Talos Vulnerability Report

TALOS-2022-1505

TCL LinkHub Mesh Wifi confctl_set_master_wlan denial of service vulnerability

AUGUST 1, 2022

CVE NUMBER

CVE-2022-27185

SUMMARY

A denial of service vulnerability exists in the confctl_set_master_wlan functionality of TCL LinkHub Mesh Wifi MS1G_00_01.00_14. A specially-crafted network packet can lead to denial of service. An attacker can send packets to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

TCL LinkHub Mesh Wifi MS1G_00_01.00_14

PRODUCT URLS

LinkHub Mesh Wifi - https://www.tcl.com/us/en/products/connected-home/linkhub/linkhub-mesh-wifi-system-3-pack

CVSSV3 SCORE

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

CWE

CWE-284 - Improper Access Control

DETAILS

The LinkHub Mesh Wi-Fi system is a node-based mesh system designed for Wi-Fi deployments across large homes. These nodes include most features standard in current Wi-Fi solutions and allow for easy expansion of the system by adding nodes. The mesh is managed solely by a phone application, and the routers have no web-based management console.

The LinkHub Mesh system uses protobuffers to communicate both internally on the device as well as externally with the controlling phone application. These protobuffers can be sent to port 9003 while on the Wi-Fi, or wired network, provided by the LinkHub Mesh in order to issue commands, much like the phone application would. Once the protobuffer is received, it is routed internally starting from the `ucloud` binary and is dispatched to the appropriate handler.

In this case, the handler is `confsrv`, which handles many message types. In this case we are interested in `WlanCfgAll`

```
enum MESH_WIFI_TYPE {
    MESH_WIFI_2G = 0;
    MESH_WIFI_5G = 1;
    MESH_WIFI_MAX = 2;
}
message WlanTimeChoice {
    repeated int32 option = 1;
}
message WlanSecChoice {
    repeated string option = 1;
}
message WlanLimitChoice {
    repeated int32 option = 1;
}
message WlanCfg {
    required MESH_WIFI_TYPE band = 1;
    required string ssid = 2;                                   [2]
    required string passwd = 3;                                 [3]
    optional string sec = 4;
    optional int32 left = 5;
    optional int32 limite = 6;
    optional int32 timeout = 7;
    optional bool enable = 8;
}
message WlanCfgAll {
    repeated WlanCfg wlan = 1;                                  [1]
    optional WlanTimeChoice timeout = 2;
    optional WlanSecChoice security = 3;
    optional WlanLimitChoice limits = 4;
    optional uint64 timestamp = 5;
    optional bool enable = 6;
    optional bool from_app = 7;
}
```

Using [1], direct control over SSID ([2]) and passwd ([3]) can be obtained, and both `WlanCfgAll` and `WlanCfg` are parsed within `confctrl_set_master_wlan`.

```
00455938  int32_t confctl_set_master_wlan(int32_t arg1, int32_t arg2, int32_t arg3)

...
00455a54          int32_t var_1d0_1 = 0
00455a74          struct WlanCfgAll* pkt = wlan_cfg_all__unpack(0, arg3, arg2)
00455a88          if (pkt == 0) {
00455aa0              puts("      wlan_cfg_all__unpack error…")
00455aac              $v0_1 = 0xffffffff
00455aac          } else {
004560d4              for (int32_t var_1bc_1 = 0; var_1bc_1 u< pkt->wlan_count;
var_1bc_1 = var_1bc_1 + 1) {
00455adc                  int32_t $v0_9 = *(*(pkt->wlan + (var_1bc_1 << 2)) + 0xc)
00455ae0                  if ($v0_9 == 0) {
00455b18                      if (*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x10) == 0) {
00455ba8                          printf("[%s][%d][Arainc] 2g ssid is null…",
"confctl_set_master_wlan", 0x17a)
00455b9c                      } else {
00455b58                          int32_t $v0_21 = set_if_changed("wl2g.ssid0.ssid",
*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x10), &var_114)                  [4]
00455b68                          int32_t $v0_22 = var_1c8
00455b78                          if ($v0_21 s>= $v0_22) {
00455b78                              $v0_22 = $v0_21
00455b78                          }
00455b7c                          var_1c8 = $v0_22
00455b7c                      }
00455bd0                      if (*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x14) == 0) {
00455dc8                          printf("[%s][%d][Arainc] 2g passwd is nu…",
"confctl_set_master_wlan", 0x19d)
00455dbc                      } else {
00455bf0                          memset(&var_94, 0, 0x80)
00455c1c                          if (sx.d(**(*(pkt->wlan + (var_1bc_1 << 2)) +
0x14)) != 0) {
00455d1c                              SetValue(name: "wl2g.ssid0.security",
input_buffer: "wpapsk")
00455d60                              int32_t $v0_50 =
set_if_changed("wl2g.ssid0.wpapsk_psk", *(*(pkt->wlan + (var_1bc_1 << 2)) + 0x14),
&var_114)        [5]
00455d70                              int32_t $v0_51 = var_1c8
00455d80                              if ($v0_50 s>= $v0_51) {
00455d80                                  $v0_51 = $v0_50
00455d80                              }
....
004560b4                  } else {
00455df8                      if (*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x10) == 0) {
00455e88                          printf("[%s][%d][Arainc] 5g ssid is null…",
"confctl_set_master_wlan", 0x1aa)
00455e7c                      } else {
00455e38                          int32_t $v0_64 = set_if_changed("wl5g.ssid0.ssid",
*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x10), &var_114)                  [6]
00455e48                          int32_t $v0_65 = var_1c8
00455e58                          if ($v0_64 s>= $v0_65) {
00455e58                              $v0_65 = $v0_64
00455e58                          }
00455e5c                          var_1c8 = $v0_65
00455e5c                      }
00455eb0                      if (*(*(pkt->wlan + (var_1bc_1 << 2)) + 0x14) == 0) {
00456070                          printf("[%s][%d][Arainc] 5g passwd is nu…",
"confctl_set_master_wlan", 0x1ca)
```

```
00456064                        } else if (sx.d(**(*(pkt->wlan + (var_1bc_1 << 2)) +
0x14)) != 0) {
00455fdc                            SetValue(name: "wl5g.ssid0.security",
input_buffer: "wpapsk")
00456020                            int32_t $v0_92 =
set_if_changed("wl5g.ssid0.wpapsk_psk", *(*(pkt->wlan + (var_1bc_1 << 2)) + 0x14),
&var_114)            [7]
00456030                            int32_t $v0_93 = var_1c8
00456040                            if ($v0_92 s>= $v0_93) {
00456040                                $v0_93 = $v0_92
00456040                            }
00456044                            var_1c8 = $v0_93
....
```

At [4], the `ssid` from the provided protobuf is used directly into the new SSID of the device. Similarly at [5] the `passwd` provided is used as the new password for the 2G wireless network. Likewise, [6] and [7] can be used to change the SSID and password associated with the 5G wireless network. While this change is the most basic example, any of the fields in the defined protobuffers can be changed using this method without authentication.

TIMELINE

2022-03-29 - Vendor Disclosure
2022-08-01 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.
