Talos Vulnerability Report

TALOS-2022-1556

Abode Systems, Inc. iota All-In-One Security Kit XCMD doDebug OS Command Injection vulnerability

OCTOBER 20, 2022

CVE NUMBER

CVE-2022-32773

SUMMARY

An OS command injection vulnerability exists in the XCMD doDebug functionality of Abode Systems, Inc. iota All-In-One Security Kit 6.9X and 6.9Z. A specially-crafted XCMD can lead to arbitrary command 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-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 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 attacker 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.

For example, one of the simplest XCMDs that can be executed is `getDev`.

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
  <mac v="B0:C5:CA:00:00:00"/>
  <cmds>
    <cmd a="getDev"/>
  </cmds>
</p>
```

One of the XCMDs, `doDebug`, appears to be intended for diagnostic access by developers and technical support. The `doDebug` XCMD can react several different ways, based on the child elments of the `<cmd>` element. For this vulnerability we focus on the `poke` element, which is intended to signal an arbitrary process with `USR1`. The manner in which this is conducted is vulnerable to an OS Command Injection.

The structure of this command might appear as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
  <mac v="B0:C5:CA:00:00:00"/>
  <cmds>
    <cmd a="doDebug">
      <poke v="log_server"/>
    </cmd>
  </cmds>
</p>
```

In this example, the XCMD handler would send the `USR1` signal to the `log_server` process.

The relevant portions of the decompilation of `doDebug` handler are included below.

```
int __fastcall doDebug(xml_related_t *xcmd, int *a2)
{
  char *poke;
  ...

  // [1] The value of `v`, of the `poke` tag, is extracted (the entire XML payload
is user controlled)
  poke = get_xcmd_param(xcmd, "poke");
  ...
  if ( poke )
  {
    // [2] The `poke` value is injected in to the cmd buffer with no sanitization
    vsnprintf_nullterm(cmd, 0x1FFu, "/bin/kill -s USR1 `pidof %s`", poke);
    log(DEBUG, XCMD, "%s", cmd);
    // [3] The command is executed as root
    system(cmd);
  }
  ...
}
```

At `[1]` the value of the `poke` tag is extracted. At `[2]` that attacker-controlled value is injected into a command without sanitization. At `[3]` the resulting command is executed via a call to `system` as the root user.

Properly formatting and submitting a `doDebug` XCMD over UDP/55050, or via authenticated POST request to `/action/xmlCmd` of the local web interface, or via the XMPP channel, will result in arbitrary command execution on the Abode iota device.

Exploit Proof of Concept

Submitting the following XCMD payload

```<?xml version=”1.0” encoding=”UTF-8”?>

```

would result in the execution of the following command:

```
/bin/kill -s USR1 -p `pidof ` || sleep 11 #`
```

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.
