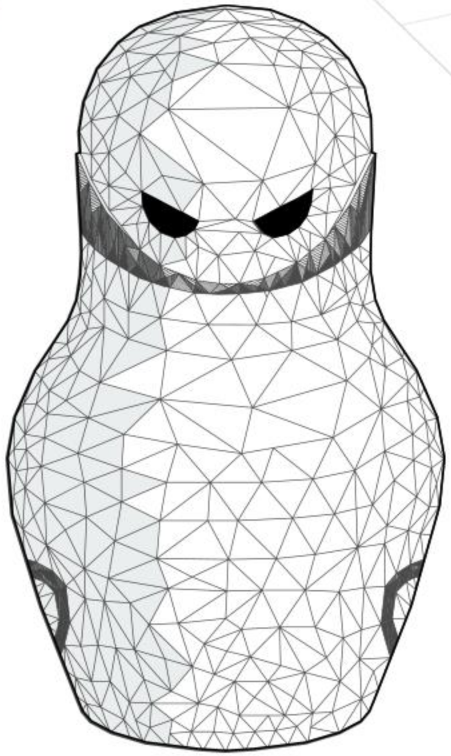


ZERO NIGHTS 2015

# Industrial switches firmware modification

Alexander Ermolov
a.ermolov@dsec.ru

# Who am I

Security researcher at Digital Security

Main interests:

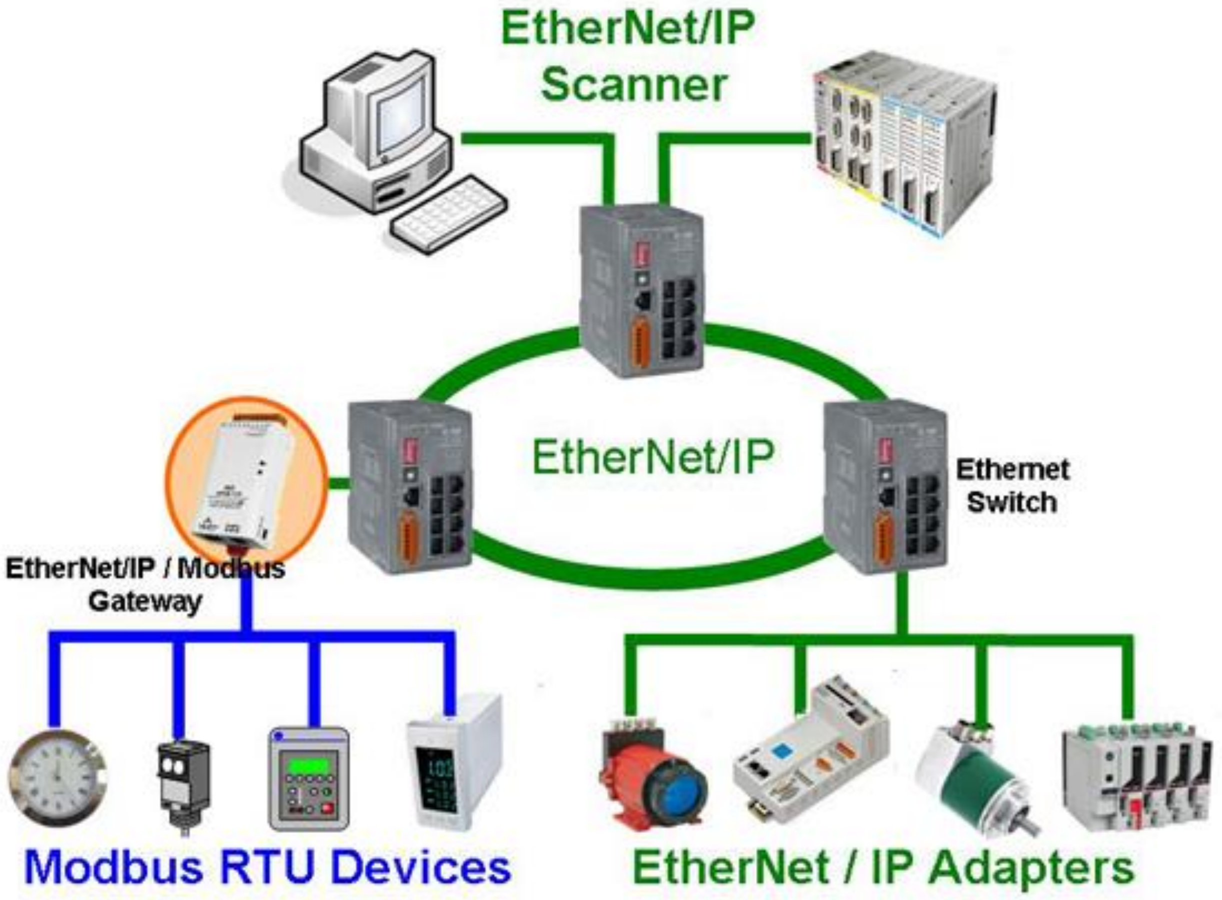
- Low-level design of computer systems
- Undocumented features

# Industrial switches

Used in industrial Ethernet

Provide communication between:

- PLC
- HMI
- field devices
- …



# Why industrial switches?

Pwned switch as a part of industrial network is capable of:

- pwning other devices (switches, field devices…)
- gathering information about technical process
- interfering with technical process

# Timeline

“Switches get stitches” workshop

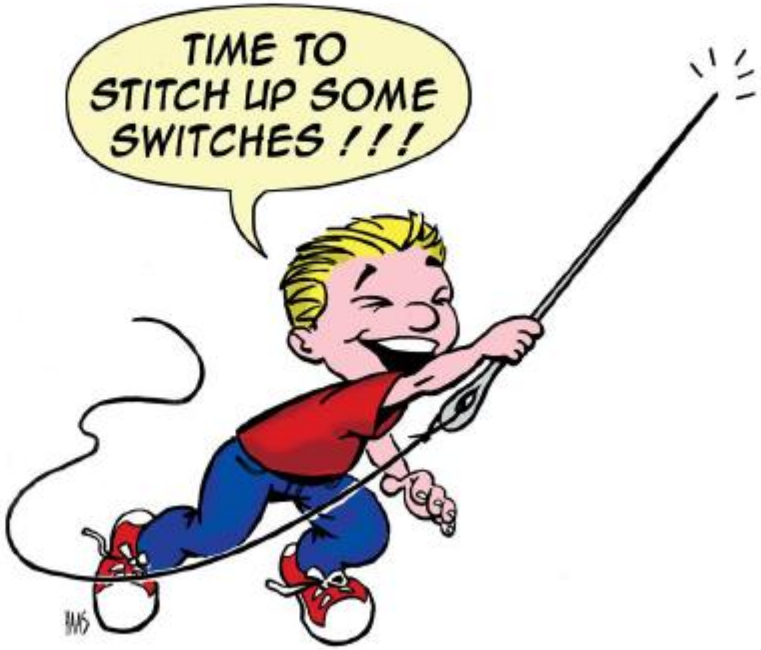
Eireann Leverett & Matt Erasmus
September 2014, 44CON

“Switches get stitches”

Eireann Leverett
December 2014, 31c3

“Switches get stitches: episode 3”

Eireann Leverett & Colin Cassidy & Robert Lee
August 2015, BlackHat



# Devices covered

## Hirschmann RS20

Managed industrial switch

External interfaces:

- USB
- V.24 (RJ11) = RS-232
- 4 x Ethernet (RJ45)

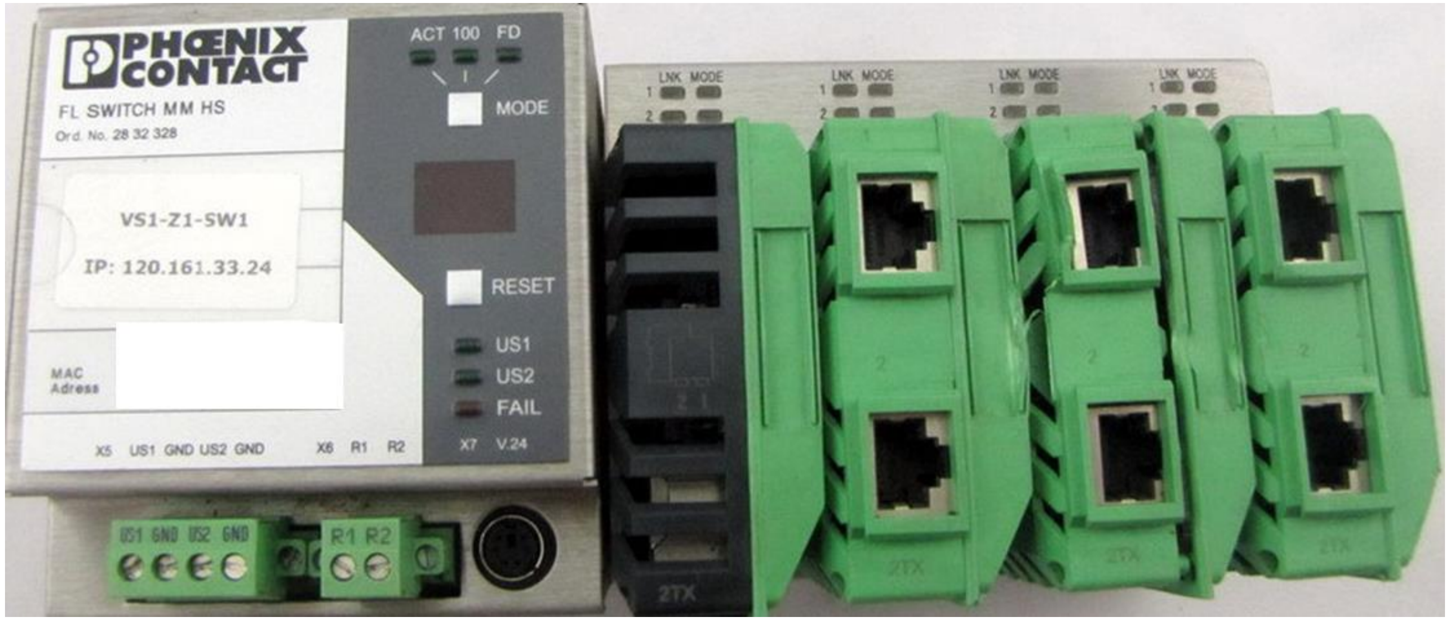
# Devices covered

## Phoenix Contact FL SWITCH MM HS

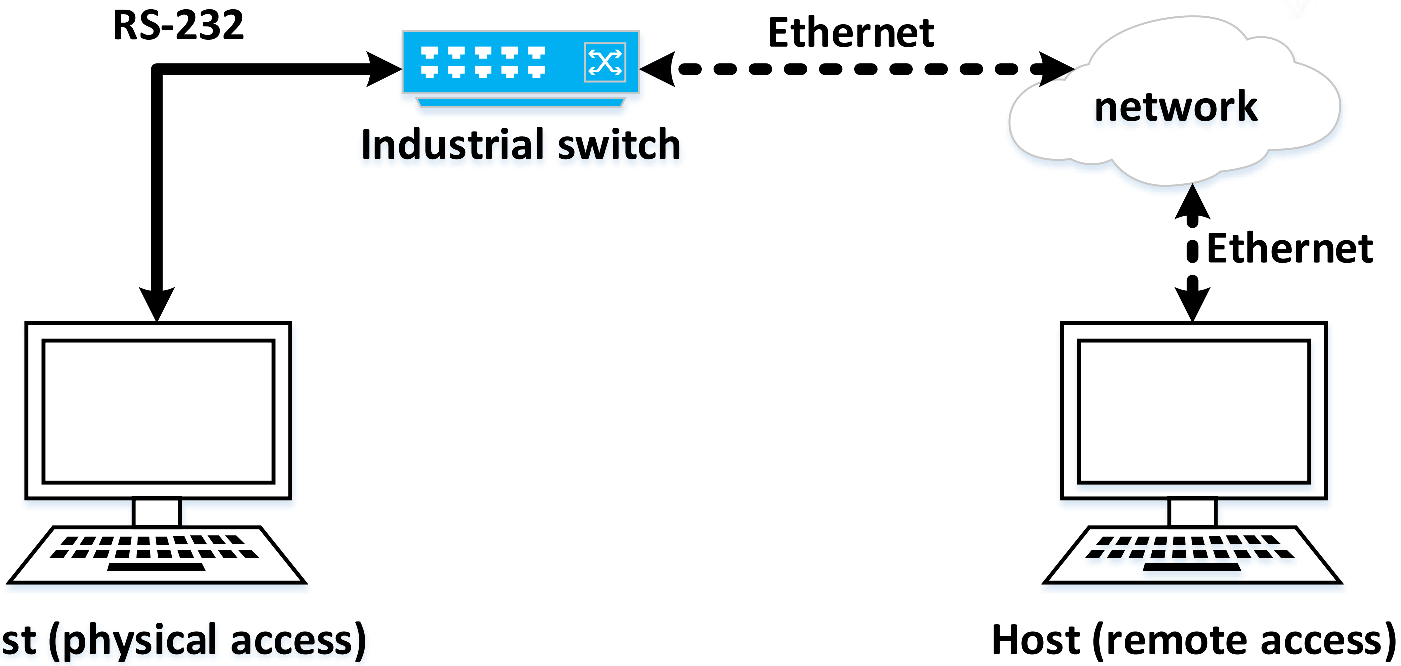
Managed industrial switch

External interfaces:

- V.24 (mini DIN) = RS-232
- 6 x Ethernet (RJ45)



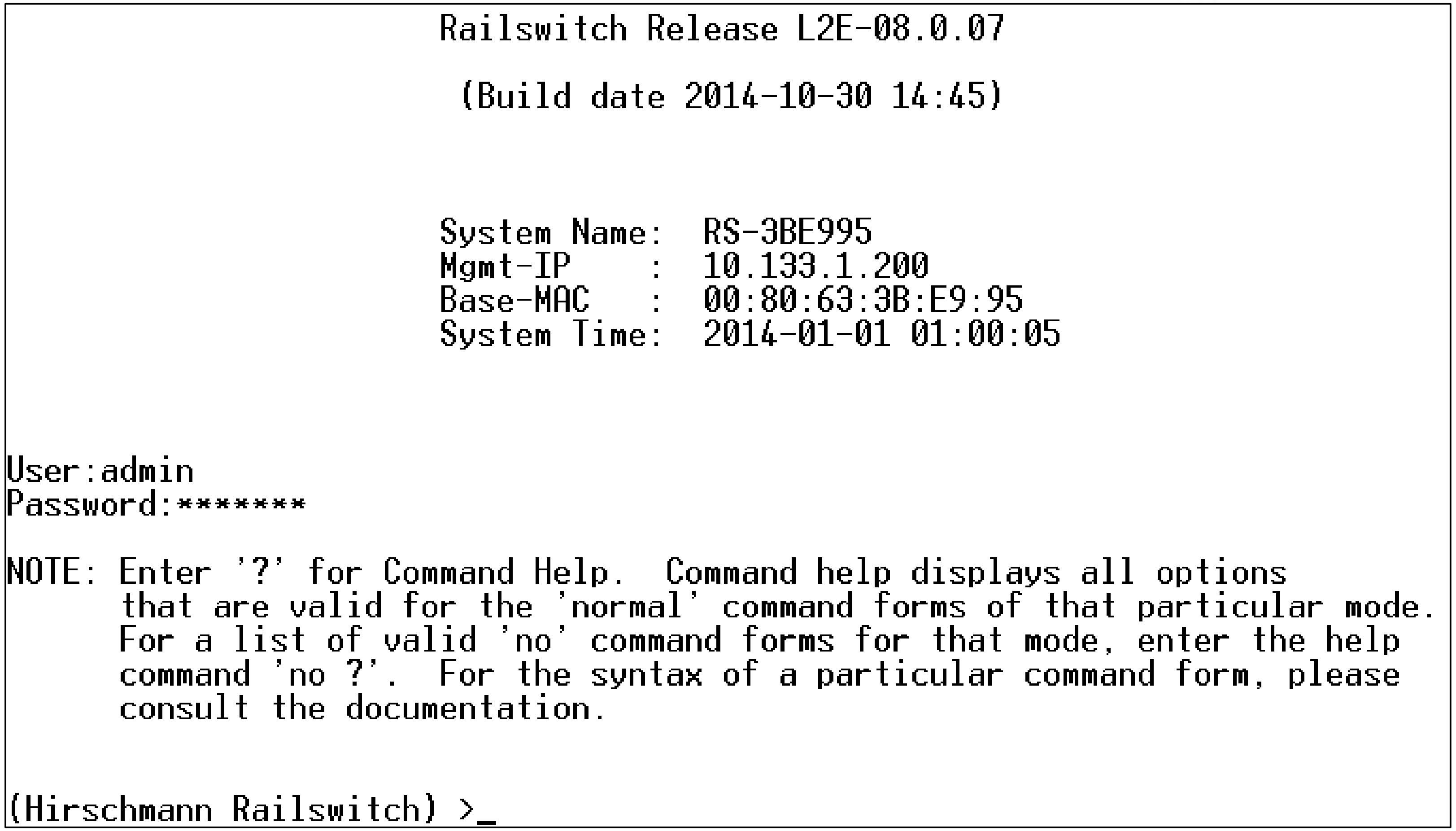
# Connecting to the switch



- Console interface

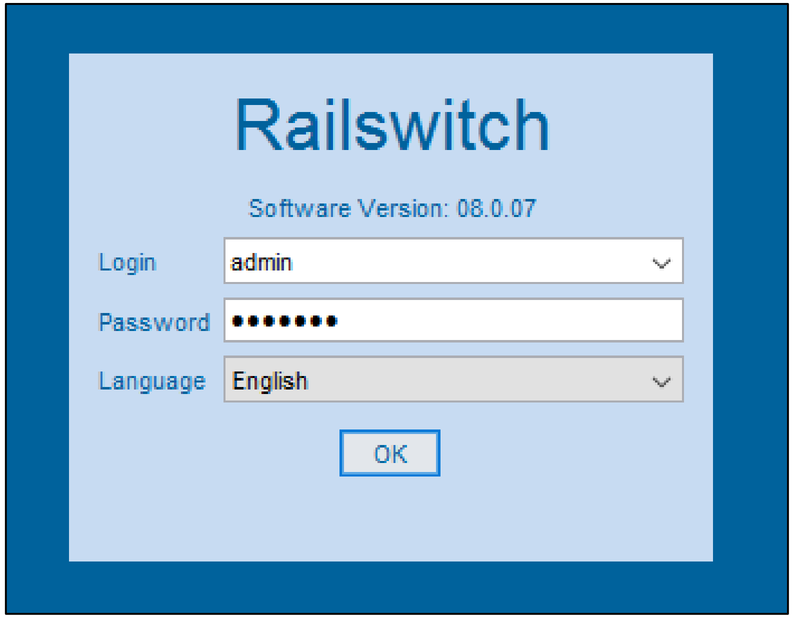
- HTTP web interface
- SNMP

# Console interface



```
                       Railswitch Release L2E-08.0.07

                        (Build date 2014-10-30 14:45)



                       System Name:  RS-3BE995
                       Mgmt-IP    :  10.133.1.200
                       Base-MAC   :  00:80:63:3B:E9:95
                       System Time:  2014-01-01 01:00:05



User:admin
Password:*******

NOTE: Enter '?' for Command Help.  Command help displays all options
      that are valid for the 'normal' command forms of that particular mode.
      For a list of valid 'no' command forms for that mode, enter the help
      command 'no ?'.  For the syntax of a particular command form, please
      consult the documentation.


(Hirschmann Railswitch) >_
```

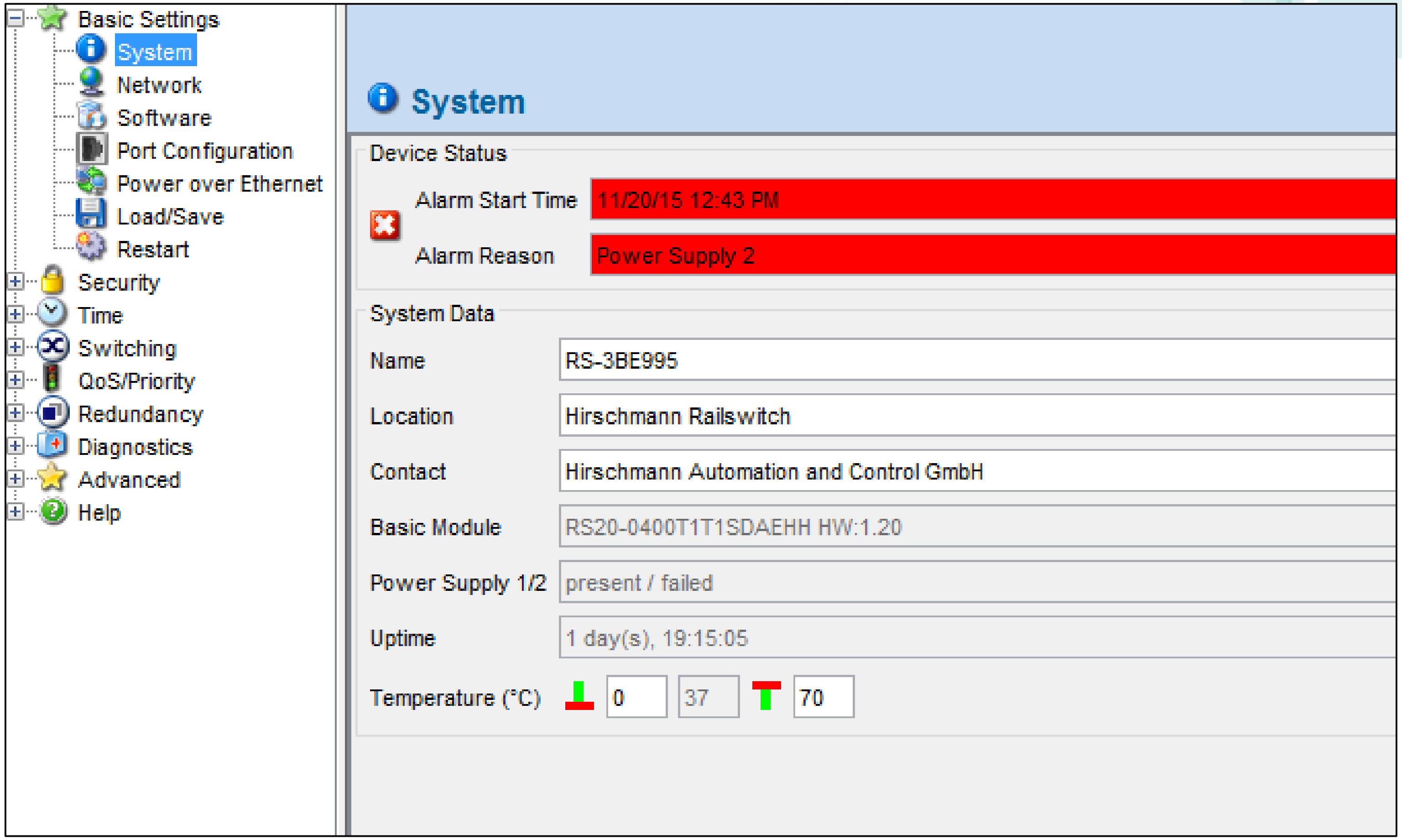
# HTTP web interface



Railswitch

Software Version: 08.0.07

| | |
|---|---|
| Login | admin |
| Password | ••••••• |
| Language | English |



System

Device Status

| | |
|---|---|
| Alarm Start Time | 11/20/15 12:43 PM |
| Alarm Reason | Power Supply 2 |

System Data

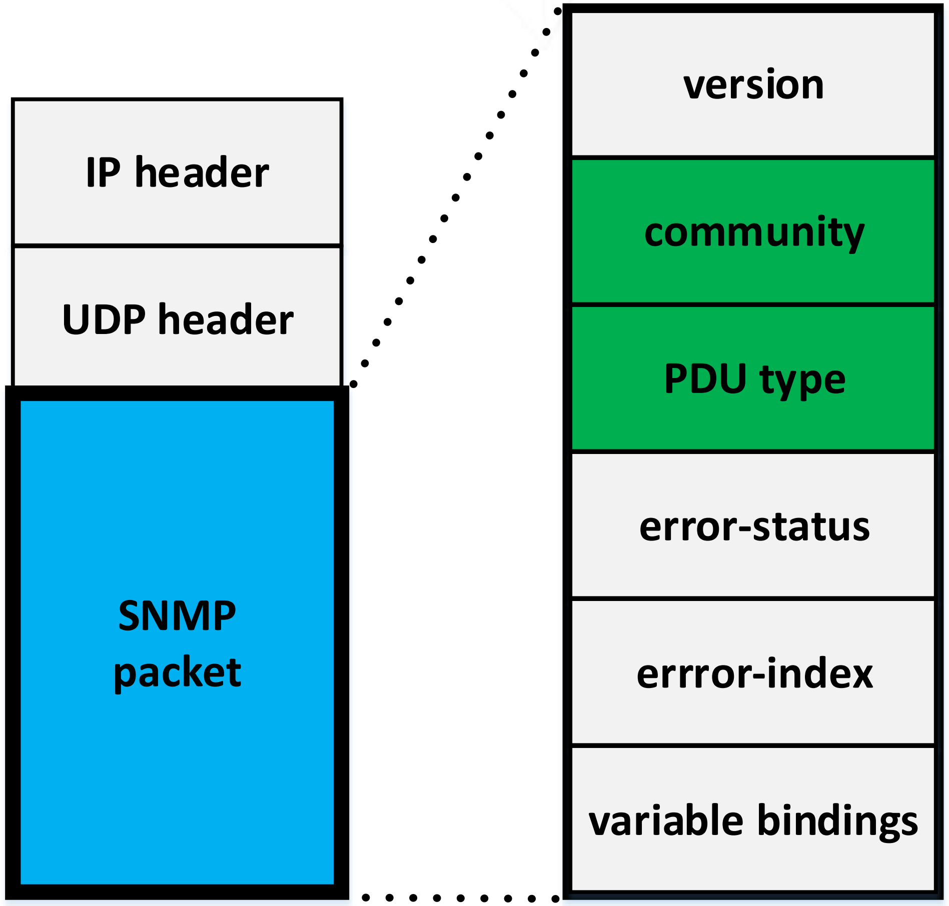
| | |
|---|---|
| Name | RS-3BE995 |
| Location | Hirschmann Railswitch |
| Contact | Hirschmann Automation and Control GmbH |
| Basic Module | RS20-0400T1T1SDAEHH HW:1.20 |
| Power Supply 1/2 | present / failed |
| Uptime | 1 day(s), 19:15:05 |
| Temperature (°C) | 0 37 70 |

# Simple Network Management Protocol (SNMP)

OSI Application layer
UDP ports: 161, 162

PDU types (commands):

- GetRequest
- SetRequest
- GetNextRequest
- GetBulkRequest
- Response
- Trap
- InformRequest



# Simple Network Management Protocol (SNMP)

- SNMP v1 used on the switches by default
- SNMP v1 uses default login/password which are not recommended (by vendor) to be changed
- SNMP v1 and SNMP v2c don’t use any encryption

# Hirschmann RS20

# Onboard hardware

**1. CPU**

Digi NET+ARM NS9360B-0-I155
ARM9 32-bit, no internal memory

**2. SDRAM**

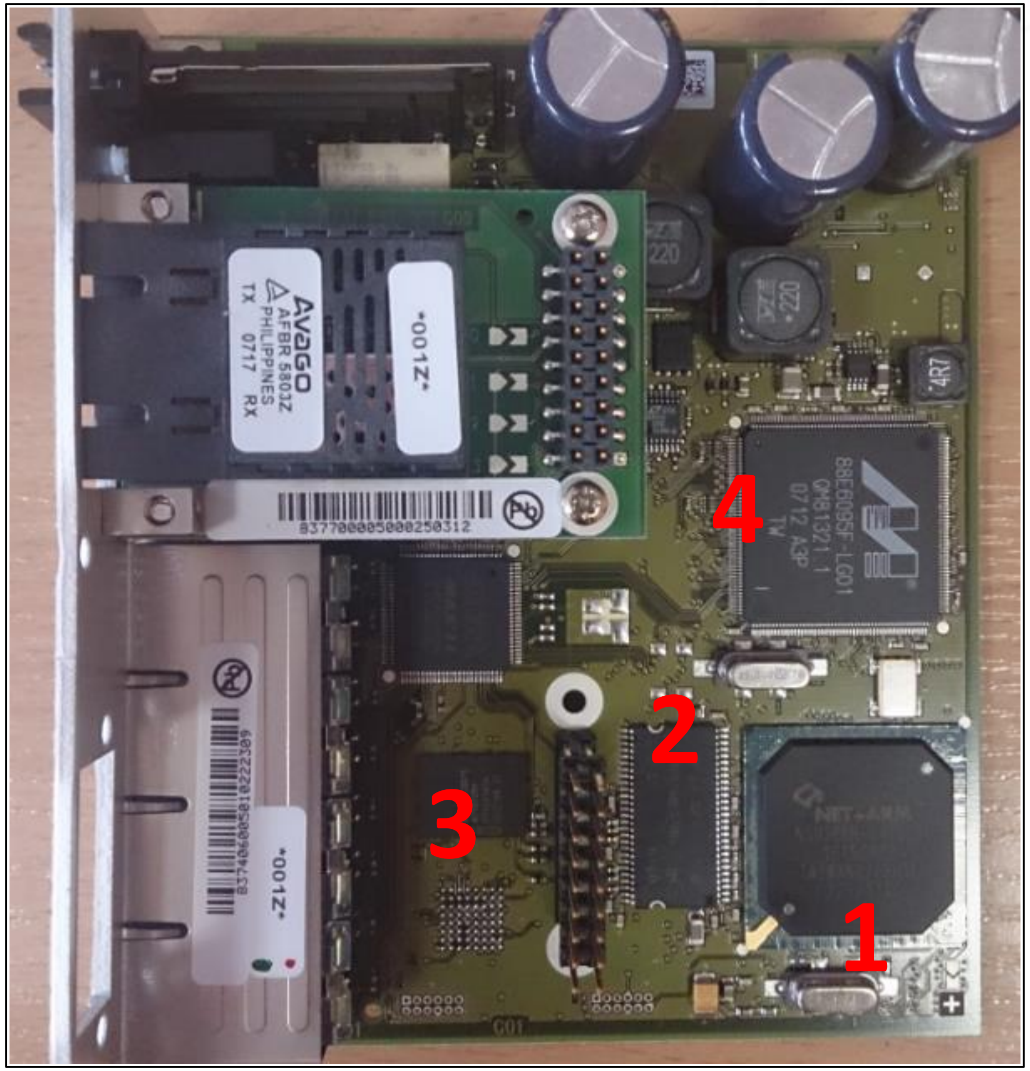
Micron MT48LC8M16A2
16 MB

**3. Flash memory**

Intel 28F640JD3D75
8 MB

**4. Ethernet switch**

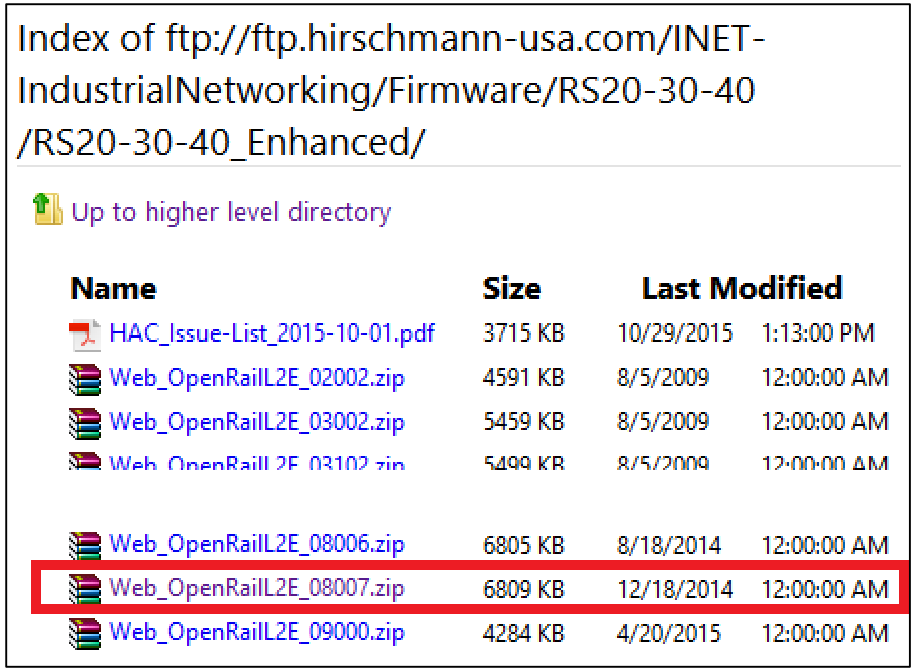
Marvell 88E6095F-LG01
CPLD



# Download firmware image

Firmware version is 8.0.07
Download from Hirschmann ftp

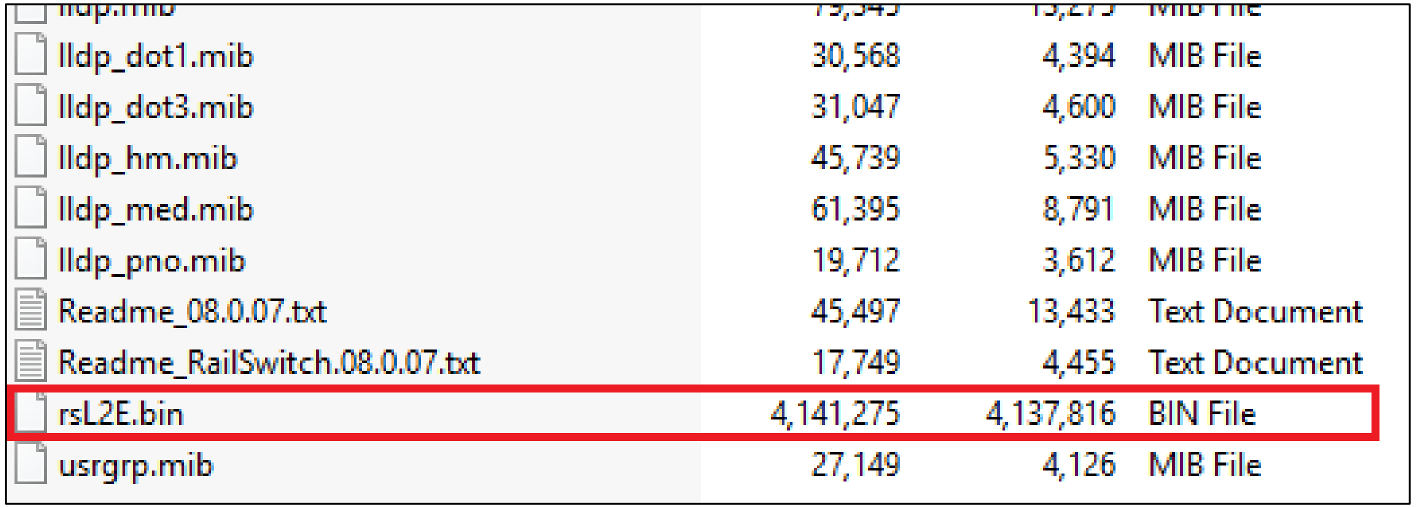
The zip archive contains firmware image (~4 Mbytes)



Index of ftp://ftp.hirschmann-usa.com/INET-IndustrialNetworking/Firmware/RS20-30-40/RS20-30-40_Enhanced/

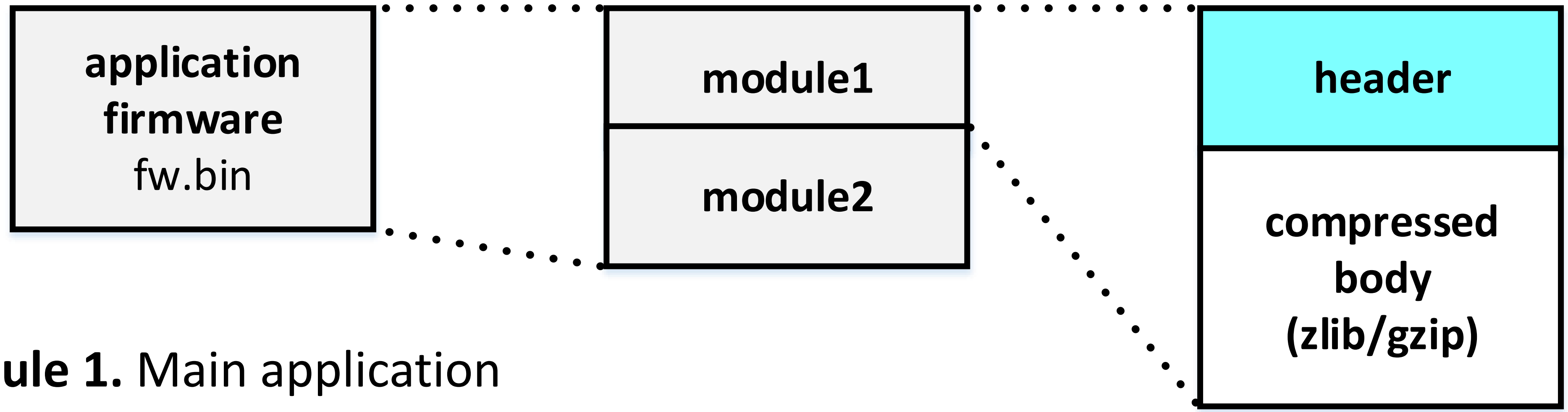
Up to higher level directory

| Name | Size | Last Modified |
|---|---|---|
| HAC_Issue-List_2015-10-01.pdf | 3715 KB | 10/29/2015 1:13:00 PM |
| Web_OpenRailL2E_02002.zip | 4591 KB | 8/5/2009 12:00:00 AM |
| Web_OpenRailL2E_03002.zip | 5459 KB | 8/5/2009 12:00:00 AM |
| Web_OpenRailL2E_03102.zip | 5499 KB | 8/5/2009 12:00:00 AM |
| Web_OpenRailL2E_08006.zip | 6805 KB | 8/18/2014 12:00:00 AM |
| Web_OpenRailL2E_08007.zip | 6809 KB | 12/18/2014 12:00:00 AM |
| Web_OpenRailL2E_09000.zip | 4284 KB | 4/20/2015 12:00:00 AM |



| | | | |
|---|---|---|---|
| lldp.mib | 79,545 | 13,275 | MIB File |
| lldp_dot1.mib | 30,568 | 4,394 | MIB File |
| lldp_dot3.mib | 31,047 | 4,600 | MIB File |
| lldp_hm.mib | 45,739 | 5,330 | MIB File |
| lldp_med.mib | 61,395 | 8,791 | MIB File |
| lldp_pno.mib | 19,712 | 3,612 | MIB File |
| Readme_08.0.07.txt | 45,497 | 13,433 | Text Document |
| Readme_RailSwitch.08.0.07.txt | 17,749 | 4,455 | Text Document |
| rsL2E.bin | 4,141,275 | 4,137,816 | BIN File |
| usrgrp.mib | 27,149 | 4,126 | MIB File |

# Firmware image structure



**Module 1.** Main application

**Module 2.** Pack200 archive -> JAR-file -> web interface applet

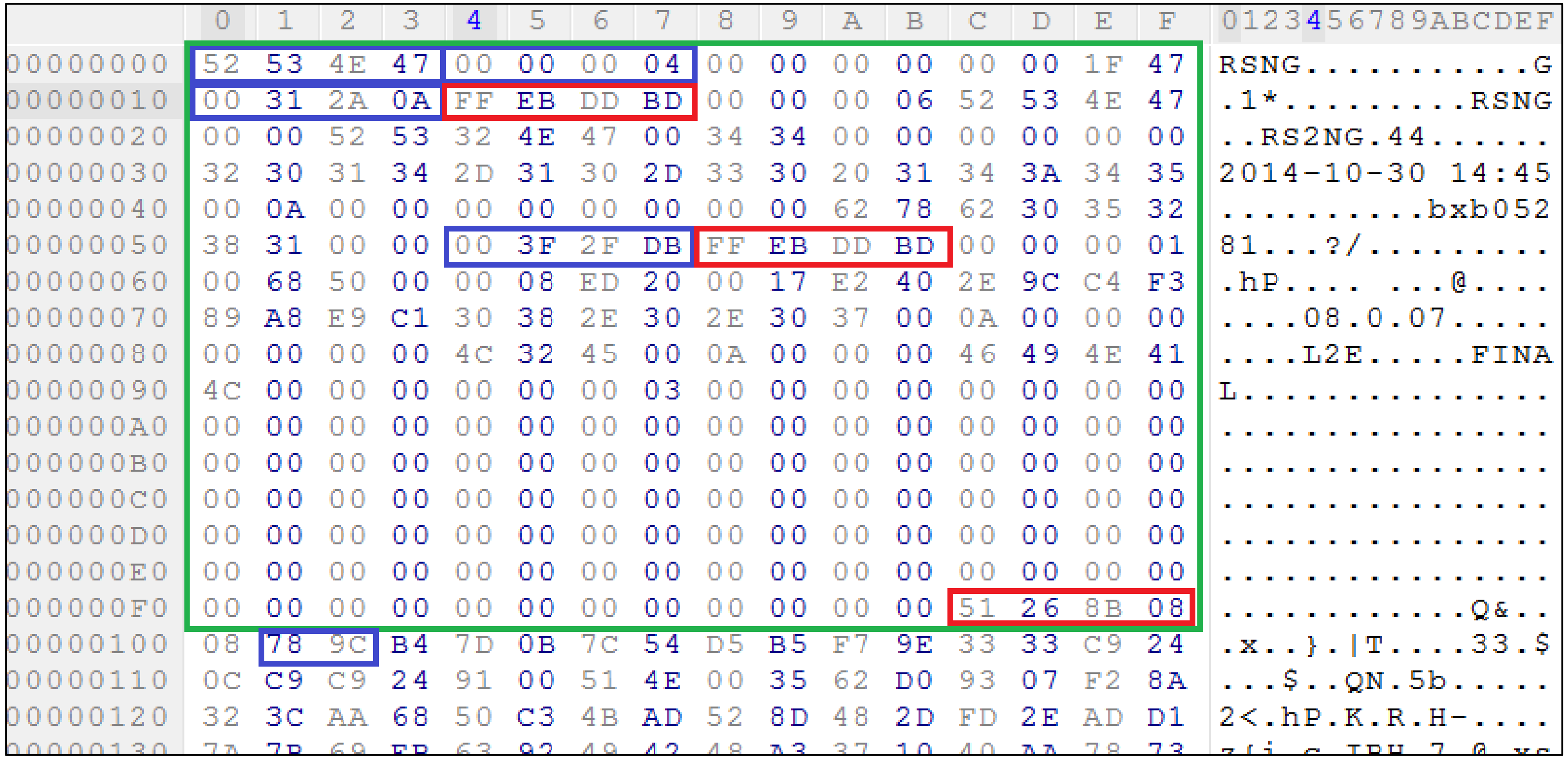
# Firmware image structure

## Module header

- 0x00 signature
- 0x04 file type
- 0x10 image size
- 0x14 image crc32

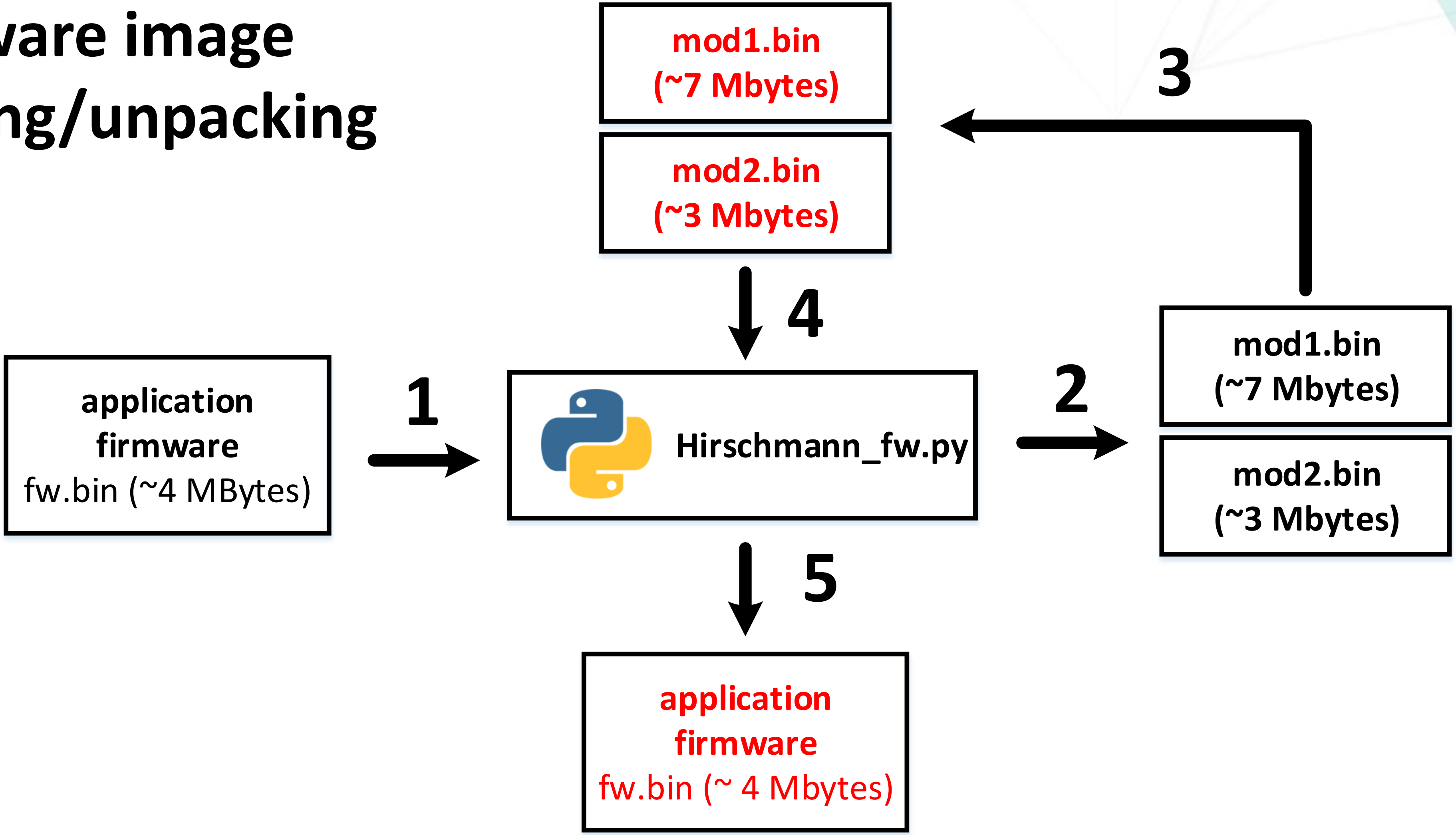
  …
- 0x54 eof offset
- 0x58 file crc32
- 0xFC header crc32

No identity verification



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
00000000  52 53 4E 47 00 00 00 04 00 00 00 00 00 00 1F 47  RSNG...........G
00000010  00 31 2A 0A FF EB DD BD 00 00 00 06 52 53 4E 47  .1*.........RSNG
00000020  00 00 52 53 32 4E 47 00 34 34 00 00 00 00 00 00  ..RS2NG.44......
00000030  32 30 31 34 2D 31 30 2D 33 30 20 31 34 3A 34 35  2014-10-30 14:45
00000040  00 0A 00 00 00 00 00 00 00 00 62 78 62 30 35 32  ..........bxb052
00000050  38 31 00 00 00 3F 2F DB FF EB DD BD 00 00 00 01  81...?/.........
00000060  00 68 50 00 00 08 ED 20 00 17 E2 40 2E 9C C4 F3  .hP.... ...@....
00000070  89 A8 E9 C1 30 38 2E 30 2E 30 37 00 0A 00 00 00  ....08.0.07.....
00000080  00 00 00 00 4C 32 45 00 0A 00 00 00 46 49 4E 41  ....L2E.....FINA
00000090  4C 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00  L...............
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 51 26 8B 08  ............Q&..
00000100  08 78 9C B4 7D 0B 7C 54 D5 B5 F7 9E 33 33 C9 24  .x..}.|T....33.$
00000110  0C C9 C9 24 91 00 51 4E 00 35 62 D0 93 07 F2 8A  ...$..QN.5b.....
00000120  32 3C AA 68 50 C3 4B AD 52 8D 48 2D FD 2E AD D1  2<.hP.K.R.H-....
00000130  7A 7B 69 FB 63 92 49 42 48 A3 37 10 40 AA 78 73  z{i.c.IBH.7.@.xs
```

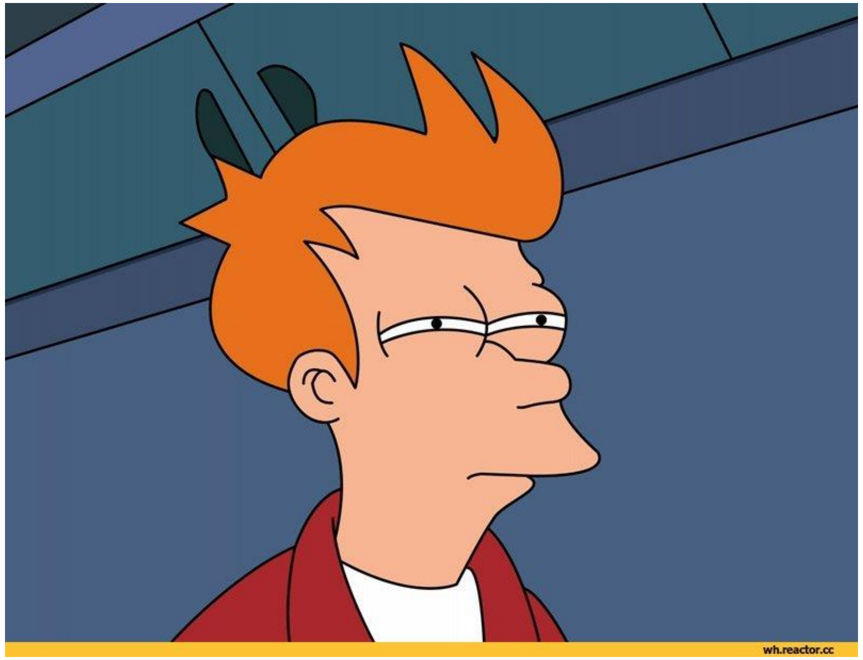
# Firmware image packing/unpacking



# Aren’t firmware modules too big?

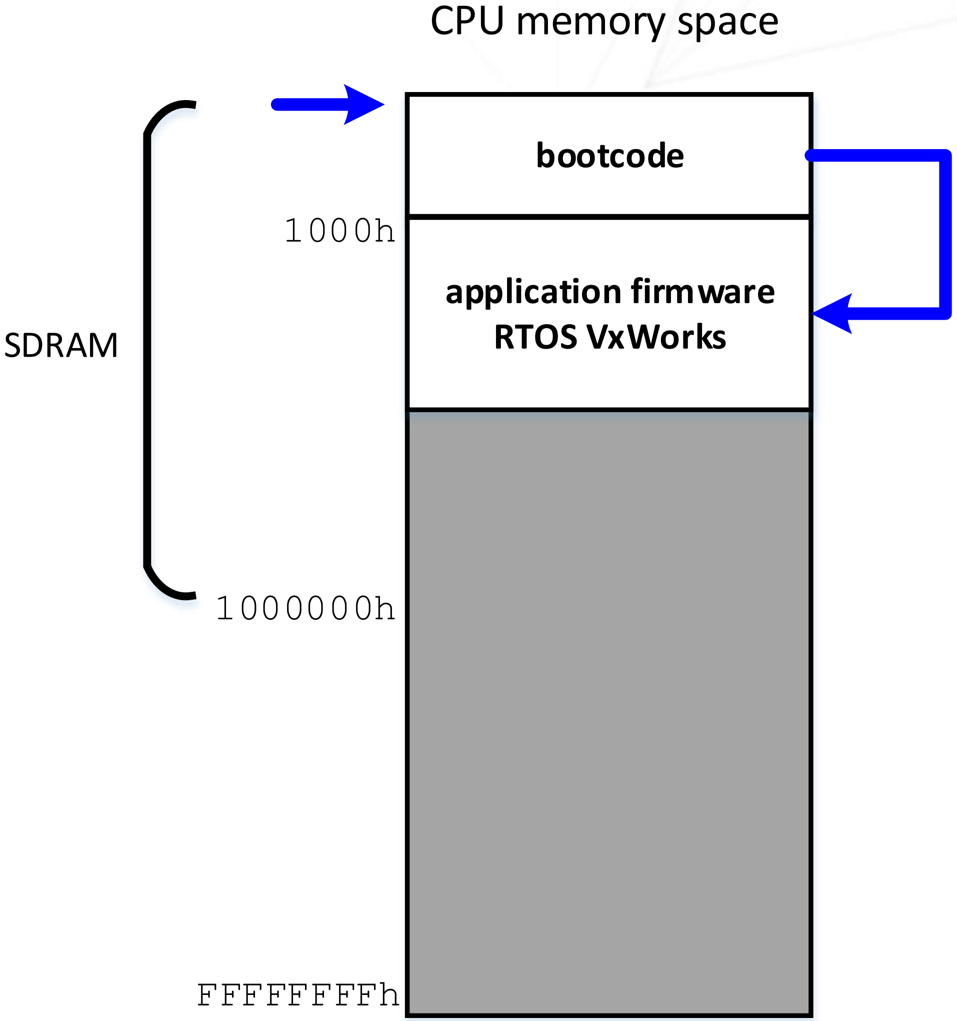
Unpacked modules are ~ 10 Mbytes
But the flash memory size is 8 Mbytes

So, there must be some kind of a bootloader...



# Firmware analysis

Booting process…



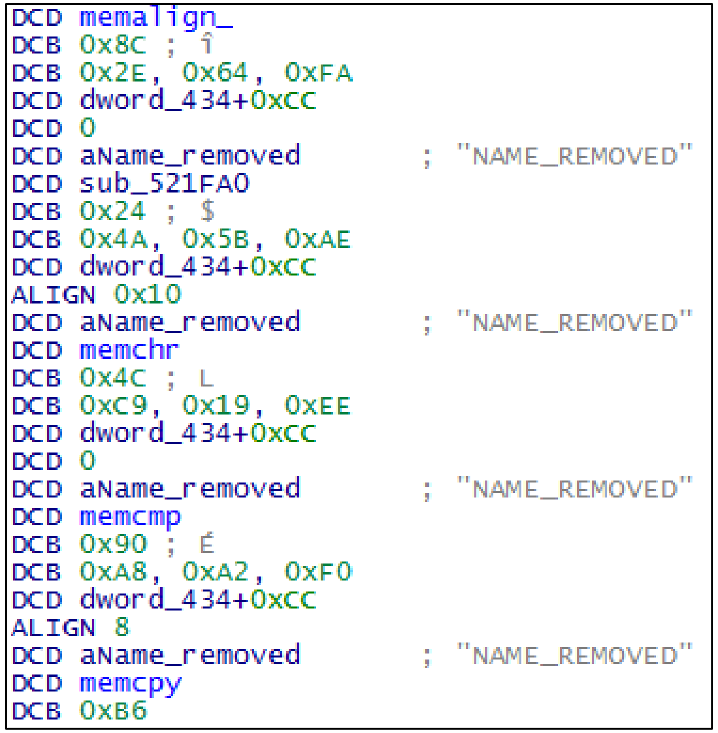
# Firmware analysis

## Operating system

RTOS VxWorks 5.4.2

old version (latest version 7)

- Found sources for 5.5
- Helps identifying libc-routines and some OS-specific routines
- Usually, VxWorks images have symbols table in the end of the image (definitely not in this case)



```
DCD memalign_
DCB 0x8C ; î
DCB 0x2E, 0x64, 0xFA
DCD dword_434+0xCC
DCD 0
DCD aName_removed         ; "NAME_REMOVED"
DCD sub_521FA0
DCB 0x24 ; $
DCB 0x4A, 0x5B, 0xAE
DCD dword_434+0xCC
ALIGN 0x10
DCD aName_removed         ; "NAME_REMOVED"
DCD memchr
DCB 0x4C ; L
DCB 0xC9, 0x19, 0xEE
DCD dword_434+0xCC
DCD 0
DCD aName_removed         ; "NAME_REMOVED"
DCD memcmp
DCB 0x90 ; É
DCB 0xA8, 0xA2, 0xF0
DCD dword_434+0xCC
ALIGN 8
DCD aName_removed         ; "NAME_REMOVED"
DCD memcpy
DCB 0xB6
```

# Firmware analysis

## Operating system

| | |
|---|---|
| DEP | no |
| Stack cookies | no |
| SafeSEH | no |
| ASLR | no |

No security technologies to protect against binary vulnerabilities exploitation

# Firmware analysis

## Operating system

Known vulnerabilities:

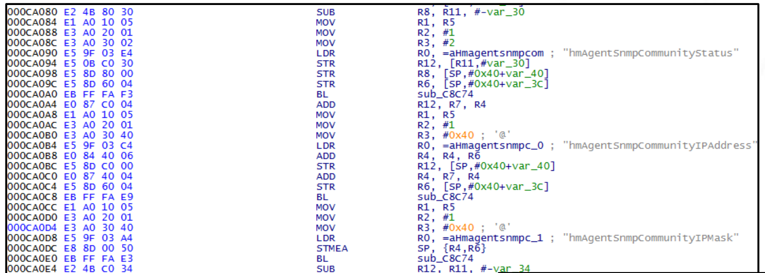
| | |
|---|---|
| CVE-2015-3963 | spoof TCP sessions |
| CVE-2010-2968 | brute-force |
| CVE-2010-2967 | obtain access |
| CVE-2010-2966 | obtain access |
| CVE-2010-2965 | RCE |
| CVE-2008-2476 | DoS |

…

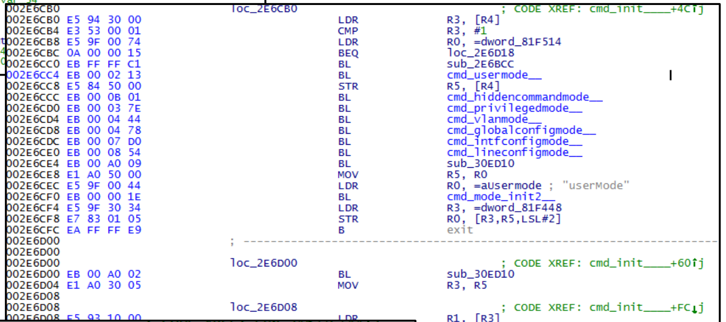
# Firmware analysis

Interesting functionality:

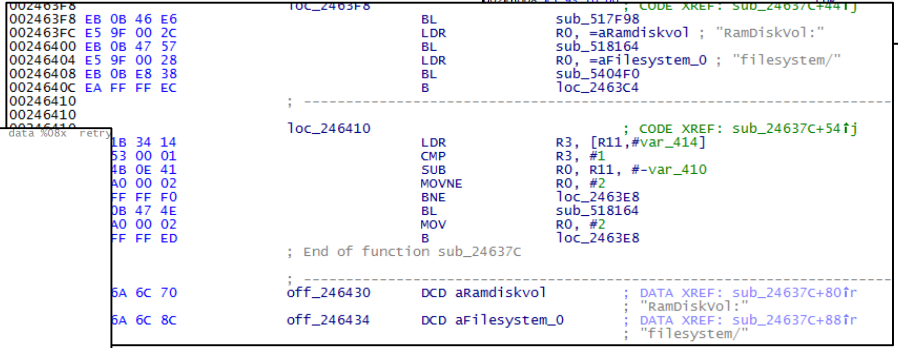
- SNMP traffic handlers
- Console commands interpreter
- Flash read/write
- Marvell CPLD flash read/write
- …



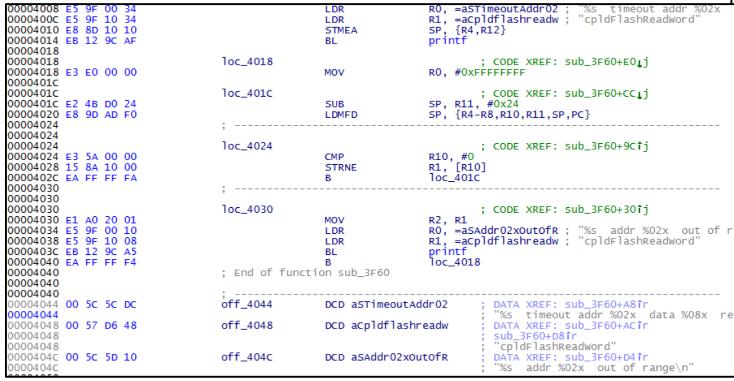
```
000CA080 E2 4B 80 30          SUB     R8, R11, #-var_30
000CA084 E1 A0 10 05          MOV     R1, R5
000CA088 E3 A0 20 01          MOV     R2, #1
000CA08C E3 A0 30 02          MOV     R3, #2
000CA090 E5 9F 03 E4          LDR     R0, =aHmagentsnmpcom ; "hmAgentSnmpCommunityStatus"
000CA094 E5 0B C0 30          STR     R12, [R11,#var_30]
000CA098 E5 8D 80 00          STR     R8, [SP,#0x40+var_40]
000CA09C E5 8D 60 04          STR     R6, [SP,#0x40+var_3C]
000CA0A0 EB FF FA F3          BL      sub_C8C74
000CA0A4 E0 87 C0 04          ADD     R12, R7, R4
000CA0A8 E1 A0 10 05          MOV     R1, R5
000CA0AC E3 A0 20 01          MOV     R2, #1
000CA0B0 E3 A0 30 40          MOV     R3, #0x40 ; '@'
000CA0B4 E5 9F 03 C4          LDR     R0, =aHmagentsnmpc_0 ; "hmAgentSnmpCommunityIPAddress"
000CA0B8 E0 84 40 06          ADD     R4, R4, R6
000CA0BC E5 8D C0 00          STR     R12, [SP,#0x40+var_40]
000CA0C0 E0 87 40 04          ADD     R4, R7, R4
000CA0C4 E5 8D 60 04          STR     R6, [SP,#0x40+var_3C]
000CA0C8 EB FF FA E9          BL      sub_C8C74
000CA0CC E1 A0 10 05          MOV     R1, R5
000CA0D0 E3 A0 20 01          MOV     R2, #1
000CA0D4 E3 A0 30 40          MOV     R3, #0x40 ; '@'
000CA0D8 E5 9F 03 A4          LDR     R0, =aHmagentsnmpc_1 ; "hmAgentSnmpCommunityIPMask"
000CA0DC E8 8D 00 50          STMEA   SP, {R4,R6}
000CA0E0 EB FF FA E3          BL      sub_C8C74
000CA0E4 E2 4B C0 34          SUB     R12, R11, #-var_34
000CA0E8 E1 A0 10 05          MOV     R1, R5
000CA0EC E3 A0 20 01          MOV     R2, #1
000CA0F0 E3 A0 30 02          MOV     R3, #2
000CA0F4 E5 9F 03 8C          LDR     R0, =aHmagent
000CA0F8 E5 8D C0 00          STR     R12, [SP,#0x4
000CA0FC E5 8D 60 04          STR     R6, [SP,#0x40
000CA100 EB FF FA DB          BL      sub_C8C74
```



```
002E6CB0                      loc_2E6CB0                        ; CODE XREF: cmd_init____+4C↑j
002E6CB0 E5 94 30 00                      LDR     R3, [R4]
002E6CB4 E3 53 00 01                      CMP     R3, #1
002E6CB8 E5 9F 00 74                      LDR     R0, =dword_81F514
002E6CBC 0A 00 00 15                      BEQ     loc_2E6D18
002E6CC0 EB FF FF C1                      BL      sub_2E6BCC
002E6CC4 EB 00 02 13                      BL      cmd_usermode__
002E6CC8 E5 84 50 00                      STR     R5, [R4]
002E6CCC EB 00 0B 01                      BL      cmd_hiddencommandmode__
002E6CD0 EB 00 03 7E                      BL      cmd_privilegedmode__
002E6CD4 EB 00 04 44                      BL      cmd_vlanmode__
002E6CD8 EB 00 04 78                      BL      cmd_globalconfigmode__
002E6CDC EB 00 07 D0                      BL      cmd_intfconfigmode__
002E6CE0 EB 00 08 54                      BL      cmd_lineconfigmode__
002E6CE4 EB 00 A0 09                      BL      sub_30ED10
002E6CE8 E1 A0 50 00                      MOV     R5, R0
002E6CEC E5 9F 00 44                      LDR     R0, =aUsermode ; "userMode"
002E6CF0 EB 00 00 1E                      BL      cmd_mode_init2__
002E6CF4 E5 9F 30 34                      LDR     R3, =dword_81F448
002E6CF8 E7 83 01 05                      STR     R0, [R3,R5,LSL#2]
002E6CFC EA FF FF E9                      B       exit
002E6D00                      ; ---------------------------------------------------------------
002E6D00
002E6D00                      loc_2E6D00                        ; CODE XREF: cmd_init____+60↑j
002E6D00 EB 00 A0 02                      BL      sub_30ED10
002E6D04 E1 A0 30 05                      MOV     R3, R5
002E6D08
002E6D08                      loc_2E6D08                        ; CODE XREF: cmd_init____+FC↓j
002E6D08 E5 93 10 00                      LDR     R1, [R3]
                                                  R2, =dword_81F448
                                                  R1, [R2,R0,LSL#2]
                                                  exit
```



```
002463F8                      loc_2463F8                        ; CODE XREF: sub_24637C+44↑j
002463F8 EB 0B 46 E6                      BL      sub_517F98
002463FC E5 9F 00 2C                      LDR     R0, =aRamdiskvol ; "RamDiskVol:"
00246400 EB 0B 47 57                      BL      sub_518164
00246404 E5 9F 00 28                      LDR     R0, =aFilesystem_0 ; "filesystem/"
00246408 EB 0B E8 38                      BL      sub_5404F0
0024640C EA FF FF EC                      B       loc_2463C4
00246410                      ; ---------------------------------------------------------------
00246410
00246410                      loc_246410                        ; CODE XREF: sub_24637C+54↑j
         1B 34 14                         LDR     R3, [R11,#var_414]
         53 00 01                         CMP     R3, #1
         4B 0E 41                         SUB     R0, R11, #-var_410
         A0 00 02                         MOVNE   R0, #2
         FF FF F0                         BNE     loc_2463E8
         0B 47 4E                         BL      sub_518164
         A0 00 02                         MOV     R0, #2
         FF FF ED                         B       loc_2463E8
                              ; End of function sub_24637C

                              ; ---------------------------------------------------------------
         6A 6C 70             off_246430  DCD aRamdiskvol       ; DATA XREF: sub_24637C+80↑r
                                                                ; "RamDiskVol:"
         6A 6C 8C             off_246434  DCD aFilesystem_0     ; DATA XREF: sub_24637C+88↑r
                                                                ; "filesystem/"
```



```
00004008 E5 9F 00 34                      LDR     R0, =aSTimeoutAddr02 ; "%s  timeout addr %02x  data %08x  retry
0000400C E5 9F 10 34                      LDR     R1, =aCpldflashreadw ; "cpldFlashReadWord"
00004010 E8 8D 10 10                      STMEA   SP, {R4,R12}
00004014 EB 12 9C AF                      BL      printf
00004018
00004018                      loc_4018                          ; CODE XREF: sub_3F60+E0↓j
00004018 E3 E0 00 00                      MOV     R0, #0xFFFFFFFF
0000401C
0000401C                      loc_401C                          ; CODE XREF: sub_3F60+CC↓j
0000401C E2 4B D0 24                      SUB     SP, R11, #0x24
00004020 E8 9D AD F0                      LDMFD   SP, {R4-R8,R10,R11,SP,PC}
00004024                      ; ---------------------------------------------------------------
00004024
00004024                      loc_4024                          ; CODE XREF: sub_3F60+9C↑j
00004024 E3 5A 00 00                      CMP     R10, #0
00004028 15 8A 10 00                      STRNE   R1, [R10]
0000402C EA FF FF FA                      B       loc_401C
00004030                      ; ---------------------------------------------------------------
00004030
00004030                      loc_4030                          ; CODE XREF: sub_3F60+30↑j
00004030 E1 A0 20 01                      MOV     R2, R1
00004034 E5 9F 00 10                      LDR     R0, =aSAddr02xOutOfR ; "%s  addr %02x  out of range\n"
00004038 E5 9F 10 08                      LDR     R1, =aCpldflashreadw ; "cpldFlashReadWord"
0000403C EB 12 9C A5                      BL      printf
00004040 EA FF FF F4                      B       loc_4018
00004040                      ; End of function sub_3F60
00004040
00004040                      ; ---------------------------------------------------------------
00004044 00 5C 5C DC          off_4044    DCD aSTimeoutAddr02   ; DATA XREF: sub_3F60+A8↑r
00004044                                                        ; "%s  timeout addr %02x  data %08x  retry"...
00004048 00 57 D6 48          off_4048    DCD aCpldflashreadw   ; DATA XREF: sub_3F60+AC↑r
00004048                                                        ; sub_3F60+D8↑r
00004048                                                        ; "cpldFlashReadWord"
0000404C 00 5C 5D 10          off_404C    DCD aSAddr02xOutOfR   ; DATA XREF: sub_3F60+D4↑r
0000404C                                                        ; "%s  addr %02x  out of range\n"
```

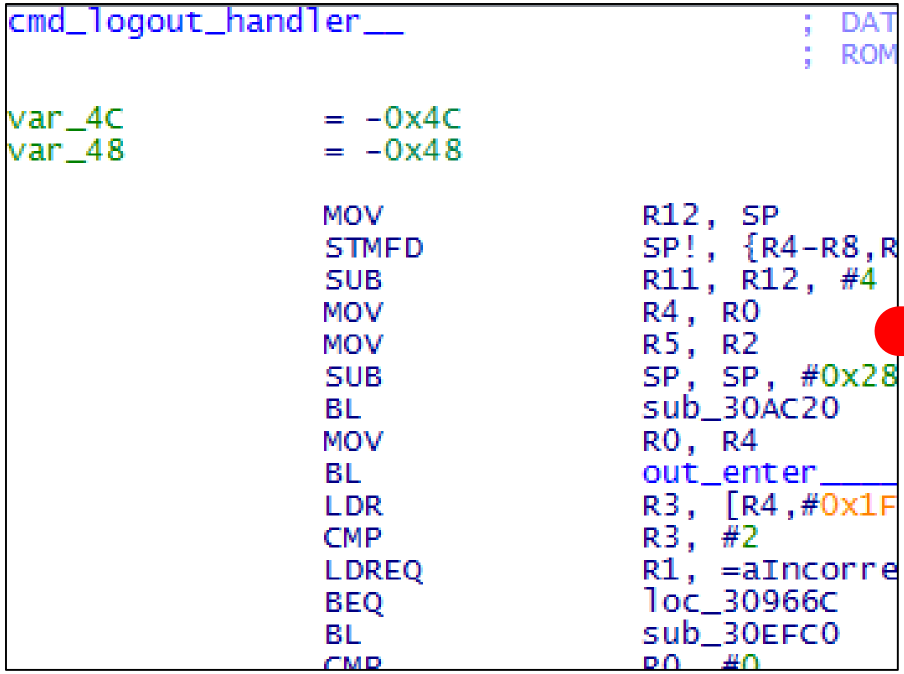
# Modifying firmware

Main requirement for testing:
the injection mustn’t brick the device

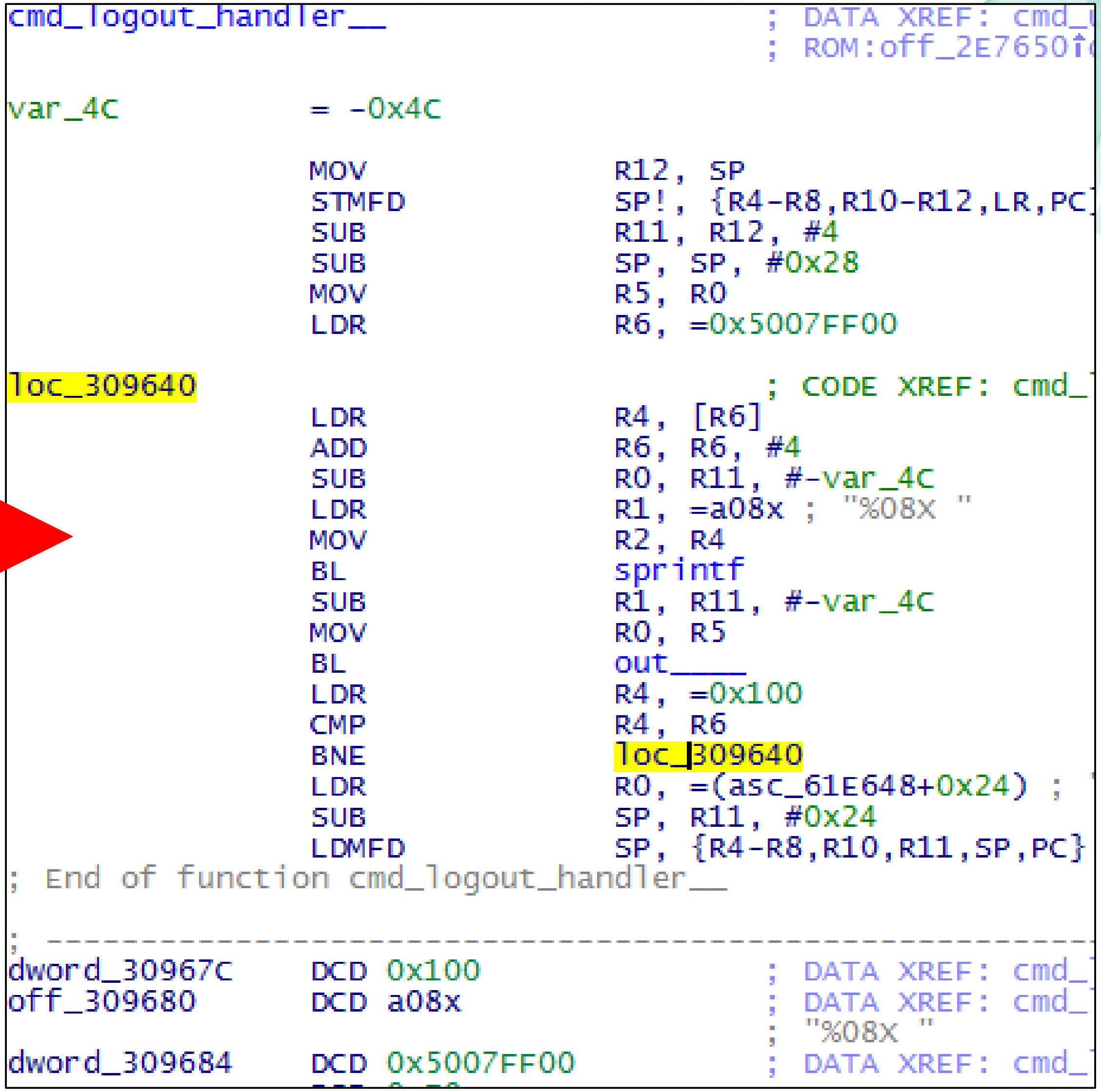
Means that the injected code must be executed on-call

Decided to pick up one of command handler:
“logout” was a good place to start…

# Modifying firmware



```
cmd_logout_handler__                                 ; DAT
                                                     ; ROM

var_4C          = -0x4C
var_48          = -0x48

                MOV             R12, SP
                STMFD           SP!, {R4-R8,R
                SUB             R11, R12, #4
                MOV             R4, R0
                MOV             R5, R2
                SUB             SP, SP, #0x28
                BL              sub_30AC20
                MOV             R0, R4
                BL              out_enter____
                LDR             R3, [R4,#0x1F
                CMP             R3, #2
                LDREQ           R1, =aIncorre
                BEQ             loc_30966C
                BL              sub_30EFC0
                CMP             R0, #0
```

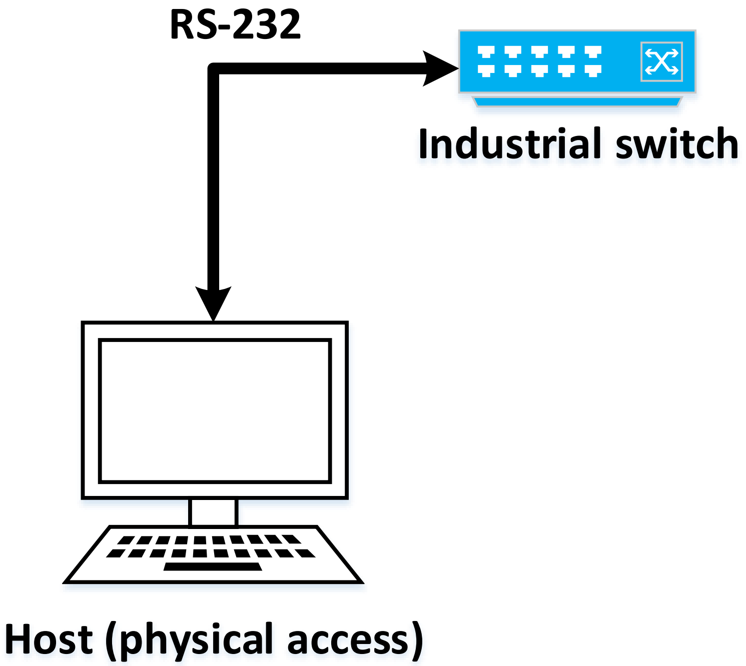


```
cmd_logout_handler__                                 ; DATA XREF: cmd_
                                                     ; ROM:off_2E7650↑

var_4C          = -0x4C

                MOV             R12, SP
                STMFD           SP!, {R4-R8,R10-R12,LR,PC}
                SUB             R11, R12, #4
                SUB             SP, SP, #0x28
                MOV             R5, R0
                LDR             R6, =0x5007FF00

loc_309640                                           ; CODE XREF: cmd_
                LDR             R4, [R6]
                ADD             R6, R6, #4
                SUB             R0, R11, #-var_4C
                LDR             R1, =a08x ; "%08X "
                MOV             R2, R4
                BL              sprintf
                SUB             R1, R11, #-var_4C
                MOV             R0, R5
                BL              out____
                LDR             R4, =0x100
                CMP             R4, R6
                BNE             loc_309640
                LDR             R0, =(asc_61E648+0x24) ;
                SUB             SP, R11, #0x24
                LDMFD           SP, {R4-R8,R10,R11,SP,PC}
; End of function cmd_logout_handler__

; ---------------------------------------------------------------------------
dword_30967C    DCD 0x100                            ; DATA XREF: cmd_
off_309680      DCD a08x                             ; DATA XREF: cmd_
                                                     ; "%08X "
dword_309684    DCD 0x5007FF00                       ; DATA XREF: cmd_
```

# DEMO 01

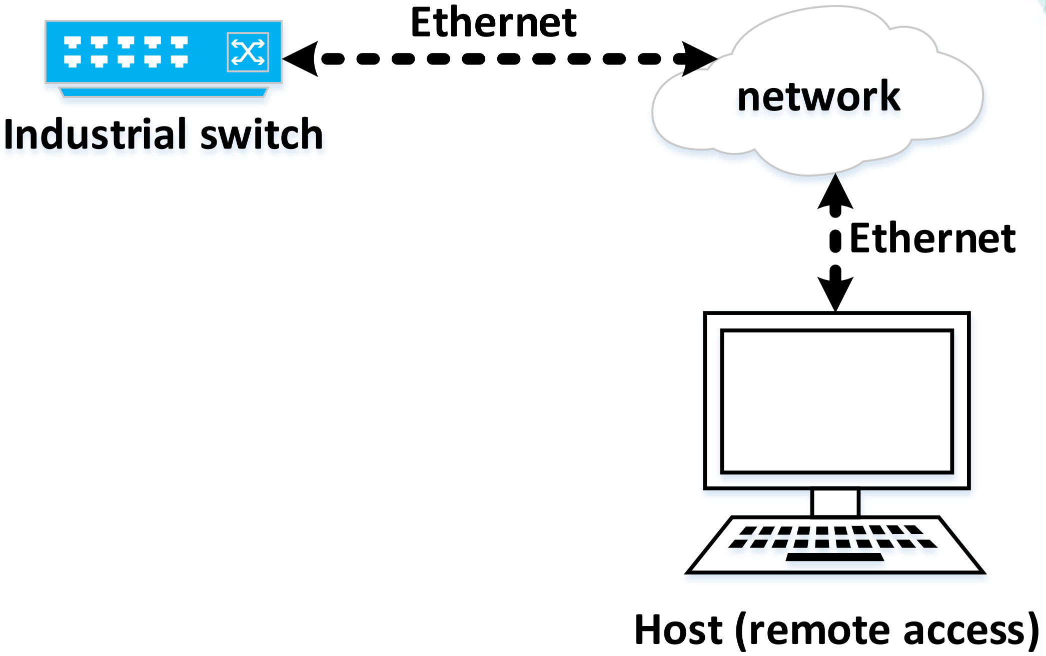
# Firmware modification scenario

- No authentication required
- Firmware image can be transferred to the switch via XMODEM protocol or USB interface



# Firmware modification scenario

- Authentication is required
  - try default login/password
  - try to brute-force
  - try to exploit vulnerability



# Firmware modification conclusion

We have a capability to modify the switch firmware:

- Execute code on the switch
- Execute code on the PC client (JVM)

The original firmware can be easily restored by standard firmware update operation

# How can the modified firmware survive the update process?

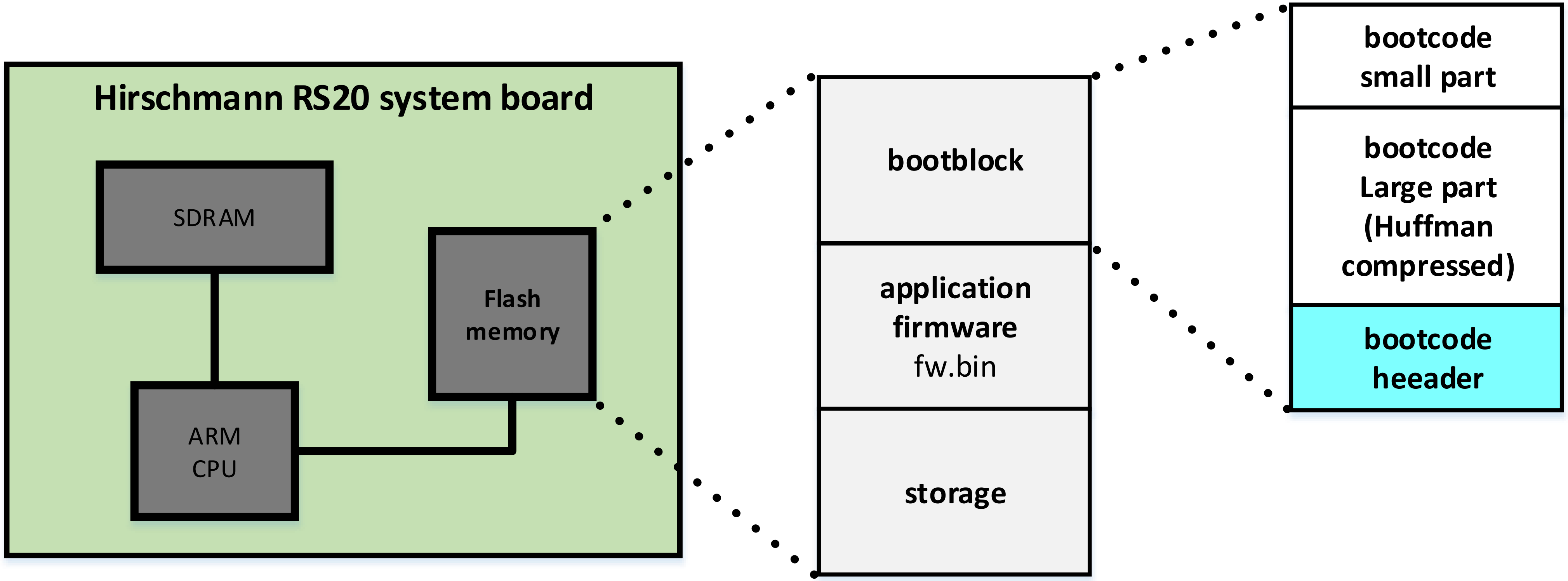
So we though of the bootcode!

# Bootcode extraction

1. Load first 1000h of SRAM
   no sign of bootcode

2. Use NVRAM read/write routines
   have full dump of the flash memory

# Bootcode structure


Hirschmann RS20 system board
SDRAM
Flash memory
ARM CPU
bootblock
application firmware fw.bin
storage
bootcode small part
bootcode Large part (Huffman compressed)
bootcode heeader

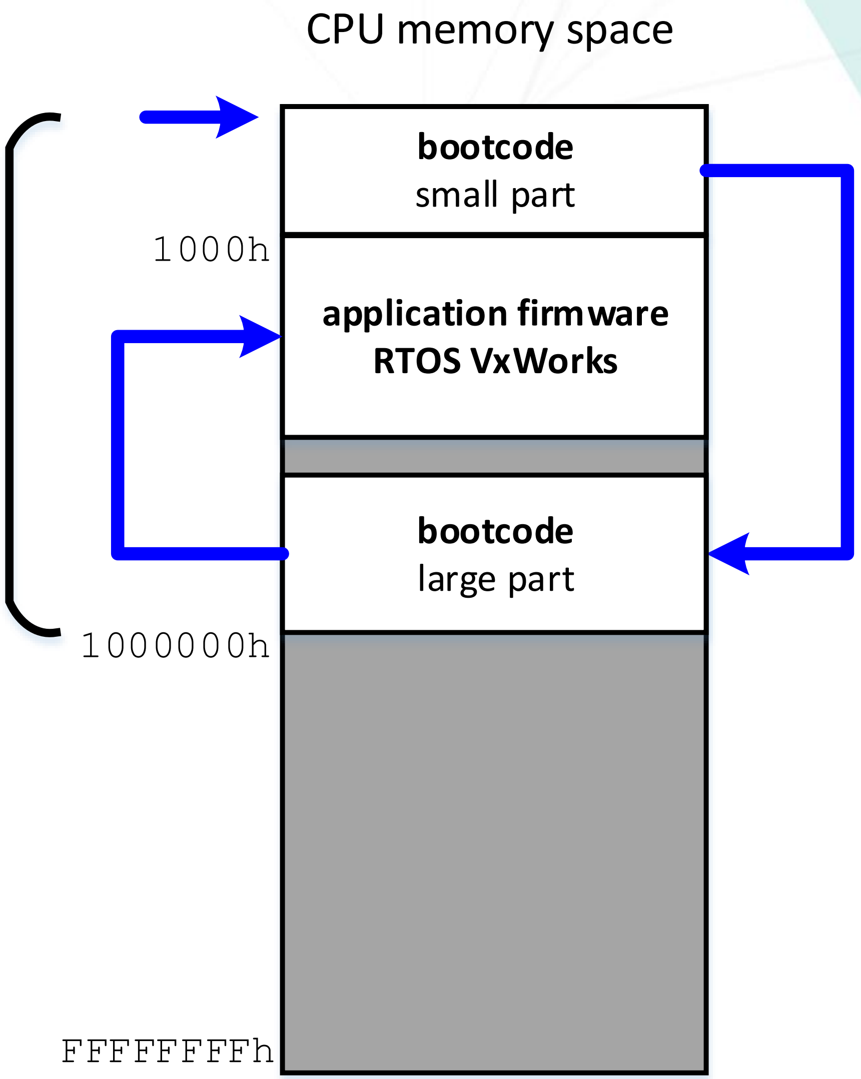
# Bootcode analysis

Small part:

- Configure memory
- Load up and execute the large part

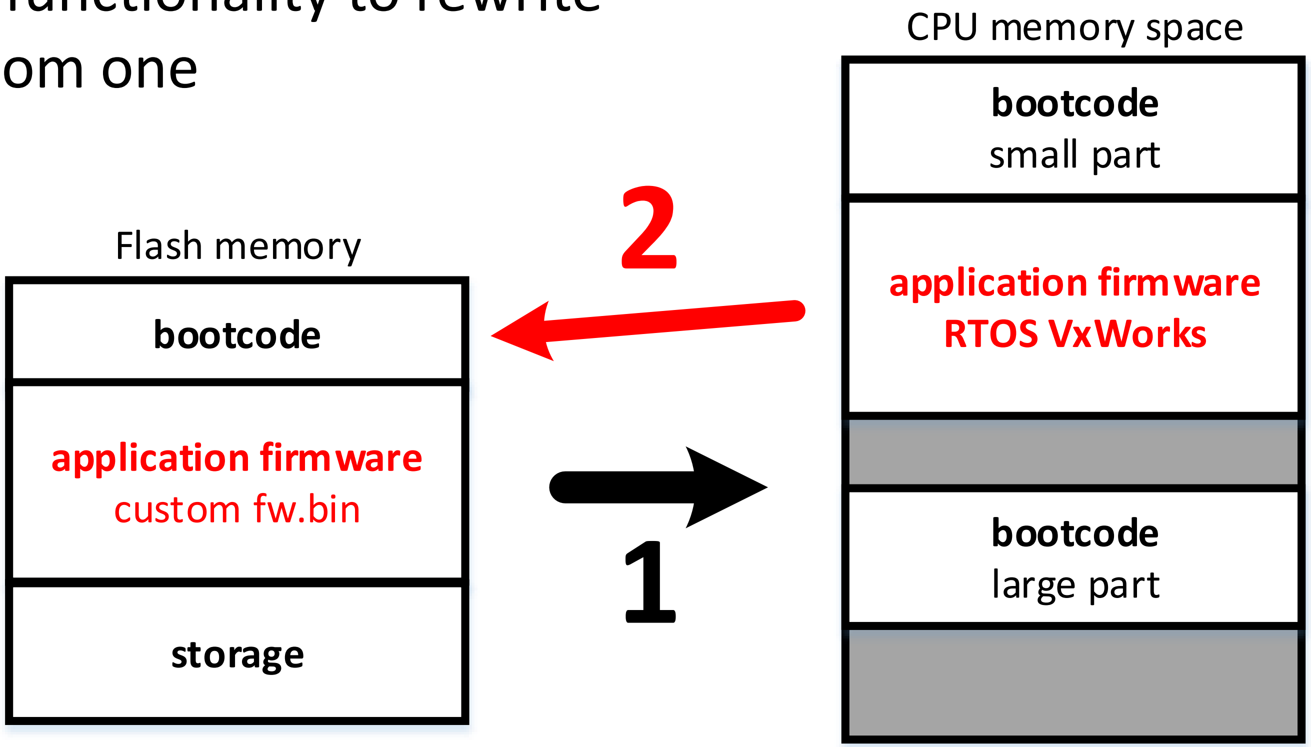
Large part:

- Initialize CPU hardware
- Configure interrupt model
- Load and execute an application firmware



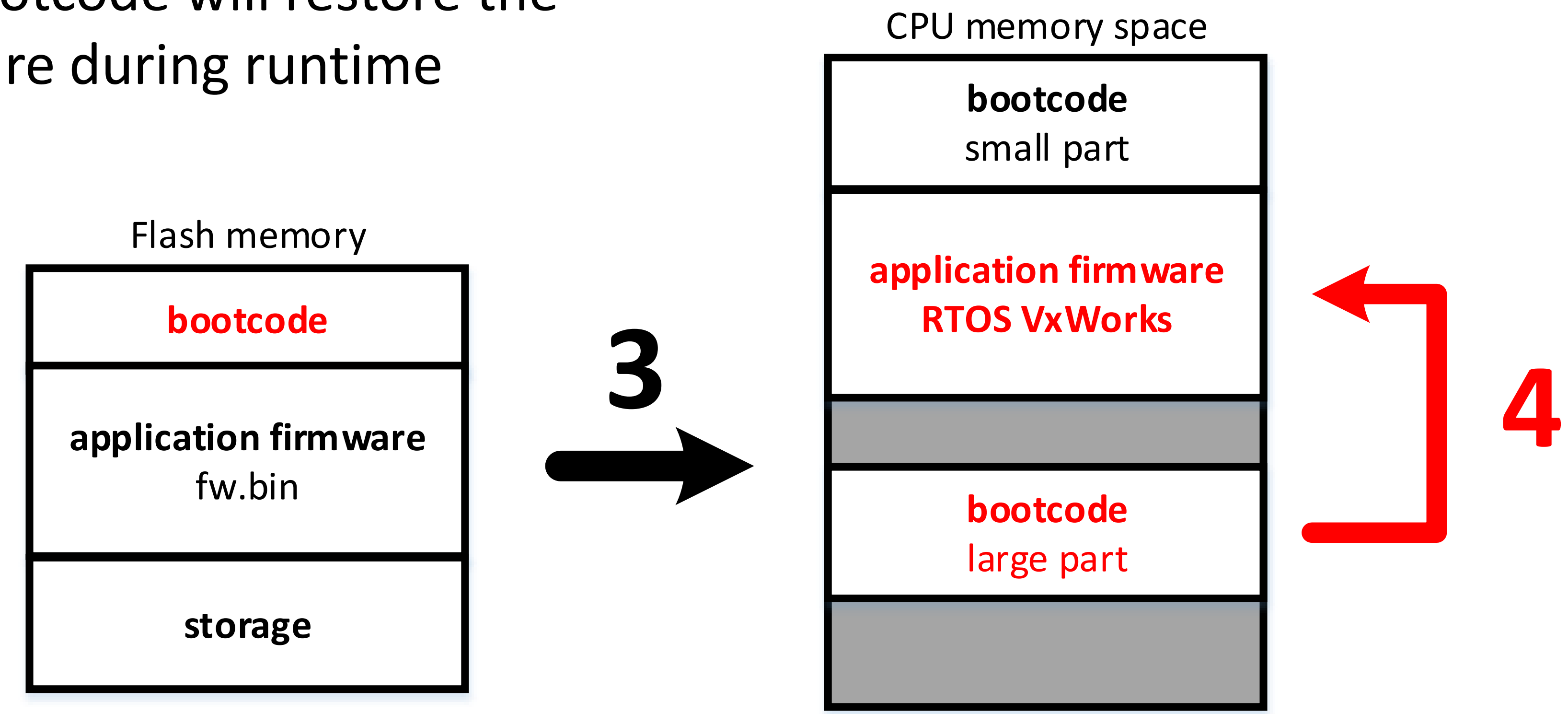
# Bootcode modification

Load up the firmware with functionality to rewrite the bootcode with the custom one



# Bootcode modification

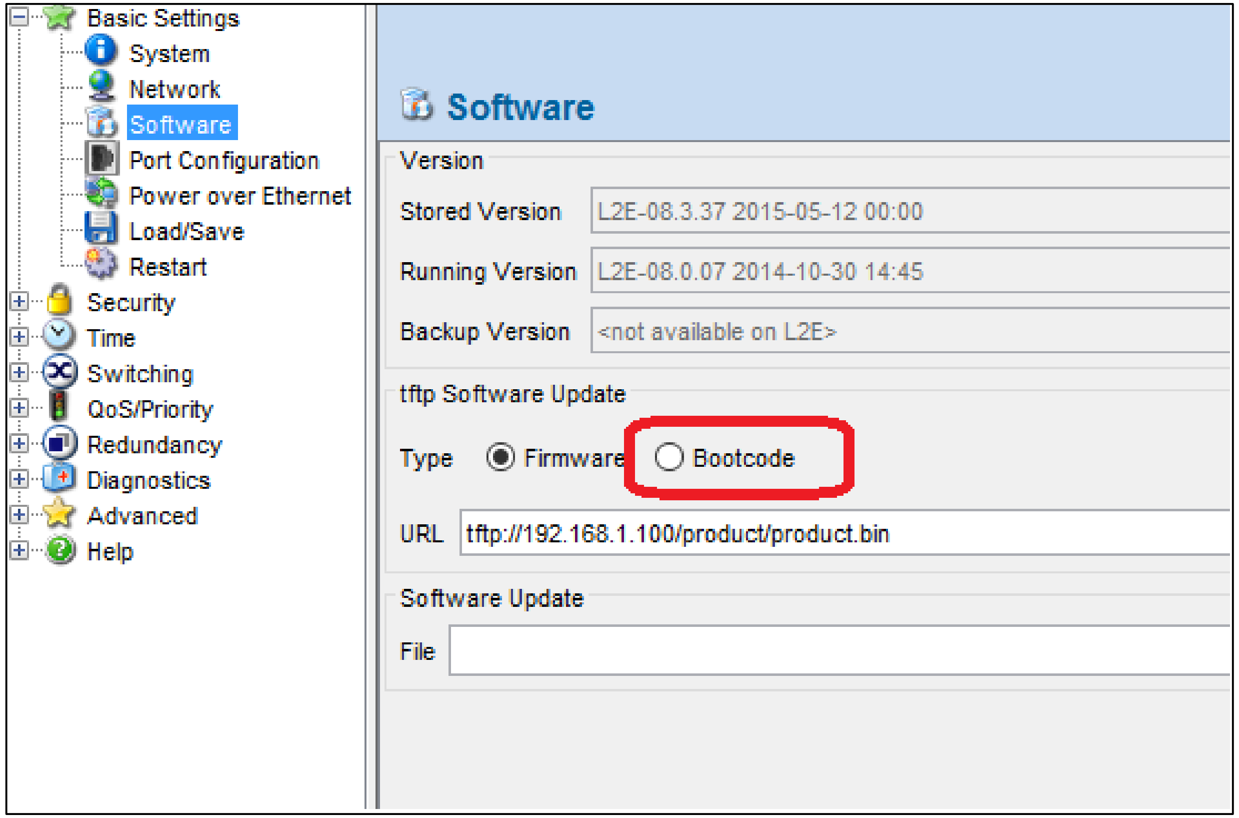
Once modified, the bootcode will restore the injection in the firmware during runtime



# Can the bootcode be legally updated?

1. Found undocumented functionality to update the bootcode from console (but it’s unused)

2. Found the capability to update the bootcode by network, but it seems to be not that simple…



Software

Version

Stored Version: L2E-08.3.37 2015-05-12 00:00

Running Version: L2E-08.0.07 2014-10-30 14:45

Backup Version: <not available on L2E>

tftp Software Update

Type ● Firmware ○ Bootcode

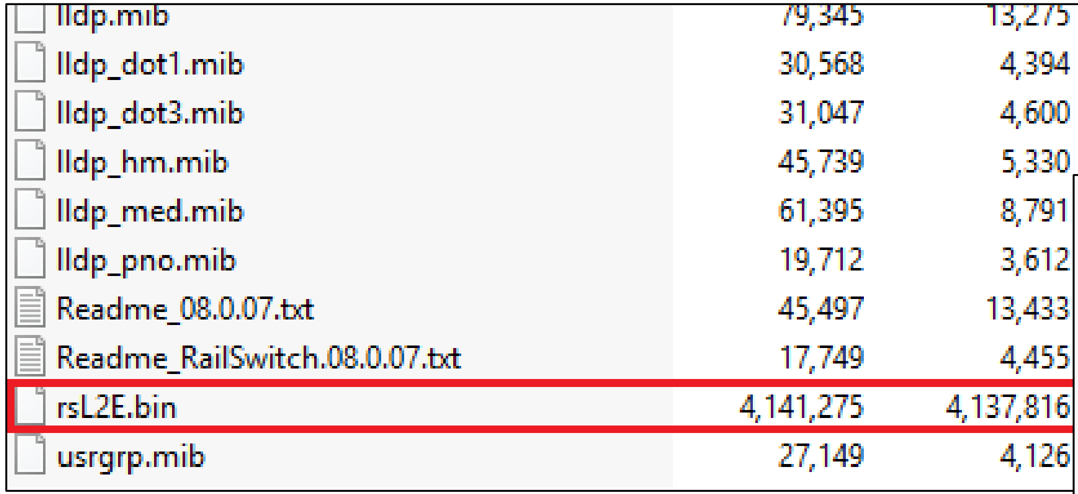
URL tftp://192.168.1.100/product/product.bin

Software Update

File

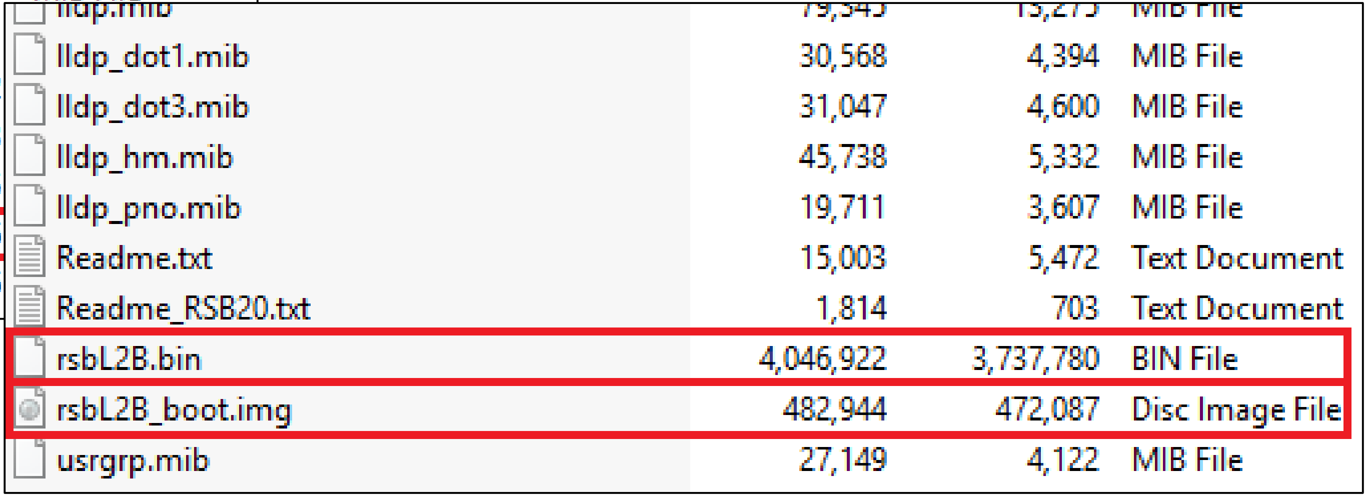
# Where to get the bootcode image?

RS20 device update archive



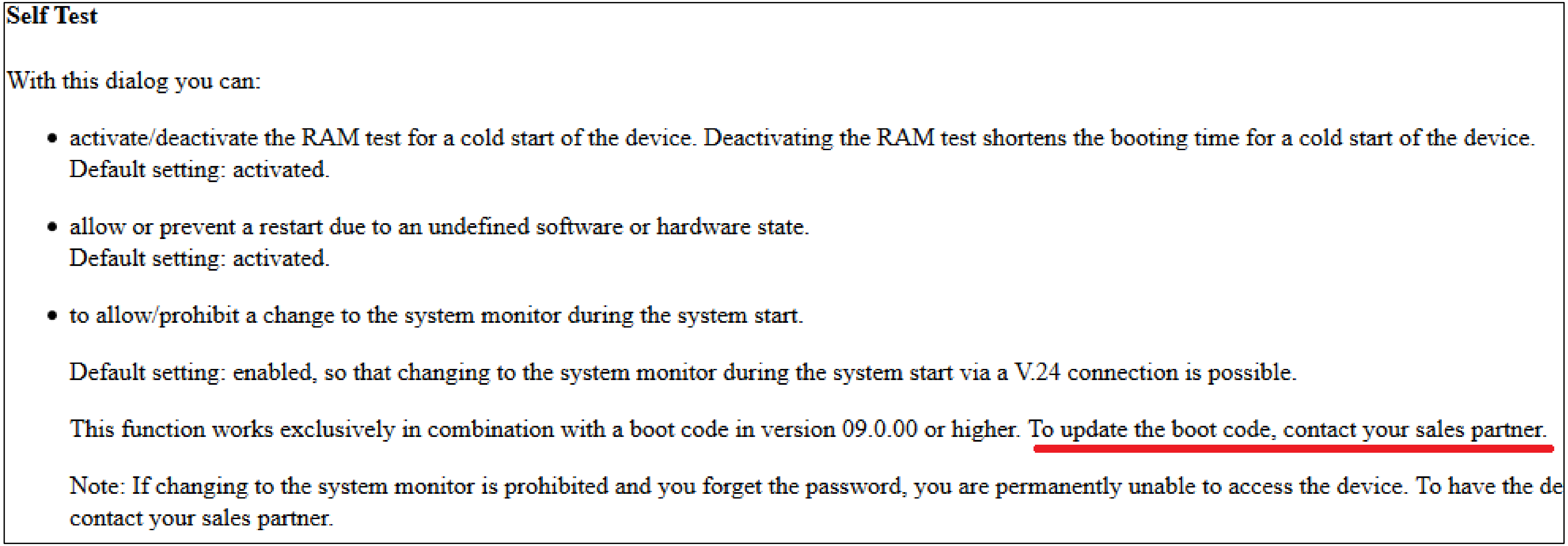
| | | | |
|---|---|---|---|
| lldp.mib | 79,345 | 13,275 | MIB File |
| lldp_dot1.mib | 30,568 | 4,394 | MIB File |
| lldp_dot3.mib | 31,047 | 4,600 | MIB File |
| lldp_hm.mib | 45,739 | 5,330 | MIB File |
| lldp_med.mib | 61,395 | 8,791 | |
| lldp_pno.mib | 19,712 | 3,612 | |
| Readme_08.0.07.txt | 45,497 | 13,433 | |
| Readme_RailSwitch.08.0.07.txt | 17,749 | 4,455 | |
| rsL2E.bin | 4,141,275 | 4,137,816 | |
| usrgrp.mib | 27,149 | 4,126 | |

RSB device update archive



| | | | |
|---|---|---|---|
| lldp.mib | 79,345 | 13,275 | MIB File |
| lldp_dot1.mib | 30,568 | 4,394 | MIB File |
| lldp_dot3.mib | 31,047 | 4,600 | MIB File |
| lldp_hm.mib | 45,738 | 5,332 | MIB File |
| lldp_pno.mib | 19,711 | 3,607 | MIB File |
| Readme.txt | 15,003 | 5,472 | Text Document |
| Readme_RSB20.txt | 1,814 | 703 | Text Document |
| rsbL2B.bin | 4,046,922 | 3,737,780 | BIN File |
| rsbL2B_boot.img | 482,944 | 472,087 | Disc Image File |
| usrgrp.mib | 27,149 | 4,122 | MIB File |

# Where to get the bootcode image?



**Self Test**

With this dialog you can:

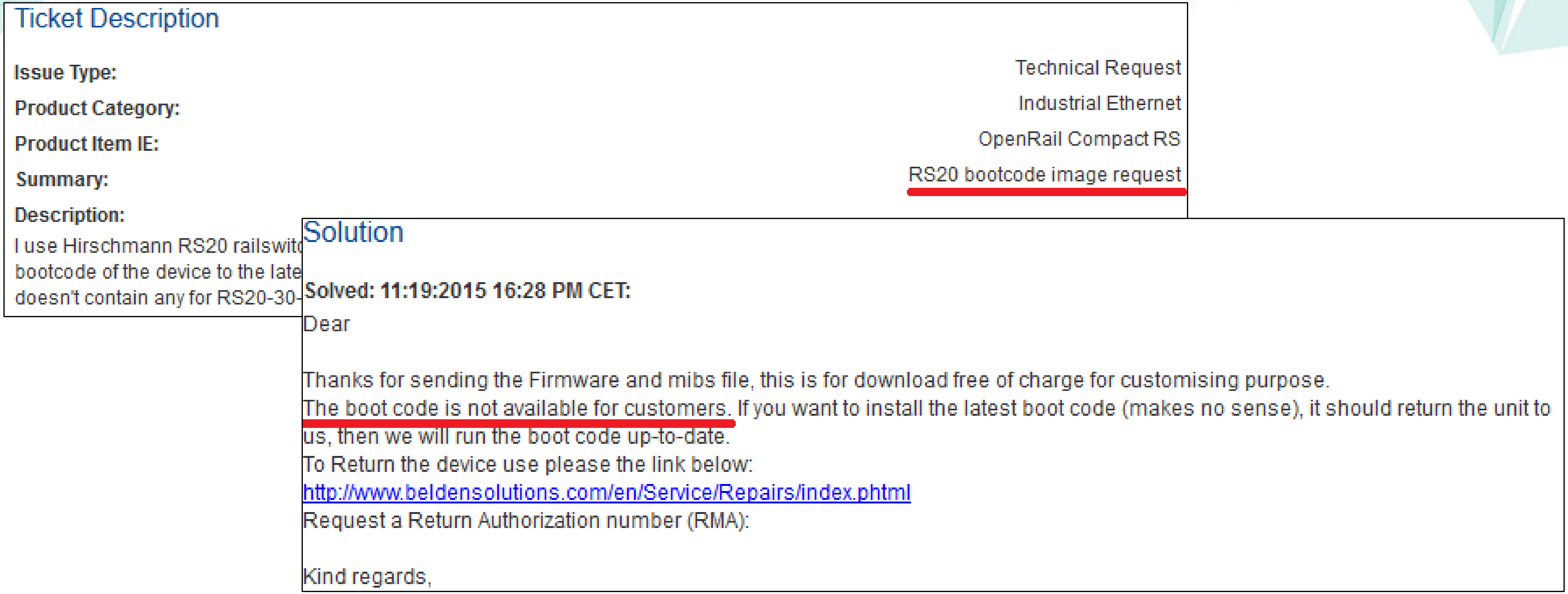
- activate/deactivate the RAM test for a cold start of the device. Deactivating the RAM test shortens the booting time for a cold start of the device.
  Default setting: activated.
- allow or prevent a restart due to an undefined software or hardware state.
  Default setting: activated.
- to allow/prohibit a change to the system monitor during the system start.

  Default setting: enabled, so that changing to the system monitor during the system start via a V.24 connection is possible.

  This function works exclusively in combination with a boot code in version 09.0.00 or higher. To update the boot code, contact your sales partner.

  Note: If changing to the system monitor is prohibited and you forget the password, you are permanently unable to access the device. To have the de
  contact your sales partner.

# Where to get the bootcode image?



Ticket Description

Issue Type: Technical Request

Product Category: Industrial Ethernet

Product Item IE: OpenRail Compact RS

Summary: RS20 bootcode image request

Description:

I use Hirschmann RS20 railswitc
bootcode of the device to the late
doesn't contain any for RS20-30-

Solution

**Solved: 11:19:2015 16:28 PM CET:**

Dear

Thanks for sending the Firmware and mibs file, this is for download free of charge for customising purpose.
The boot code is not available for customers. If you want to install the latest boot code (makes no sense), it should return the unit to us, then we will run the boot code up-to-date.
To Return the device use please the link below:
http://www.beldensolutions.com/en/Service/Repairs/index.phtml
Request a Return Authorization number (RMA):

Kind regards,

# Bootcode modification conclusion

We have capability to:

- hide in the bootcode
- restore any injections into firmware during boot

Theoretically, it can be restored the original image

# How to survive the bootcode update process?

Let’s try to dig in a bit deeper…

# CPLD flash modification capability

CPLD (Complex Programmable Logic Device) is type of a PLD (Programmable Logic Device)

Logic is defined via hardware description language (VHDL, Verilog, …)

Has a flash configuration memory

# Phoenix Contact
# FL SWITCH MM HS

# Onboard hardware

1. **CPU**

   PMC RM5231A

   MIPS IV 32-bit, no internal memory

2. **SDRAM**

   Micron MT48LC8M16A2

   16 MB 2x = 32MB

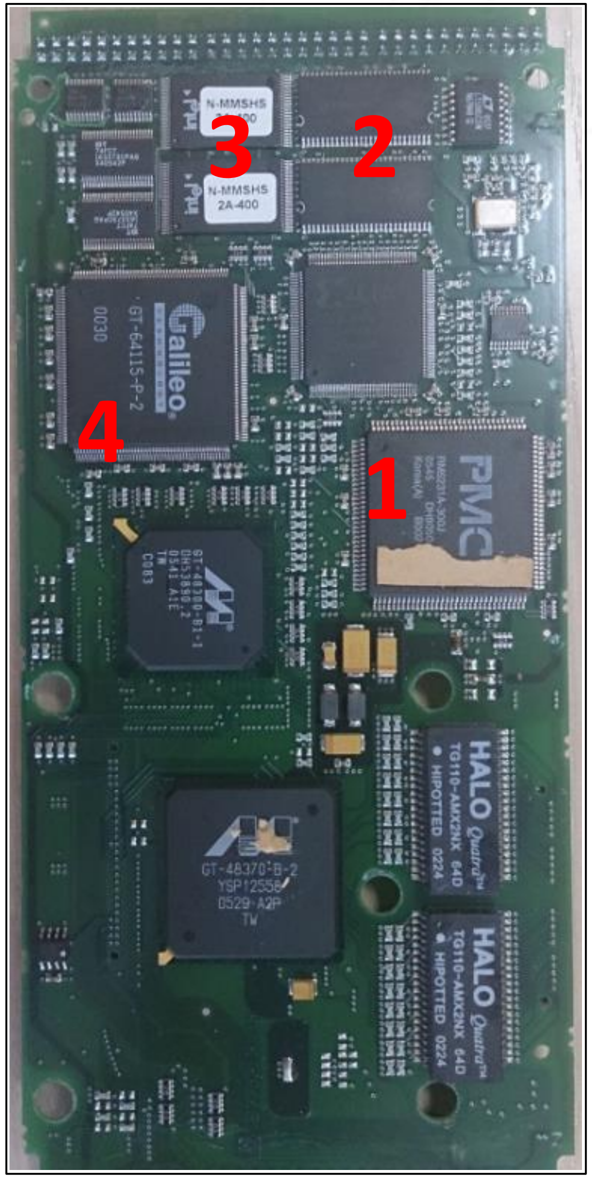
3. **Flash memory**

   Intel ????

   NAND

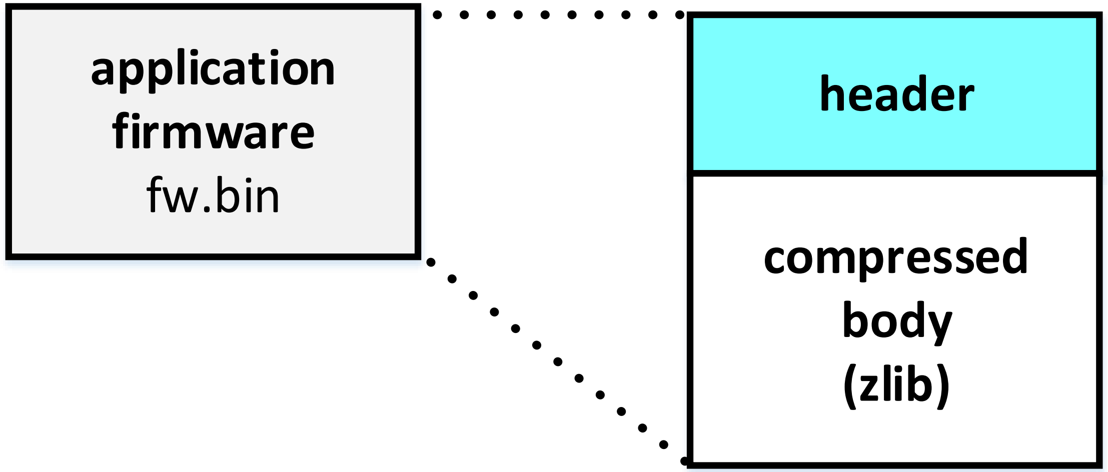
4. **Chipset**

   Galileo GT-64115



# Firmware image structure

Downloaded from
Phoenix Contact official


**application firmware**
fw.bin

**header**

**compressed body (zlib)**

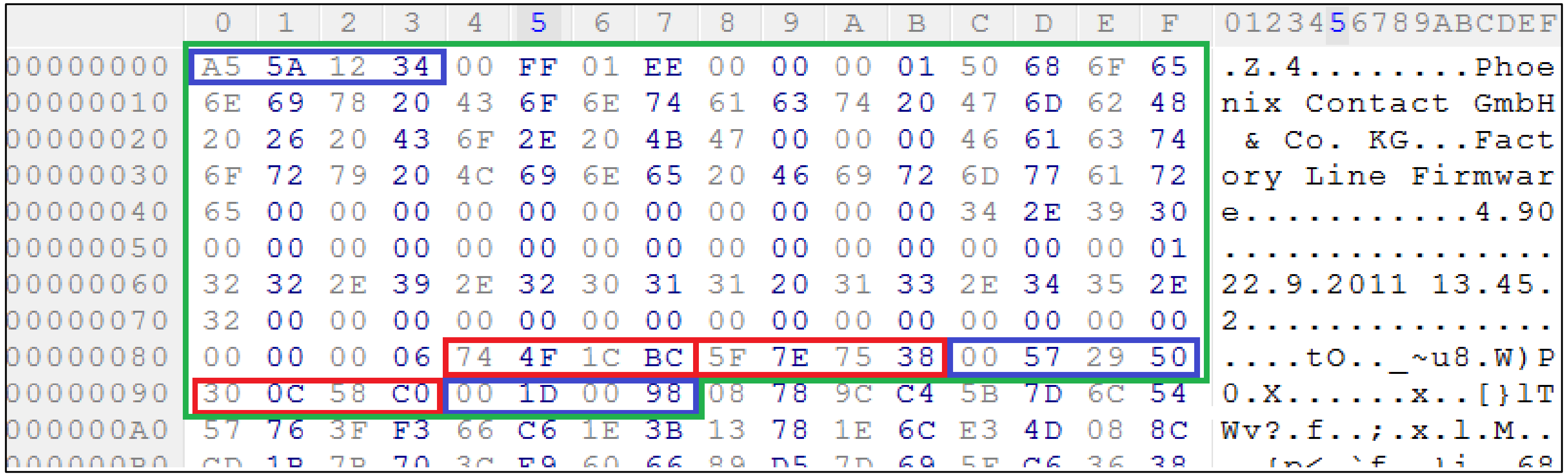
Main firmware – ELF executable

# Firmware image structure

- 0x00 signature

  …
- 0x84 header adler32
- 0x88 decompressed adler32
- 0x8C decompressed size
- 0x90 compressed adler32
- 0x94 compressed size

No identity verification



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
00000000  A5 5A 12 34 00 FF 01 EE 00 00 00 01 50 68 6F 65  .Z.4........Phoe
00000010  6E 69 78 20 43 6F 6E 74 61 63 74 20 47 6D 62 48  nix Contact GmbH
00000020  20 26 20 43 6F 2E 20 4B 47 00 00 00 46 61 63 74   & Co. KG...Fact
00000030  6F 72 79 20 4C 69 6E 65 20 46 69 72 6D 77 61 72  ory Line Firmwar
00000040  65 00 00 00 00 00 00 00 00 00 00 00 34 2E 39 30  e...........4.90
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01  ................
00000060  32 32 2E 39 2E 32 30 31 31 20 31 33 2E 34 35 2E  22.9.2011 13.45.
00000070  32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  2...............
00000080  00 00 00 06 74 4F 1C BC 5F 7E 75 38 00 57 29 50  ....tO.._~u8.W)P
00000090  30 0C 58 C0 00 1D 00 98 08 78 9C C4 5B 7D 6C 54  0.X......x..[}lT
000000A0  57 76 3F F3 66 C6 1E 3B 13 78 1E 6C E3 4D 08 8C  Wv?.f..;.x.l.M..
000000B0  CD 1B 7B 70 3C F9 60 66 89 D5 7D 69 5F C6 36 38
```

# Firmware analysis

## Operating system

VxWorks®

WIND RIVER

RTOS VxWorks 6.1

old version (latest version 7)

- No protection from binary vulnerabilities exploitation

# Firmware analysis

## Operating system

Known vulnerabilities:

| | |
|---|---|
| CVE-2015-3963 | spoof TCP sessions |
| CVE-2013-0714 | DoS/RCE |
| CVE-2013-0714 | DoS |
| CVE-2010-2968 | brute-force |
| CVE-2010-2967 | obtain access |
| CVE-2010-2966 | obtain access |
| CVE-2010-2965 | RCE |
| CVE-2008-2476 | DoS |

…

# Firmware and bootcode modification

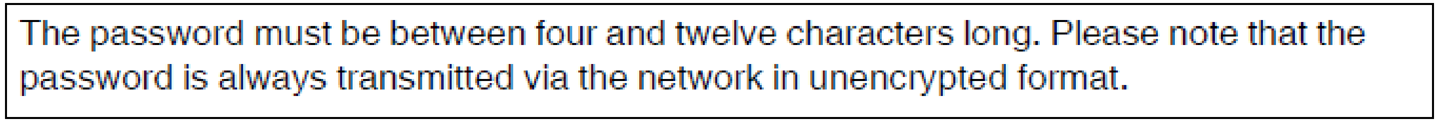
Firmware can be modified via:

- RS-232 (XMODEM) console, no auth
- HTTP Web interface, auth required

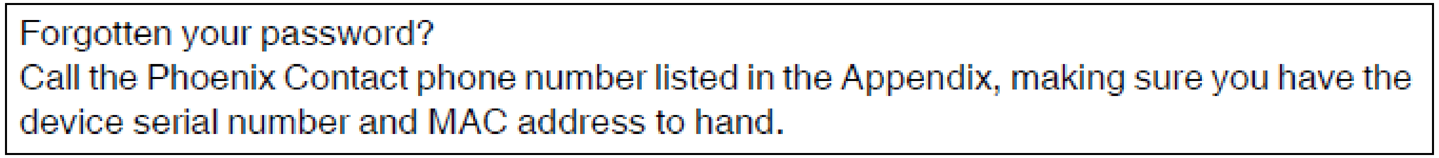
Bootcode is present on the flash and can also be rewritten

# Firmware analysis

- Engineer password



The password must be between four and twelve characters long. Please note that the password is always transmitted via the network in unencrypted format.



Forgotten your password?
Call the Phoenix Contact phone number listed in the Appendix, making sure you have the device serial number and MAC address to hand.

- No bootcode update mechanism

- Web interface can be reached without any auth (though, to make changes you will need a password)

# DEMO 02

# Conclusion

- Authorization requirements are not enough: firmware can be illegally updated
- No identity protection of firmware image: firmware (bootcode, CPLD…) can be modified
- No security technologies to protect against binary vulnerability exploitation

# Mitigation

Users:

- Do not use default security configurations
- Update firmware to the latest versions

Developers:

- Must pay more attention to the security model of their products

# Any questions?

# Thank You