360 | SKY-GO 360 VEHICLE CYBER SECURITY

Security Research Report on Mercedes-Benz Cars

1. Abstract

Nowadays, more and more intelligent and connectivity functionalities have been introduced to modern cars, which also brings more attack surfaces to the cars. As a car security research team, we would like to learn more about the connected cars' design and development, since they have more intelligent and connectivity functionalities, we initiated the research on Mercedes-Benz in 2018.

In this paper, we discuss how to perform security research on the intelligent car. First of all, we talk about how to build a testbench with relevant intelligent components at a low cost. Second, we design an attack chain from the outside to the inside of the vehicle based on this testbench. Third, we perform the attack chain in a genuine car. This paper explains how we researched a Mercedes-Benz E-Class car and found the vulnerabilities. By exploiting these vulnerabilities, we can remotely unlock the door and start the engine; and they potentially impact all Mercedes-Benz connected cars in China (estimated over 2 million).

2. Abbreviations and acronyms

This Paper uses the following abbreviations and acronyms:

CAN	Controller Area Network
ECC	Error Correction Code
ECU	Electronic Control Unit
EIS	Electronic Ignition System
HERMES	Hardware for Enhanced Remote-, Mobility- & Emergency Services
HU	Head-Unit
IAP	In-Application Programming
MCU	Micro Controller Unit
OOB	Out of Band
SLC	Single-Level Cell
TCU	Telematics Control Unit

3. Introduction

Sky-Go is a professional security research team on connected cars, organized in 2014, and we have performed a lot of car security-related researches. We collaborate with car manufacturers to help them strengthen their car security. Many car manufacturers and suppliers are our customers, such as FAW, Changan, Tesla, Dongfeng Nissan, Geely, BOSCH, BYD.

In 2018, we begin research on Mercedes-Benz, since it is one of the most famous car brands in the world and an industry benchmark in the automotive industry. We analyze the security of Mercedes-Benz cars.

There are so many models from Mercedes-Benz, and we finally chose the research target on Mercedes-Benz E-Class, since the E-Class's in-vehicle infotainment system has the most connectivity functionalities of all.

In this technical paper, we describe the research methodology. In order to protect the intellectual property of Daimler, we disclose limited security designs and limited code details.

Figure: Test Cars in the Research

4. Build the Testbench

In this Chapter, we describe the procedure that how to build the testbench.

4.1. The architecture of Telematics

The first step in testbench building procedure, we need to reveal the architecture of targeted system. Based on this architecture, we can figure out the key components.

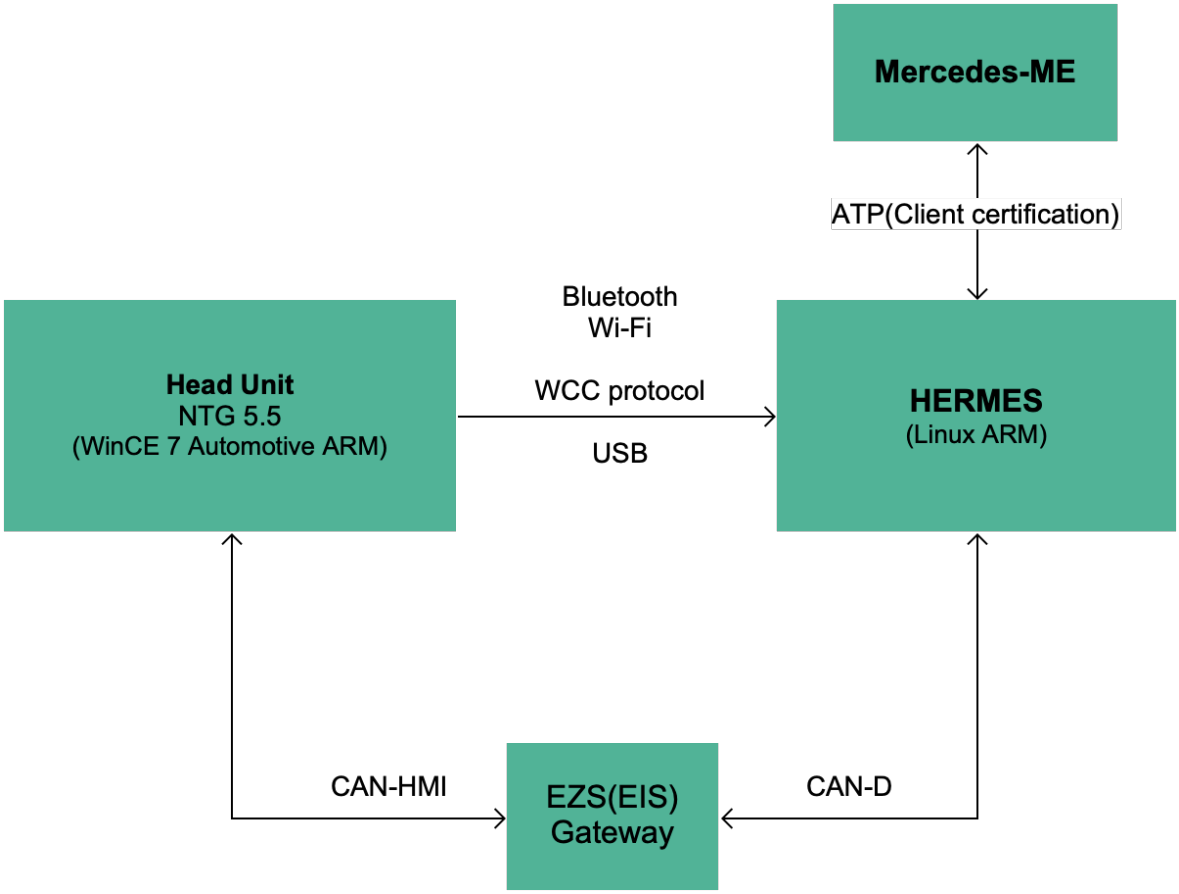


Figure: Architecture of Connectivity Functionality

The Key components for our testbench are as follows:

- Head-Unit
- HERMES
- Screen
- Center Control Media Button Mouse

4.2. Obtain Key ECUs

The first step is to dismantle the control panel to tear down ECUs.

Figure: HERMES 1.5 in the E300L

Figure: NTG 5.5 Head-Unit

After obtaining the target devices, it is necessary to collect relevant information such as network topology, pin definitions, chip model and enable signals in the car.

Therefore, we need to disassemble the center panel in the car and find out the wiring connections between the ECUs.

It is also necessary to know the CAN messages that could enable the ECUs, such as wake-up status messages and ignition status messages.

4.3. Head-Unit

The 2018 Mercedes-Benz E300L uses a Head-Unit code-named NTG-55, which is designed by Mitsubishi Electronics. The FCC ID of this device is UJHNTG55HUE. The operating system running in the NTG-55 is Windows Automotive 7, which based on Windows embedded compact 7. The main control chip used by NTG55 is Renesas R-Car H2 SoC. It uses an octa-core ARM-v7 architecture and is specially designed for in-vehicle infotainment systems.

There are also some security protection mechanisms in the Head-Unit, such as secure boot, storage media encryption (SD card & HDD), and anti-theft system.

The most challenging part is that this master SH-2A does not disclose the Datasheet. We can only analyze it with limited knowledge.

Figure: Cover of Head-Unit

4.4. HERMES

HERMES is a Telematics Control Unit and it is equipped in all Mercedes-Benz connected cars. The full name of it is Hardware for Enhanced Remote-, Mobility- & Emergency Services.

It handles emergency calls, information calls, with support for remote diagnosis, local diagnosis, which can communicate with each ECU. Besides, it is responsible for the Internet access function of the Head-Unit and supports 2.4GHz and 5GHz WLAN networking. The CAN transceiver is connected to the CAN bus 500k, and the LIN line is connected to the Airbag.

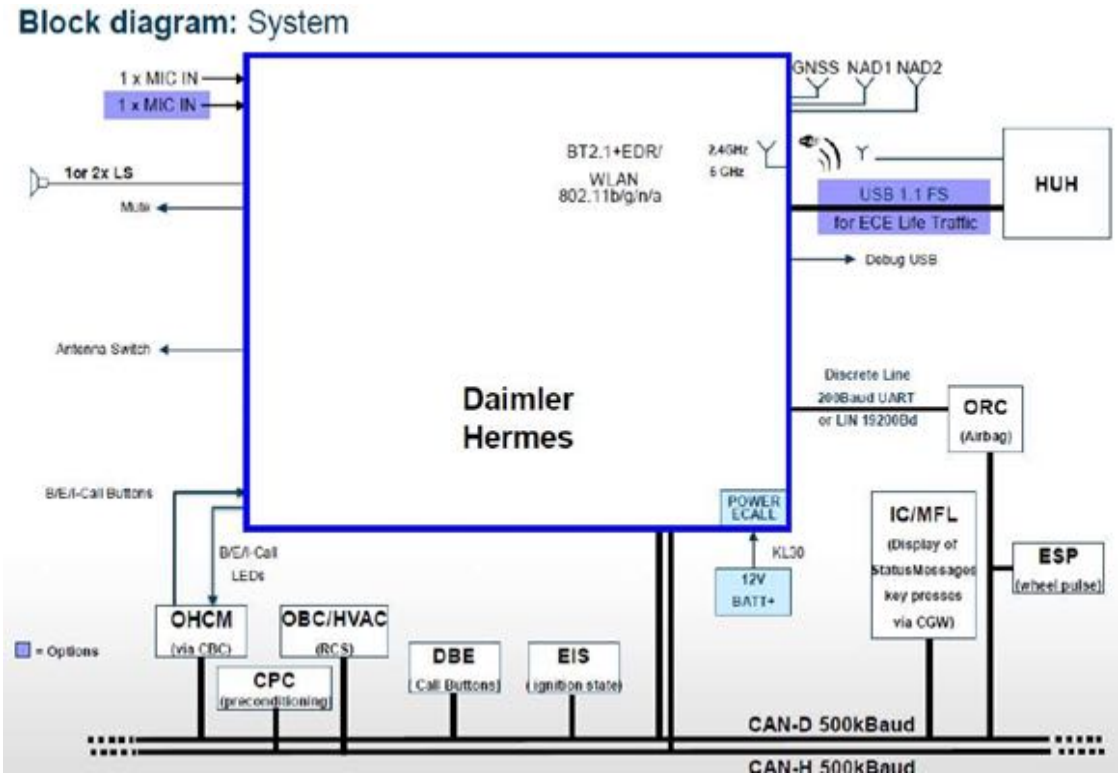


Figure: HERMES system block diagram

The core of HERMES is the communication module, which supports 3G & 4G network. The module can set up a wireless network for the Head-Unit, and the network could be Wi-Fi or Bluetooth.

This solution is called OpenCPU in China. The performance of the communication module is higher than MCU, so it is responsible for calculating data and running the operating system. The primary operating system of the communication module is Linux, and the throughput performance of the module can meet the working requirements. Some 4G routers also use this solution.

The communication module communicates with the MCU through the UART and is responsible for control instructions and software upgrades. SH2A MCU is responsible for managing peripheral chips, including LIN transceiver, CAN transceiver, and power management.

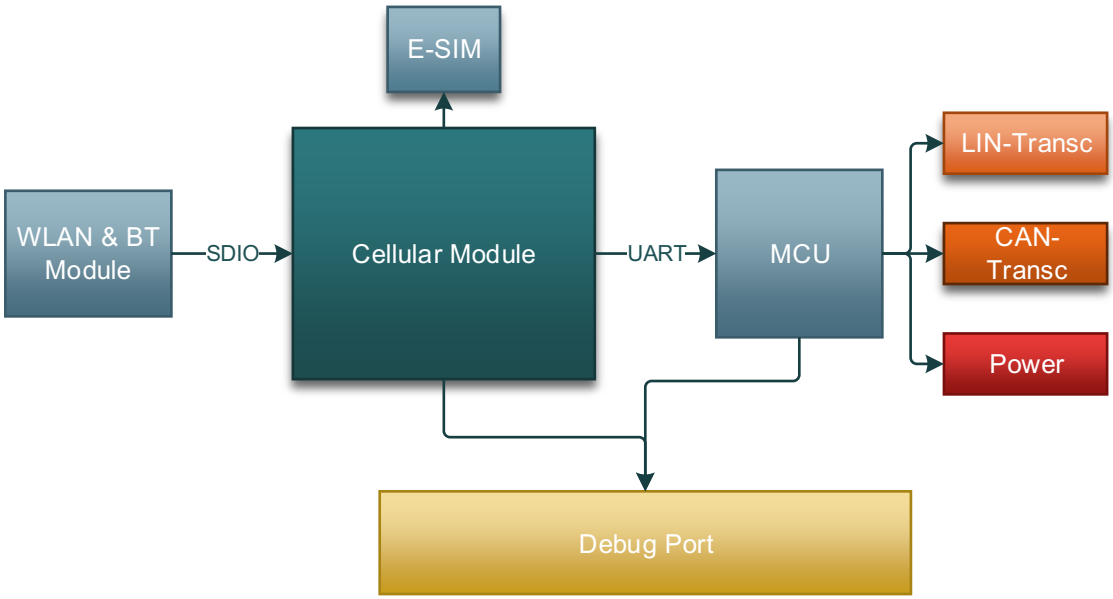


Figure: HERMES hardware block diagram

Basebands of communication modules vary in different regions.

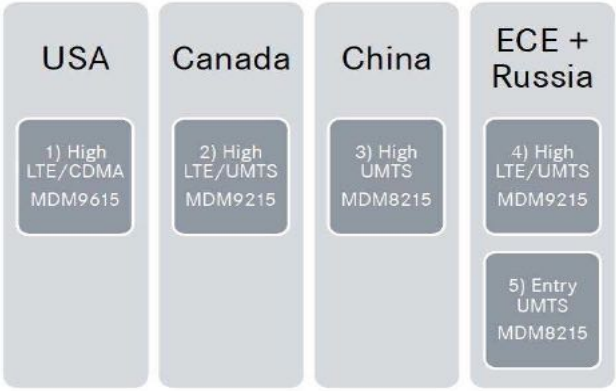


Figure: Network modes in different regions

In this research, we analyze 4 versions of HERMES.

Version	Communication Modules	Models of the Car
HERMES v1.1	ME809Tu UTMS	All Mercedes-Benz connected cars
HERMES v1.2	ME909Tu LTE	All Mercedes-Benz connected cars
HERMES v1.5	ME919bs	All Mercedes-Benz connected cars
HERMES v2.1	ME919bs	All Mercedes-Benz connected cars

Table: Information for HERMES

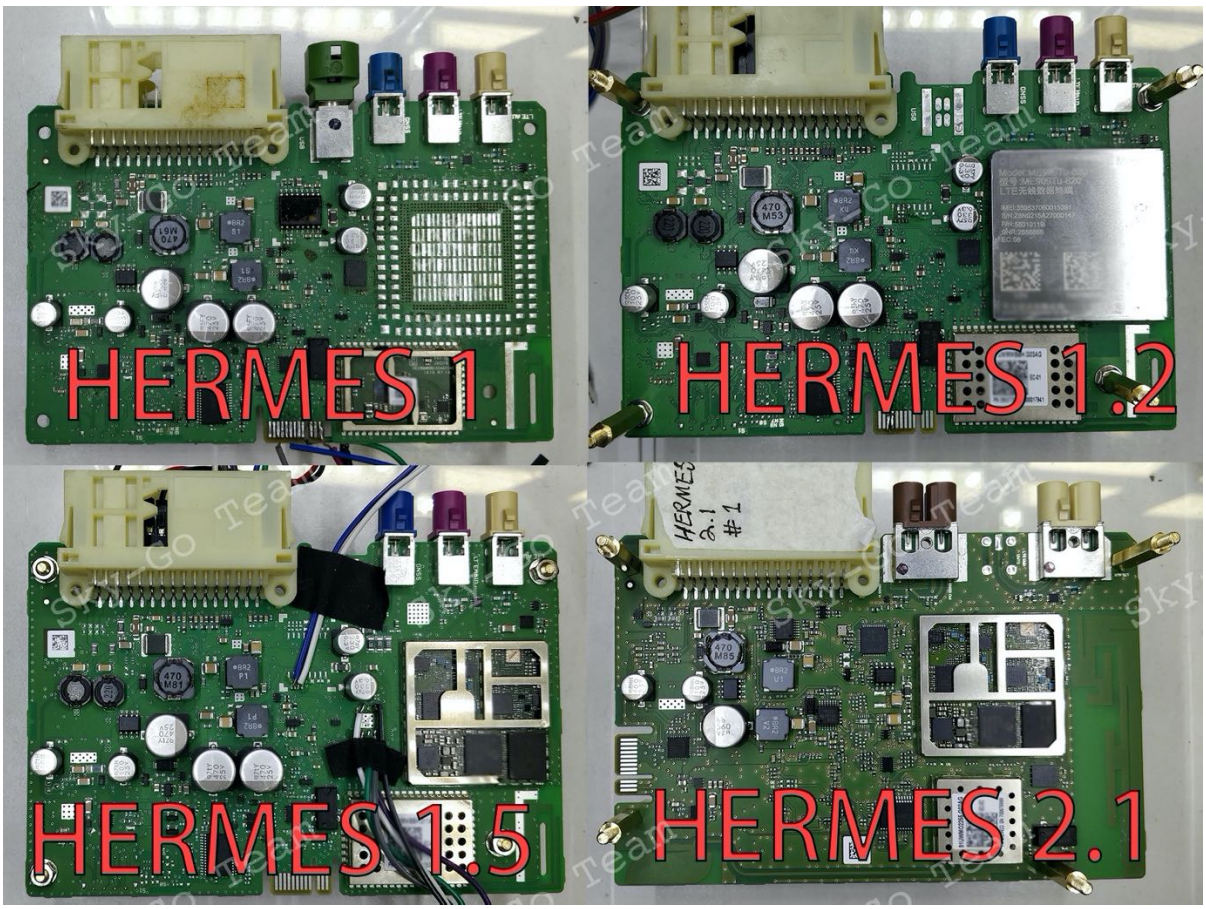


Figure: Comparation of HERMES

The HERMES v1 PCB has a USB interface.

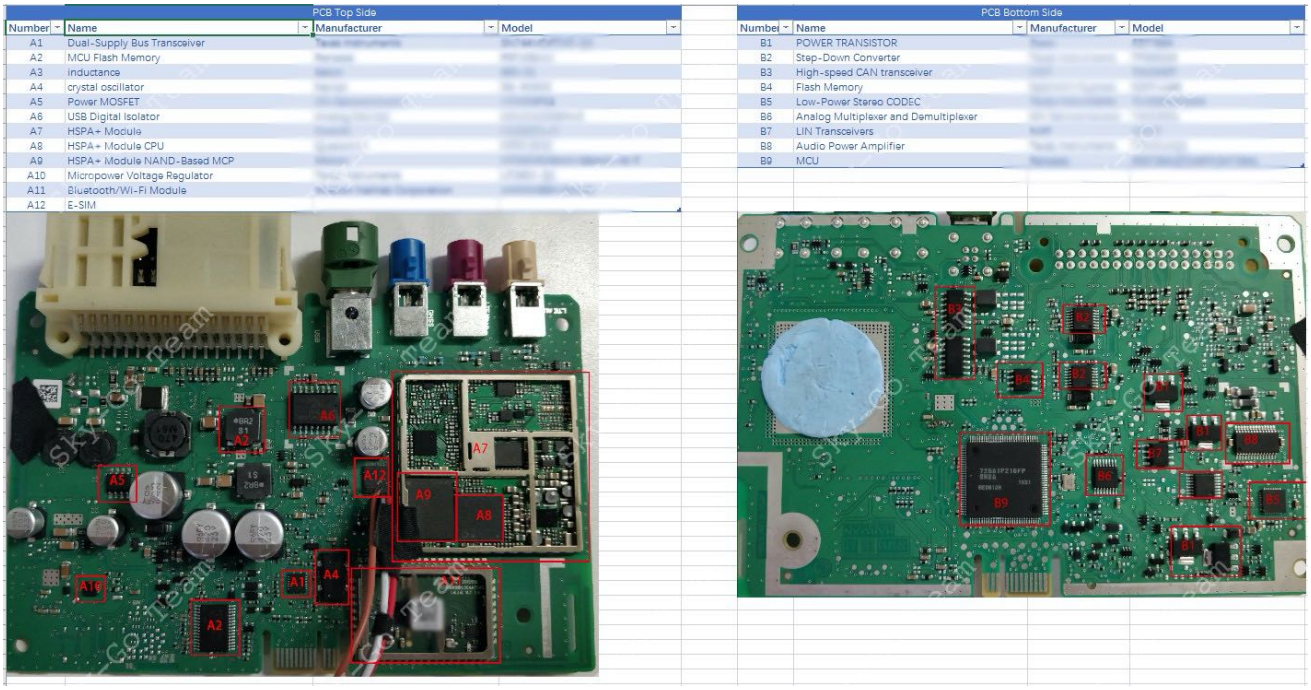
The vias of the HERMES v1.2 PCB are covered with solder masks, and the USB interface is removed.

The HERMES v1.5 uses the ME919Bs communication module with a GPS module.

The HERMES v2.1 is different. The debug port is moved from the bottom to the left. There are two FAKRA LTE antennas to ensure signal stability, and a GNSS interface to receive GPS information. The USB interface is used to provide network functions for the Head-Unit.

4.5. Pinout Definition

It's necessary to list the information of all ICs on the PCB. This work is very similar to copying a PCB. The purpose is to understand the working principle of the device at the hardware level.



PCB Top Side

Number	Name	Manufacturer	Model
A1	Dual-Supply Bus Transceiver		
A2	MCU Flash Memory		
A3	Inductance		
A4	crystal oscillator		
A5	Power MOSFET		
A6	USB Digital Isolator		
A7	HSPA+ Module		
A8	HSPA+ Module CPU		
A9	HSPA+ Module NAND-Based MCP		
A10	Micropower Voltage Regulator		
A11	Bluetooth/WI-FI Module		
A12	E-SIM		

PCB Bottom Side

Number	Name	Manufacturer	Model
B1	POWER TRANSISTOR		
B2	Step-Down Converter		
B3	High-speed CAN transceiver		
B4	Flash Memory		
B5	Low-Power Stereo CODEC		
B6	Analog Multiplexer and Demultiplexer		
B7	LIN Transceivers		
B8	Audio Power Amplifier		
B9	MCU		

Figure: Chip information of HERMES board

It is not easy to find the debug port of the chip in the mass-produced version of the PCB. If no silkscreen is found, you can only test the connection between the test pads you think by multimeter according to the tracks or chip pin assignment.

LTE modules with HERMES version higher than 1.5 have more pins, making testing more laborious.

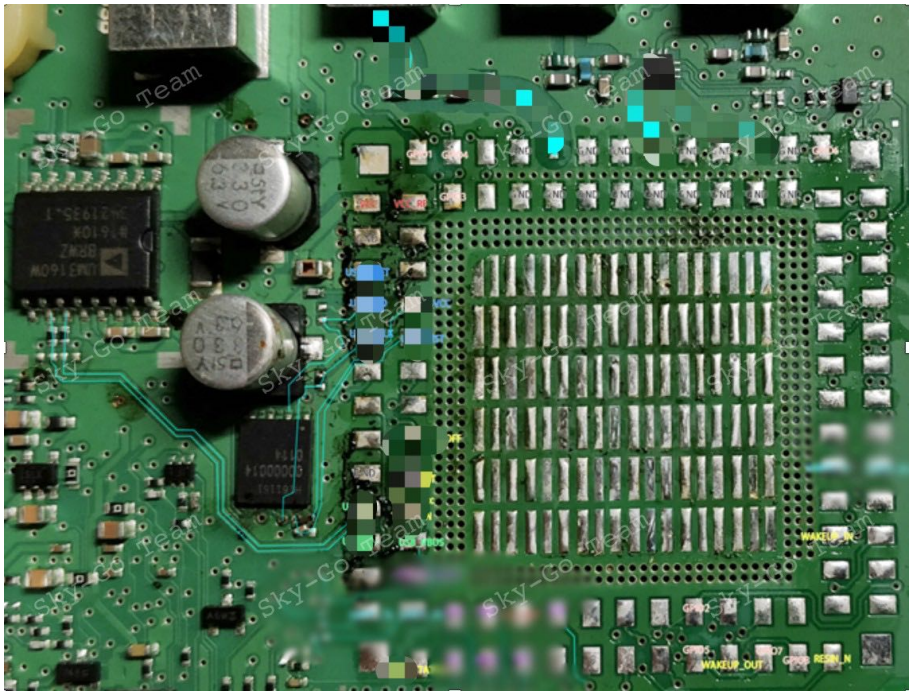


Figure: Definition of Communication module pinout

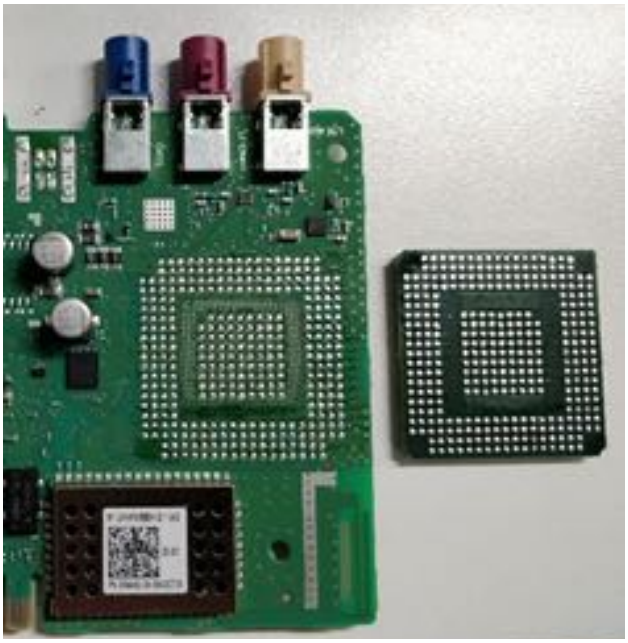


Figure: The LGA paddings of LTE module

Finally, we found a useful test pad.

Figure: Definition of Debug interface

To analyze whether the module exists a chip debug port, we scan the SoC with X-Ray to figure out the pins, which avoided damage caused by disassembling the equipment.

For example, we can find out the debug port on the processor, then check if there is a corresponding pad in the LGA pads.

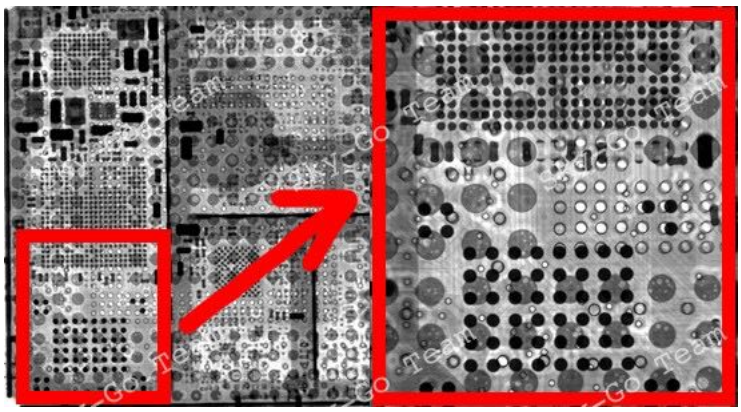


Figure: X Ray Image

Finally, we're able to sort out the pin definition of the System Connector.

Pin	Abbreviation	Description	Pin	Abbreviation	Description
1	CAN-D H	Diagnostic	17	CAN-L H	Hybrid ME08
[illegible]	[illegible]	[illegible]	18	CAN-L L	Hybrid ME08
			19	USB_VBUS[1]	NAD-Debug interface USB-VBus
			20	USB_DM[1]	NAD-Debug Interface USB-DM
5	USB_GND	NAD-Debug Interface USB-GND	21	USB_DP[1]	NAD-Debug Interface USB-DP
6	ICall_LED				
7	RCall_LED				
8	ECall_LED		24	ECall_Btn#	ECall-Button
[illegible]			25	NC	
[illegible]			26	31	GND
11	LIN_BUS		27	[illegible]	VCC
12	Mute	SoundSystem or HU	28	RF_Switch	
13	MicIn-[1]		29	Mic-[0]	HeadUnit GND
14	MicIn+[1]		30	Mic+[0]	HeadUnit
15	NC		31	Shield	Microphone shielding
16	ECallSpeaker+	ECallSpeaker-	32	ECallSpeaker+	Center instrument panel speake

Figure: Connector Pin assignment of HERMES

There is an easier way to find the definition of the connectors. A User Manual of HERMES is leaked in online publicly available databases, which has the connector pin definition.

The HU needs to connect to the TCU to access Internet. HU has three ways to connect with TCU: USB UTC, Bluetooth DUN and Wi-Fi. The configuration file in the system determines the actual connection method. Before establishing connection between HU and TCU, they need to negotiate protocols through CAN-A. After the connection is established, they manage the network through WCC protocol.

HU connects to two CAN-buses: the first is CAN-D, which is a comfortable can bus, the other way is CAN-HMI.

The Communication module has 2-ways of PDP

PDP1: Applications on the HU need to get real-time car status, real-time road condition. Communicate with the server. Communicate data to USB ECM.

PDP2: Set up a local area network with Head-Unit as a gateway. There are two networking modes which described as follows:

1: HU set an AP hotspot, the Communication module connects to the Wi-Fi AP as a STA, all the traffic to the Internet pass thought the LTE module.

2: Bluetooth: Only the old HERMES uses the Bluetooth datalink to access the Internet.

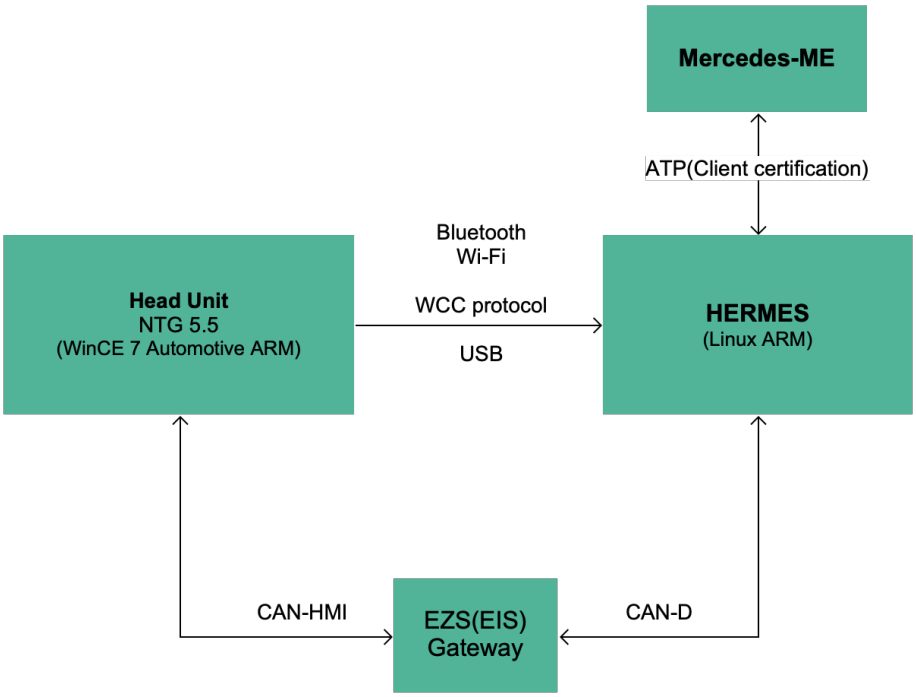


Figure: ECU Functions Connection diagram

4.6. Bypass Anti-Thefts

Mercedes-Benz has applied anti-theft technology since the 1990s. Thus, their development has been very experienced after many iterations and the system became very robust. Our version of NTG55 triggered anti-theft. Multiple anti-theft modes are implemented in the system. Among these, the highest-level mode cannot be cracked by external means unless unlocked by a dealer shop.

There are 3 levels to activate Anti-Theft in the Head-Unit as follows.

Level-A: The heartbeat messages error. You need to switch the ignition on, restart the system.

Figure: restart warning

Level-B: The VIN isn't matched. You need to ask your dealer to cancel the anti-theft.

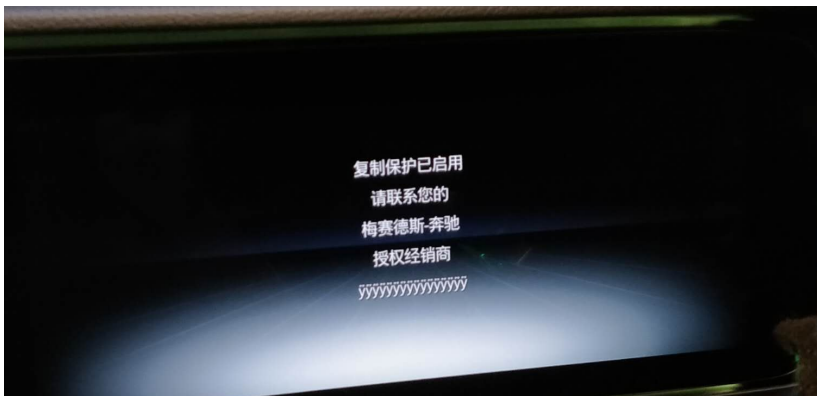


Figure: anti-copy protection warning

Level-C: Unexpected messages. You need to call the Mercedes-Benz Service Center.

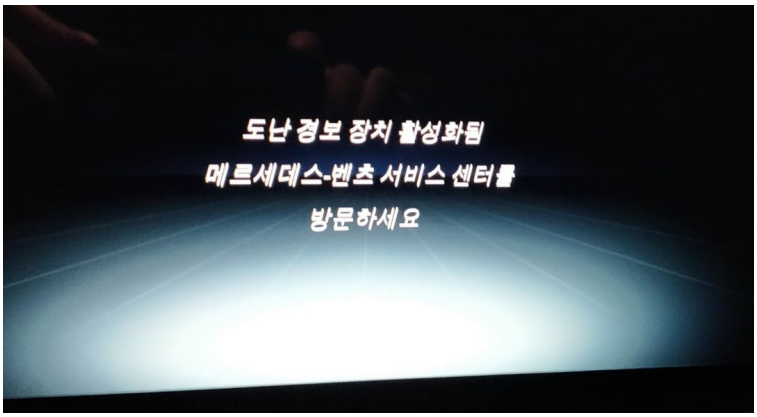


Figure: anti-theft warning

It's evident that if you replay the CAN-bus messages in a new car, the anti-theft could be activated.

However, we learned that if we block some corresponding data at CAN-bus for capturing the anti-theft Level-A deactivate message, the anti-theft Level-C will be enabled.

After some analysis work done in CAN-bus, we found some CAN-bus messages which can keep Head-Unit running. Two different heartbeat messages are needed to bypass the HU anti-theft system and make it run on the bench.

Use the collected information above to build a bench environment. The research preparation is completed, and the next step is to discover and analyze the loopholes.

For the Head-Unit that has triggered anti-theft, the anti-theft related data is stored in the SD card. However, the SD card is locked. The area where the anti-theft data stored is a file system developed by Daimler and runs in WinCE. Therefore, we need to disassemble the Head-Unit, reverse the file system driver, unlock the SD card and modify the anti-theft configuration, and then restart to disarm the anti-theft. The cost of this operation is much higher than going to a 4S shop to disarm theft.

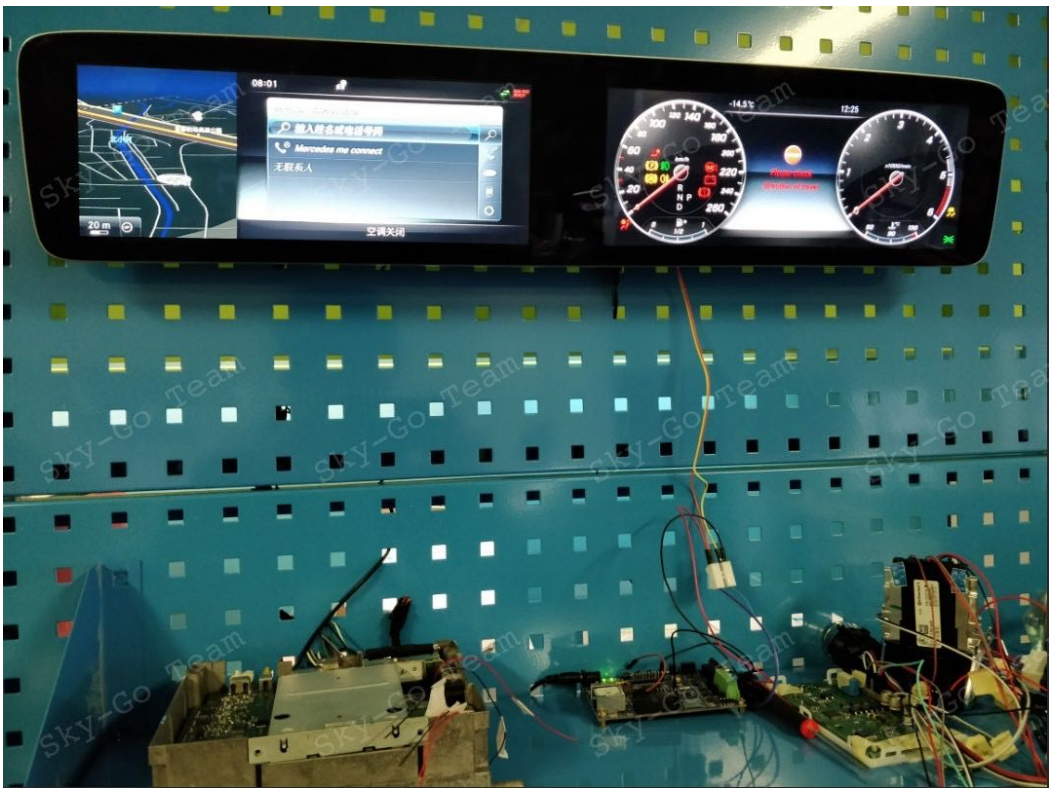


Figure: Testbench

5. Analysis Procedure

After analyzing the attack surfaces of the connected car, TCU is in the most crucial component in the whole system, since it is the communication module between the external network and the in-vehicle network.

5.1. Collecting the Network Information

Here're 4 APN configurations information which correspond to different environments.

```
]:  [info:] Loaded APN1 settings: URL: "[redacted].CLFU.NJM2MAPN", user: [redacted]
]:  [info:] APN with index 2 is not configured
]:  [info:] Loaded APN2 settings: URL: "[redacted].CLFU.NJM2MAPN", user: [redacted]
]:  [info:] APN with index 3 is not configured
]:  [info:] Loaded APN3 settings: URL: "DefaultValue1", user: [redacted]
]:  [info:] APN with index 4 is not configured
]:  [info:] Loaded APN4 settings: URL: "DefaultValue2", user: [redacted]
]:  [info:] >>>>[getInstance][6973]
```

Figure: APN configuration

There is the boot log to show the procedure when HERMES connects to car backend.

```
27 05:36:48.395 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... URL]
27 05:36:48.395 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... ]
27 05:36:48.395 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... ]
27 05:36:48.396 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... RL]
27 05:36:48.399 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... RL]
27 05:36:48.399 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... URL]
27 05:36:48.399 OMADM[1052]:  [info:] TCUReadCb read value [https:// ... for param ... URL]
```

Figure: Connections of TCU

We use the Qualcomm misconfiguration to open the debug function by connecting to the USB port.

Once we connected the HERMES to the PC, the device message showed the serials numbers and USB devices info.

```
[  141.272232] usb 1-4.4: new full-speed USB device number 20 using xhci_hcd
[  142.206792] usb 1-4.4: not running at top speed; connect to a high speed hub
[  142.232510] usb 1-4.4: New USB device found, idVendor=12d1, idProduct=1573, bcdDevice= 2.28
[  142.232522] usb 1-4.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  142.232528] usb 1-4.4: Product: KOM1
[  142.232535] usb 1-4.4: Manufacturer: DAG
[  142.232541] usb 1-4.4: SerialNumber: 359537060015391
[  142.419120] audit: type=1130 audit(1542617697.397:108): pid=1 uid=0 auid=4294967295 ses=4294
comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
[  142.419124] audit: type=1131 audit(1542617697.397:109): pid=1 uid=0 auid=4294967295 ses=4294
comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
[  142.424267] usbcore: registered new interface driver option
[  142.424551] usbserial: USB Serial support registered for GSM modem (1-port)
[  142.464430] cdc_ether 1-4.4:1.0 usb0: register 'cdc_ether' at usb-0000:00:14.0-4.4, CDC Ether
[  142.464561] usbcore: registered new interface driver cdc_ether
[  142.464941] option 1-4.4:1.2: GSM modem (1-port) converter detected
[  142.465241] usb 1-4.4: GSM modem (1-port) converter now attached to ttyUSB1
[  142.465466] option 1-4.4:1.3: GSM modem (1-port) converter detected
[  142.465668] usb 1-4.4: GSM modem (1-port) converter now attached to ttyUSB2
[  142.465882] option 1-4.4:1.4: GSM modem (1-port) converter detected
[  142.466093] usb 1-4.4: GSM modem (1-port) converter now attached to ttyUSB3
[  142.477550] cdc_ether 1-4.4:1.0 enp0s20f0u4u4: renamed from usb0
[  142.507558] IPv6: ADDRCONF(NETDEV_UP): enp0s20f0u4u4: link is not ready
```

Figure: USB devices on Linux

It reveals to be 6-devices in the Microsoft Windows system for debugging.

Figure: USB devices on Windows

We can use the AT command to operate the Communication module to get APN configurations. Also, we can flash new firmware into the module.

```
OK
AT+CGDCONT?
+CGDCONT: 1,"IPV4V6","[redacted]CLFU.NJM2MAPN","0.0.0.0",0,0
+CGDCONT: 15,"IP","[redacted]CLFU.NJM2MAPN","0.0.0.0",0,0
+CGDCOAT+CIND?
+CIND: 0,2,1,1,0,0,1,0
```

Figure: APN configuration

We cannot impact the backend merely with the vulnerabilities above. We need to analyze the vulnerabilities in communication.

5.2. Dumping the Firmware

In previous Connected Car researches, dumping firmware would normally be our first step. In this research, we did the same.

At first, we try to tear down the NAND Flash from Cellular Module to dump data from NVM.

The HERMES < 1.5 use the Qualcomm. The SoC is MDM9615. There's no internal RAM, so the RAM is in the same package as ROM. The memory flash is the Micron SLC NAND, and the package is the BGA 137 MCP.

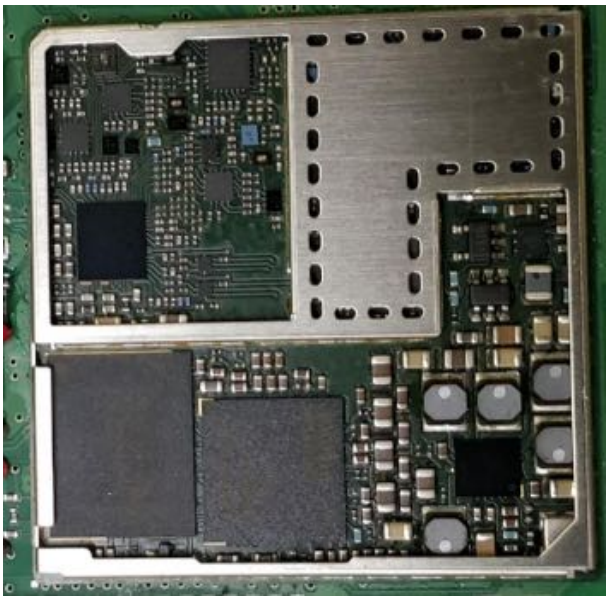


Figure: LTE module PCB

We use the BGA Rework Station to disassemble the NAND Flash.

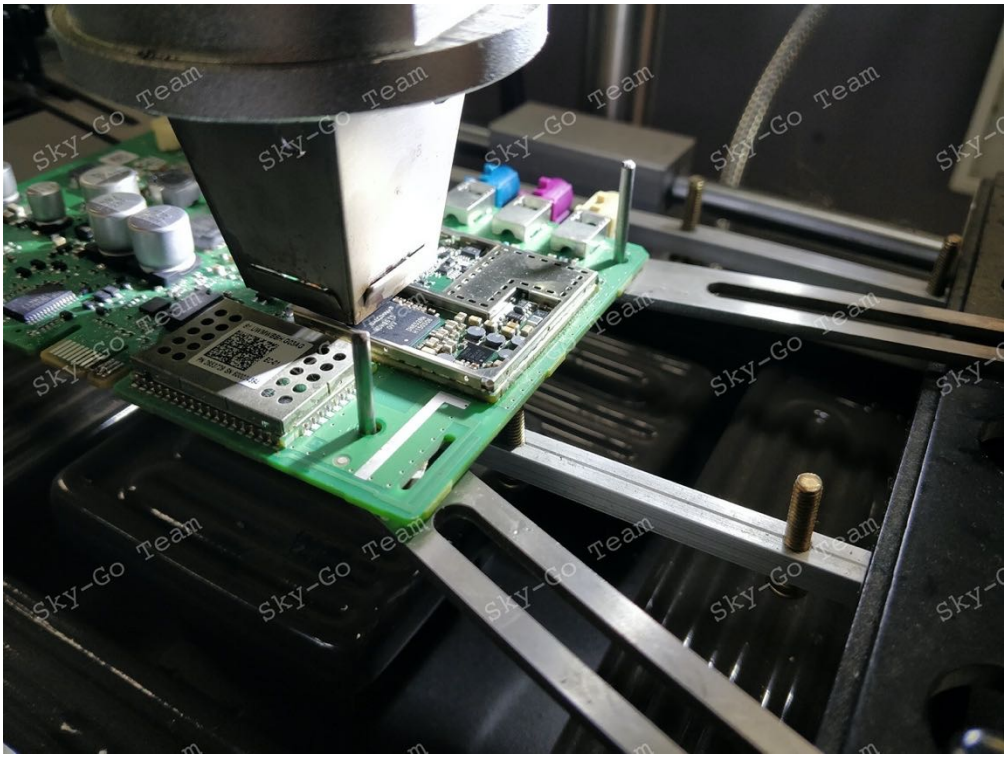


Figure: BGA rework station

Unfortunately, we had no available NAND Flash Adapter at that time. As an option, we jumped wires from the NAND Flash footprint to the TSOP-48 Adapter according to BGA-137 pin assignments.

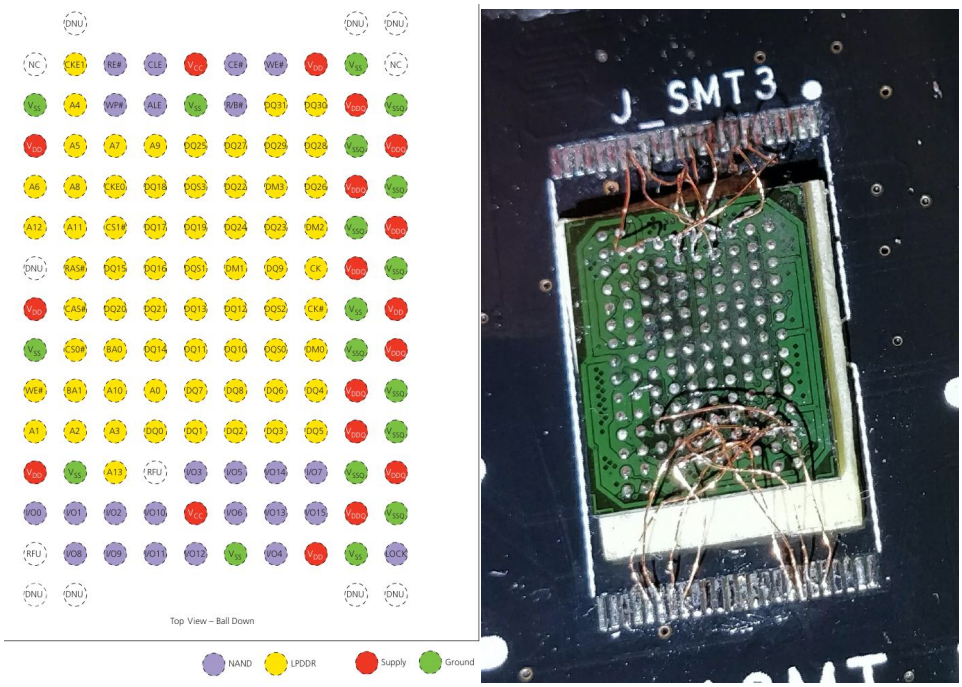


Figure: NAND Flash pinout

The NAND flash consists of blocks, and the blocks consist of pages. In general, it can skip the OOB. But as to the 4G modules, the spare area including relevant data: Bad block information, ECC bytes, Erase Block mapping info. So, we must dump the NAND Flash with the OOB area.

Figure 7: Array Organization – MT29F4G08 (x8)

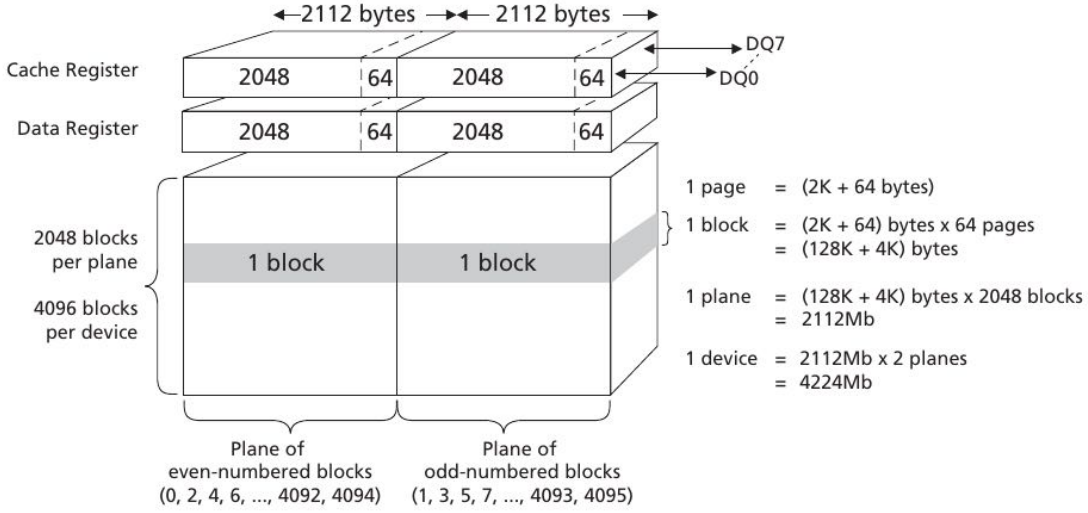


Figure: NAND array organization

We have another way to dump the raw NAND flash. Although the entrances that transport the data are different, the principles are the same: operate the CPU to read pages data from peripheral memory devices to the RAM, then dump the RAM data to the host (PC or the emulators)

Connect wires to the debug ports, set breakpoints at the appropriate time. The external signals were set to prevent watchdog timing out. Otherwise, the watchdog restarts the SoC when you are reading the RAM data.

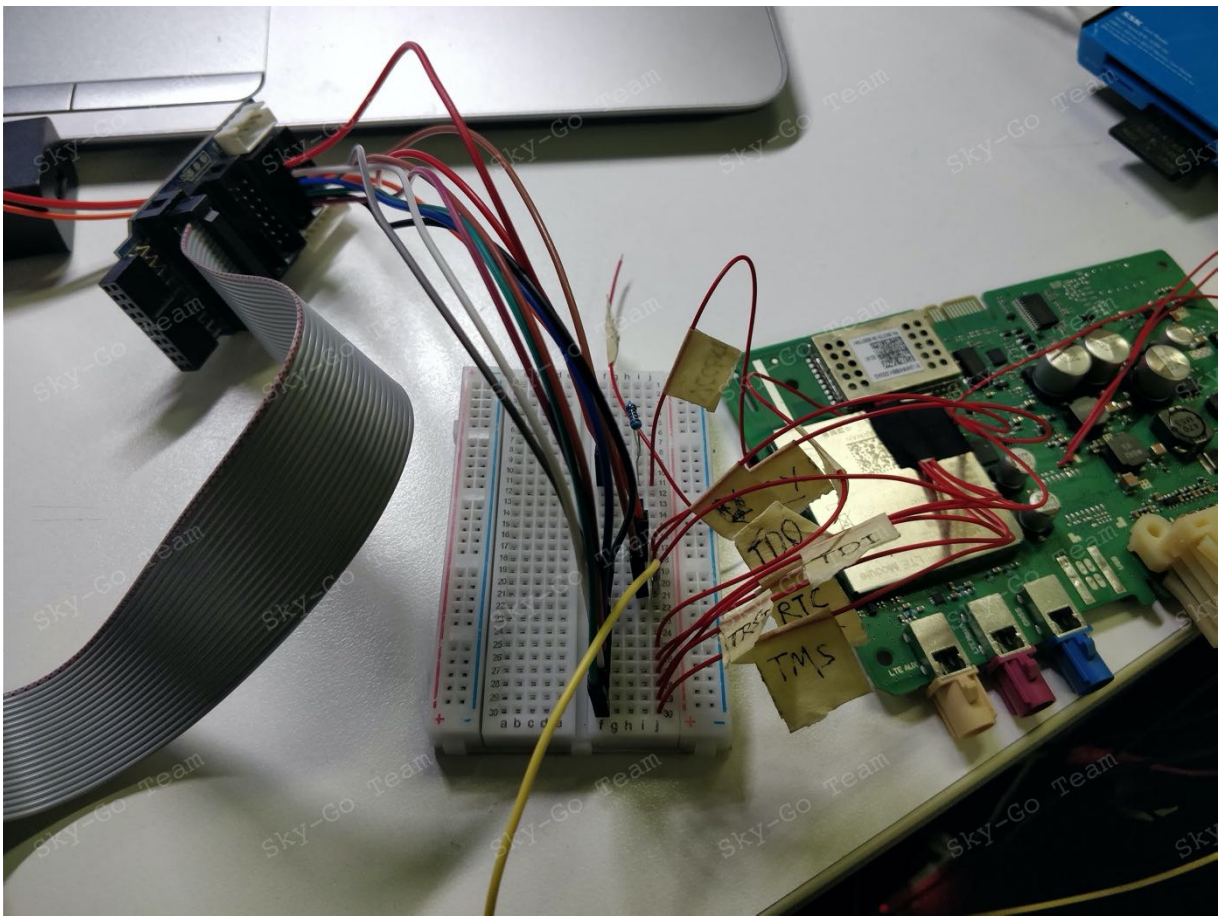


Figure: JTAG Connector

In this research, we use the OpenOCD with FT2232 to operate the debug interface.

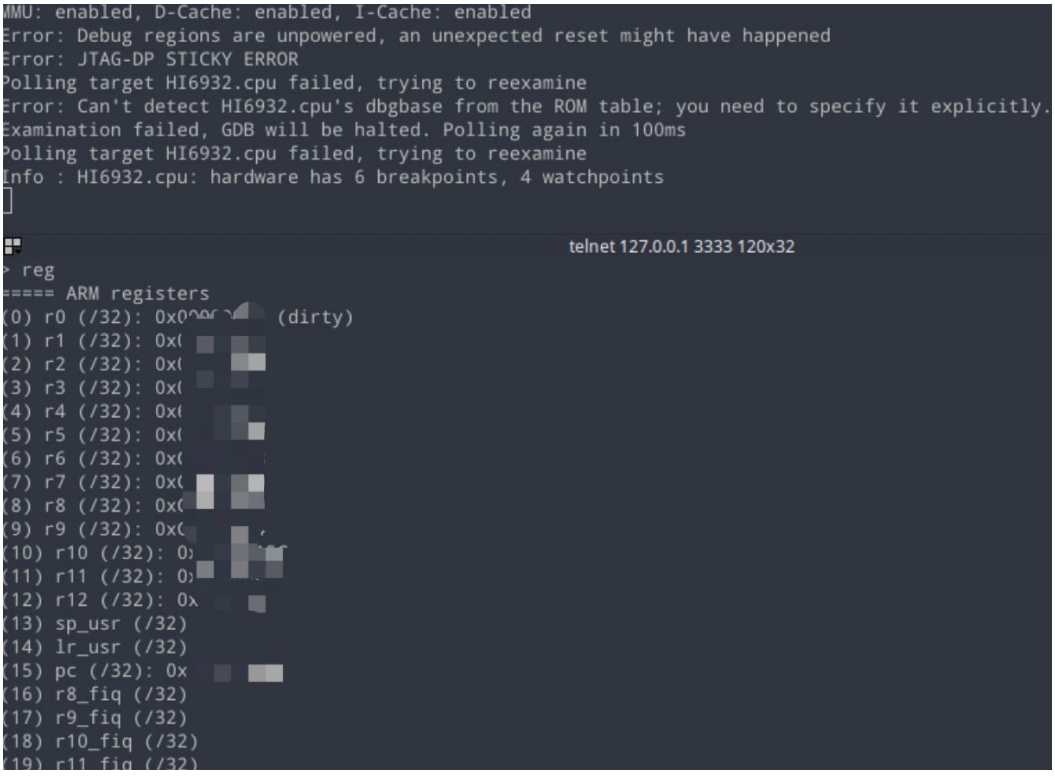


Figure: JTAG debugging for Hi6932

We can read or write the NAND Flash by IAP: run a DLOAD program in the RAM, the DLOAD program will read the NAND pages to the RAM, then we can read the RAM via the debug port or dump the RAM to a USB disk.

The raw NAND Flash file is not similar to the Flash with the routers which can be extracted by Binwalk.

After referring to the chip's datasheet, we figured out the spare area distribution, and thus, we can read the regular partitions by skipping the OOB area.

Figure 78: Spare Area Mapping (x8)

Max Byte Address	Min Byte Address	ECC Protected	Area	Description
1FFh	000h	Yes	Main 0	User data
3FFh	200h	Yes	Main 1	User data
5FFh	400h	Yes	Main 2	User data
7FFh	600h	Yes	Main 3	User data
801h	800h	No		Reserved
803h	802h	No		User metadata II
807h	804h	Yes	Spare 0	User metadata I
80Fh	808h	Yes	Spare 0	ECC for main/spare 0
811h	810h	No		Reserved
813h	812h	No		User metadata II
817h	814h	Yes	Spare 1	User metadata I
81Fh	818h	Yes	Spare 1	ECC for main/spare 1
821h	820h	No		Reserved
823h	822h	No		User metadata II
827h	824h	Yes	Spare 2	User metadata I
82Fh	828h	Yes	Spare 2	ECC for main/spare 2
831h	830h	No		User data
833h	832h	No		User metadata II
837h	834h	Yes	Spare 3	User metadata I
83Fh	838h	Yes	Spare 3	ECC for main/spare 3

Bad Block Information	ECC Parity	User Data (Metadata)
2 bytes	8 bytes	6 bytes

Figure: spare area mapping for common NAND flash

For the HERMES < 1.5, we decoded the partition tables of Qualcomm, and extract these partitions.

```
Partition Version: 3
Partition Number: 50
Partition       Start          Size           Start(int)      Size(int)
MIBIB           00000000       0000000a       0x0             0x140000
OEMINFO         0000000a       00000050       0x140000        0xa00000
SBL2            0000005a       0000000c       0xb40000        0x180000
SBL2BACKUP      00000066       0000000c       0xcc0000        0x180000
CONTROL         00000072       0000000c       0xe40000        0x180000
SECURITY        0000007e       0000000c       0xfc0000        0x180000
RPM             0000008a       0000000c       0x1140000       0x180000
RPMBACKUP       00000096       0000000c       0x12c0000       0x180000
EFS2            000000a2       00000058       0x1440000       0xb00000
EFS2_A          000000fa       00000058       0x1f40000       0xb00000
EFSBACKUP1      00000152       00000038       0x2a40000       0x700000
EFSBACKUP2      0000018a       00000038       0x3140000       0x700000
APPSBL          000001c2       0000000c       0x3840000       0x180000
APPSBL_A        000001ce       0000000c       0x39c0000       0x180000
APPS            000001da       00000040       0x3b40000       0x800000
APPS_A          0000021a       00000040       0x4340000       0x800000
MTCHUB          0000025a       00000018       0x4b40000       0x300000
USERDATA        00000272       00000030       0x4e40000       0x600000
```

Figure: system partitions of old HERMES

It's easy to extract some partitions, but some partitions are hard.

The spare area defined in the chip datasheet is just as recommended; it's not an enforced standard. The spare area is related to the NAND controller. The SLC has sub-pages. Each page has ECC, so we can analyze the regulation of the differences of the pages to figure out the OOB area distribution.

```
* Layout with ECC enabled:
*
* |----------------------|  |----------------------------------|
* |           xx.......yy|  |           *********xx.......yy|
* |    DATA   xx..ECC..yy|  |    DATA     **SPARE**xx..ECC..yy|
* |   (516)   xx.......yy|  |  (516-n*4)  **(n*4)**xx.......yy|
* |           xx.......yy|  |           *********xx.......yy|
* |----------------------|  |----------------------------------|
*     codeword 1,2..n-1              codeword n
*  <---(528/532 Bytes)-->   <-------(528/532 Bytes)--------->
```

Figure: OOB area distribution

The user data, applications and system partitions use the YaFFS filesystem.

YaFFS is designed for NAND and NOR Flash, and It has a wear-leveling feature. The NAND controller messes up the block order for longer life expectancy.

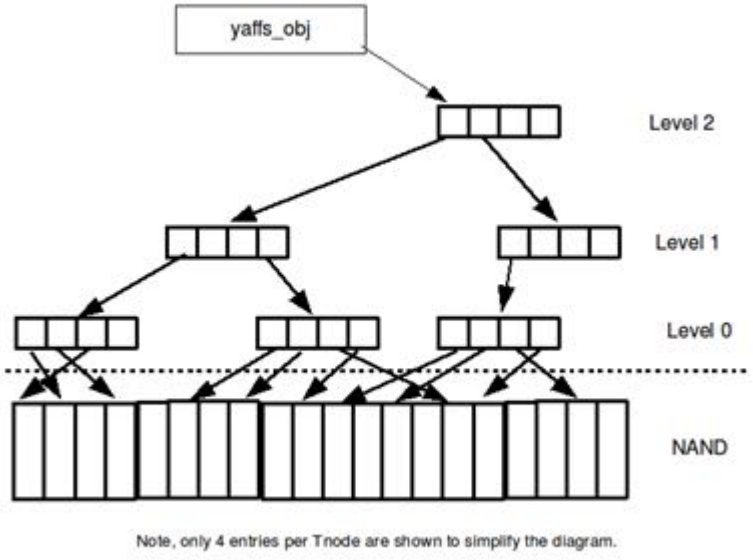


Figure: Yaffs blocks mapping

Skipping the ECC of OOB data, with only 16-bytes OOB data left for Yaffs, it allows us to recover the sequential NAND filesystem and extract files from it.

```
if obj_type == YAFFS_OBJECT_TYPE_DIRECTORY:
    file_path = root_path + get_path(obj_id_list, parent_id, file_name)
    if not os.path.exists(file_path):
        try:
            os.makedirs(file_path)
        except:
            pass
elif obj_type == YAFFS_OBJECT_TYPE_FILE:
    file_path = root_path + get_path(obj_id_list, parent_id, file_name)
    print(file_path, largest_index)
    if not os.path.exists(file_path):
        try:
            obj.writeVersion(largest_index, file_path)
        except:
            pass
```

Figure: codes for extracting Yaffs

Write a script to extract files from system partitions. It turns out to be a Linux system.

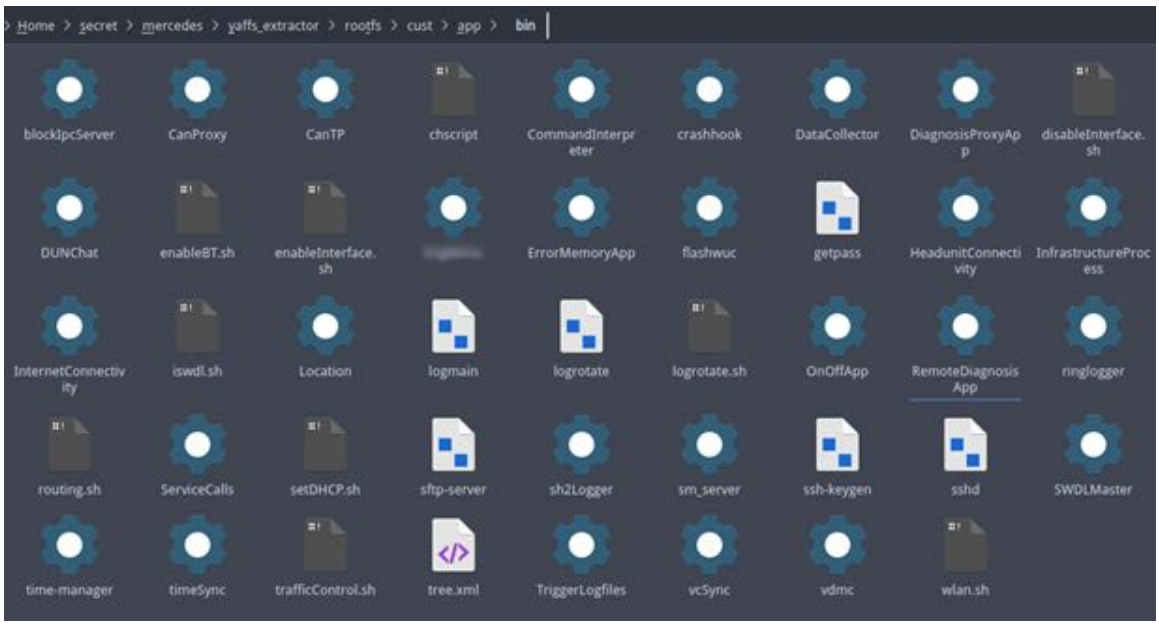


Figure: Applications of HERMES

For Hermes 1.5~2.1, the Cellular module is ME919, and the SoC is Hi6932, the USB cable had been removed from PCB.

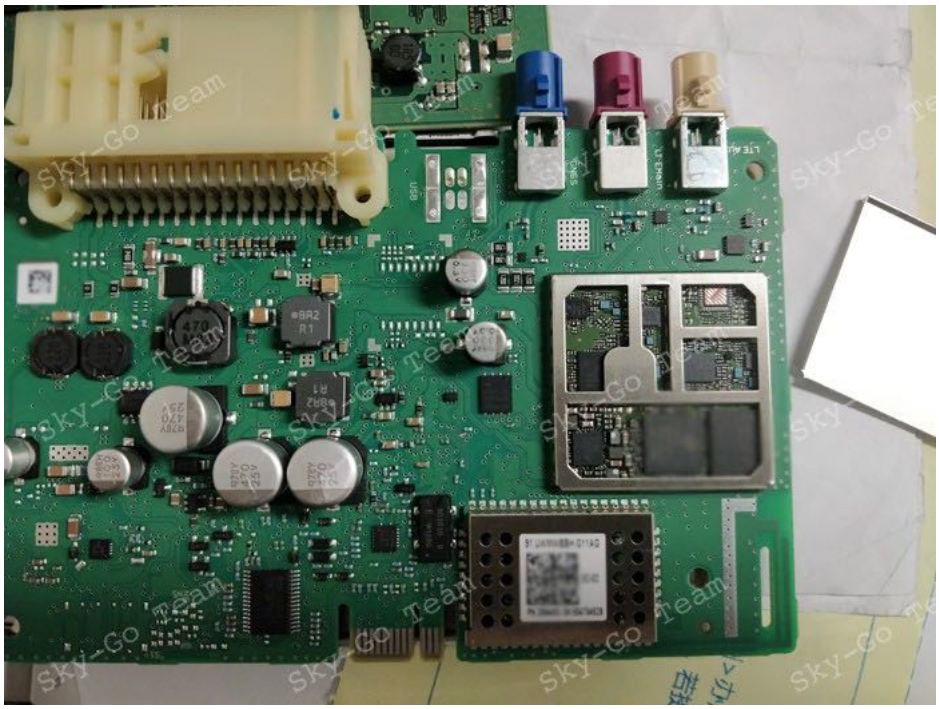


Figure: HERMES 1.5

The NAND Flash in a newer version of HERMES is using the BGA63 package.

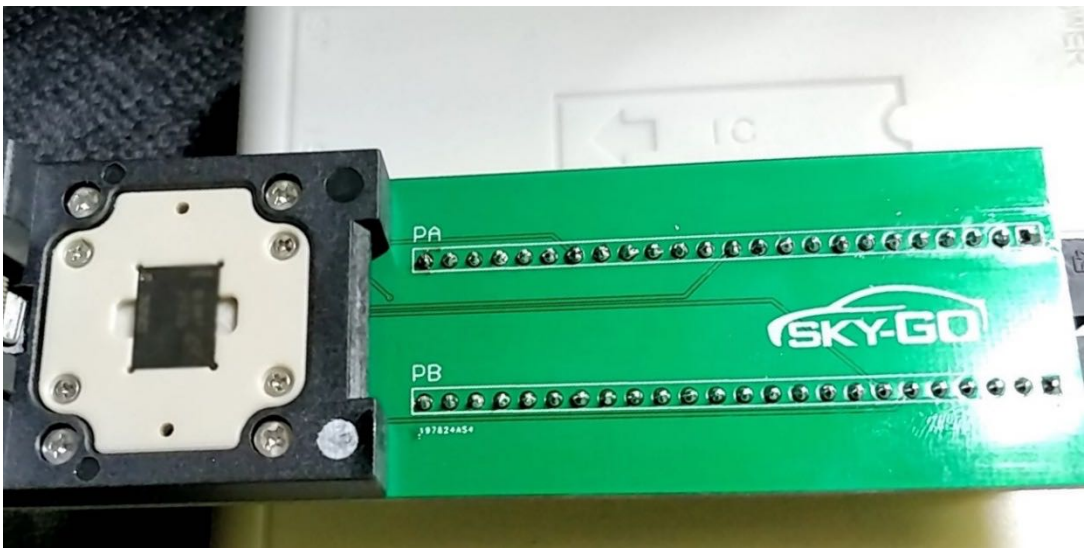


Figure: NAND flash adapter

Fortunately, we get rid of the soldering work because we have the Flash Adapter.

Then we decode the partition table for the ME919 as follows:

HISI Dumper: ptable 1.00 V7R22_TELEMATIC

name	offset	size	loadaddr	type	property
m3boot	0×00000000	0×00040000	0×00000000	IMAGE_M3BOOT	MTD
fastboot	0×0004[illegible]0000	0×00[illegible]0000	0×afcff000	IMAGE_FASTBOOT	MTD
fastbootbk	0×0004[illegible]0000	0×00[illegible]0000	0×afcff000	IMAGE_FASTBOOTBK	MTD
oeminfo	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_OEMINFO	MTD
nvbacklte	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVBACKLTE	Protected,MTD
nvbackltebk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVBACKLTEBK	Protected,MTD
nvdefault	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVFACTORY	MTD
nvimg	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVIMG	MTD
nvsys	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVDLD	MTD
nvdload	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVDLD	MTD
control	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_CONTROL	MTD
security	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_SECURITY	MTD
m3image	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_M3IMAGE	MTD
m3imagebk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_M3IMAGEBK	MTD
teeos	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_TEEOS	MTD
teeosbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_TEEOSBK	MTD
dts	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_DTS	MTD
dtsbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_DTSBK	MTD
hifi	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_HIFI	MTD
modem_fw	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_MODEM_FW	YAFFS,MTD
boot	0×0004[illegible]0000	0×00[illegible]0000	0×afdff000	IMAGE_KERNER	MTD
bootbk	0×0004[illegible]0000	0×00[illegible]0000	0×afdff000	IMAGE_KERNELBK	MTD
nvimgbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVIMGBK	MTD
nvsysbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVDLDBK	MTD
nvdloadbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_NVDLDBK	MTD
hifibk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_HIFIBK	MTD
modem_fwbk	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_MODEM_FWBK	YAFFS,MTD
system	0×0004[illegible]0000	0×00[illegible]0000	0×00000000	IMAGE_SYSTEM	YAFFS,MTD

Figure: System partitions of new HERMES

But things didn’t go smoother in research work. We found in some Brands of the chip have the bit-flipping. This problem affected the data we extracted.

Figure: bit-flipping error

The bit-flipping is a NAND Flash features. If the key jump instructions are affected by bit-flipping, our research may have headed in a wrong direction.

On the other side, there're many peripheral buses, thus we cannot set up a simulation environment by QEMU. If we have to debug the TCU client programs dynamically, we need to tamper the filesystem to get an interactive shell with ROOT privileges.

Because of the bit-flipping, if we writeback the wrong data we have read to the NAND flash, it will occur an unexpected error that the ECC algorithm will make the correct data to the wrong data.

Combining our previous research experience, we recovered the ECC algorithm for this NAND controller.

```
NandIPCommon nandSet = new NandIPV600();
int oobSize = 64;
int pageSize = PageType.Size2K.getPageSize();
FlashOption option = new FlashOption(ECCType.Ecc4bit,PageType.Size2K, oobSize);
byte[] pagebuf = new byte[0x800];
byte[] emptyPage = new byte[0x800];
int n;
while ((n=rawfileStream.read(pagebuf))!=-1){
    byte[] pageData = new byte[0x840];
    if (pagebuf != emptyPage) {
        System.arraycopy(pagebuf, srcPos: 0, pageData, destPos: 0, pageSize);
        Arrays.fill(pageData, pageSize, toIndex: pageSize+oobSize, (byte)0xFF);
        pageData = nandSet.genPageData(pageData, option, oobSize);
    }
    outfileStream.write(pageData);
}
```

Figure: Code for generating ECC

So, we can generate and decode the ECC data and remove the OOB area from a raw NAND file.

```
public class ECCBase {
    private static final int MAX_LFSR_BITS = 2048;
    private static final String POLY_[redacted]
    private static final String POLY_[redacted]
    private static final String POLY_[redacted]
    private static final String POLY_[redacted]
    private static final String POLY_[redacted]
    int lfsr_len;
    byte[] lfsr_poly = new byte[2048];
    byte[] lfsr_value = new byte[2048];
```

Figure: Polynomial code

We generate a raw NAND file with ECC from the NAND file without the OOB area. The comparing results between the old file are the same with the new NAND file we generated.

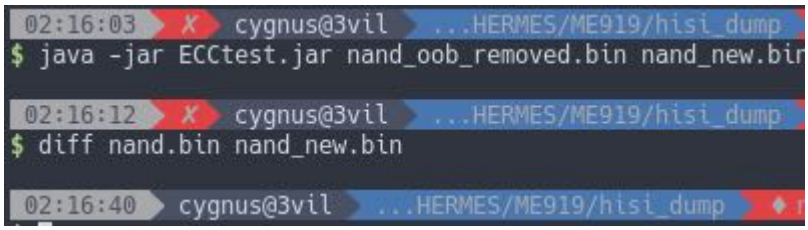


Figure: Correct ECC

Then we tamper the filesystem by adding an interactive shell with ROOT privileges. We found an engineer mode program for debugging the TCU system, with access to the CAN bus via operating the MCU. Thus, we can perform some operations for example, lock or unlock the doors.

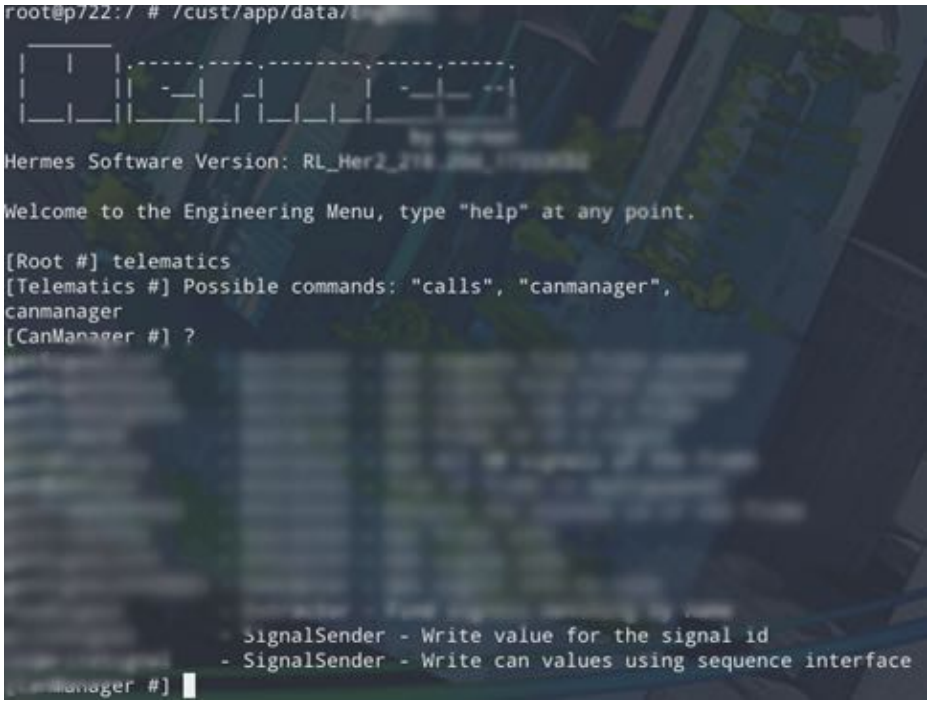


Figure: Engineering menu application

5.3. Client Certificates

TCU file systems stores the pkcs12 client certificates, passwords and CA certificates for the car backend server.

Figure: certificates and key pair

The files with suffix ".passwd" are the password files, encrypting with AES_256_CBC and the key hardcoded.

```
CEncryptionInterface::readEncrypted((int)&plain_text_passwd, (int)&xx);
std::string::operator=((int)xxflag, (int)&plain_text_passwd);
std::string::~string((std::string *)&plain_text_passwd);
std::string::~string((std::string *)&xx);
if ( std::string::compare(xxflag, (const char *)&unk_1F037) )
{
  v7 = (std::string *)(v2 + 20);
  if ( checkFile(regular_passwd, 0, 1) )
  {
    std::string::string((std::string *)&xx, (const std::string *)&regular_passwd);
    CEncryptionInterface::readEncrypted((int)&plain_text_passwd, (int)&xx);
    std::string::operator=((int)v7, (int)&plain_text_passwd);
    std::string::~string((std::string *)&plain_text_passwd);
    std::string::~string((std::string *)&xx);
```

Figure: Decryption code

```
str_cpy(&iv, (int)"[redacted]", (int)"");
AES_set_decrypt_key((int)"C9A2[redacted]", 256, (int)&key);
AES_cbc_encrypt(*in, (int)out, in[1] - *in, (int)&key, iv, 0);
```

Figure: Hardcode AES key

The key of the certificate is encrypted to a file, we can get the certificate key by compiling the decrypting tool with OpenSSL, obtaining the password of the certificate key. After decryption, the passwords of client certificate including ECE, AMS, and CHN region can be obtained

Chinese region certificate is using a week password.

5.4. Protocol Analysis

The HERMES-Backend protocol design doesn't seem vulnerable. Not only the ISP and SSL provide protection, but also the Mercedes's own secure communications make the MITM impossible.

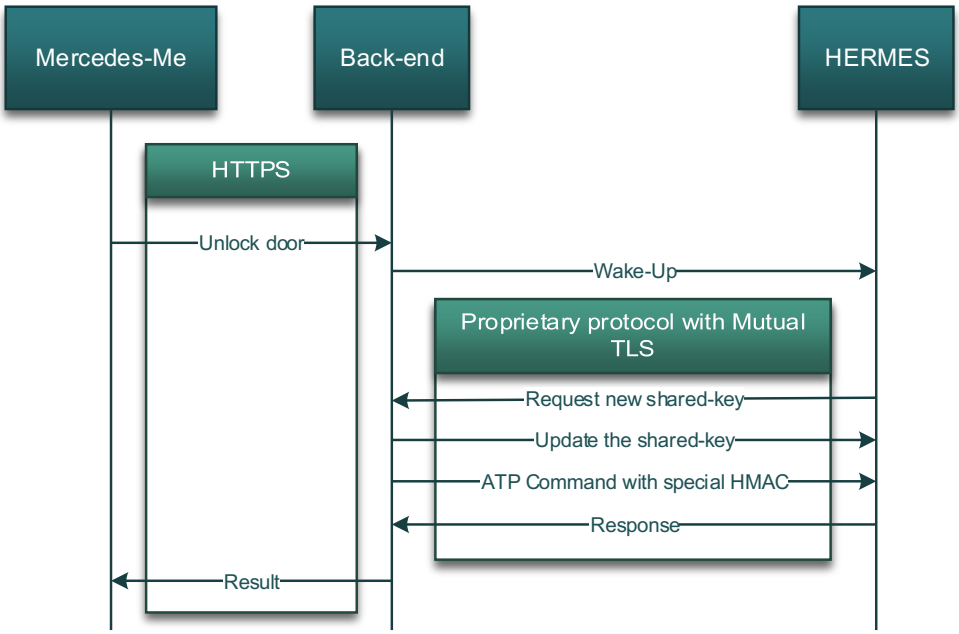


Figure: Communications of Telematics

1. The user controls the vehicle through "Mercedes ME" APP. The car control request is sent from the mobile phone to the server.

2. The backend server checks the validity of the request. If the car control request is valid, it is sent to the TCU of the corresponding vehicle by SMS. SMS is using a protocol called ATP. The ATP protocol uses a shared key to encrypt data fields and uses a hash algorithm for authentication. The encryption method is AES. Each time the TCU boots, it requests a new shared key from the key server using the HTTPS protocol. The protocol validates if there's a need for a new key and then proceeds with a process to get the new key, and both TCU and backend databases store the shared-key. The control message is bound to be unforgeable through the shared-key and digest fields.

```
AuthData        struc ; (sizeof=0x25, mappedto_129)
                                        ; XREF: UnpackATPMessage/r
Length          DCD ?
securityLevel   DCD ?
digest_algorithm DCD ?
field_C         DCD ?
digest_length   DCB ?
digest_content  DCB 20 dup(?)
AuthData        ends
```

Figure: Message struct

3. After the TCU receiving the SMS, it uses the shared key to decrypt the SMS data field and verify its validity. TCU sends actual control instructions via CAN-D.

4. A few moments later, the TCU receives a response message from the EIS or other ECU, which contains the execution result of the control instruction. As soon as TCU receives these results, TCU uses HTTPS to feedback the execution result to the ATP server.

The security mechanism of this protocol has been upgraded several times and is much more secure than most of car manufacturers.

5.5. Access the Backend

Car Backend is the core of Connected Cars. As long as Car Backends' services can be accessed externally, it means that car backend is at risk of being attacked. The vehicles connecting to this Car Backend are in danger, too. So, our next step is to try to access Car Backend.

For accessing the APN networks of backend, one possibility would be using the e-sim of car-parts since the sim account wouldn't log out automatically.

After tearing down this eSIM, we put it into the 4G router.

Figure: eSIM

Applying the APN information obtained before, dial-up to access the Internet.

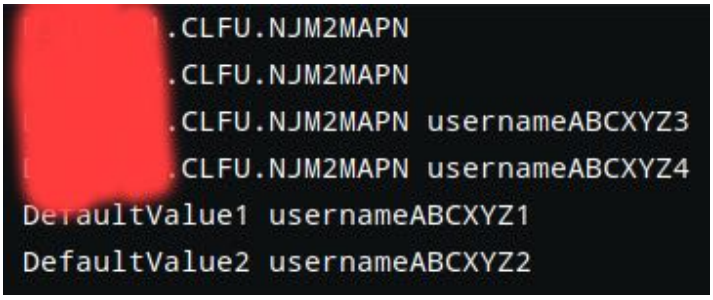


Figure: APNs

The security strategy of ISP detects the relationship between ICCID and IMEI; if changed, the SIM account would be frozen. In order to access the 4G network, we modified the IMEI of the 4G router and configured the APN information to the 4G router.

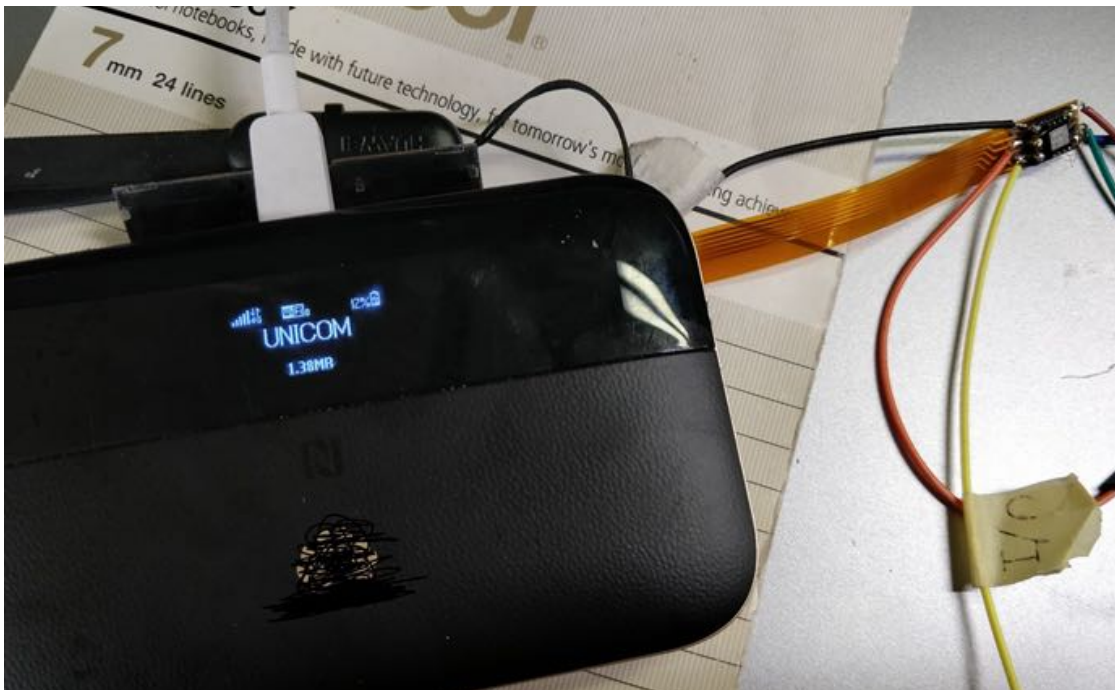


Figure: E-SIM Jump wiring

The interface IP address belongs to the APN the intranet. It can be hard to trace the attack source.

```
eth_x     Link encap:Ethernet  HWaddr 58:02:03:04:05:06
          inet addr:10.232.231.5  Bcast:10.255.255.255  Mask:255.25
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:288098 errors:0 dropped:0 overruns:0 frame:0
          TX packets:238666 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:322761113 (307.8 MiB)  TX bytes:24312369 (23.1 M
```

Figure: Access the ISP intranet

We can obtain an IPv4 address of the ISP intranet (Not the belong to the Car Backend network).

A device of CHN region has three certificates of three different regions, which has been deployed to most of the backends, considered as an authentication mechanism for all of the backend servers.

You have certificates from these organizations that identify you

Certificate Name	Security Device	Serial Number	Expires On
▾ DAIMLER			
[redacted]0060001	Software Security Device	23:BC:54:5A:00:00:00:00:6B:E1	November 18, 2039
[redacted]0000008	Software Security Device	10:FC:14:52:00:00:00:00:01:55	February 5, 2036
▾ Daimler AG			
[redacted]0000001	Software Security Device	16:0E:C9:B4:00:00:00:00:00:0E	August 4, 2040

Figure: 3 different certificates

Once the certificate is obtained, certain access to backend servers was possible by utilizing the certificate.

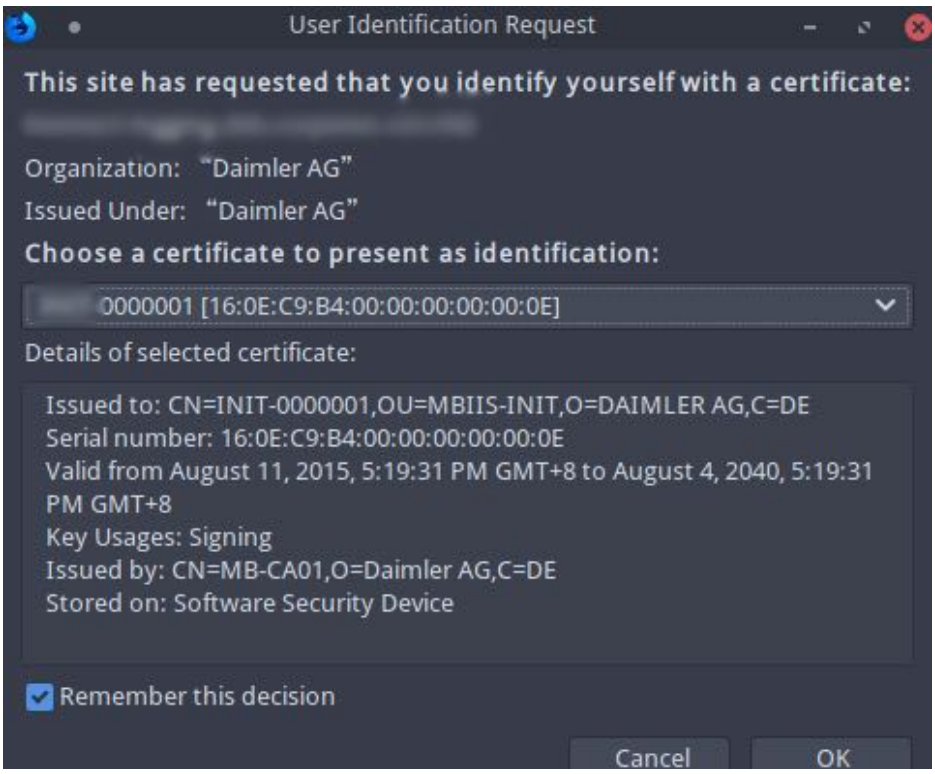


Figure: the basic information of cert

5.6. Social Plugins SSRF

We can scan the QR-code on the social plugins of the Head-Unit, which is actually a web application. We can bind the social accounts to the car. A SSRF vulnerability occurred in the backend service, as the image provider failed to filter the parameters we input.

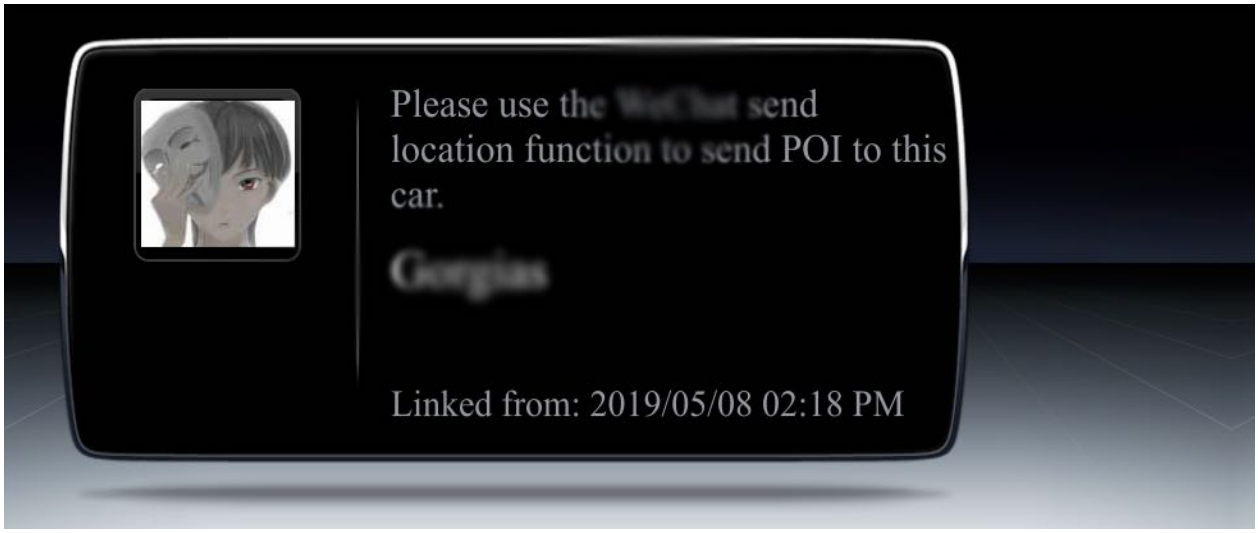


Figure: Social plugin page

The plugin developers have less consideration of the requested URL. For example, if we submit a local URL to the image provider, it'll return the contents we requested.

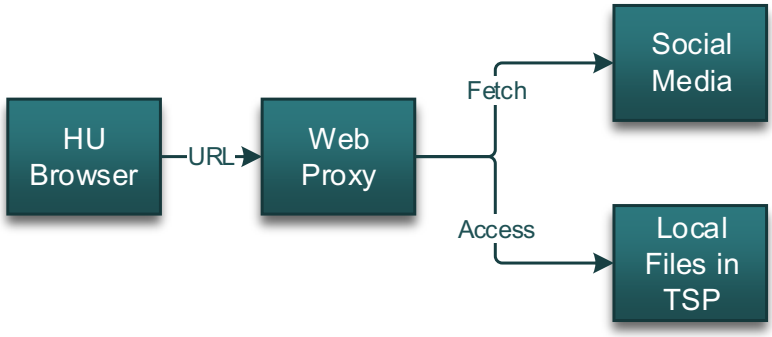


Figure: SSRF data stream

Figure: System file leaks

6. Vulnerabilities List

No.	Description	Components	Reference
1	Reserved due to security concerns.	HERMES	CVE-2019-19556 CVE-2019-19560 CVE-2019-19562
2		HERMES	Reserved
3		HERMES	CVE-2019-19557 CVE-2019-19561 CVE-2019-19563
4		HERMES	Reserved
5		HERMES	Reserved
6		HERMES	Reserved
7		Operations	N/A but fixed
8		Operations	N/A but fixed
9		Head-Unit	Reserved
10		Backend	N/A but fixed
11		Backend	CVE-2019-19558
12		Backend	N/A but fixed
13		Backend	N/A but fixed
14		Backend	N/A but fixed
15		Backend	N/A but fixed
16		Backend	N/A but fixed
17		Backend & ISP	N/A but fixed
18		Backend	N/A but fixed
19		Backend	N/A but fixed

Table: Vulnerabilities found in this Research

7. Disclosure Timeline

Sky-Go Team follows the "Responsible Disclosure" and work together with Mercedes-Benz Security Team on vulnerability fixing.

In the joint work for fixing the vulnerabilities Sky-Go Team shared valuable information on the findings. All vulnerabilities that allowed access were promptly fixed.

Aug 21, 2019: The findings reported to Daimler AG (360)

Aug 23, 2019: The services shutdown: preventing further effect on MB cars (Mercedes-Benz)

Aug 26, 2019: Initial fix (Mercedes-Benz)

Sep 12, 2019: All access vulnerabilities fixed (Mercedes-Benz)

Oct 23, 2019: Joint workshop (360 & Mercedes-Benz)

Feb 28, 2020: RSA Conference Publication (360 & Mercedes-Benz)

July 20, 2020: Research Report Publication (360)

8. Conclusion

In this report, we describe how to do a security research on connected cars. Based on Mercedes-Benz case, we show that how we build a testbench and what analysis works we have done. Then we disclosed the vulnerabilities with limited detail due to security concerns.

In the joint work for fixing the vulnerabilities Sky-Go Team shared valuable information on the findings. All vulnerabilities that allowed access were promptly fixed.

During the research and joint workshop, we see so many security designs in Mercedes-Benz Connected Cars and these designs are protecting the cars from various attacks.

The capability of a car company to work jointly with researchers contributes to the overall security of our cars.

8.1. Data Protection

The Head-Unit adopts WinCE Automotive 7 as its operating system, and less security research has been carried out comparing to the widely used operating systems such as Linux, QNX, and Android, that many privilege escalation vulnerabilities have been found.

Daimler has designed a mutual authentication proxy for pipes, by AES256 and HMACSHA256, and they also have many security countermeasures are implemented in the architecture, appearing in the OS, communication, data protection and secure boot.

The security considerations of Daimler make the cars hard to be attacked; however, the shortcomings exist. By fully utilizing them, it allowed us to dump the firmware from NAND flash, decrypted the certificates, do some reverse engineering work of the communication protocols.

8.2. Lifecycle Management

Basically, car manufacturer should have their own lifecycle management system which can monitor the state of cars and components. Once cars or components are decommissioned, lifecycle management should restrict their functionalities. However, it is very hard to monitor the state of cars and components for the car manufacturer, since decommissioning can occur without their knowledge and in such a way that decommissioning procedures cannot be enforced.

8.3. Anti-Theft Protection

Anti-theft Protection does make research more difficult. It’s possible to stop your research when you are building the testbench. Such as the FBS4 implemented on the key programming, it makes the key numerical and robust enough in encryption.

Even if the arbitrary CAN message sending privilege is acquired, the start-up of the car is also prohibited.

8.4. Communication Security

The structure is quite a scalability and well-formed, with HTTPS to authenticate mutually, an elaborated defense design for Tier 4, independent keys for each car which can be replaced at any time, replay attack prevented, isolation of core service and telematics, the remote startup is prohibited in default.

8.5. Intranet Security

Meanwhile, shortcomings also showed in the car backend. Their security depends on is the client certification and had some weaknesses in internal mechanisms. Besides, third-party suppliers have caused some other

security problems. Make every backend component secure all the time is hard. No company can make this perfect.

At the early attack stage, it comes some abnormal logs. If automakers monitoring these checkpoint and warning in real-time, they are in an advantageous position.

Annex A
(Informative)
Letter from Daimler Group

October 18, 2019

ATT:

Dear Mr. Yan,

Dear colleagues from Sky-Go team,

With this letter, we would like to thank you for the excellent cooperation and collaboration between our companies. Proactive identification of vulnerabilities is essential to protect our customers and their vehicles. Therefore, Mercedes-Benz has always attached great importance to the relevant work, and values the support and collaboration from industry experts such as Qihoo 360 with the Sky-Go team.

During our cooperation, we have always experienced your company as a valuable and inspiring partner. This applies especially to the field of vehicle cyber security led by your Sky-Go team.

Taking this opportunity, we would like to emphasize that we highly appreciate your expertise and the effort you have put to help further secure our vehicles and we look forward to our joint workshop and further discussions.

With my warmest regards,

Adi Ofek

CarIT Security Mandate, Mercedes-Benz Cars

Annex B
(Informative)
Sky-Go Team

Figure: Mercedes-Benz Research Project Members

Sky-Go Team is a professional connected car security research team from 360 Group. Since we established in 2014, we have done many research cases, such as Tesla Telematics System, Tesla Autopilot System, BYD Telematics System. Sky-Go Team provides security evaluation service, consulting service and product for the car industry, and we also attend standardization work, such as ISO, ITU-T, China National Standards (GB) and industry standards.