Talos Vulnerability Report

TALOS-2022-1503

TCL LinkHub Mesh Wifi confctl_get_guest_wlan information disclosure vulnerability

AUGUST 1, 2022

CVE NUMBER

CVE-2022-27633

SUMMARY

An information disclosure vulnerability exists in the confctl_get_guest_wlan functionality of TCL LinkHub Mesh Wifi 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.

```
00456a24  int32_t confctl_get_guest_wlan(int32_t arg1, int32_t arg2, int32_t arg3,
int32_t* arg4, int32_t* arg5)

00456a44      arg_0 = arg1
00456a48      arg_4 = arg2
00456a4c      arg_8 = arg3
00456a54      int32_t var_154 = 0
00456a60      int32_t var_14c = 0
00456a68      void* const var_148 = wlan_cfg_all__descriptor
...
00456ad4      void var_108
00456ad4      memset(&var_108, 0, 0x100)
00456aec      int32_t $v0 = malloc(8)
00456b00      int32_t $v0_2
00456b00      if ($v0 == 0) {
00456b18          puts("djc__WlanCfg alloc memory Failed")
00456b24          $v0_2 = 0xffffffff
00456b24      } else {
00456b48          memset($v0, 0, 8)
00456b58          int32_t var_13c_1 = 2
00456b68          int32_t $v0_4 = malloc(0x78)
00456b7c          if ($v0_4 == 0) {
00456b94              puts("djc___WlanCfg array alloc memory…")
00456ba4              var_154 = 0xffffffff
00456ba4          } else {
00456bc4              memset($v0_4, 0, 0x78)
00456bd4              int32_t var_118_1 = 1
00456bf0              GetValue(name: "wl.guest.dhcps_enable", output_buffer:
&var_14c)
00456c20              if (strcmp(&var_14c, "1") != 0) {
00456c48                  guest_enable_flag = 0
00456c4c                  int32_t var_114_2 = 0
00456c4c              } else {
00456c30                  guest_enable_flag = 1
00456c38                  int32_t var_114_1 = 1
00456c38              }
00456c50              int32_t var_150_1 = 0
00456d4c              while (true) {
00456d4c                  if (var_150_1 s>= 2) {
00456d60                      int32_t $v0_27 = malloc(0x14)
00456d74                      if ($v0_27 == 0) {
00456d9c                          _td_snprintf(3, "api/wifi_module.c", 0x2cf,
"WlanTimeChoice array alloc memor…", 0x4ae4b0)
00456dac                          var_154 = 0xffffffff
00456dac                      } else {
00456dcc                          memset($v0_27, 0, 0x14)
00456de4                          wlan_time_choice__init($v0_27)
00456e10                          *($v0_27 + 0x10) = malloc(0xc)
00456e1c                          if (*($v0_27 + 0x10) == 0) {
00456e28                              var_154 = 0xffffffff
00456e28                          } else {
00456e40                              **($v0_27 + 0x10) = 0x3840
00456e54                              *(*($v0_27 + 0x10) + 4) = 0x7080
00456e68                              *(*($v0_27 + 0x10) + 8) = 0xffffffff
00456e74                              *($v0_27 + 0xc) = 3
00456e7c                              int32_t var_134_1 = $v0_27
00456ea4                              if (GetValue(name: "sys.cfg.stamp",
output_buffer: &var_108) != 0) {
```

```
00456ebc                                    int32_t var_128_2 = 1
00456ed0                                    int32_t $v0_44
00456ed0                                    int32_t $v1_8
00456ed0                                    $v0_44, $v1_8 = atoll(&var_108)
00456edc                                    int32_t var_120_1 = $v0_44
00456ee0                                    int32_t var_11c_1 = $v1_8
00456ee0                                } else {
00456eac                                    int32_t var_128_1 = 0
00456eac                                }
00456f08                                *arg5 =
wlan_cfg_all__get_packed_size(&var_148)
00456f34                                *arg4 = malloc(*arg5)
00456f40                                if (*arg4 != 0) {
00456f74                                    wlan_cfg_all__pack(&var_148, *arg4)
00456f5c                                } else {
00456f4c                                    var_154 = 0xffffffff
00456f4c                                }
00456f4c                            }
00456f90                            sub_454a98($v0_27)
00456f90                        }
00456d74                        break
00456d74                    }
00456c60                    int32_t $v0_8 = var_150_1 << 2
00456c80                    wlan_cfg__init($v0_4 + ($v0_8 << 4) - $v0_8)
00456c90                    int32_t $v0_12 = var_150_1 << 2
00456cc8                    var_154 = wlan_get_master_cfg(var_150_1, 1, $v0_4 +
($v0_12 << 4) - $v0_12)                     [1]
00456ce0                    int32_t $v0_19 = var_150_1 << 2
00456cf4                    *($v0 + (var_150_1 << 2)) = $v0_4 + ($v0_19 << 4) - $v0_19
00456cfc                    if (var_154 != 0) {
00456d24                        printf("djc______%s(%d)\n", "confctl_get_guest_wlan",
0x2c5)
00456d30                        break
00456d30                    }
00456d40                    var_150_1 = var_150_1 + 1
00456d3c                }
00456fb0                sub_4549e0(&var_148)
00456fc8                free($v0_4)
00456fc8            }
00456fe4            free($v0)
00456ff0            $v0_2 = var_154
00456ff0        }
00457004        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 of the Guest WLAN.

TIMELINE

2022-03-29 - Vendor Disclosure

2022-08-01 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.

VULNERABILITY REPORTS | PREVIOUS REPORT | NEXT REPORT

TALOS-2022-1504 | TALOS-2022-1502
