouwjaar (vanaf)

Alle | Koop | Lease | Prijs (tot) | Brandstof

Ga door met zoeken

Te koop

Meer filters

255.547 auto's

Verkoop je huidige auto

Vul je kenteken in en plaats je advertentie of check wat het waard is.

NL

Check waarde Plaats

Recent bekeken

Vind wat bij je past

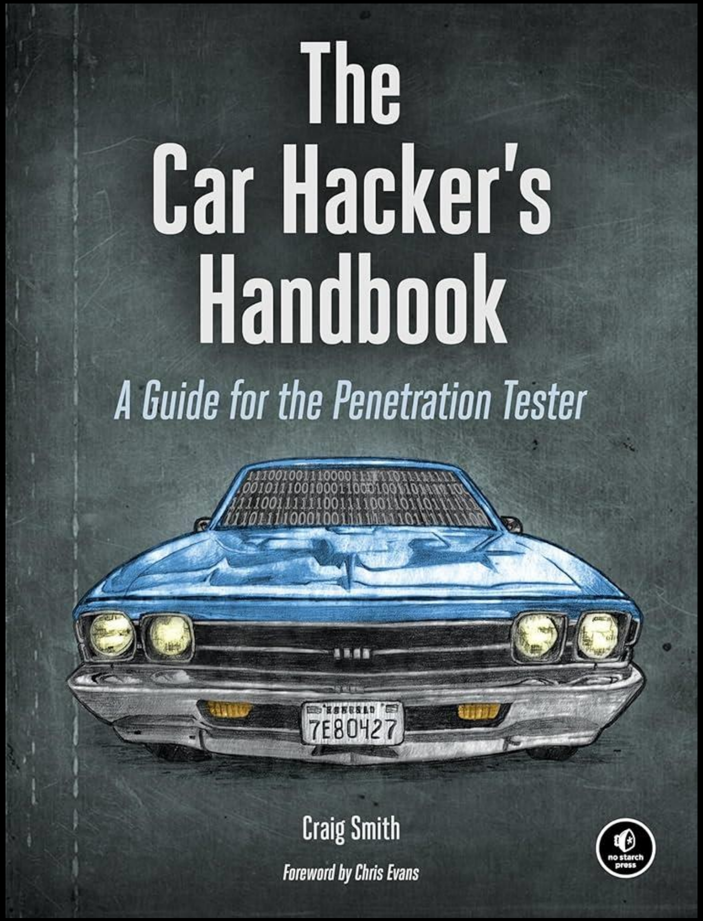
Tesla in a box_

Image source
https://nl.wikipedia.org/wiki/Tesla_Model_3

Resources_

Some additional resources:

- reddit.com/r/CarHacking
- Book: The Car Hacker's Handbook
- Whitepaper: Car Hacking: For Poories
- Defcon: Car Hacking Village
- …



BSides Groningen & Amsterdam_

Thanks_

Hacking Automotive Systems
Workshop by Warpnet

Do you have any questions?
roald.nefs@warpnet.nl
linkedin.com/in/roaldnefs/
warpnet.nl