Talos Vulnerability Report

TALOS-2021-1396

Sealevel Systems, Inc. SeaConnect 370W HandleSeaCloudMessage out-of-bounds write vulnerabilities

FEBRUARY 1, 2022

CVE NUMBER

CVE-2021-21970,CVE-2021-21969

Summary

Two out-of-bounds write vulnerabilities exists in the HandleSeaCloudMessage functionality of Sealevel Systems, Inc. SeaConnect 370W v1.3.34. A specially-crafted MQTT payload can lead to an out-of-bounds write. An attacker can perform a man-in-the-middle attack to trigger these vulnerabilities.

Tested Versions

Sealevel Systems, Inc. SeaConnect 370W v1.3.34

Product URLs

SeaConnect 370W - https://www.sealevel.com/product/370w-a-wifi-to-form-c-relays-digital-inputs-a-d-inputs-and-1-wire-bus-seaconnect-multifunction-io-edge-module-powered-by-seacloud/

CVSSv3 Score

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

CWE

CWE-120 - Buffer Copy without Checking Size of Input (‘Classic Buffer Overflow’)

Details

The SeaConnect 370W is a Wi-Fi connected IIoT device offering programmable cloud access and control of digital and analog I/O and a 1-wire bus.

This device offers remote control via several means including MQTT, Modbus TCP and a manufacturer-specific protocol named “SeaMAX API”.

The device is built on top of the TI CC3200 MCU with built-in Wi-Fi capabilities.

One of the role of the SeaConnect 370W is as MQTT client. Different functionality is supported and controlled remotely from the “Sealevel SeaCloud”. When a message is pushed, in the SeaConnect 370W’s subscribed topic, it receive the message, parses it and performs the related action based on the message content.

A specially-crafted MQTT payload can lead to out-of-bound write due to a missing size check.

The `HandleIncomingSeaCloudMessage` function is responsible for parsing the MQTT message. The message should be in the following form:

```
{
    "name": "<functionality>",
    "payload": "<data>"
}
```

Here is the `HandleIncomingSeaCloudMessage` function:

```
void HandleIncomingSeaCloudMessage(incoming_message_struct *param_1)

{
  [...]

  puVar1 = read_volatile_4(fname);
  dVar6 = param_1->topic_element->element_content;
  Report(s_FSeaConnect_%s_topic_%.*s:_%.*s_2000ecbf + 1,(dword)puVar1,
         param_1->topic_element->element_size,dVar6);
  ppcVar2 = (char **)read_volatile_4(p_incomingTopic);
  memset(*ppcVar2,0,0x81);
  ppcVar3 = (char **)read_volatile_4(p_incomingMessage);
  memset(*ppcVar3,0,0x201);
  p_name_ = read_volatile_4(p_name);
  memset(p_name_,0,0x80);
  p_payload_ = read_volatile_4(p_payload);
  memset(p_payload_,0,0x100);
  topic_elem = param_1->topic_element;
  if ((int)topic_elem->element_size < 0x80) {
    size = topic_elem->element_size;
  }
  else {
    size = 0x80;
  }
  strncpy(*ppcVar2,(char *)topic_elem->element_content,size);
  message_elem = param_1->message_element;
  size = 0x201;
  if (*(uint *)&message_elem->element_size < 0x201) {                                    [1]
    size = *(size_t *)&message_elem->element_size;
  }
  strncpy(*ppcVar3,(char *)message_elem->element_content,size);                          [2]
  Report(aSeaconnectSGIn,(dword)puVar1,(dword)*ppcVar2,dVar6);
  incoming_message = *ppcVar3;
  Report(aSeaconnectSGIn_0,(dword)puVar1,(dword)incoming_message,dVar6);
  json_incoming_message_parser_ = json_parser_init(&jParser,*ppcVar3);
  if (json_incoming_message_parser_ == -1) {
    Report(aErrorSeaconnec_2,(dword)puVar1,(dword)incoming_message,dVar6);
  }
  else {
    puVar4 = read_volatile_4(pPrintCallback);
    json_parser_dump(&jParser,puVar4);
    is_error_ = json_object_get_string(&jParser,json_incoming_message_parser_,aName,p_name_);          [3]
    if ((is_error_ != 0) &&
       (payload_string_ = p_payload_,
       iVar5 = json_object_get_string(&jParser,json_incoming_message_parser_,aPayload,p_payload_),     [4]
       iVar5 != 0)) {
      Report(aSNameS,(dword)puVar1,(dword)p_name_,(dword)payload_string_);
      Report(s_F%s_payload:_%s_2000ed4f + 1,(dword)puVar1,(dword)p_payload_,(dword)payload_string_);
      HandleSeaCloudPayload(p_name_,p_payload_);
    }
    json_parser_deinit(&jParser);
  }
  return;
}
```

The function only copies at most 0x201 element from the MQTT message. This is guaranteed with the combination of the condition at `[1]` and the `strncpy` at `[2]`. The `HandleIncomingSeaCloudMessage` function extracts the function name and the payload data respectively at `[3]` and `[4]` using the function `json_object_get_string`.

Here is the function `json_object_get_string`:

```
char * json_object_get_string(jsonParser *jParser, jsonObj jObj, char *tagName, char *str)
{
      char              *cmpStr;
      int       cmpStrLen = 0;
      jsonString        jString;

      jString = (jsonString)get_object_by_type(jParser, jObj, tagName, JSON_STRING_OBJECT);
      if(jString == JSON_INVALID_OBJECT)
      {
          return NULL;
      }
      else
      {
          cmpStr = jParser->jsonStream + jParser->tokenList[jString].start;
          cmpStrLen = jParser->tokenList[jString].end - jParser->tokenList[jString].start;
          strncpy(str, cmpStr, cmpStrLen);
          str[cmpStrLen] = '\0';
          return str;
      }
}
```

This function will fill the `str` variable with the value corresponding to the key specified in `tagName`. So, at `3` it will fill `str` with the value corresponding to the `name` key, and at `[4]` with the one corresponding to `payload`.

CVE-2021-21969 - json payload key value out-of-bound write

The `HandleIncomingSeaCloudMessage` function uses at `[4]` the `json_object_get_string` to populate the `p_payload` global variable. The `p_payload` is only 0x100 bytes long, and the total MQTT message could be up to 0x201 bytes. Because the function `json_object_get_string` will fill `str` based on the length of the json's value and not the actual `str` size, this would result in a possible out-of-bounds write.

CVE-2021-21970 - json name key value out-of-bound write

The `HandleIncomingSeaCloudMessage` function uses at `[3]` the `json_object_get_string` to populate the `p_name` global variable. The `p_name` is only 0x80 bytes long, and the total MQTT message could be up to 0x201 bytes. Because the function `json_object_get_string` will fill `str` based on the length of the json's value and not the actual `str` size, this would result in a possible out-of-bounds write.

Timeline

2021-10-26 - Vendor disclosure
2022-02-01 - Public Release

CREDIT

Discovered by Francesco Benvenuto and Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1395

NEXT REPORT TALOS-2021-1397