Vendor of the products: D-Link

Affected Device: DI-8200G、DI-7200G+V2

Affected Device: D-Link DI-8200G、DI-7200G+V2

Firmware Download: http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-8200G

http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-7000G%20V2%E7%B3%BB%E5%88%97

Vulnerability Description: A command injection vulnerability was discovered in D-Link DI_8200G-17.12.20A1 and DI_7200G+V2-24.04.18D1, triggered by the path parameter in upgrade_filter.asp. Attackers can exploit this vulnerability by crafting malicious packets to execute arbitrary commands, thereby gaining full control of the target device.

POC:

Vulnerability Effect:

It can be observed that the router receives the request and successfully executes the command.

```
/tmp # ls
001                 006                 jhttpd_state        upgrade
002                 automount.log       mnt                 var
003                 df_file             ping
004                 etc                 pppoe_route_table
005                 home                radius_state
/tmp # cat 006
bin
dev
etc
etc_ro
firmadyne
hd
hd_share
home
init
lib
lost+found
media
mnt
proc
root
run
sbin
sys
tmp
usr
var
/tmp #
```

Vulnerability Cause:

The issue resides in the jhttpd component. In jhttpd, the program invokes the sub_46FC20 function to handle requests related to upgrade_filter.asp. The program first retrieves the value of the path parameter via httpd_get_parm and stores it in the parm variable.

Next, the program performs a security check on the value of the path parameter using the commandInjectionCheck function. The checked value is then concatenated into a variable via the sprintf function, which is ultimately executed by the jhl_system function.

```
1  // upgrade_filter.asp
2  int __fastcall sub_46FC20(int a1)
3  {
4    int parm; // $s1
5    int v3; // $s2
6    int v4; // $v0
7    int v5; // $v0
8    int n6684672; // $a0
9    int n20; // $a2
10   char ___ret__:0___msg__:__ok___[512]; // [sp+18h] [-208h] BYREF
11   const char *v10; // [sp+218h] [-8h]
12
13   parm = httpd_get_parm(a1, "path");
14   v3 = httpd_get_parm(a1, "time");
15   v10 = (const char *)commandInjectionCheck(parm);
16   nvram_set("tzk_upgrade_info", "");
17   nvram_set("tzk_state", "0");
18   v4 = jiffies_get();
19   mod_timer(a1 + 103056, v4 + 100000);
20   sprintf(___ret__:0___msg__:__ok___, "wys wget download %s", v10);
21   system(___ret__:0___msg__:__ok___);
22   v5 = nvram_get("tzk_state");
23   if ( J_atoi(v5) )
24   {
25     n20 = sprintf(___ret__:0___msg__:__ok___, "{\"ret\":1,\"msg\":\"%s\"}", byte_661C50);
26   }
27   else
28   {
29     n6684672 = 6684672;
30     if ( v3 )
31       nvram_set("tzk_time", v3);
32     jhl_parm_commit(n6684672);
33     n20 = 20;
34     strcpy(___ret__:0___msg__:__ok___, "{\"ret\":0,\"msg\":\"ok\"}");
35   }
36   return httpd_cgi_ret(a1, ___ret__:0___msg__:__ok___, n20, 4);
37 }

0006FC60 sub_46FC20:13 (46FC60)
```

However, the commandInjectionCheck function only filters characters such as `&`, `|`, and `;`. Attackers can bypass the check and execute arbitrary commands to fully control the device by constructing malicious parameters.

```
1  int *__fastcall commandInjectionCheck(int a1)
2  {
3    _BYTE *v2; // $v0
4    _BYTE *v3; // $v0
5    _BYTE *v4; // $v0
6
7    memset(&dword_6879AC, 0, 1024);
8    strncpy(&dword_6879AC, a1, 1024);
9    v2 = (_BYTE *)strchr(&dword_6879AC, '&');
10   dword_687DAC = (int)v2;
11   if ( v2 )
12     *v2 = 0;
13   v3 = (_BYTE *)strchr(&dword_6879AC, '|');
14   dword_687DAC = (int)v3;
15   if ( v3 )
16     *v3 = 0;
17   v4 = (_BYTE *)strchr(&dword_6879AC, ';');
18   dword_687DAC = (int)v4;
19   if ( v4 )
20     *v4 = 0;
21   return &dword_6879AC;
22 }
```