Talos Vulnerability Report

TALOS-2022-1504

TCL LinkHub Mesh Wifi confctl_get_master_wlan information disclosure vulnerability

AUGUST 1, 2022

CVE NUMBER

CVE-2022-27630

SUMMARY

An information disclosure vulnerability exists in the confctl_get_master_wlan functionality of TCL LinkHub Mesh Wi-Fi MS1G_00_01.00_14. A specially-crafted network packet can lead to information disclosure. 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

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

CWE

CWE-200 - Information Exposure

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 don't actually need a specific protobuffer at all to achieve the information disclosure.

```
004565e8  int32_t confctl_get_master_wlan(int32_t arg1, int32_t arg2, int32_t arg3,
int32_t* arg4, int32_t* arg5)

...
00456690      void var_108
00456690      memset(&var_108, 0, 0x100)
004566a8      int32_t $v0 = malloc(8)
004566bc      int32_t $v0_2
004566bc      if ($v0 == 0) {
004566e4          _td_snprintf(3, "api/wifi_module.c", 0x21c, "WlanCfg alloc memory
Failed\n", 0x4ae4b0)
004566f0          $v0_2 = 0xffffffff
004566f0      } else {
00456714          memset($v0, 0, 8)
00456724          int32_t var_13c_1 = 2
00456734          int32_t $v0_4 = malloc(0x78)
00456748          if ($v0_4 == 0) {
00456770              _td_snprintf(3, "api/wifi_module.c", 0x226, "WlanCfg array
alloc memory Faile…", 0x4ae4b0)
00456780              var_154 = 0xffffffff
00456780          } else {
004567a0              memset($v0_4, 0, 0x78)
004567ac              int32_t var_118_1 = 0
004567b0              int32_t var_150_1 = 0
004568ac              while (true) {
004568ac                  if (var_150_1 s>= 2) {
004568d8                      if (GetValue(name: "sys.cfg.stamp", output_buffer:
&var_108) != 0) {
004568f0                          int32_t var_128_2 = 1
00456904                          int32_t $v0_27
00456904                          int32_t $v1_7
00456904                          $v0_27, $v1_7 = atoll(&var_108)
00456910                          int32_t var_120_1 = $v0_27
00456914                          int32_t var_11c_1 = $v1_7
00456914                      } else {
004568e0                          int32_t var_128_1 = 0
004568e0                      }
0045693c                      *arg5 = wlan_cfg_all__get_packed_size(&var_148)
00456968                      *arg4 = malloc(*arg5)
00456974                      if (*arg4 != 0) {
004569a8                          wlan_cfg_all__pack(&var_148, *arg4)
00456990                      } else {
00456980                          var_154 = 0xffffffff
00456980                      }
00456974                      break
00456974                  }
004567c0                  int32_t $v0_7 = var_150_1 << 2
004567e0                  wlan_cfg__init($v0_4 + ($v0_7 << 4) - $v0_7)
004567f0                  int32_t $v0_11 = var_150_1 << 2
00456828                  var_154 = wlan_get_master_cfg(var_150_1, 0, $v0_4 +
($v0_11 << 4) - $v0_11)                [1]
00456840                  int32_t $v0_18 = var_150_1 << 2
00456854                  *($v0 + (var_150_1 << 2)) = $v0_4 + ($v0_18 << 4) - $v0_18
0045685c                  if (var_154 != 0) {
00456884                      printf("%s(%d)\n", "confctl_get_master_wlan", 0x237)
00456890                      break
00456890                  }
004568a0                  var_150_1 = var_150_1 + 1
```

```
0045689c                }
004569c8                sub_4549e0(&var_148)
004569e0                free($v0_4)
004569e0            }
004569fc            free($v0)
00456a08            $v0_2 = var_154
00456a08        }
00456a1c        return $v0_2
```

As seen above, there is no protobuf parsing occuring from the data received, but at [1] `wlan_get_master_cfg` retrieves sensitive data to send back as a response. This response includes various information, but notable fields include the SSID and password in plaintext.

TIMELINE

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

CREDIT

Discovered by Carl Hurd of Cisco Talos.
