The Making of the Kosher Phone

RECon - 2014

Assaf Nativ

- Presented here in 2010 about Memory Analysis techniques
  Youtube movie of it is on my blog
- A security researcher

Disclaimers

- Everything presented here was done by me and many other people
- I now work for the security startup Sentinel
- I’m not a religious person

The story

About 4 years ago

By the end of this talk

- You will be able to make a Kosher firmware.
- But you are not going to...

What can feature phone do?
(AKA Dumb phone)

- Can:
  - Make phone calls
  - SMS
  - MMS
  - Calendar
  - FM Radio
  - Play annoying ringtones
  - GPRS or 3G using dis-functional web browser
  - Bluetooth
  - ~2MPixel camera
  - Utilities and Games
- On some newer models:
  - Limited Facebook, Twitter and WhatsApp

What a Kosher phone can do
(AKA a phone that suffers from severe retardation)

- Can:
  - Make phone calls
  - ~~SMS~~ 8==D
  - ~~MMS~~
  - Calendar
  - ~~FM Radio~~
  - *Play annoying hasidic ringtones*
  - ~~GPRS or 3G using dis-functional web browser~~
  - *Bluetooth to earpiece only*
  - ~~~2MPixel camera~~
  - Utilities and ~~Games~~
- On some newer models:
  - ~~Limited Facebook, Twitter and WhatsApp~~

Kosher Phone's new app

The Jewish calendar

Fun fact about the Jewish calendar:

- Follows both the moon and the sun
- Every year has either 12 or 13 months
- Day is of no fixed length
- Inaccurate by 1 day every 216 years
- Strange

Planning ahead

1. Choose a phone
2. Get the company who makes it remove some features
3. Sell it for more money

Nokia, connecting people (sort of)

I needed a phone that is:

- Cheap
- Reliable
- In mass production

Choose a model

Nokia software series

S30

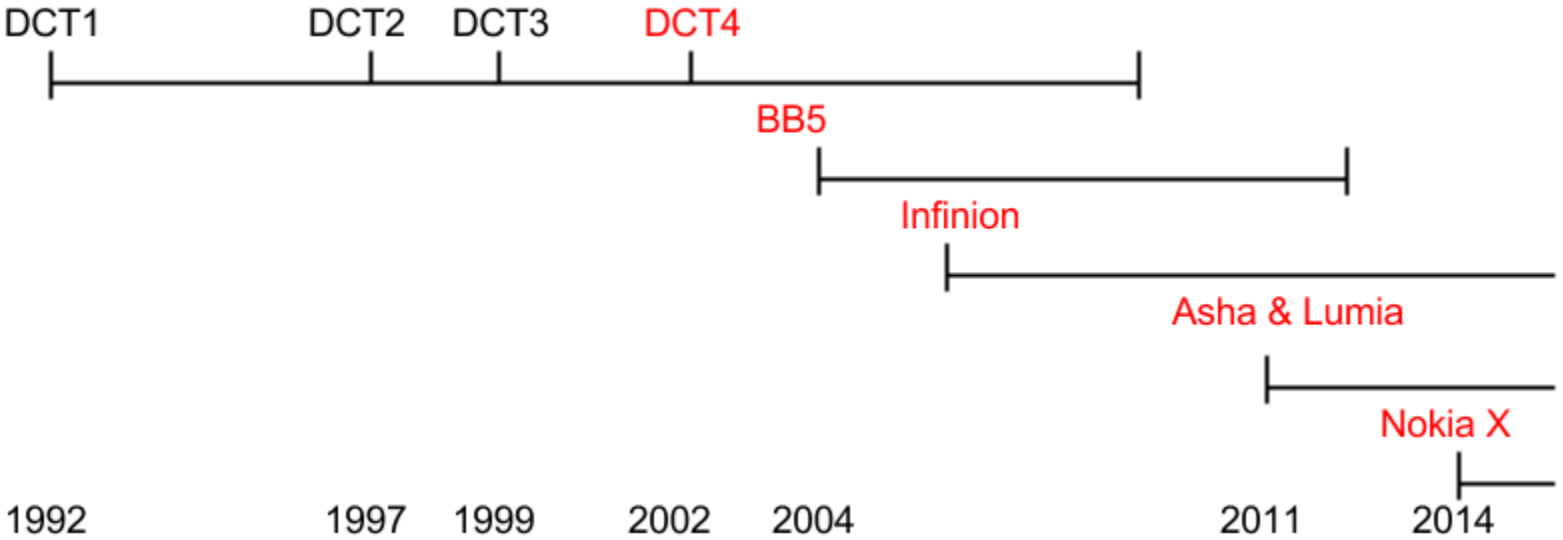
S40

Nokia software series

Symbian S60 (Died on 2010)

Symbian^3

Hardware versions



Hardware Samples

DCT1

DCT3

DCT4

BB5

Asha

First Kosher phones

- Nokia 1208, 2680, 2720
- All DCT4
- All S40

Nokia 1208 - Ugly Candybar

Nokia 2680 - Ugly Slider

Nokia 2720 - Super ugly Clamshell

Ask Nokia

Plz Nokia,

I can haz no Interwebz and camera and file transfer?

No, but...

You can do whatever helps you sell more of our phones

Patching

DCT4+ firmware

- Flashable
  - OS (MCUSW)
  - Localization strings and gfx (PPM)
  - General purpose file system - Operator FAT16 (CNT / IMAGE)
- SecureROM
  - ?

Why would Nokia care

Patching the MCUSW might allow you to:

- Change the IMEI
- Break the SIMlock
- And more...

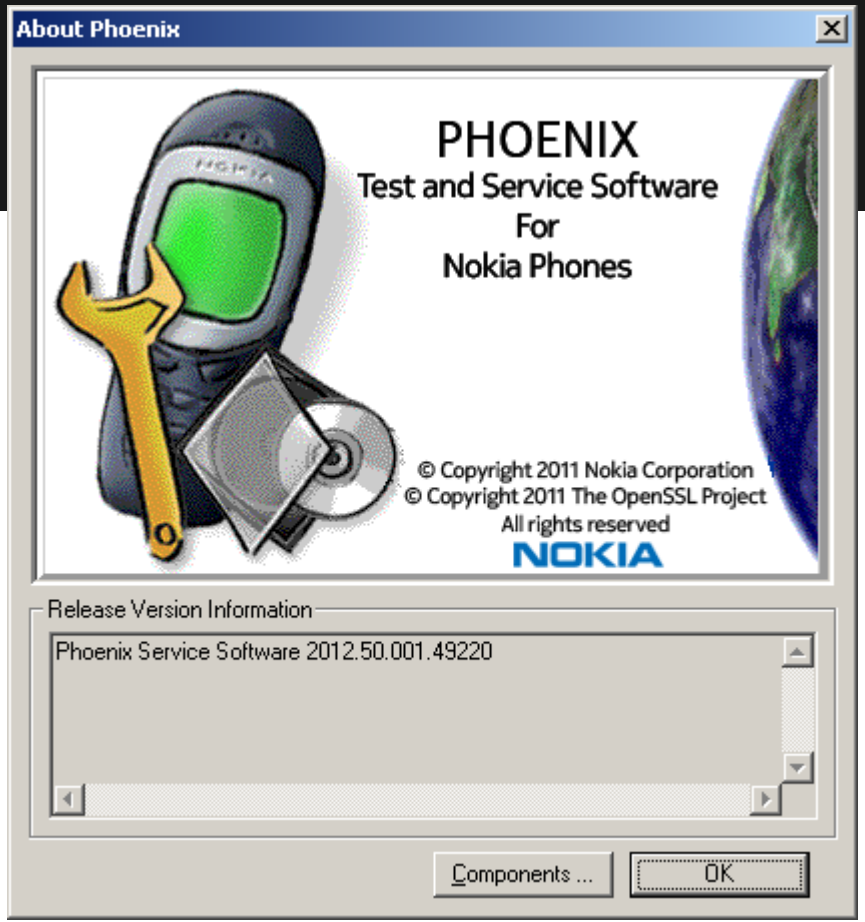
How to obtain the files

Just  it

How to flash a phone

3 options

Phoenix


About Phoenix
PHOENIX
Test and Service Software
For
Nokia Phones
© Copyright 2011 Nokia Corporation
© Copyright 2011 The OpenSSL Project
All rights reserved
NOKIA
Release Version Information
Phoenix Service Software 2012.50.001.49220
Components ...
OK

A box


TURBO Flasher
WWW.GU-TEK.EU


cyclone
COMPUTER
AC


SERVICE CABLE
PW
TX
RX
USB
j.o.p.


mt-box
www.mtbox.org
POWERED BY GRIFFIN TEAM
MT-BOX


www.z3x-team.com


INFINITY BOX


SPT BOX
WWW.SPTBOX.COM

And the right cable

FBus - Connector

USB

At this point

- A phone
- Firmware files
- A way to flash firmwares

File format - First layer

After a short header:

- 1 Byte: Type (Always 0x14)
- 4 Bytes: Address
- 3 Bytes: Length
- 1 Byte: Header checksum
- 1 Byte: Data Xor checksum

File format - 2nd layer

```
01000000  AD 7E B6 1B 23 10 03 40 C6 05 E4 01 20 A2 00 00   .~¶.#..@ֶ.ה. ¢..
01000010  00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF   ...........
01000020  FF FF FF FF F8 1F BD FA 50 65 61 4B FF FF FF FF   ת½.ךPeaK
01000030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000060  FF FF FF FF FF FF FF FF FF FF FF FF 94 72 48 92   ″rH′
01000070  A6 87 2A FE 00 02 00 00 01 00 01 00 00 00 00 00   ¦‡*.............
01000080  00 00 00 00 AB ED A4 43 5C 0D 53 A3 BD 70 EC 37   ....«םפC\.S£½p7ל
01000090  D1 AE CD 20 AE 54 47 F7 59 B4 A2 36 B4 85 BB 2B   ®ֽ ®TGקY´¢6´…»+
010000A0  B3 62 22 05 2E 16 13 1E C6 EE F2 2F AC CF CC 11   ³b".....ֶמע/¬ֿ.
010000B0  50 C8 B9 82 FE BC 8B C7 0E 58 91 9D 32 28 E8 B3   Pָ,¼<ַ.X`.2(³ט
010000C0  D4 1D AF 3E ED 0C 50 AB E0 F9 E5 09 69 D7 33 CE   ם<¯.ו.P«אשו.i׳3′
010000D0  62 CC D1 E2 3B DB 77 1E 64 7E AE 8A D4 AA BE CE   bגֹ;w.d~®.״¾×וו
010000E0  97 9E 24 23 40 05 9A 1C A0 37 41 30 58 9D 2A 3D   —.$#@...  7A0X.*=
010000F0  41 F5 85 AF 67 A1 42 60 02 8E E9 59 8C BE 43 F5   Aץ…¯g¡B`..יY.¾Cץ
01000100  56 4D EE F6 55 C3 DC B1 DB 14 72 74 43 A5 47 8F   VMצמU±ֳ.rtC¥G.
```

Obfuscation

- Starting from offset 0x84
- There is a lot of pieces of information about it spread around GSM forums

Thema: Merry X-Mas: DCT4Crypter

25-12-2005, 10:18

g3gg0
[RevEngineer]
Super Moderator

Dabei seit: Aug 2002
Ort: bavaria
Beiträge: 5.295

Merry X-Mas: DCT4Crypter

Hiho,

anbei ne kleinigkeit zu weihnachten :)
ich hoffe, die gefaellt euch.

[EDIT: Fixed CrypterX encryption problem]

Code:

```
          ----------------------------------
                      DCT4Crypter
          ----------------------------------
                    by nok5rev & g3gg0

 Yes, this package contains the necessary routines and
 even some apps to decrypt DCT4 FlashFiles and also to
 encrypt again after you applied some changes. We must
 admit, this stuff is not "hot" anymore - it was coded
 in about 2 months between 01/2004 and 03/2004. That's
 now nearly 2 years. But it still should be a somewhat
 interesting X-Mas present for all the GSM-Modders out
 there...


 Why this was done?
-----------------------
 Why? I think it was just fun :)
 But i dont remember anymore who of us had the idea
 to start analyzing the encryption algorithm.
 I just remember, we both suddenly sat in front of
 many bits (really MANY!) and stared at them to
 find out how the data was encrypted.

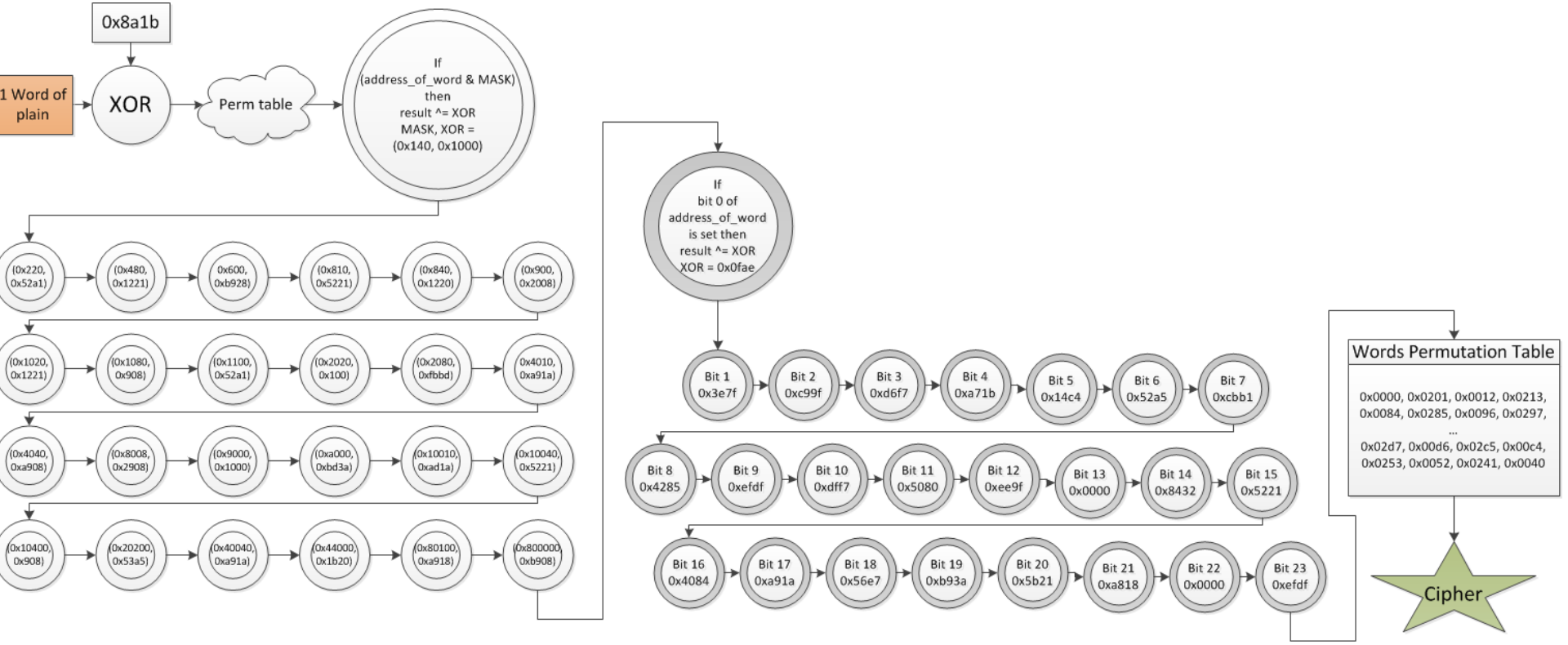
 How this was done?
-----------------------
 Heh, just open your notepad.exe, paste some 100
 lines of 11001001 10101010 11100100 11100001...
 and you know what we've done in these 2 months ;)
 We didnt have any access to neither the flash device,
```

Angehängte Dateien
Merry Xmas (fixed).zip (44,5 KB, 170x aufgerufen)

G3gg0 and nok5rev work

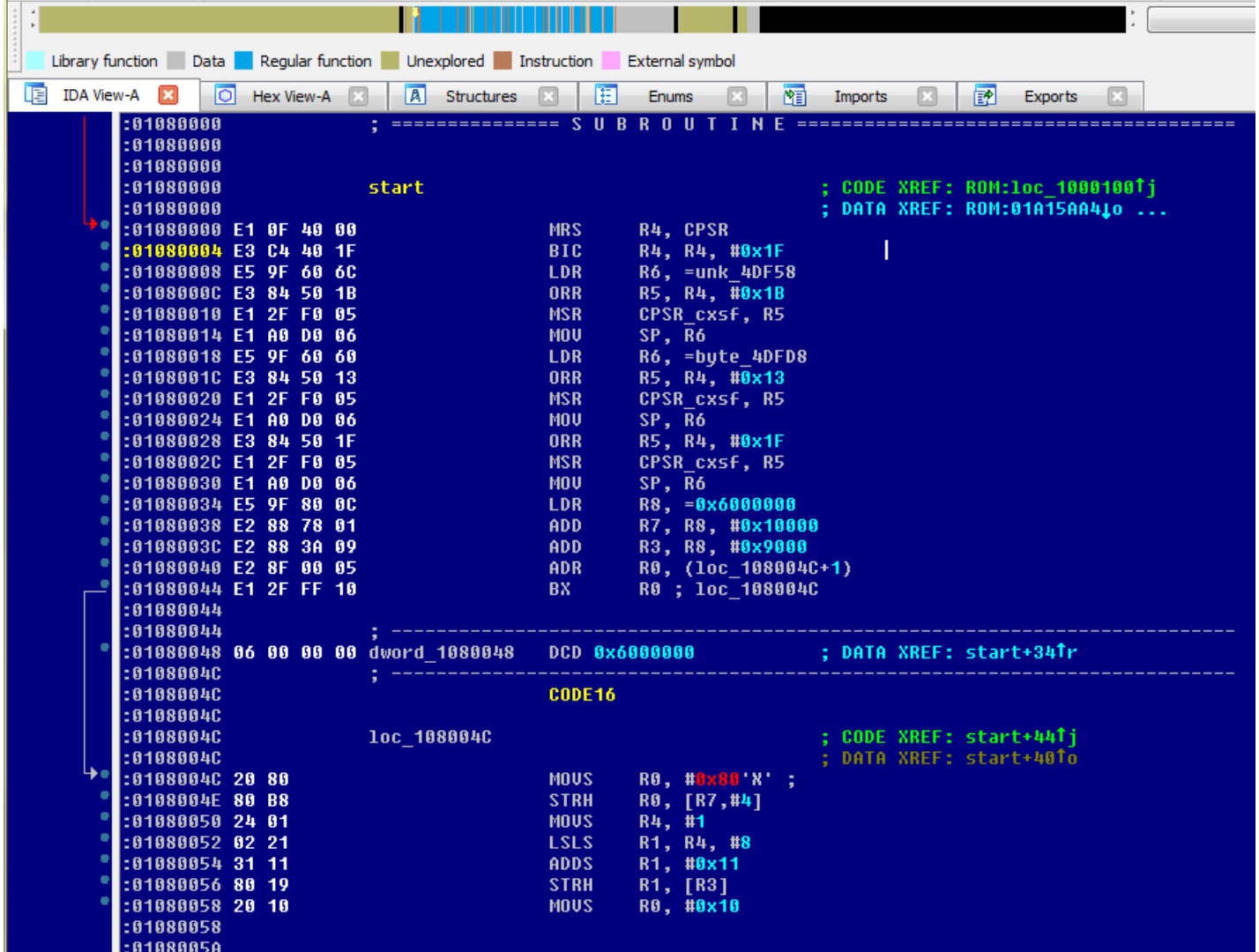
They reversed the obfuscation

- Just because it was fun
- They simply stared at the bits for 2 months until it made sense
- They published it for free :)


0x8a1b
1 Word of plain
XOR
Perm table
If (address_of_word & MASK) then result ^= XOR MASK, XOR = (0x140, 0x1000)
(0x220, 0x52a1)
(0x480, 0x1221)
0x600, 0xb928)
(0x810, 0x5221)
(0x840, 0x1220)
(0x900, 0x2008)
(0x1020, 0x1221)
(0x1080, 0x908)
(0x1100, 0x52a1)
(0x2020, 0x100)
(0x2080, 0xfbbd)
0x4010, 0xa91a)
(0x4040, 0xa908)
(0x8008, 0x2908)
(0x9000, 0x1000)
(0xa000, 0xbd3a)
(0x10010, 0xad1a)
(0x10040, 0x5221)
(0x10400, 0x908)
(0x20200, 0x53a5)
(0x40040, 0xa91a)
(0x44000, 0x1b20)
(0x80100, 0xa918)
(0x800000, 0xb908)
If bit 0 of address_of_word is set then result ^= XOR XOR = 0x0fae
Bit 1 0x3e7f
Bit 2 0xc99f
Bit 3 0xd6f7
Bit 4 0xa71b
Bit 5 0x14c4
Bit 6 0x52a5
Bit 7 0xcbb1
Bit 8 0x4285
Bit 9 0xefdf
Bit 10 0xdff7
Bit 11 0x5080
Bit 12 0xee9f
Bit 13 0x0000
Bit 14 0x8432
Bit 15 0x5221
Bit 16 0x4084
Bit 17 0xa91a
Bit 18 0x56e7
Bit 19 0xb93a
Bit 20 0x5b21
Bit 21 0xa818
Bit 22 0x0000
Bit 23 0xefdf
Words Permutation Table
0x0000, 0x0201, 0x0012, 0x0213, 0x0084, 0x0285, 0x0096, 0x0297, ... 0x02d7, 0x00d6, 0x02c5, 0x00c4, 0x0253, 0x0052, 0x0241, 0x0040
Cipher

File format - 3rd layer

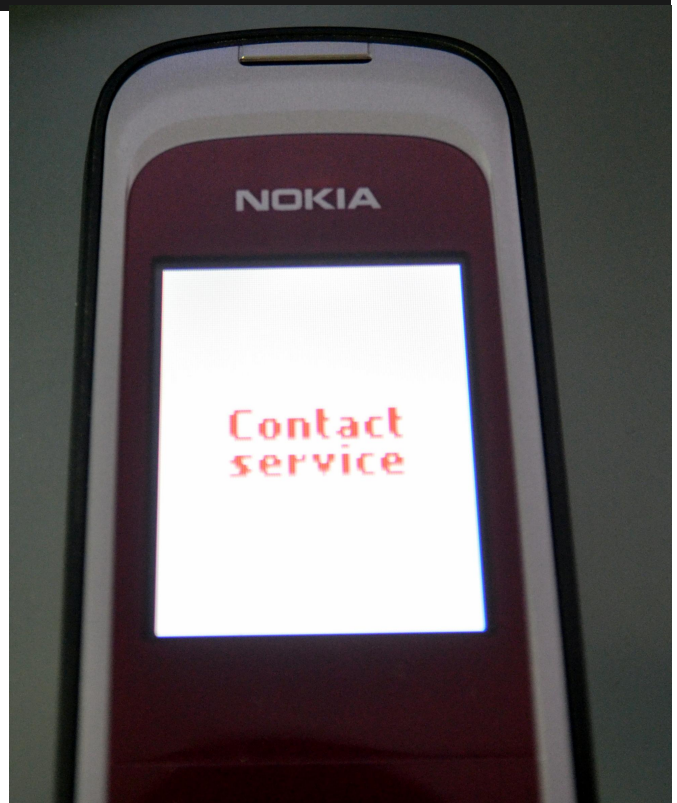
```
01000000  AD 7E B6 1B 23 10 03 40 C6 05 E4 01 20 A2 00 00  .~¶.#..@.ה. ¢..
01000010  00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF  ...........
01000020  FF FF FF FF F8 1F BD FA 50 65 61 4B FF FF FF FF  ת½.רPeaK
01000030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01000060  FF FF FF FF FF FF FF FF FF FF FF FF 94 72 48 92  ”rH’
01000070  A6 87 2A FE 00 02 00 00 01 00 01 00 00 00 00 00  ¦‡*.............
01000080  00 00 00 00 FF FF FF FF FF FF FF FF 01 CE 00 00  .....▪..
01000090  03 00 00 00 00 04 CC A2 00 04 CC A3 FF FF FF FF  ......¢..£
010000A0  00 00 F1 EF 89 33 EB 2D 1F 09 3B DA C7 C0 3D 9F  ..כ‰3ן-..;╪.
010000B0  BB D3 29 98 01 C8 BC B0 06 6E A8 11 0E D1 69 67  »:)˜.¼°.n¨..ig
010000C0  A4 A3 9A A5 BF 7B 27 5A E6 C7 61 2D F7 B8 70 9C  ₪£.¥¿{'Zיa-ק.p.
010000D0  D4 1C 09 96 AF 5B F2 05 20 92 49 DF D5 0B FC DE  ע]¯-..וו. ’Iוי.
010000E0  A8 30 B7 39 34 59 13 7D E7 BD 72 3F C7 CF B3 5A  ¨0·94Y.}ת½r?³Z
010000F0  60 2C 5E 7D 63 17 56 C4 9F 6C C5 1A 01 BF B7 DF  `,^}c.V.l..¿·
01000100  EA 01 FF BE 00 FE 6A 84 EA 50 20 20 20 20 6A 04  .¾.רj„רP    j.
```


Library function
Data
Regular function
Unexplored
Instruction
External symbol
IDA View-A
Hex View-A
Structures
Enums
Imports
Exports
:01080000 ; ============== S U B R O U T I N E =======================================
:01080000
:01080000
:01080000 start ; CODE XREF: ROM:loc_1000100↑j
:01080000 ; DATA XREF: ROM:01A15AA4↓o ...
:01080000 E1 0F 40 00 MRS R4, CPSR
:01080004 E3 C4 40 1F BIC R4, R4, #0x1F
:01080008 E5 9F 60 6C LDR R6, =unk_4DF58
:0108000C E3 84 50 1B ORR R5, R4, #0x1B
:01080010 E1 2F F0 05 MSR CPSR_cxsf, R5
:01080014 E1 A0 D0 06 MOV SP, R6
:01080018 E5 9F 60 60 LDR R6, =byte_4DFD8
:0108001C E3 84 50 13 ORR R5, R4, #0x13
:01080020 E1 2F F0 05 MSR CPSR_cxsf, R5
:01080024 E1 A0 D0 06 MOV SP, R6
:01080028 E3 84 50 1F ORR R5, R4, #0x1F
:0108002C E1 2F F0 05 MSR CPSR_cxsf, R5
:01080030 E1 A0 D0 06 MOV SP, R6
:01080034 E5 9F 80 0C LDR R8, =0x6000000
:01080038 E2 88 78 01 ADD R7, R8, #0x10000
:0108003C E2 88 3A 09 ADD R3, R8, #0x9000
:01080040 E2 8F 00 05 ADR R0, (loc_108004C+1)
:01080044 E1 2F FF 10 BX R0 ; loc_108004C
:01080044
:01080044 ; ---
:01080048 06 00 00 00 dword_1080048 DCD 0x6000000 ; DATA XREF: start+34↑r
:0108004C ; ---
:0108004C CODE16
:0108004C
:0108004C loc_108004C ; CODE XREF: start+44↑j
:0108004C ; DATA XREF: start+40↑o
:0108004C 20 80 MOVS R0, #0x80'X' ;
:0108004E 80 B8 STRH R0, [R7,#4]
:01080050 24 01 MOVS R4, #1
:01080052 02 21 LSLS R1, R4, #8
:01080054 31 11 ADDS R1, #0x11
:01080056 80 19 STRH R1, [R3]
:01080058 20 10 MOVS R0, #0x10
:01080058
:0108005A

Attempt to patch

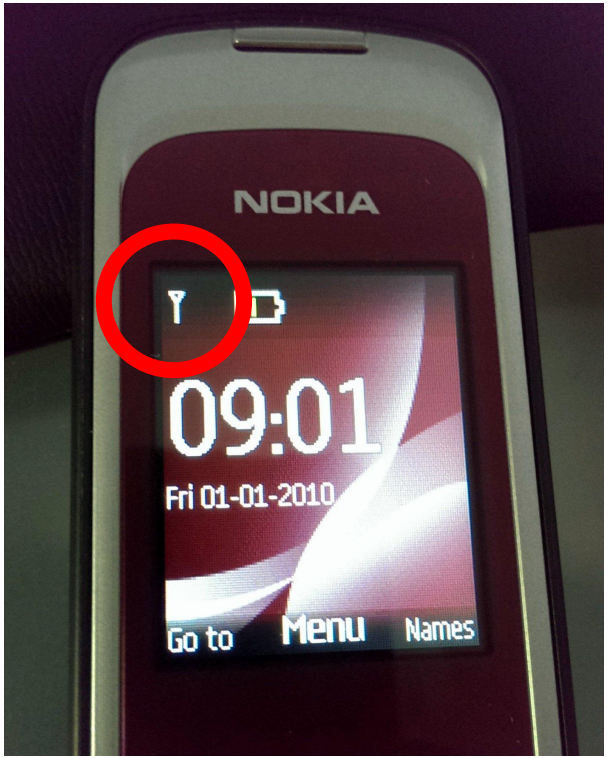
Error type 1

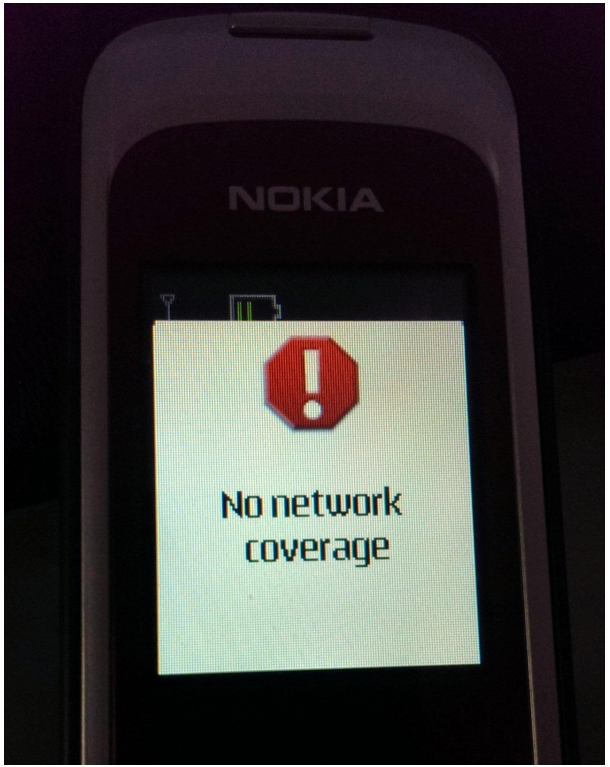
Contact service



Error type 2

No signal

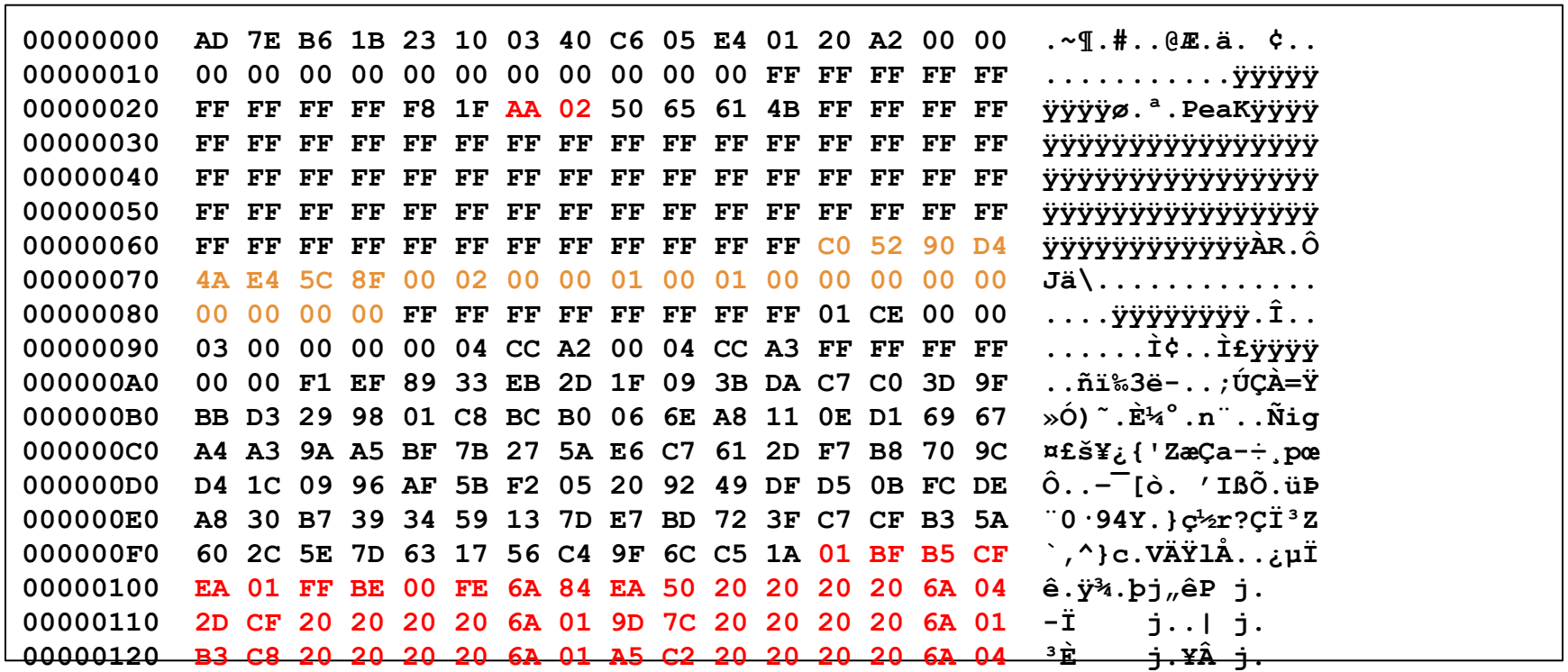




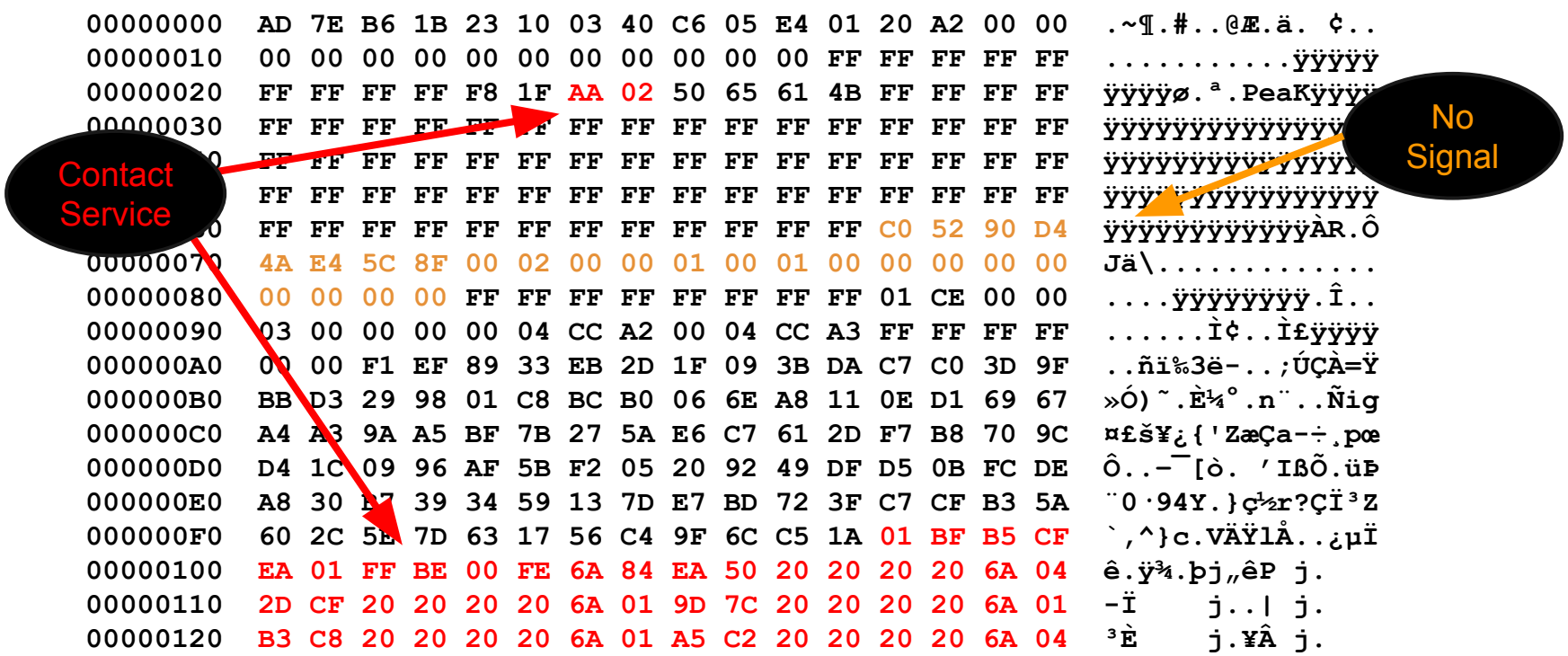
2 types of errors

1. Contact Service
2. No Signal

Patching doesn’t work map



Patching doesn’t work map

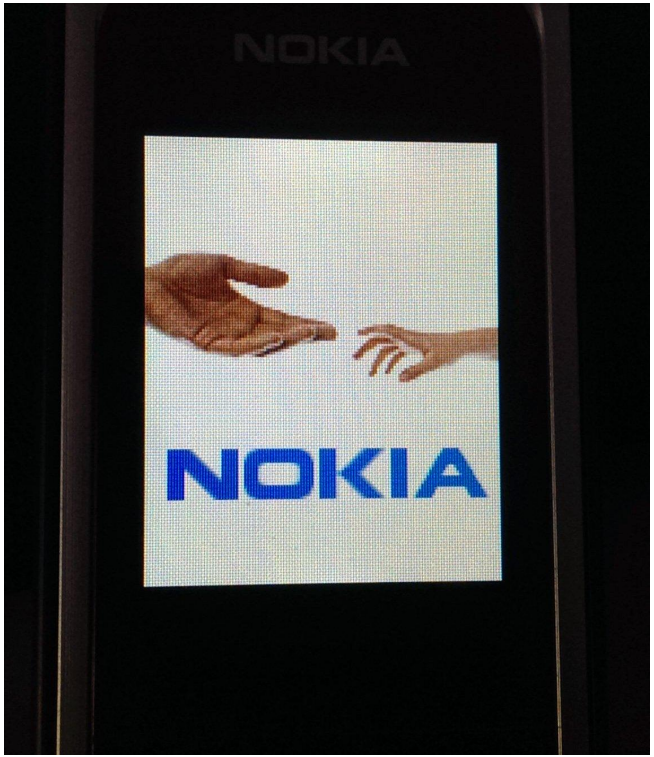


Contact Service Error

Simple checksum 16bit

Error type 3

Reboot



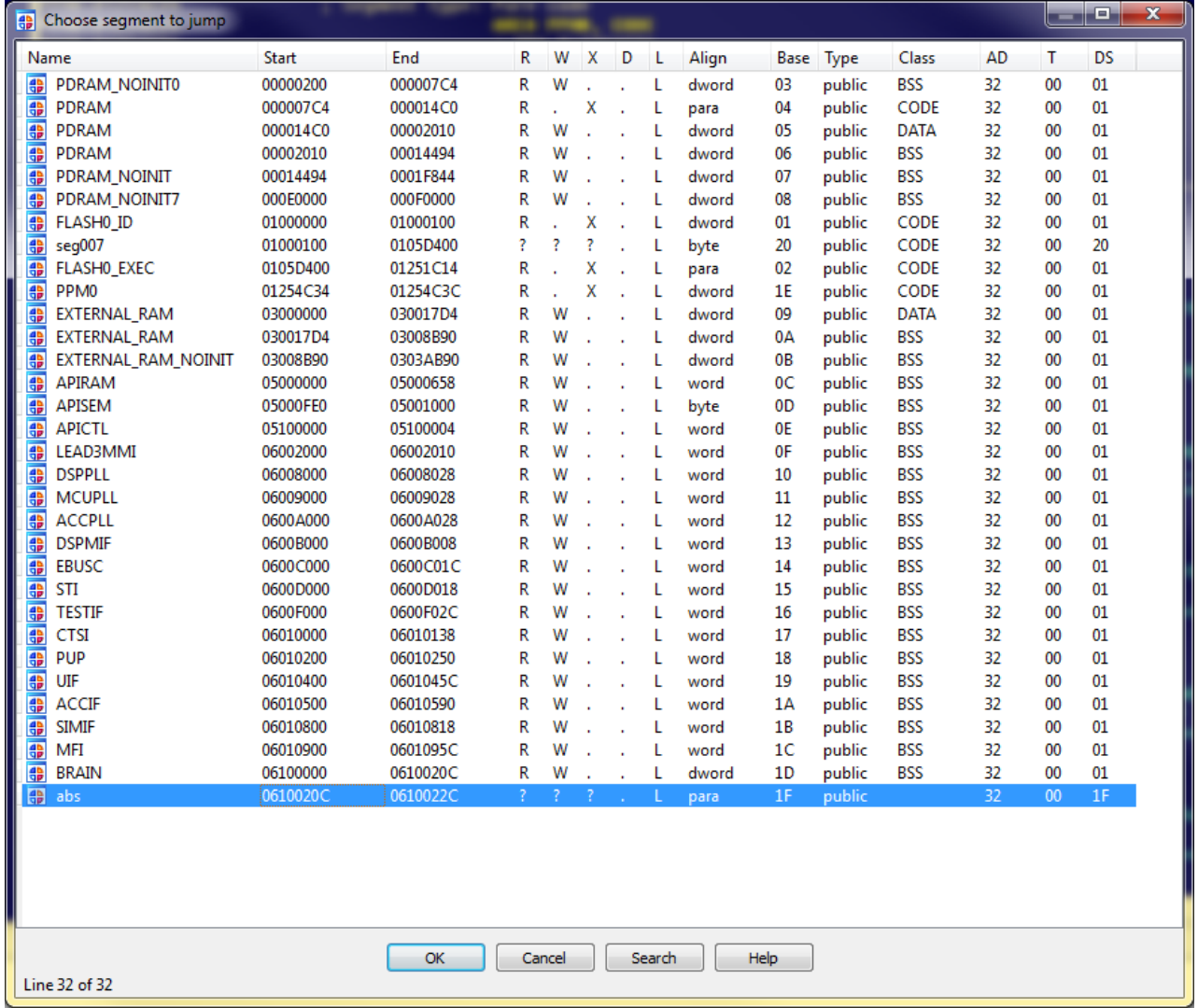
2 types of errors

1. ~~Contact Service~~
2. No Signal
3. Reboot

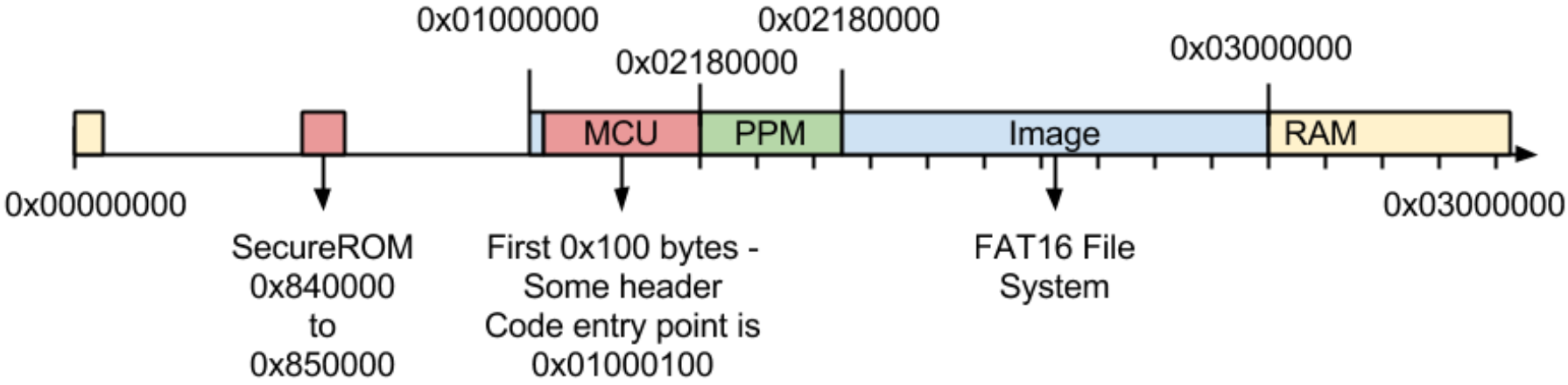
SRE the checks

Finding the memory map

Only one leak of debug symbols
Nokia 1650 rm305_05.530.out


Choose segment to jump
Name Start End R W X D L Align Base Type Class AD T DS
PDRAM_NOINIT0 00000200 000007C4 R W . . L dword 03 public BSS 32 00 01
PDRAM 000007C4 000014C0 R . X . L para 04 public CODE 32 00 01
PDRAM 000014C0 00002010 R W . . L dword 05 public DATA 32 00 01
PDRAM 00002010 00014494 R W . . L dword 06 public BSS 32 00 01
PDRAM_NOINIT 00014494 0001F844 R W . . L dword 07 public BSS 32 00 01
PDRAM_NOINIT7 000E0000 000F0000 R W . . L dword 08 public BSS 32 00 01
FLASH0_ID 01000000 01000100 R . X . L dword 01 public CODE 32 00 01
seg007 01000100 0105D400 ? ? ? . L byte 20 public CODE 32 00 20
FLASH0_EXEC 0105D400 01251C14 R . X . L para 02 public CODE 32 00 01
PPM0 01254C34 01254C3C R . X . L dword 1E public CODE 32 00 01
EXTERNAL_RAM 03000000 030017D4 R W . . L dword 09 public DATA 32 00 01
EXTERNAL_RAM 030017D4 03008B90 R W . . L dword 0A public BSS 32 00 01
EXTERNAL_RAM_NOINIT 03008B90 0303AB90 R W . . L dword 0B public BSS 32 00 01
APIRAM 05000000 05000658 R W . . L word 0C public BSS 32 00 01
APISEM 05000FE0 05001000 R W . . L byte 0D public BSS 32 00 01
APICTL 05100000 05100004 R W . . L word 0E public BSS 32 00 01
LEAD3MMI 06002000 06002010 R W . . L word 0F public BSS 32 00 01
DSPPLL 06008000 06008028 R W . . L word 10 public BSS 32 00 01
MCUPLL 06009000 06009028 R W . . L word 11 public BSS 32 00 01
ACCPLL 0600A000 0600A028 R W . . L word 12 public BSS 32 00 01
DSPMIF 0600B000 0600B008 R W . . L word 13 public BSS 32 00 01
EBUSC 0600C000 0600C01C R W . . L word 14 public BSS 32 00 01
STI 0600D000 0600D018 R W . . L word 15 public BSS 32 00 01
TESTIF 0600F000 0600F02C R W . . L word 16 public BSS 32 00 01
CTSI 06010000 06010138 R W . . L word 17 public BSS 32 00 01
PUP 06010200 06010250 R W . . L word 18 public BSS 32 00 01
UIF 06010400 0601045C R W . . L word 19 public BSS 32 00 01
ACCIF 06010500 06010590 R W . . L word 1A public BSS 32 00 01
SIMIF 06010800 06010818 R W . . L word 1B public BSS 32 00 01
MFI 06010900 0601095C R W . . L word 1C public BSS 32 00 01
BRAIN 06100000 0610020C R W . . L dword 1D public BSS 32 00 01
abs 0610020C 0610022C ? ? ? . L para 1F public 32 00 1F
OK
Cancel
Search
Help
Line 32 of 32

Memory map


0x01000000
0x02180000
0x02180000
0x03000000
MCU
PPM
Image
RAM
0x00000000
0x03000000
SecureROM
0x840000
to
0x850000
First 0x100 bytes -
Some header
Code entry point is
0x01000100
FAT16 File
System

1st MB

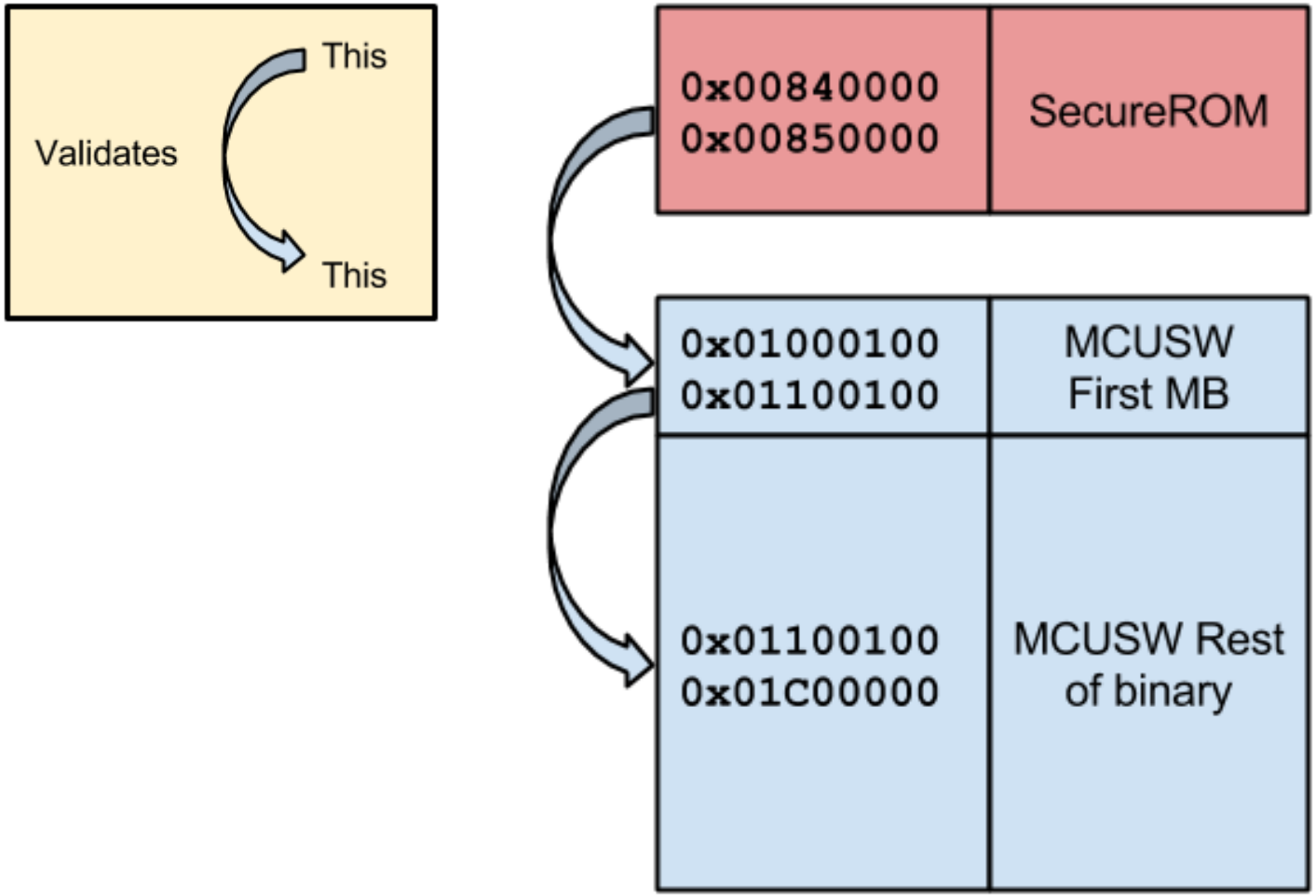
Calling to the 1st MB validation and activate GSM secROM function

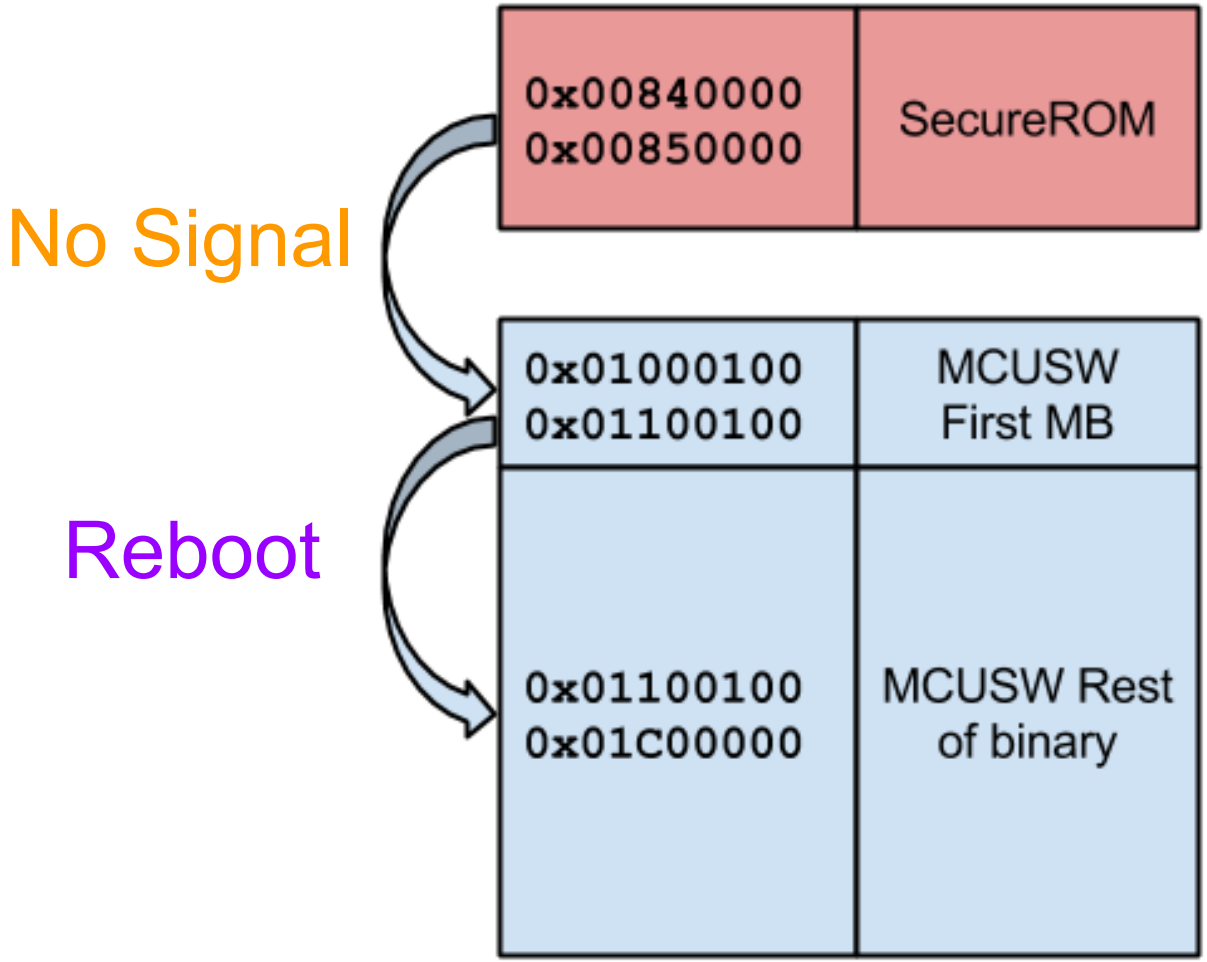
```
LDR     R0, =0x900003A
LDR     R1, =0x84000F
ADDS    R0, #0x32 ; '2'
ADDS    R1, R1, #6
BL      br_to_R1_thumb
```

Encryption disable bit

Address 0x900003a = 0x100003a | 0x8000000

The 0x8000000 is a flag that disables the firmware encryption / decryption for that address.


This
Validates
This
0x00840000
0x00850000
SecureROM
0x01000100
0x01100100
MCUSW
First MB
0x01100100
0x01C00000
MCUSW Rest
of binary

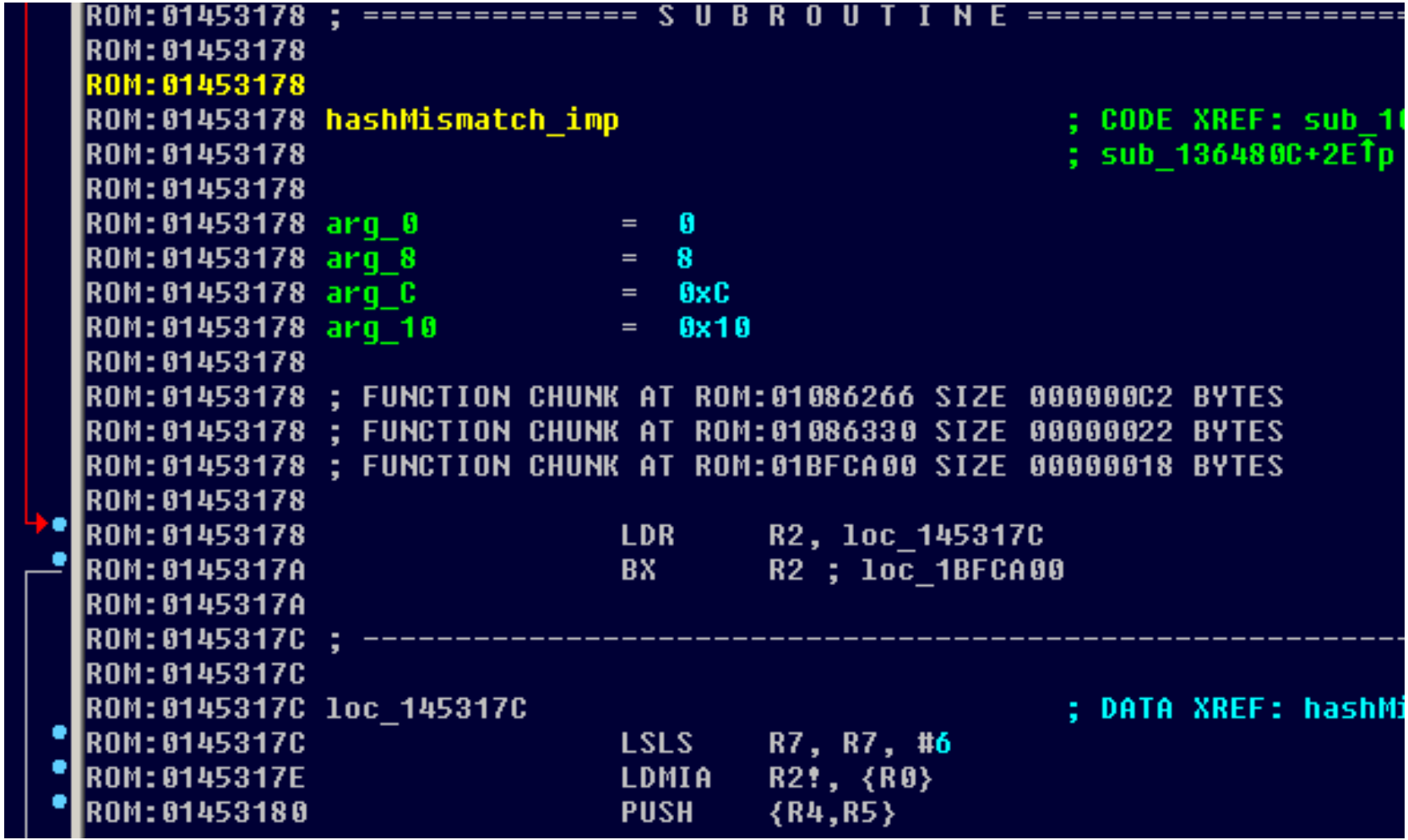

No Signal
Reboot
0x00840000
0x00850000
SecureROM
0x01000100
0x01100100
MCUSW
First MB
0x01100100
0x01C00000
MCUSW Rest
of binary

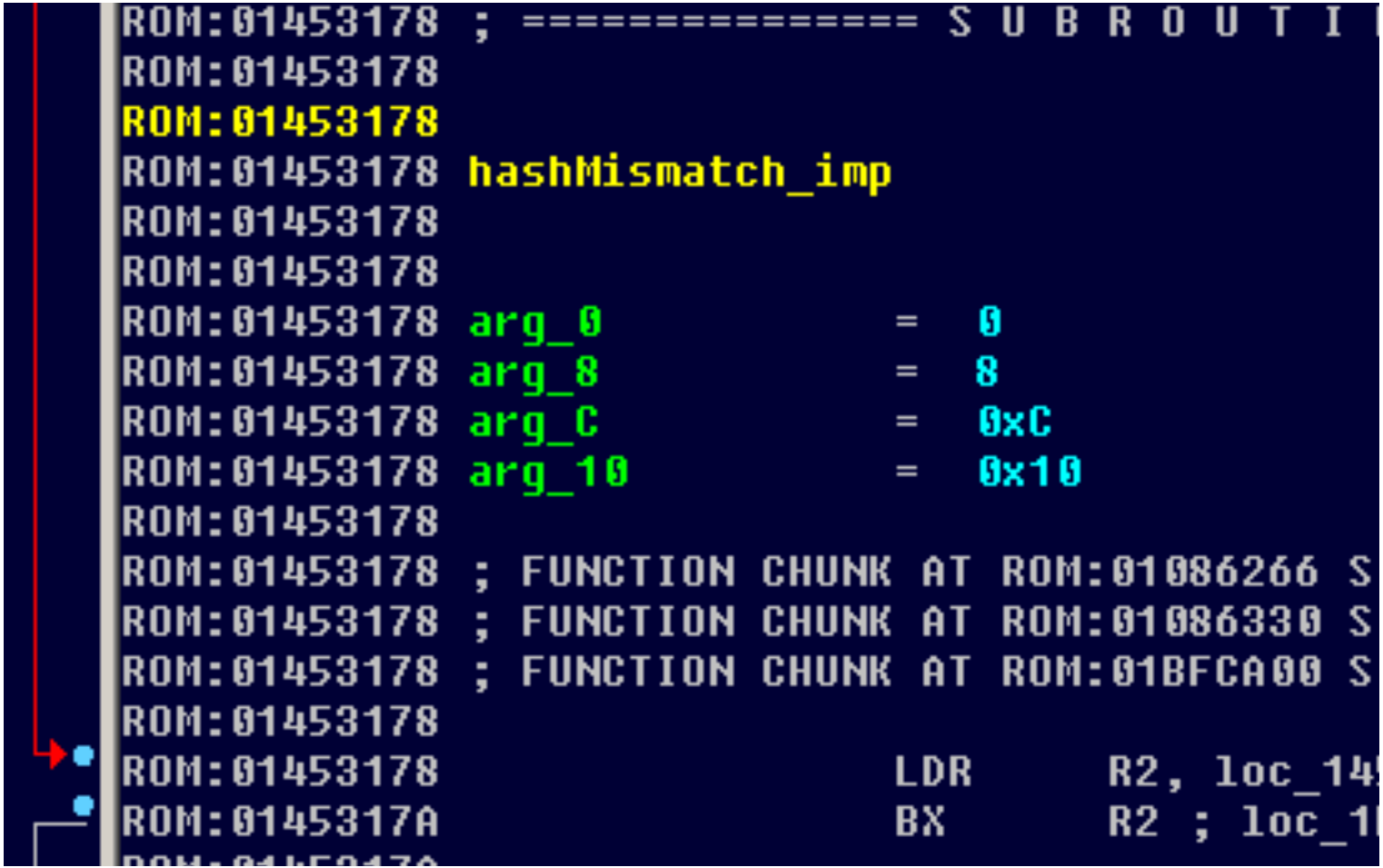
```
ROM:01086268                    MOVS    R1, #0x1C
ROM:0108626A                    LDRB    R0, [R2,R0]
ROM:0108626C                    MULS    R1, R0
ROM:0108626E                    LDR     R0, =SHA1_check_related
ROM:01086270                    SUBS    R0, #0x80 ; 'Ç'
ROM:01086272                    ADDS    R0, R1, R0
ROM:01086274                    MOVS    R4, R0
ROM:01086276                    ADDS    R0, #0x80 ; 'Ç'
ROM:01086278 R1 = Start
ROM:01086278                    LDR     R1, [R0,#0xC]
ROM:0108627A                    LDR     R2, [R0,#0x10]
ROM:0108627C                    LDR     R0, [R0,#0xC]
ROM:0108627E DataLength = DataStart - DataEnd;
ROM:0108627E                    SUBS    R3, R2, R0
ROM:01086280                    ADD     R2, SP, #0x38+hashLength
ROM:01086282                    STR     R2, [SP,#0x38+hashLengthCopy]
ROM:01086284                    LDRB    R0, [R6,#8]
ROM:01086286 DataLength += 1;
ROM:01086286                    ADDS    R3, R3, #1
ROM:01086288                    ADDS    R7, R7, R3
ROM:0108628A R2 = DataLength;
ROM:0108628A                    MOVS    R2, R3
ROM:0108628C                    ADD     R3, SP, #0x38+hashToCompare
ROM:0108628E                    BL      hashInitUpdateNDigest_j
ROM:01086292                    CMP     R0, #0
ROM:01086294                    BNE     loc_10862A4
```

```
ROM:01089DD4 SHA1_check_related DCD 0xB5213665       ; DATA XREF: SHA1_check:loc_108616A↑
ROM:01089DD4                                         ; SHA1_check+9E↑o ...
ROM:01089DD8                 DCD 3
ROM:01089DDC SHA1_check_info DCD 0x200400AA          ; DATA XREF: SHA1_check+44↑r
ROM:01089DE0 #1
ROM:01089DE0                 DCD loc_1100100
ROM:01089DE4                 DCD loc_13AFFFE+1
ROM:01089DE8                 DCD 0xEE41347A, 0x8C88F02F, 0x563BB973, 0x40E1233, 0x8C03AF
ROM:01089DFC #2
ROM:01089DFC                 DCD loc_13B0000
ROM:01089E00                 DCD loc_165FFFE+1
ROM:01089E04                 DCD 0xCC29F881, 0xA441D8CD, 0x7CEF5FEF, 0xC35FE703, 0x8BD3D
ROM:01089E18 #3
ROM:01089E18                 DCD loc_1660000
ROM:01089E1C                 DCD loc_190FFFC+3
ROM:01089E20                 DCD 0x77439E9B, 0x530F0029, 0xA7490D5B, 0x4E621094, 0xC7844
ROM:01089E34 #4
ROM:01089E34                 DCD loc_1910000
ROM:01089E38                 DCD dword_1BFB5C8+7
ROM:01089E3C                 DCD 0xA87ABFB7, 0xFB44D95E, 0xC3E95DCA, 0xE190ECCA, 0x9D100
ROM:01089E50                 DCD 0
ROM:01089E54                 DCD 0
```

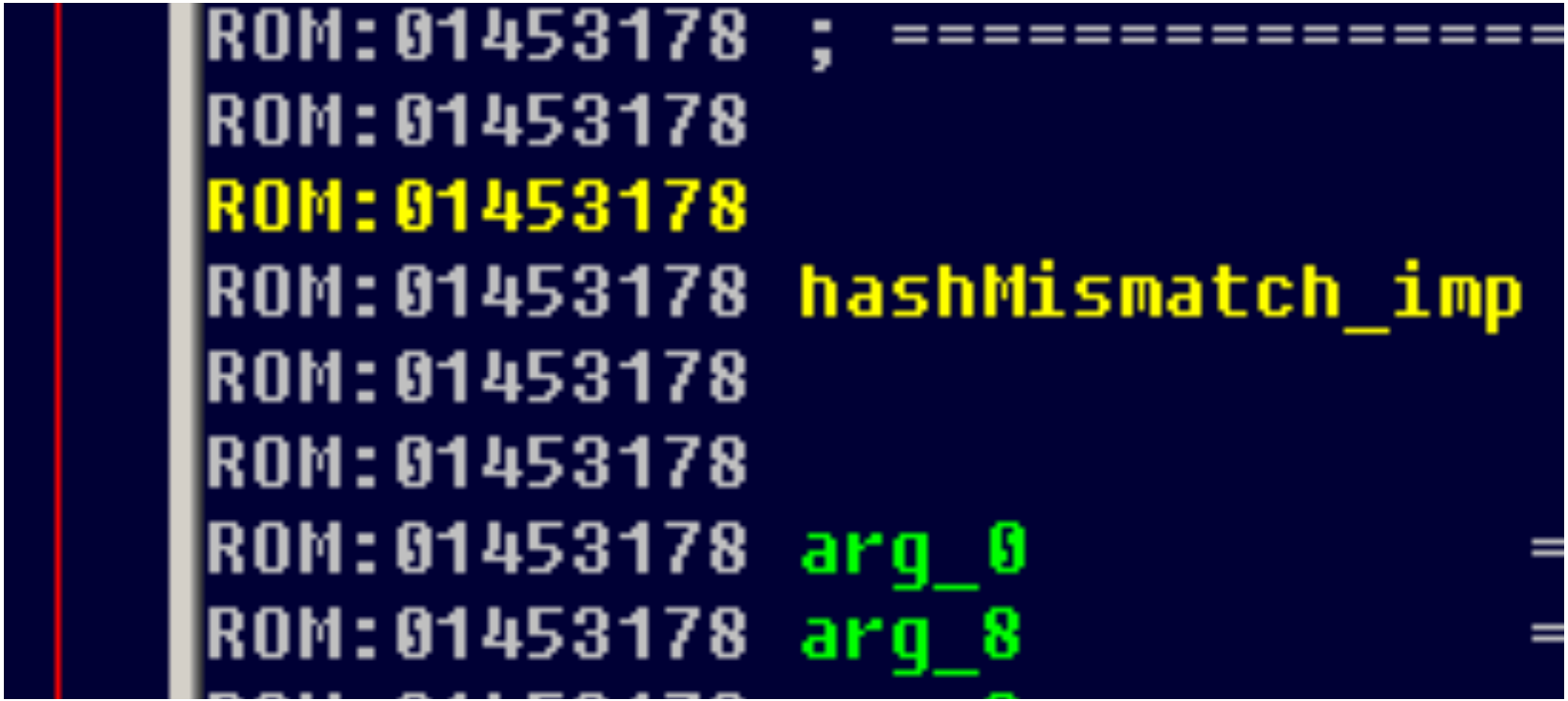
```
ROM:010862E0 ; ---------------------------------------------------------------
ROM:010862E0 for( i = 0; i < hashLength; ++i ) {
ROM:010862E0
ROM:010862E0 loc_10862E0                                ; CODE XREF: SHA1_check+
ROM:010862E0                 ADDS    R3, R4, R0
ROM:010862E2                 ADDS    R3, #0x80 ; 'Ç'
ROM:010862E4                 ADD     R2, SP, #0x38+hashToCompare
ROM:010862E6                 LDRB    R2, [R2,R0]
ROM:010862E8                 LDRB    R3, [R3,#0x14]
ROM:010862EA     if (hash[i] != hashToCompare[i])
ROM:010862EA                 CMP     R2, R3
ROM:010862EC                 BEQ     loc_10862F0
ROM:010862EE                 MOVS    R5, #1
ROM:010862F0
ROM:010862F0 loc_10862F0                                ; CODE XREF: SHA1_check+
ROM:010862F0                 ADDS    R0, R0, #1
ROM:010862F2
ROM:010862F2 loop                                       ; CODE XREF: SHA1_check+
ROM:010862F2                 CMP     R0, R1
ROM:010862F4 }
ROM:010862F4                 BCC     loc_10862E0
ROM:010862F6                 CMP     R5, #1
ROM:010862F8                 BNE     loc_1086308
ROM:010862FA                 LDR     R0, =0x7D0005
ROM:010862FC                 BL      hashMismatch
ROM:01086300                 MOVS    R0, #4
ROM:01086302                 BL      reset
ROM:01086306                 B       loc_1086310
ROM:01086308 ; ---------------------------------------------------------------
```

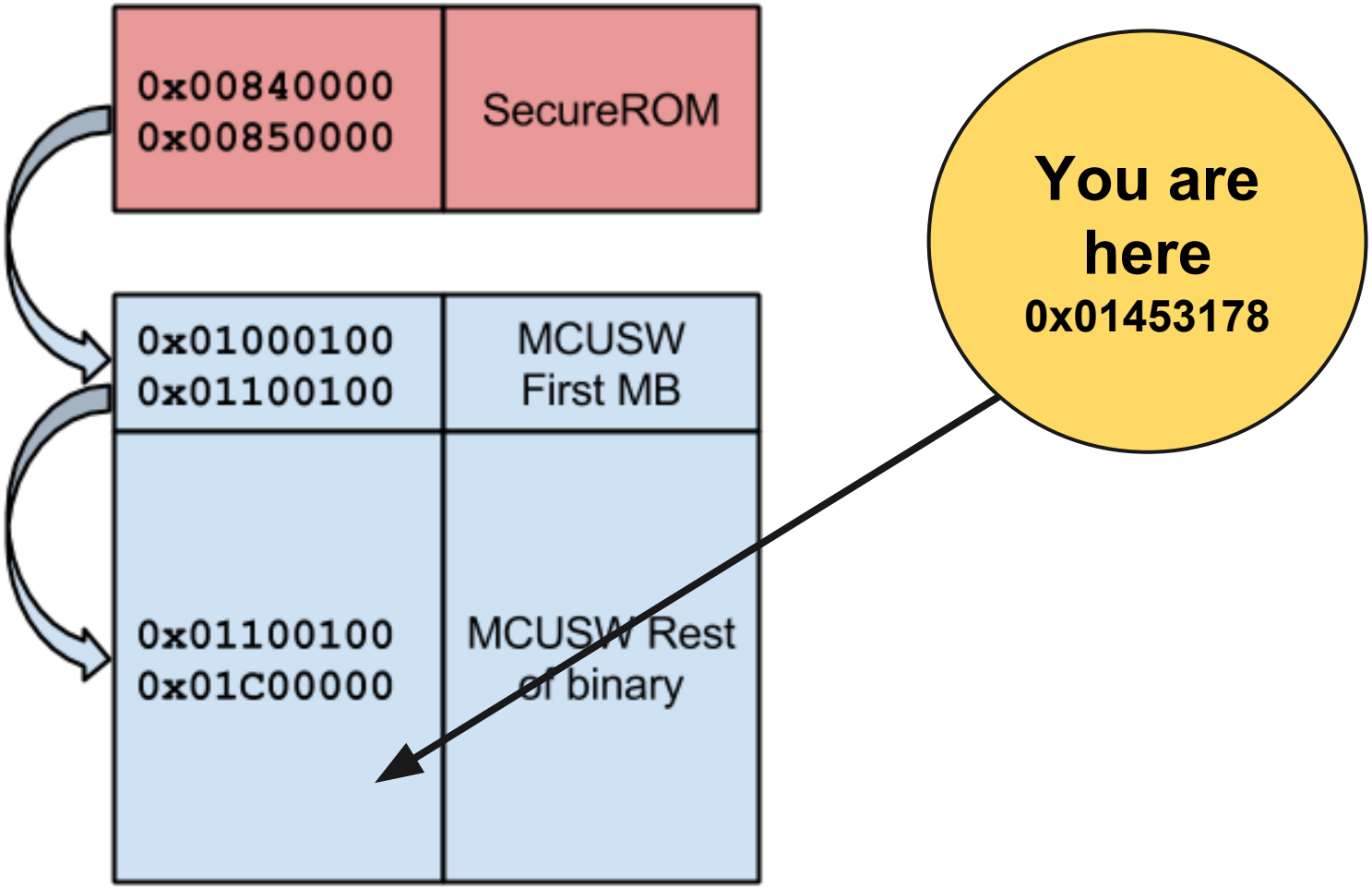
```
ROM:01085320 ; Attributes: thunk
ROM:01085320
ROM:01085320 hashMismatch                            ; CODE XREF: sub_1084232+38↑p
ROM:01085320                                         ; sub_1085B6C+6C↓p ...
ROM:01085320                 BX      PC
ROM:01085320 ; ---------------------------------------------------------------------
ROM:01085322                 ALIGN 4
ROM:01085322 ; End of function hashMismatch
ROM:01085322
ROM:01085324                 CODE32
ROM:01085324
ROM:01085324 ; =============== S U B R O U T I N E =======================================
ROM:01085324
ROM:01085324
ROM:01085324 sub_1085324                             ; CODE XREF: hashMismatch↑j
ROM:01085324                 LDR     R12, =(hashMismatch_imp+1)
ROM:01085328                 BX      R12 ; hashMismatch_imp
ROM:01085328 ; End of function sub_1085324
ROM:01085328
ROM:01085328 ; ---------------------------------------------------------------------
ROM:0108532C off_108532C     DCD hashMismatch_imp+1  ; DATA XREF: sub_1085324↑r
ROM:01085330                 CODE16
ROM:01085330
```

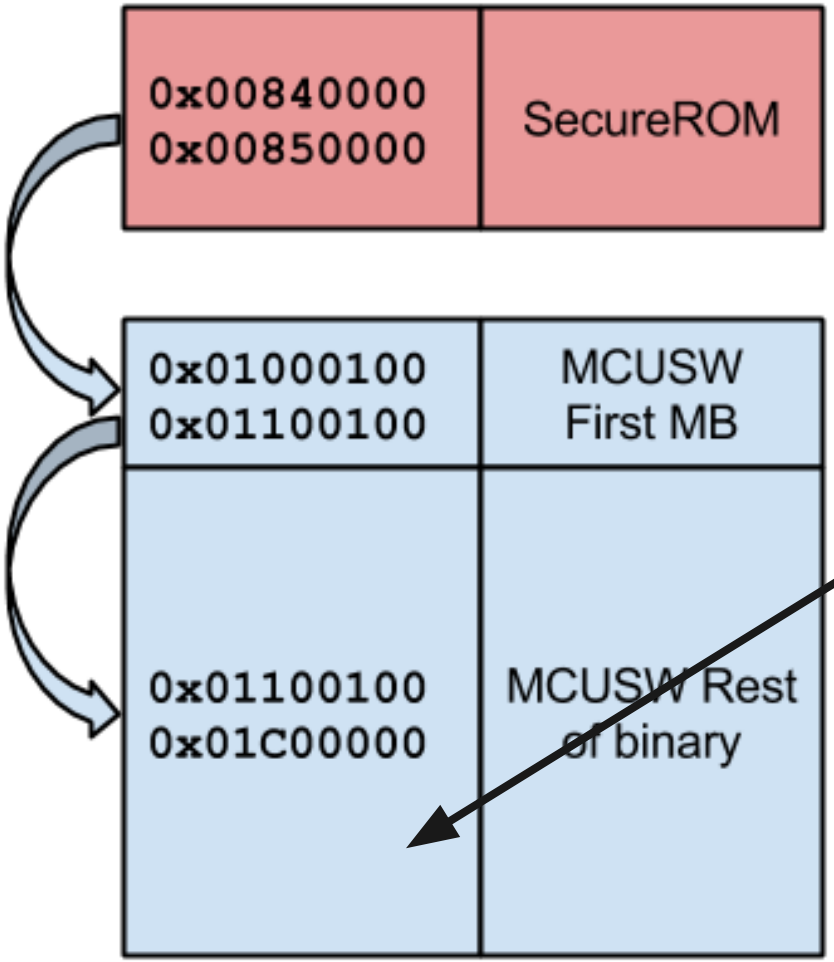

ROM:01453178 ; =============== S U B R O U T I N E ===================
ROM:01453178
ROM:01453178
ROM:01453178 hashMismatch_imp ; CODE XREF: sub_1
ROM:01453178 ; sub_136480C+2E↑p
ROM:01453178
ROM:01453178 arg_0 = 0
ROM:01453178 arg_8 = 8
ROM:01453178 arg_C = 0xC
ROM:01453178 arg_10 = 0x10
ROM:01453178
ROM:01453178 ; FUNCTION CHUNK AT ROM:01086266 SIZE 000000C2 BYTES
ROM:01453178 ; FUNCTION CHUNK AT ROM:01086330 SIZE 00000022 BYTES
ROM:01453178 ; FUNCTION CHUNK AT ROM:01BFCA00 SIZE 00000018 BYTES
ROM:01453178
ROM:01453178 LDR R2, loc_145317C
ROM:0145317A BX R2 ; loc_1BFCA00
ROM:0145317A
ROM:0145317C ; ---
ROM:0145317C
ROM:0145317C loc_145317C ; DATA XREF: hashMi
ROM:0145317C LSLS R7, R7, #6
ROM:0145317E LDMIA R2!, {R0}
ROM:01453180 PUSH {R4,R5}


ROM:01453178 ; =============== S U B R O U T I
ROM:01453178
ROM:01453178
ROM:01453178 hashMismatch_imp
ROM:01453178
ROM:01453178
ROM:01453178 arg_0 = 0
ROM:01453178 arg_8 = 8
ROM:01453178 arg_C = 0xC
ROM:01453178 arg_10 = 0x10
ROM:01453178
ROM:01453178 ; FUNCTION CHUNK AT ROM:01086266 S
ROM:01453178 ; FUNCTION CHUNK AT ROM:01086330 S
ROM:01453178 ; FUNCTION CHUNK AT ROM:01BFCA00 S
ROM:01453178
ROM:01453178 LDR R2, loc_14
ROM:0145317A BX R2 ; loc_1

Zoom x3


ROM:01453178 ; ==========
ROM:01453178
ROM:01453178
ROM:01453178 hashMismatch_imp
ROM:01453178
ROM:01453178
ROM:01453178 arg_0 =
ROM:01453178 arg_8 =


0x00840000
0x00850000
SecureROM
0x01000100
0x01100100
MCUSW
First MB
0x01100100
0x01C00000
MCUSW Rest
of binary
You are
here
0x01453178



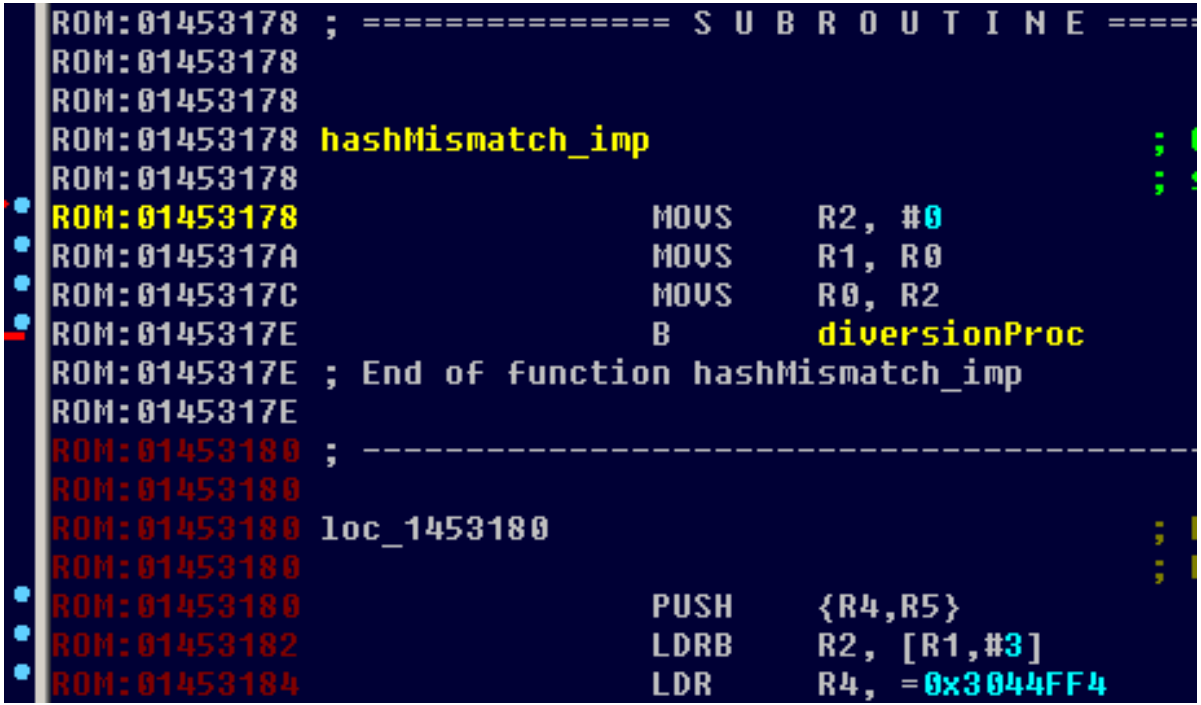
You are here
0x01453178

Meaning

When the check fails, it uses the code that it just failed to validate!

Exploiting

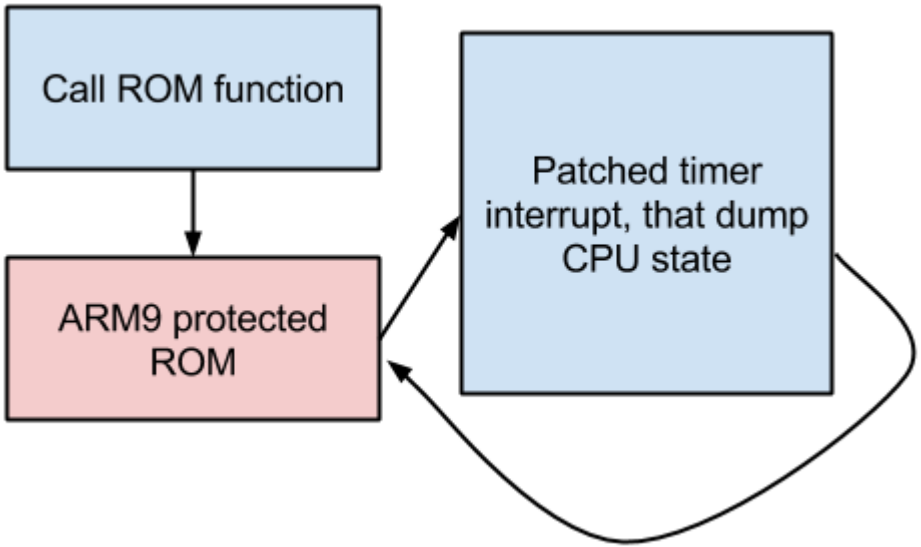
Allow us to overcome the validation of binary after the 1st MB (Reboot error)



POC

https://www.youtube.com/watch?v=i0NJZ_J5c6g&feature=youtu.be

Overcoming the SecureROM



First 1MB check (No Signal)

Example for log recovering

R0:00000000 R1:00000005 R2:00000002

R0:00000007 R1:00000005 R2:00000002

R0:00000009 R1:00000005 R2:00000002

R0:0000000A R1:00000005 R2:00000002

Make it Kosher

Disable Internet

- Remember that time is money

Disable Internet

- “GET” -> “BET”
- “POST” -> “MOST”

No web server would ever answer you again.

Hardware patches

Circumcising a phone

Bugs

- GSM connection
- FM Radio
- Bad factory reset

BB5

Not on this presentation :(

Asha phones protection

This one has

- Whatsapp
- Facebook
- Twitter
- And all kind of other things...

Three steps of trust chain

1. PBL
2. SBL
3. Firmware

RSA 1024 - SHA1

Signed

- Flashable
  - OS (MCUSW) - Signed
  - Localization strings and gfx (PPM) - Signed
  - General purpose file system - Operator
    - FAT16 (CNT / IMAGE) - Open
- SecureROM
  - ?

In the Image

- Menusettings.xml
- Java apps
- Startup / Shutdown animations
- Ringtones
- Many kinds of other settings

Blocking SMS & MMS

When I have no control over the OS

A few things about FAT16

- Table that defines chains of sectors
- Hard-links are possible
- Two files with the same name are possible

Blocking SMS

- Find where messages are stored
- Delete that folder
- Create a file with the same name

pyFAT16

Parsing FAT16 with Python is fun

Making of the Samsung Kosher Phone

How to

1. Download the firmware
2. Patch whatever you like
3. Flash it

* If it’s hard to find what to patch, Samsung are leaking binaries with debug symbols everywhere!

Release

Phosher framework:

https://phosher.googlecode.com/svn/trunk

* It includes the FAT16 parser

Thanks

- Friends who prefer to stay Anonymous
- AT
- Ildis, Rubi, Yuval, Nitzan, Oren & Budo
- G3gg0, Krish and Nok5rev
- The good people of GSM Forum
- Wife
- My daughter for stress testing the hardware

Thank you

Questions?