SYNACKTIV

0-Click RCE on the Tesla Infotainment Through Cellular Network

OffensiveCon

May 11th 2024

Who are we

David BERARD

Security expert

@p0ly

Vincent DEHORS

Security expert

@vdehors

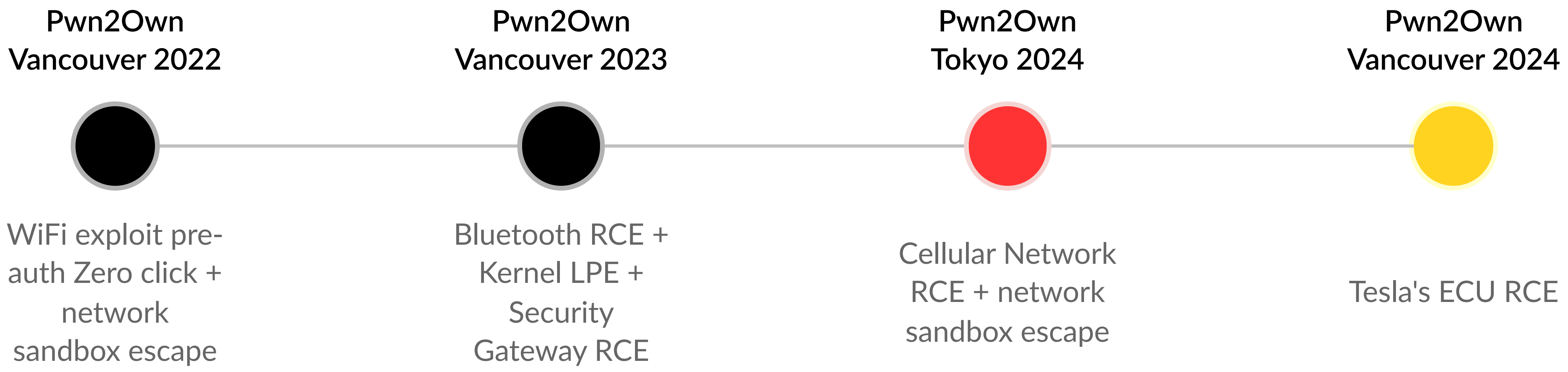
Synacktiv

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

Reverse Engineering team

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

Previous work



Pwn2Own Vancouver 2022

WiFi exploit pre-auth Zero click + network sandbox escape

Pwn2Own Vancouver 2023

Bluetooth RCE + Kernel LPE + Security Gateway RCE

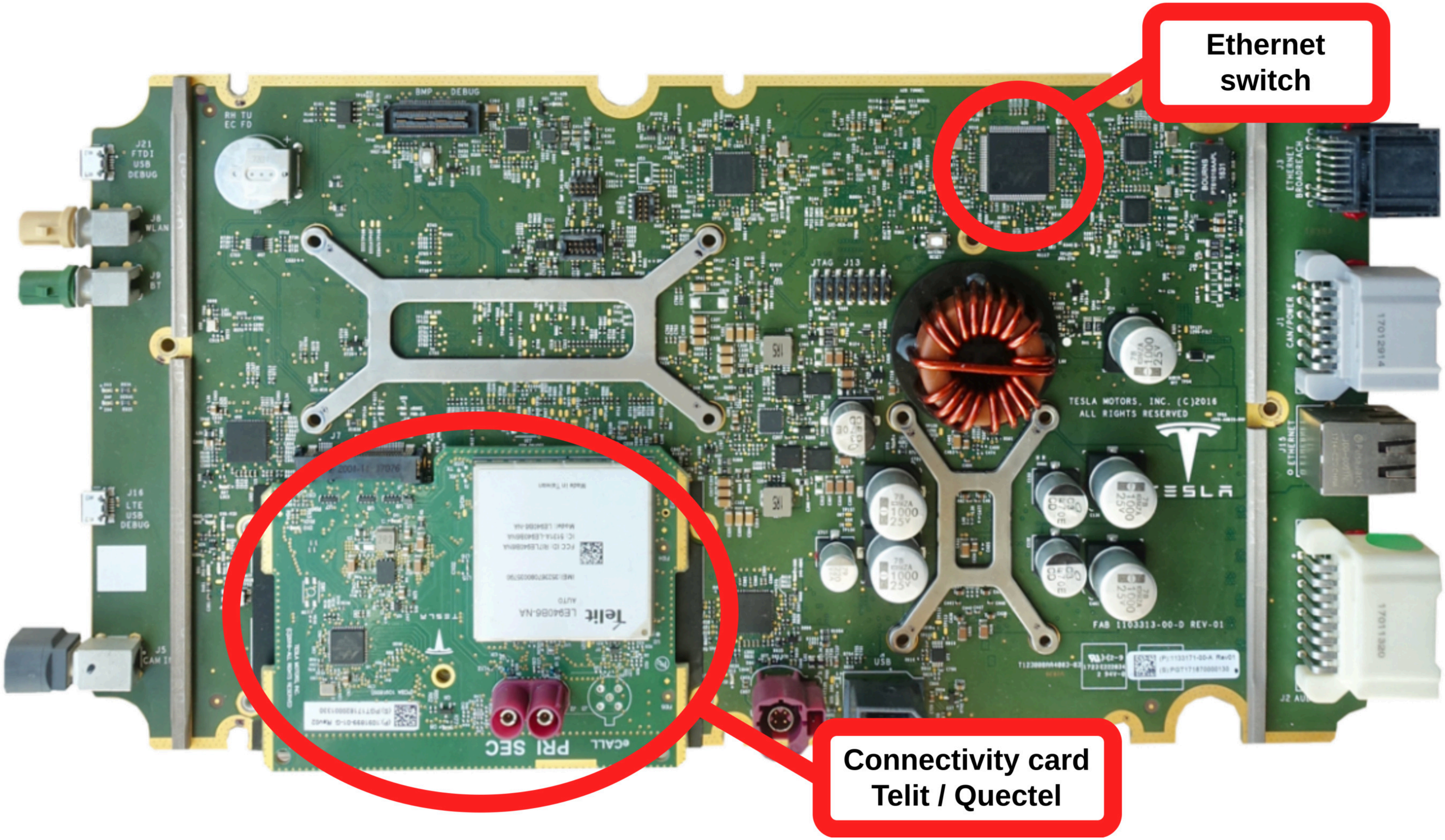
Pwn2Own Tokyo 2024

Cellular Network RCE + network sandbox escape

Pwn2Own Vancouver 2024

Tesla's ECU RCE

Hardware architecture

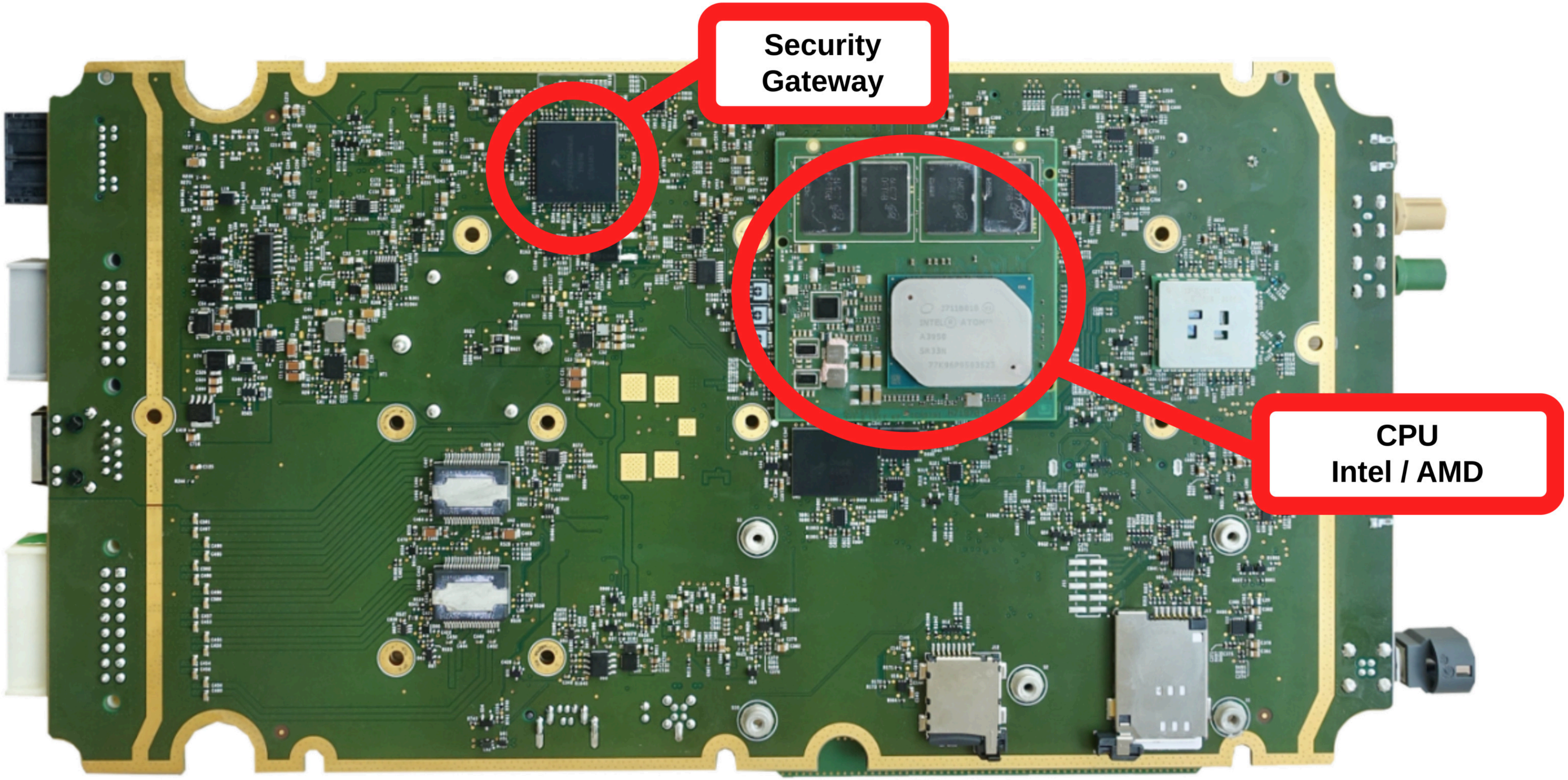


Ethernet switch

Connectivity card
Telit / Quectel

Hardware architecture

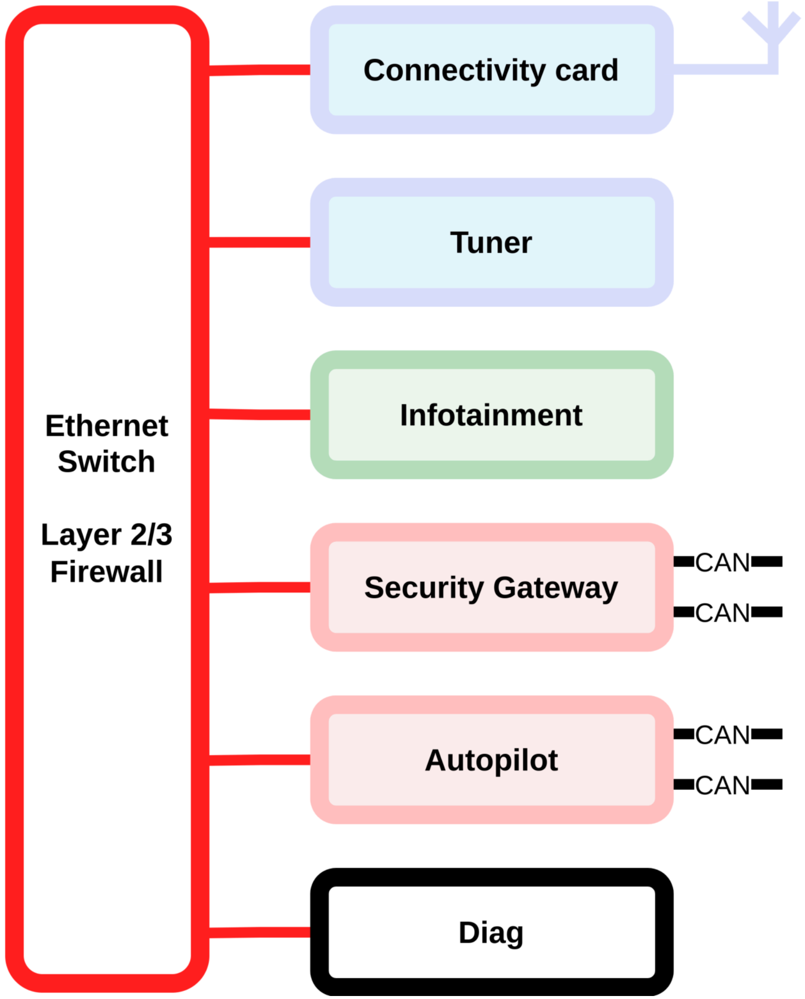
SYNACKTIV



Security Gateway

CPU
Intel / AMD

Network architecture



LTE connectivity

- Provided to Infotainment & Autopilot through Ethernet network
- Setup by Ofono software on the Infotainment through AT commands over TCP
- VLAN on the Ethernet network for data channel

Firewall

- Filtered at various level:
  - switch
  - infotainment
  - connectivity card

Connectivity card RCE

Connectivity card: System

Studied version: Quectel

- Qualcomm Baseband
- ARM Application processor
  - Linux System
  - Tasty mix of Yocto, Android, and Ubuntu distribution
- Newer versions also provide WLAN and Bluetooth connectivity to the infotainment

Free root shell on the UART test points

```
sa415m login: root
~ # id
uid=0(root) gid=0(root) groups=0(root),3003(inet) context=u:r:shell:s0
```

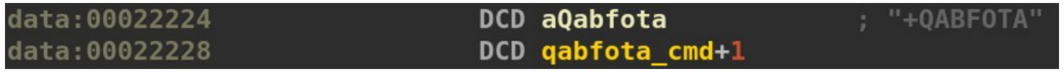
- Very useful for debugging
- `/etc` partition is R/W → add your ssh key and profit

Connectivity card: Network

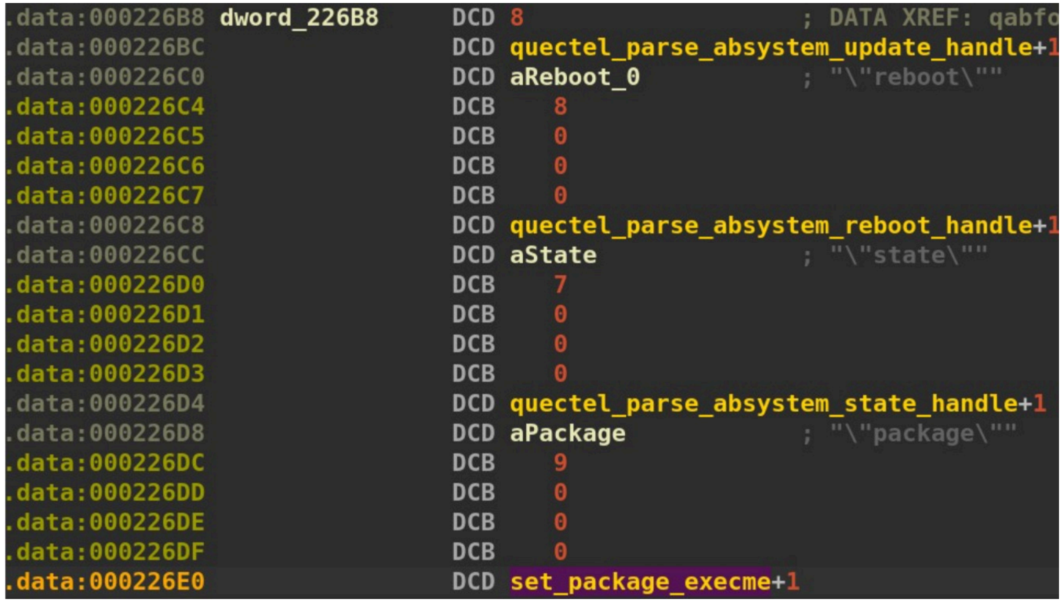
- **rmnet0** → LTE/GPRS interface
  - **IP address dynamically allocated by the cellular network**
- **eth0** → Interface connected to the internal Ethernet switch
  - **bridge0** `192.168.90.60`
  - **bridge20** `192.168.20.1` **VLAN data**
- Trafic is NAT'ed from **bridge20** to **rmnet0**
- AT commands
  - **Ofono** (on the infotainment) sends AT commands to the card over TCP
  - **ql_atfwd** process is responsible of handling AT commands
  - Listen on **192.168.90.60:50950**

Connectivity card: Command injection

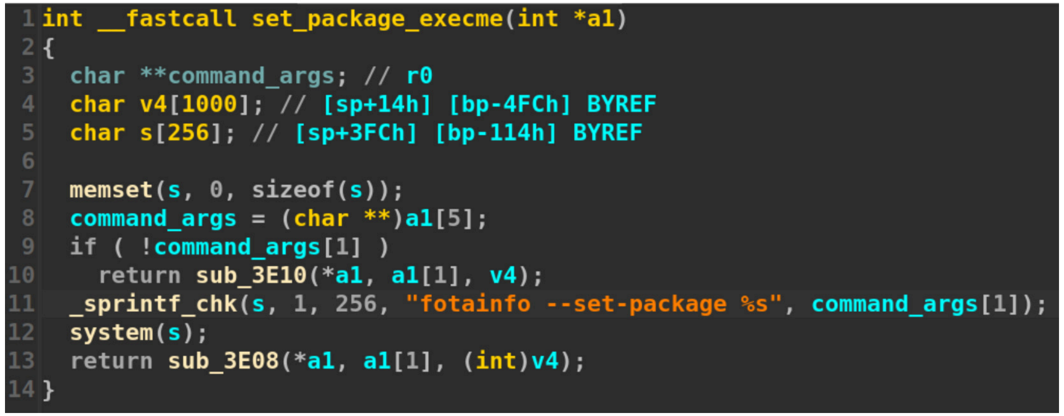
- **ql_atfwd** vulnerability in one of the AT command handler



```
data:00022224                 DCD aQabfota            ; "+QABFOTA"
data:00022228                 DCD qabfota_cmd+1
```



```
.data:000226B8 dword_226B8     DCD 8                   ; DATA XREF: qabfo
.data:000226BC                 DCD quectel_parse_absystem_update_handle+1
.data:000226C0                 DCD aReboot_0           ; "\"reboot\""
.data:000226C4                 DCB    8
.data:000226C5                 DCB    0
.data:000226C6                 DCB    0
.data:000226C7                 DCB    0
.data:000226C8                 DCD quectel_parse_absystem_reboot_handle+1
.data:000226CC                 DCD aState              ; "\"state\""
.data:000226D0                 DCB    7
.data:000226D1                 DCB    0
.data:000226D2                 DCB    0
.data:000226D3                 DCB    0
.data:000226D4                 DCD quectel_parse_absystem_state_handle+1
.data:000226D8                 DCD aPackage            ; "\"package\""
.data:000226DC                 DCB    9
.data:000226DD                 DCB    0
.data:000226DE                 DCB    0
.data:000226DF                 DCB    0
.data:000226E0                 DCD set_package_execme+1
```



```
int __fastcall set_package_execme(int *a1)
{
  char **command_args; // r0
  char v4[1000]; // [sp+14h] [bp-4FCh] BYREF
  char s[256]; // [sp+3FCh] [bp-114h] BYREF

  memset(s, 0, sizeof(s));
  command_args = (char **)a1[5];
  if ( !command_args[1] )
    return sub_3E10(*a1, a1[1], v4);
  _sprintf_chk(s, 1, 256, "fotainfo --set-package %s", command_args[1]);
  system(s);
  return sub_3E08(*a1, a1[1], (int)v4);
}
```

```
AT+QABFOTA="package","$(injected command)"
```

- Should be reachable only from the internal network as **ql_atfwd** listen only on **192.168.90.60:50950** !
- But …

Connectivity card: IP configuration

- **rmnet0** → LTE/GPRS interface
  - **IP address dynamically allocated by the cellular network**
  - **Address advertised from the network is not verified**
  - Local IP can be affected by the network
  - As **ql_atfwd** listen on **192.168.90.60:50950** it may be reached from the cellular network
- **But an Iptables rule prevents that :(**
  - Could be cool to have the firewall disabled no ?

Connectivity card: Firewall bypass

- By chance we observed that at times the firewall is not active after a reboot
- While booting `systemd` starts two processes that use the iptables lock
  - `firewall` that loads the defaults iptables rules
  - `QCMAP_ConnectionManager` process responsible for dynamically adding iptables rules
- If `firewall` can't take the lock, the default rules are not loaded, and exits properly

```
sa415m firewall[1005]: Another app is currently holding the xtables lock. Perhaps you want to use the -w
option?
```

- This situation occurs in about 25% of the connectivity card boots
- We have to find a way to have this lack of firewall situation remotely, on normal operation the connectivity card doesn't reboot

Connectivity card: Firewall bypass

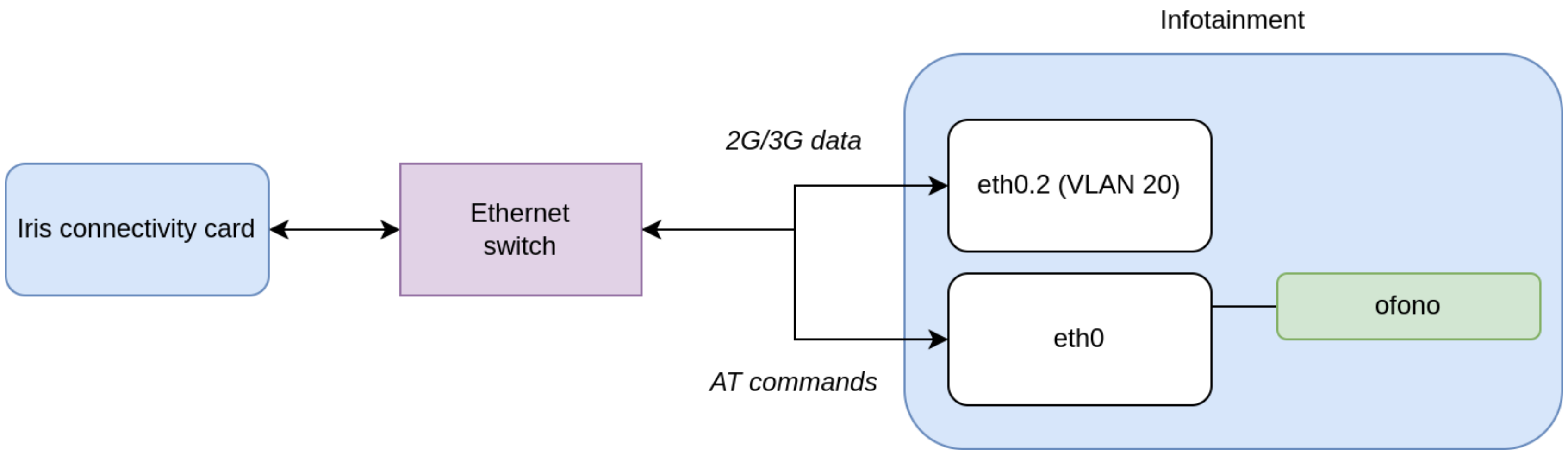
- A connectivity card reboot mechanism is implemented on the Infotainment
  - When the LTE connectivity is established, the Infotainment checks the Internet access
  - If the Internet check fails 3 times, the connectivity card is rebooted
  - Checks are based on HTTP requests, so the Cellular network can make this test fail
  - Reboots are limited to 4 reboots per 30 minutes, but are based on local time
  - The infotainment sends NTP requests, so the Cellular network can change the time to bypass this limit and make the board reboot more than 4 times

Connectivity card: exploitation sumup

1. The attacker PC runs the base station and affects the local IP to the client
2. If the firewall is detected to be active, the board is rebooted by making the connectivity check fail
3. If the firewall is inactive, the attacker PC replies to the connectivity check to keep the board UP
4. As AT commands can be sent from the cellular network when firewall is not active, the command injection vulnerability is used to execute arbitrary commands as root on the connectivity card
   - Firewall is disabled permanently (by writting to `/etc`)
   - An SSH key is added to connect remotely to the board through SSH
   - **Next stage**: exploit the Infotainment from the connectivity card

From Modem to Infotainment

Attack surface from the Modem



Network

- eth0.2 for the mobile network data
- eth0 for infotainment/modem communication
  - 192.168.90.60 : Modem
  - 192.168.90.100 : Infotainment

Iptables rules

- Network input is filtered using IP addresses (checked by the switch)

```
-A INPUT -s 192.168.90.60/32 -i eth0 ! -p icmp -j MODEM_INPUT
```

- Network output is filtered using process UID

```
-A OUTPUT -m owner --uid-owner 2000 -j OFONO
```

Ofono

- OpenSource, hosted in git.kernel.org
- Manage modem using AT commands
- Usual channel : Serial link or USB ACM
- SMS, GPRS, Location, SIM management, Voice calls, …
- Standard implementation + plugins for modems custom feature

Usage in Tesla car

- Two plugins added : **Tesla** (for Telit modem) and **Iris** (for Quectel modem)
- AT commands transmitted over **TCP** (port 50950)

AT Commands

- Text protocol
  1. Infotainment issues commands
  2. The modem answers
- Line ends with `\r\n`
- One command at a time
- Multiple lines for response
  - Ends when a terminator is received
  - `OK`, `ERROR`, …
- Notifications messages

```
ATE0
        ATE0
        OK
AT+CMEE=2
        OK
AT+QADC=0
        +QADC: 0,961
        OK
AT+CGMM
        AG525RGL
        OK
```

Quectel custom command

Some modem data are read at initialization

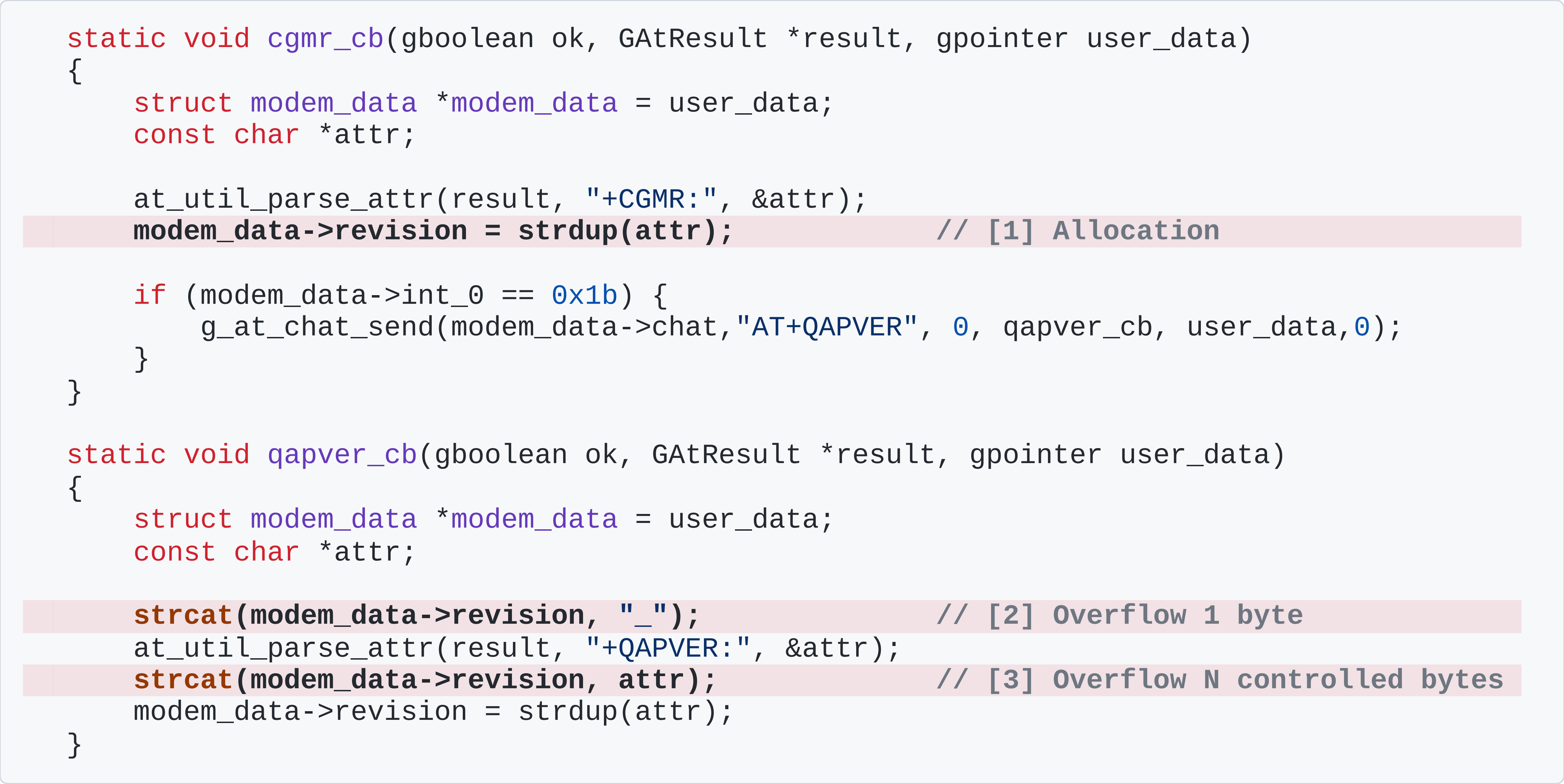
1. `AT+CGMI` : Manufacturer
2. `AT+CGMM` : Model
3. `AT+CGMR` : Revision

Quectel added a new one : `AT+QAPVER`

For Iris modem :

```
AT+CGMR
    AG525RGLAAR01A16M4G_OCPU
    OK
AT+QAPVER
    +QAPVER: 02.003.10.003
    OK
```

Vulnerability in Iris plugin



```
static void cgmr_cb(gboolean ok, GAtResult *result, gpointer user_data)
{
    struct modem_data *modem_data = user_data;
    const char *attr;

    at_util_parse_attr(result, "+CGMR:", &attr);
    modem_data->revision = strdup(attr);            // [1] Allocation

    if (modem_data->int_0 == 0x1b) {
        g_at_chat_send(modem_data->chat,"AT+QAPVER", 0, qapver_cb, user_data,0);
    }
}

static void qapver_cb(gboolean ok, GAtResult *result, gpointer user_data)
{
    struct modem_data *modem_data = user_data;
    const char *attr;

    strcat(modem_data->revision, "_");              // [2] Overflow 1 byte
    at_util_parse_attr(result, "+QAPVER:", &attr);
    strcat(modem_data->revision, attr);             // [3] Overflow N controlled bytes
    modem_data->revision = strdup(attr);
}
```

Heap-based buffer overflow exploitation

✓ Bug primitive

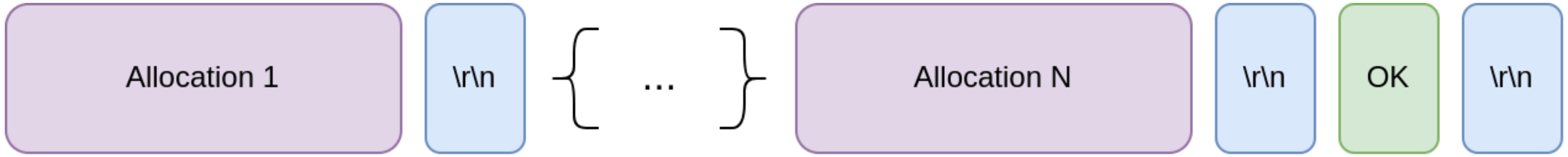
- Heap overflow
- Controlled overflow size
- Controlled allocation size
- Controlled content but bad characters : `\x00` , `\n` , `\r`

! Difficulties

- No null byte in the overflow
- No **`shaping primitive`** : mostly no allocation kept between commands
- TCP buffering of the line reception

Heap shaping

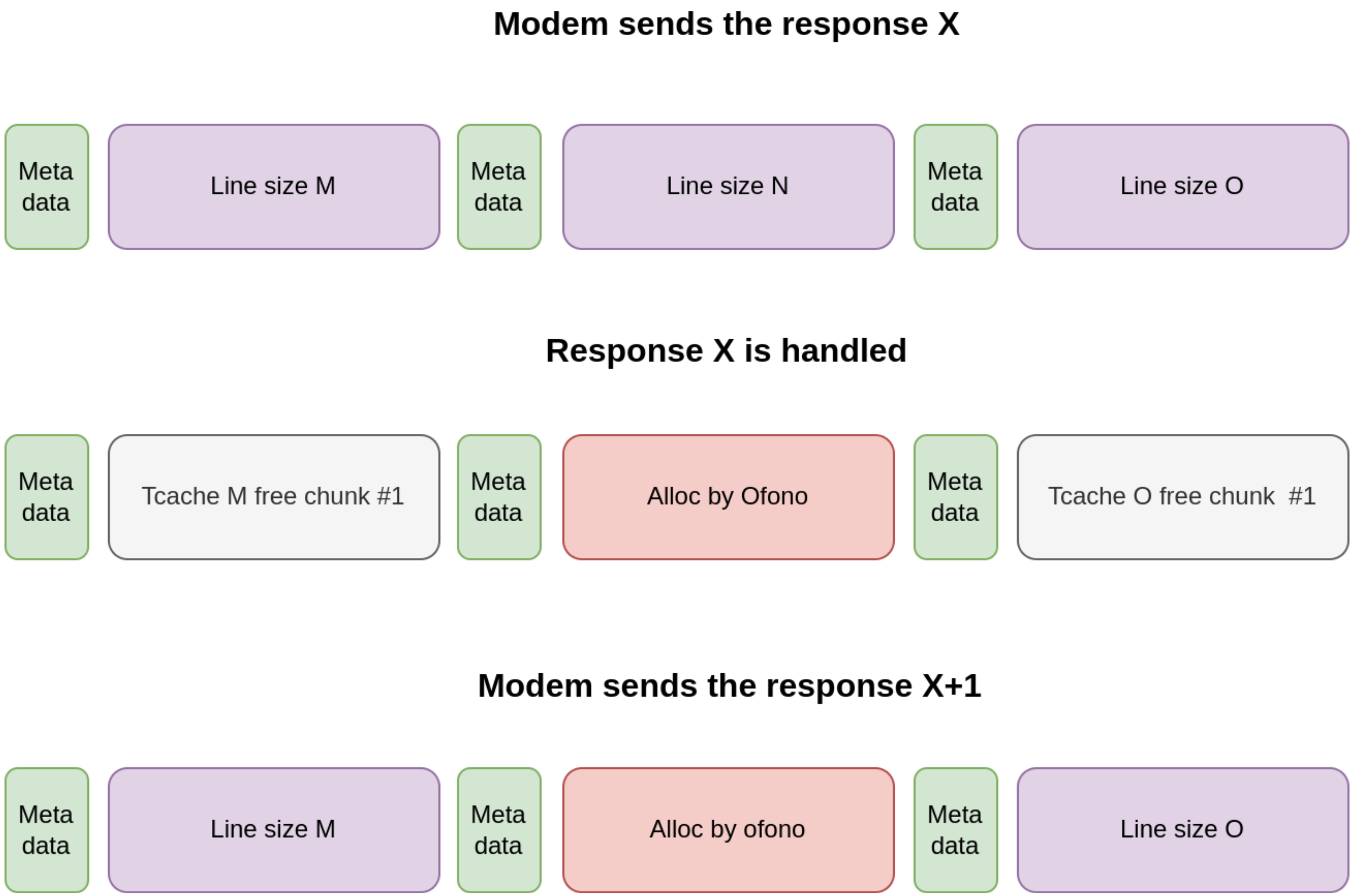
For each line in the command response until `OK`, there is an allocation with controlled size and content.



- Ofono uses **tcache** with only one thread
- These chunks are freed after the command response is handled
- Tcache prevents from merging them.

→ By playing with sizes, it is possible to place precisely an allocation.

Shaping with free tcache chunk



Modem sends the response X

Meta data | Line size M | Meta data | Line size N | Meta data | Line size O

Response X is handled

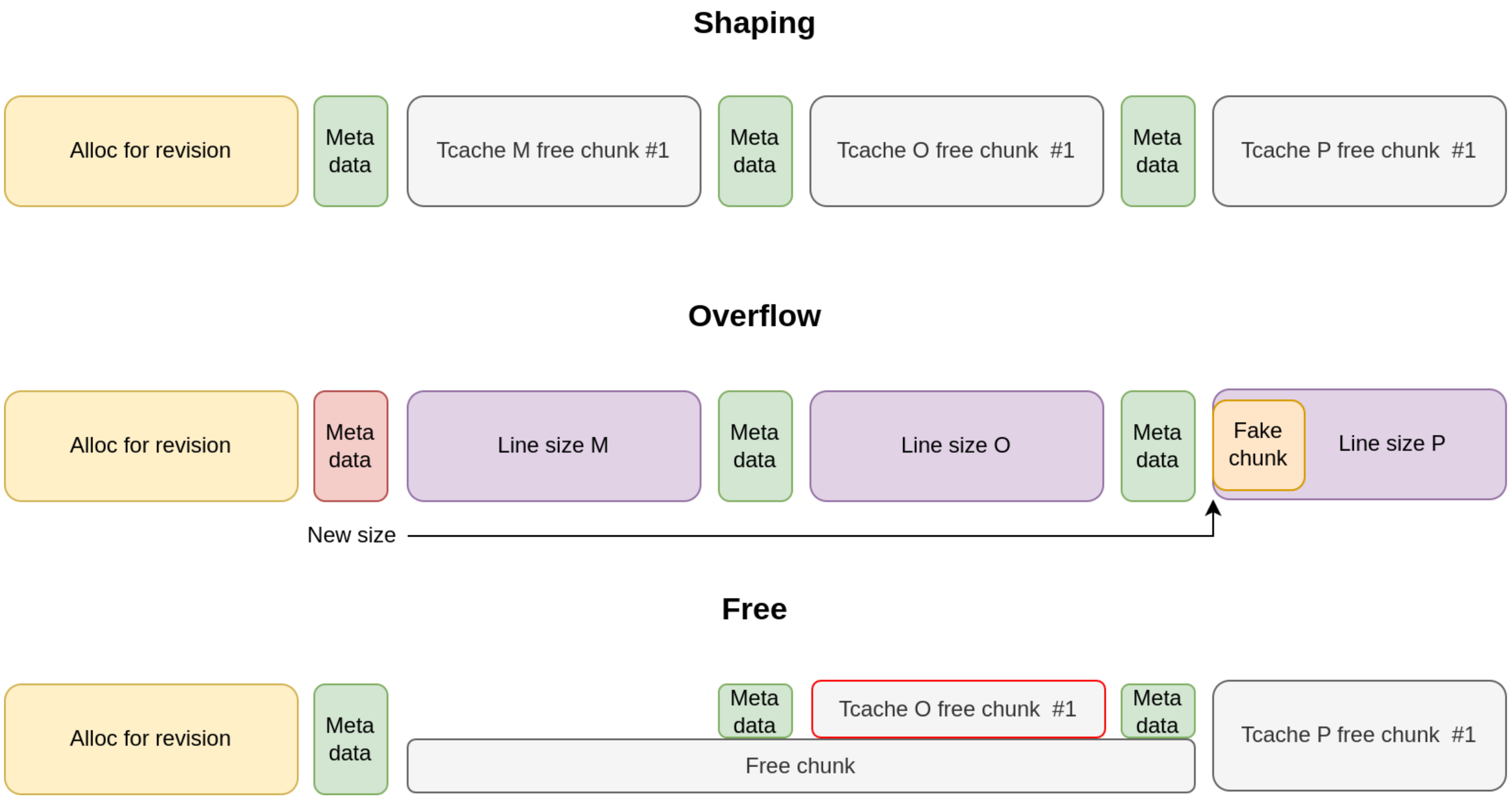
Meta data | Tcache M free chunk #1 | Meta data | Alloc by Ofono | Meta data | Tcache O free chunk #1

Modem sends the response X+1

Meta data | Line size M | Meta data | Alloc by ofono | Meta data | Line size O

From heap overflow to chunk overlap

SYNACKTIV



From overlap to arbitrary read

1. Use the chunk overlap to take over a Ofono structure
2. During initialization, some *files* in the SIM card are read
3. Modifying a structure `sim_fs_op` allows exfiltrating memory

```
struct sim_fs_op {
    // ...
    unsigned char *buffer;      // Exfiltrated data
    // ...
    int length;                 // Size of exfiltrated data
    // ...
    gboolean is_read;           // Change the READ operation to a WRITE
};
```

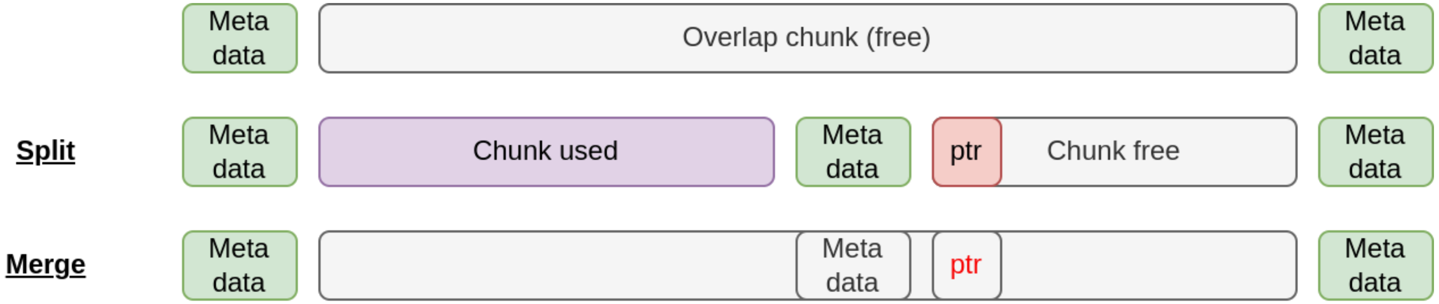
Heap content exfiltration

SYNACKTIV

But where to read ?

- ASLR on all mappings
- Need to place a valid pointer in `buffer`

Generic solution to write a heap pointer in the heap :



Leak obtained :

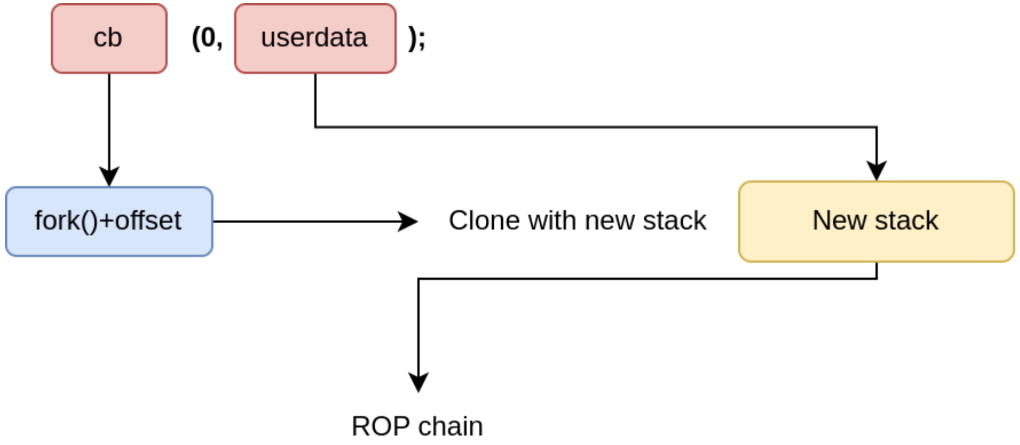
```
AT+CRSM=214,-1866667136,0,0,4096,"414141414141414141414141..."
```

Code execution

- Controlling `sim_fs_op` also gives execution flow control with `cb`

```
struct sim_fs_op {
    //...
    gconstpointer cb;
    void *userdata;
};
```

- Arbitrary call with second argument controlled (`userdata`)
- Jump in the libc clone implementation → ROP

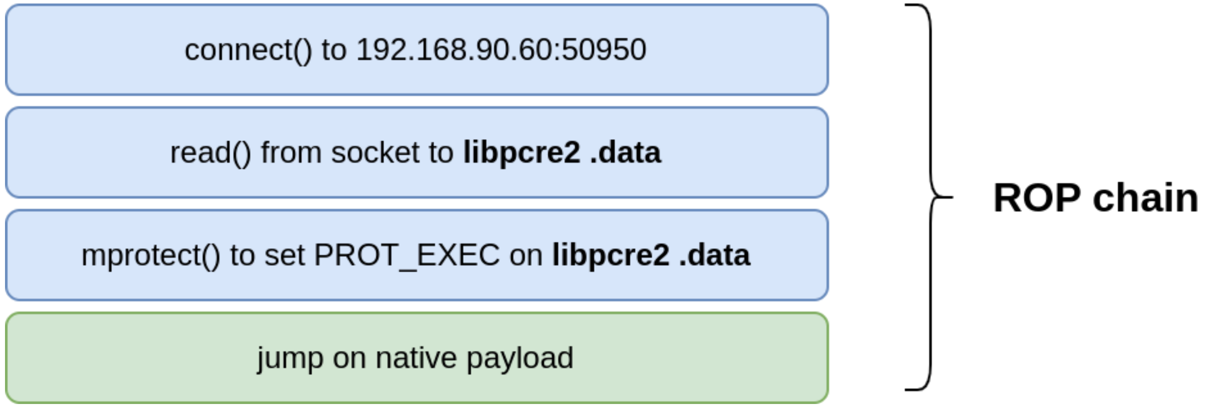


XPIN

- Tesla developed a new LSM called *XPIN*
- Hooks memory management syscalls
- Prevents process from configuring executable mapping with untrusted data
- The hook for `mprotect` adds checks :
  - The mapping needs to be **backed by a file**
  - The file must be in a FS protected with **dmverity**
- SELinux has a similar feature

XPIN Bypass

- Linux uses a Copy-On-Write mechanism (COW) when a mapping is set writable
  1. The page is shared until a write operation
  2. On write, the fault handler allocates a new page with the modified content
  3. The information about the file is kept (`vma->vm_file`)
- SELinux detects when a mapping is modified but XPIN does not
  - The page is **anonymous** (`vma->anon_vma`) after a modification
- The exploit uses a `.data` section of a library to execute code



connect() to 192.168.90.60:50950

read() from socket to **libpcre2 .data**

mprotect() to set PROT_EXEC on **libpcre2 .data**

jump on native payload

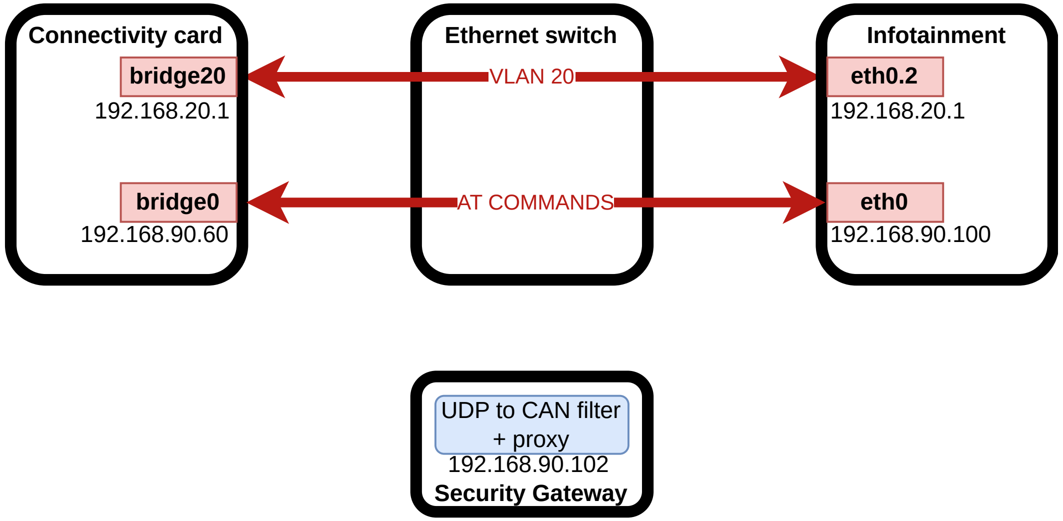
ROP chain

Network isolation bypass

Ofonod attack surface

- Heavily sandboxed process
  - Minijail: runs in a dedicated namespace with a dedicated Linux UID
  - SecComp (Kafel): Only syscalls used during normal operation are allowed
  - Apparmor: Limit access to files to the minimum required
  - Iptables: Only the AT command TCP connection is allowed
- Have the `CAP_NET_ADMIN` capability !
  - Used to manage the state of the data interface (UP/DOWN)
- Sandboxes allow Netlink socket !
  - Used by some Ofono modems (not used on Tesla but are enabled)
  - Used by the `udev` ofono interface (not used on Tesla, but enabled at build time)

Ofonod attack surface



Ethernet switch enforce firewalling

- Connectivity card can only:
  - Communicate with the infotainment with tagged VLAN 20
  - Communicate with the infotainment for the TCP AT commands
- Infotainment can send CAN over UDP messages to the GTW

Routing packets

Two Iptables rules are interesting

```
-A FORWARD -s 192.168.10.2/32 -i vtap -o eth0.2 -j ACCEPT
-A POSTROUTING -s 192.168.10.0/24 -o eth0.2 -j MASQUERADE
```

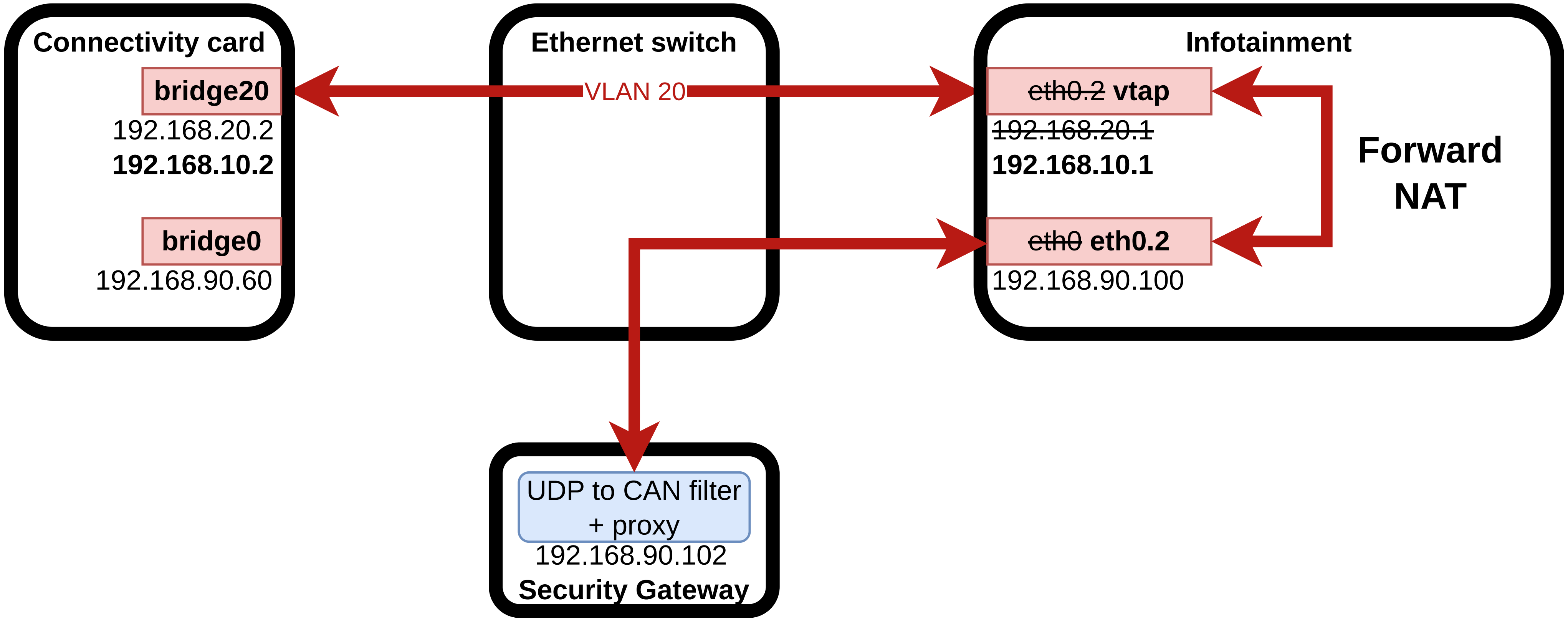
- Used to give Internet access to the Steam (games) virtual machine

Ofono is `CAP_NET_ADMIN` and can open Netlink sockets

- Can rename network interfaces
- Can change the IP configuration
- We can take advantage of these Iptables rules to forward packets to the security gateway from the connectivity card

Routing packets

Send CAN packets from the connectivity card



```
-A FORWARD -s 192.168.10.2/32 -i vtap -o eth0.2 -j ACCEPT
-A POSTROUTING -s 192.168.10.0/24 -o eth0.2 -j MASQUERADE
```

Conclusion

- Not so long of a work
  - Strong knowledge of the Tesla cars architecture
  - Got very lucky to spot the iptables race condition
  - Command injection was found before by someone else on another Quectel device[1]
- Future Infotainment exploit will be harder
  - Sandboxes are hard to bypass
  - Native code execution will be much harder in the future (XPIN)
- Great support from Tesla
  - Provides Infotainment and connectivity card
  - Version freeze 1 month before the event
  - Thanks to them
- Was fun
- Some Pwn2Own Automotive targets were much easier

[1] https://github.com/closethe/AG550QCN_CommandInjection_ql_atfwd

SYNACKTIV

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

https://twitter.com/synacktiv

https://synacktiv.com