Cisco RV Authentication Bypass / Code Execution

Authored by T Shiomitsu | Site iot-inspector.com
Posted Apr 20, 2021

Cisco RV-series routers suffer from an authentication bypass vulnerability. The RV34X series are also affected by a command injection vulnerability in the sessionid cookie, when requesting the /upload endpoint. A combination of these issues would allow any person who is able to communicate with the web interface to run arbitrary system commands on the router as the www-data user. Vulnerable versions include RV16X/RV26X versions 1.0.01.02 and below and RV34X versions 1.0.03.20 and below.

tags | exploit, web, arbitrary, bypass
systems | cisco
advisories | CVE-2021-1472, CVE-2021-1473
SHA-256 | f3c8685d841186aca43bc22f8ed2b32e8512c7730129f2ed6fe20f360378fa91 Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
IoT Inspector Research Lab Security Advisory IOT-20210414-0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
              title: Cisco RV series Authentication Bypass and Remote Command
                     Execution
     vendor/product: Cisco (https://www.cisco.com/)
 vulnerable version: RV16X/RV26X: 1.0.01.02 & below.
                     RV34X: 1.0.03.20 & below.
      fixed version: RV16X/RV26X: 1.0.01.03.
                     RV34X: 1.0.03.21.
         CVE number: CVE-2021-1472, CVE-2021-1473
             impact: 5.3 (medium) CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N
                     8.8 (high)   CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L
           reported: 2021-01-02
        publication: 2021-04-14
                 by: T Shiomitsu, IoT Inspector Research Lab
                     https://www.iot-inspector.com/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Vendor description:
-------------------
The RV series devices are Cisco's line of small business routers with extra
functionality, including VPN and other security measures.


Vulnerability overview/description:
-----------------------------------
All Cisco RV-series routers suffer from an authentication bypass vulnerability.
The RV34X series are also affected by a command injection vulnerability in the
sessionid cookie, when requesting the /upload endpoint. A combination of these
issues would allow any person who is able to communicate with the web
interface to run arbitrary system commands on the router as the www-data user.


Root Cause Analysis:
--------------------
CVE-2021-1472: /upload Authentication Bypass Vulnerability

While Cisco has noted that this issue also affects the RV160, I will provide
a RCA for only the RV34X series here.

The RV340 web interface is served by nginx on port 443. The nginx configuration
(found in files in /etc/nginx) is such that requests made to the web interface
URIs /upload, /form-file-upload and
/api/operations/ciscosb-file:form-file-upload are all proxied to a CGI binary
called upload.cgi. Depending on which URI is requested, the behaviour of the
binary will be slightly different.

While some attempt was introduced in recent firmware revisions to prevent
unauthenticated access to the functionality available at the /upload endpoint,
the authentication check is incomplete. An attacker simply has to pass any
generic Authorization header as part of the request to bypass the authorization
check. This can be seen in web.upload.conf:

[...snip...]
location /upload {
        set $deny 1;

        if ($http_authorization != "") {
                set $deny "0";
        }

        if (-f /tmp/websession/token/$cookie_sessionid) {
                set $deny "0";
        }

        if ($deny = "1") {
                return 403;
        }
[...snip...]

As can be seen, the $deny is set to 0 if the $cookie_sessionid is valid (i.e.
that the authorization file exists on the system). But it also set to 0 if the
$http_authorization value (i.e. the Authorization header) is not blank.
Therefore, passing any value to an Authorization header can allow an attacker
access to the /upload endpoint.

CVE-2021-1473: /upload sessionid Command Injection Remote Code Execution

Within the main() function in upload.cgi, the HTTP_COOKIE environmental
variable is read, and the value from the sessionid cookie is extracted using
a simple series of strtok_r and strstr. This specific sessionid-reading logic
is notable because, due to the strtok_r call, it's not possible to use ";"
characters in any injection, as it will prematurely terminate the injection
string. In pseudocode, it looks like this:

if (HTTP_COOKIE != (char *)0x0) {
     StrBufSetStr(cookie,HTTP_COOKIE);
     cookie = StrBufToStr(cookie);
     cookie = strtok_r(cookie, ";", &saveptr);
     while (cookie != 0x0) {
       cookie = strstr(cookie, "sessionid=");
       if (cookie != 0x0) {
         sessionid_cookie_value = pathparam_ + 10;
       }
     }
   }

Because our HTTP request is made to the /upload URI, the main() function in
upload.cgi calls a function at 000124a4, which I've named handle_upload().
This function takes a pointer to the sessionid cookie value as its first
argument.

void handle_upload(char *sessionId, char *destination, char *option,
    char *pathparam, char *fileparam, char *cert_name, char *cert_type,
    char *password)

It also takes several other arguments, each of which are populated by the
multipart request parsing that takes place in the main() function. The names
I've given these arguments roughly align with the names of the parameters
that this multipart ingesting logic looks for.

(Depending on what string is passed as the pathparam parameter, slightly
different code paths will be taken, which means that slightly different checks
must be bypassed to be able to reach the vulnerable code. In this example, I
am using a request with the pathparam set to "Configuration", so the pseudocode
```

Follow us on Twitter

Subscribe to an RSS Feed

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

- Red Hat 180 files
- Ubuntu 78 files
- Debian 24 files
- LiquidWorm 23 files
- malvuln 12 files
- nu11secur1ty 10 files
- Gentoo 9 files
- Google Security Research 8 files
- T. Weber 4 files
- Julien Ahrens 4 files

File Tags

ActiveX (932)
Advisory (79,733)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,924)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,601)
Encryption (2,349)
Exploit (50,358)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (820)
Kernel (6,290)
Local (14,201)
Magazine (586)
Overflow (12,418)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,043)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,776)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,294)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,448)
Slackware (941)
Solaris (1,607)

```
I'm showing reflects this.)

Within handle_upload(), a curl command is constructed with a call to sprintf,
the resulting buffer of which is then passed directly to popen:

ret = strcmp(pathparam, "Configuration");
 if (ret == 0) {
   config_json = upload_Configuration_json(destination,fileparam);
   if (config_json != 0) {
     post_data = json_object_to_json_string(config_json);
     sprintf(command_buf, "curl %s --cookie \'sessionid=%s\' -X POST -H \'Content-Type: application/json\' -
d\'%s\' ", jsonrpc_cgi, sessionId , post_data);
     debug("curl_cmd=%s",command_buf);
     __stream = popen(command_buf, "r");
     if (__stream != (FILE *)0x0) {
       [...snip...]
   }

The sessionid cookie value that we have passed in our request is passed
directly into this sprintf() call. With a crafted sessionid value, we would
therefore be able to inject arbitrary commands into this command buffer. This
will run the command with the privileges of the upload.cgi process which, in
this case, is www-data.


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Vulnerable / tested versions:
-----------------------------
Cisco RV16X, RV26X and RV34X series devices.


Solution:
---------
Apply Cisco-supplied patch. For RV16X/26X, 1.0.01.03. For RV34X, 1.0.03.21.


Advisory URL:
-------------
https://www.iot-inspector.com/blog/advisory-cisco-rv34x-authentication-bypass-remote-command-execution/


Vendor contact timeline:
------------------------
2021-01-02: Initial disclosure made to Cisco PSIRT.
2021-01-07: Confirmation of receipt of disclosure from Cisco PSIRT.
2021-01-27: Confirmation that issue is valid from Cisco PSIRT.
2021-02-12: Update from Cisco PSIRT.
2021-03-23: We contact Cisco PSIRT for timeline update and CVE IDs.
2021-03-23: Cisco PSIRT respond giving us timeline and CVE IDs.
2021-04-07: Cisco release advisory.


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The IoT Inspector Research Lab is an integrated part of IoT Inspector.

IoT Inspector is a platform for automated security analysis and compliance
checks of IoT firmware. Our mission is to secure the Internet of Things. In
order to discover vulnerabilities and vulnerability patterns within IoT devices
and to further enhance automated identification that allows for scalable
detection within IoT Inspector, we conduct excessive security research in the
area of IoT.

Whenever the IoT Inspector Research Lab discovers vulnerabilities in IoT
firmware, we aim to responsibly disclose relevant information to the vendor
of the affected IoT device as well as the general public in a way that
minimizes potential harm and encourages further security analyses of IoT
systems.

You can find our responsible disclosure policy here:
https://www.iot-inspector.com/responsible-disclosure-policy/


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Interested in using IoT Inspector for your research or product?

Mail: research at iot-inspector dot com
Web: https://www.iot-inspector.com
Blog: https://www.iot-inspector.com/blog/
Twitter: https://twitter.com/iotinspector

EOF T Shiomitsu / @2021
```

Login or Register to add favorites