Arbitrary Code Injection

Affecting convert-svg-core package, versions **<0.6.3**

INTRODUCED: 6 JUN 2022 CVE-2022-24429 CWE-94 FIRST ADDED BY SNYK

Share

How to fix?

Upgrade `convert-svg-core` to version 0.6.3 or higher.

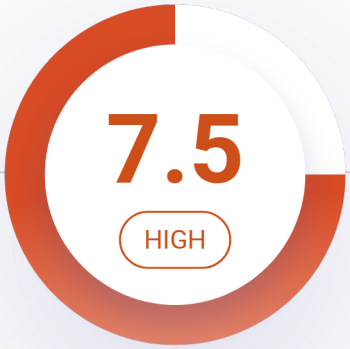
Overview

convert-svg-core is a package that supports converting SVG into another format using headless Chromium.

Affected versions of this package are vulnerable to Arbitrary Code Injection when using a specially crafted `SVG` file. An attacker can read arbitrary files from the file system and then show the file content as a converted `PNG` file.

PoC

```
const { convert } = require("convert-svg-to-
png"); const express = require("express"); const
fileSvg = ` `; const app = express();
app.get("/poc", async (req, res) => { try { const
png = await convert(fileSvg); res.set("Content-
Type", "image/png"); res.send(png); } catch (e) {
```



Snyk CVSS

Exploit Maturity **Proof of concept**

Attack Complexity **Low**

Confidentiality HIGH

See more

NVD 7.8 HIGH

Do your applications use this vulnerable package?

In a few clicks we can analyze your entire application and see what

components are vulnerable in your application, and suggest you quick fixes

```
res.send(""); } }); app.listen(3000, () => {
console.log("started"); });
```

Details

A Directory Traversal attack (also known as path traversal) aims to access files and directories that are stored outside the intended folder. By manipulating files with "dot-dot-slash (../)" sequences and its variations, or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system, including application source code, configuration, and other critical system files.

Directory Traversal vulnerabilities can be generally divided into two types:

- **Information Disclosure**: Allows the attacker to gain information about the folder structure or read the contents of sensitive files on the system.

`st` is a module for serving static files on web pages, and contains a vulnerability of this type. In our example, we will serve files from the `public` route.

If an attacker requests the following URL from our server, it will in turn leak the sensitive private key of the root user.

```
curl
http://localhost:8080/public/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2
e%2e/root/.ssh/id_rsa
```

Note `%2e` is the URL encoded version of `.` (dot).

- **Writing arbitrary files**: Allows the attacker to create or replace existing files. This type of vulnerability is also known as `Zip-Slip`.

One way to achieve this is by using a malicious `zip` archive that holds path traversal filenames. When each filename in the zip archive gets concatenated to the target extraction folder, without validation, the final path ends up outside of the target folder. If an executable or a configuration file is overwritten with a file containing malicious code, the problem can turn into an arbitrary code execution issue quite easily.

The following is an example of a `zip` archive with one benign file and one malicious file. Extracting the malicious file will result in traversing out of the target folder, ending up in `/root/.ssh/` overwriting the `authorized_keys` file:

Snyk ID **SNYK-JS-CONVERTSVGCORE-2859212**

Published **6 Jun 2022**

Disclosed **6 Jun 2022**

Credit **Donggyu Kim**

```
2018-04-15 22:04:29 ..... 19 19 good.txt 2018-04-15
22:04:42 ..... 20 20
../../../../../../root/.ssh/authorized_keys
```

References

- Github Commit

FIND US ONLINE

TRACK OUR DEVELOPMENT

Join the >> community

Registered in England and Wales. Company number: 09677925

Registered address: Highlands House, Basingstoke Road, Spencers Wood, Reading, Berkshire, RG7 1NT.