Talos Vulnerability Report

TALOS-2022-1506

TCL LinkHub Mesh Wi-Fi confctl_set_wan_cfg denial of service vulnerability

AUGUST 1, 2022

CVE NUMBER

CVE-2022-27178

SUMMARY

A denial of service vulnerability exists in the confctl_set_wan_cfg functionality of TCL LinkHub Mesh Wi-Fi 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

9.6 - CVSS:3.0/AV:A/AC:L/PR:N/UI:N/S:C/C:H/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 `WanCfg`

```
message AdslCfg {
    required string uname = 1;
    required string passwd = 2;
    optional int32 mode = 3;
    optional WanDnsCfg dns = 4;
    optional int32 mtu = 5;
    optional string service_name = 6;
    optional string server_name = 7;
}
message StaticCfg {
    required string ipaddr = 1;
    required string mask = 2;
    required string gateway = 3;
    optional WanDnsCfg dns = 6;
}
message DynamicCfg {
    optional WanDnsCfg dns = 1;
}
message WanPortCfg {
    required int32 idx = 1;
    required int32 mode = 2;                                    [2]
    optional AdslCfg adsl = 3;
    optional StaticCfg static_info = 4;
    optional DynamicCfg dhcp = 5;
    optional RussiaAdslCfg rsadsl = 6;
    optional RussiaPPTPCfg rsapptp = 7;
    optional RussiaL2tpCfg rsal2tp = 8;
    optional IpnetCfg cfg = 9;
}
message WanCfg {
    repeated WanPortCfg wan = 1;                                [1]
    optional uint64 timestamp = 2;
    optional int32 double_wan = 3;
}
message WanDnsCfg {
    required bool automic = 1;
    optional string dns1 = 2;
    optional string dns2 = 3;
}
message IpnetCfg {
    optional bool automic = 1;
    optional string ipaddr = 2;
    optional string mask = 3;
    optional string gateway = 4;
    optional string dns1 = 5;
    optional string dns2 = 6;
}
```

Using [1] we have control over `wan` in the packet. The parsing of the data in the protobuf is done in `confctl_set_wan_cfg`. We also have control over the `WanPortCfg`. For this example, most important is `mode` at [2].

```
00448b74  int32_t confctl_set_wan_cfg(int32_t arg1, uint8_t* data, int32_t len)

00448b94      arg_0 = arg1
00448ba0      int32_t $a3
00448ba0      arg_c = $a3
00448ba8      int32_t $v0_1
00448ba8      if (data == 0) {
00448bd0          printf("[%s][%d][luminais] invalid param…", "confctl_set_wan_cfg",
0xe61)
00448bdc          $v0_1 = 0xffffffff
00448bdc      } else {
...
00448c90          struct WanCfg* $v0_3 = wan_cfg__unpack(0, len, data)
00448ca4          if ($v0_3 == 0) {
00448ccc              printf("[%s][%d][luminais] wan_cfg__unpa…",
"confctl_set_wan_cfg", 0xe71)
00448cd8              $v0_1 = 0xffffffff
00448cd8          } else {
00448cfc              if (confctl_module_debug_en(module_id: 0xb) != 0) {
00448d10                  print_wan_cfg($v0_3)
00448d10              }
00448d40              if (GetValue(name: "sys.workmode", output_buffer: &var_cc) ==
0) {
00448d68                  memcpy(&var_cc, "router", 7)
00448d5c              }
00448d78              uint32_t $v0_8 = $v0_3->wan_ctr
00448dcc              int32_t var_f4_1
00448dcc              for (var_f4_1 = 0; var_f4_1 u< $v0_8; var_f4_1 = var_f4_1 + 1)
{
00448dac                  if (*(*($v0_3->wan + (var_f4_1 << 2)) + 0x10) == 0x10) {
[3]
00448dac                      break
00448dac                  }
00448dac              }
00448de8              if (var_f4_1 == $v0_8) {
...
00449870              } else if (strcmp(&var_cc, "router") == 0) {
00448e34                  SetValue(name: "sys.workmode", input_buffer: "ap")
[4]
00448e4c                  sub_448a38()
00448e4c              }
```

The most basic example of this functionality is being able to swap the working mode of the device from `router` to `ap`. This swap will completely disable the network connectivity if the device is in `router` mode. At [3] the function parses the `mode` provided in the protobuffer. If the value is 0x10, then the swap to `ap` mode will occur. There is significantly more functionality you can do with this same protobuffer packet, modifying all of the WAN network configurations seen in the `WanPortCfg` protobuffer definition.

TIMELINE

2022-03-29 - Vendor Disclosure

2022-08-01 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2022-1507

NEXT REPORT TALOS-2022-1505