07 - XSS - Stored Lab

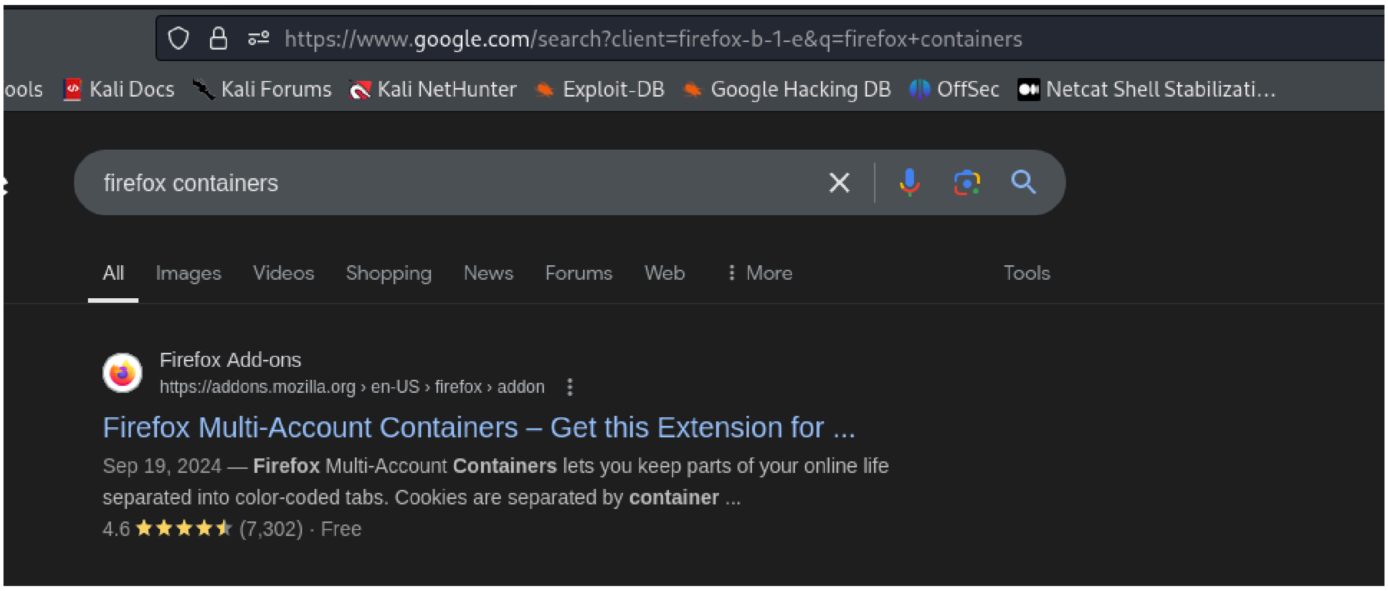
So, here we are going to set up Containers for our testing, which allow us to have multiple sessions opened, and test accross different user accounts. This could be particularly interesting when testing for authorizations issues, for example, to see if it is possible to update a user account using another user's session token.

But, in this scenario, we want t check that cross site scripting is indeed stored and not just showing up for the same user that posts the payload.

It is also possible to do this using multiple browsers or sessions in private mode.

To install Containers:

Google "Firefox containers" >



https://www.google.com/search?client=firefox-b-1-e&q=firefox+containers

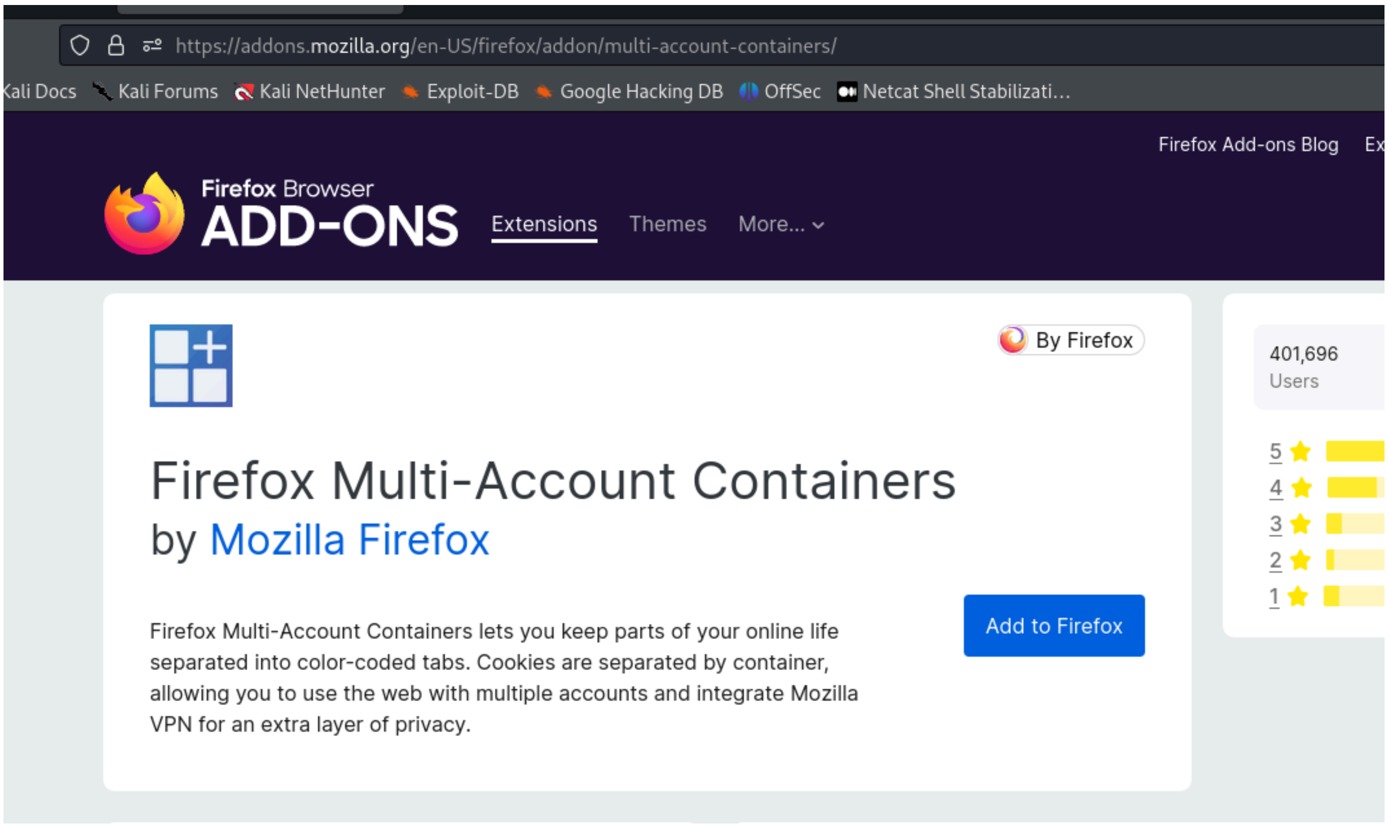
Firefox Add-ons
https://addons.mozilla.org › en-US › firefox › addon

Firefox Multi-Account Containers – Get this Extension for ...

Sep 19, 2024 — **Firefox** Multi-Account **Containers** lets you keep parts of your online life separated into color-coded tabs. Cookies are separated by **container** ...

4.6 ★★★★★ (7,302) · Free

That should do it > Open the website > Click "Add to "Firefox".


https://addons.mozilla.org/en-US/firefox/addon/multi-account-containers/

Kali Docs · Kali Forums · Kali NetHunter · Exploit-DB · Google Hacking DB · OffSec · Netcat Shell Stabilizati...

Firefox Add-ons Blog

Firefox Browser ADD-ONS · Extensions · Themes · More...

By Firefox

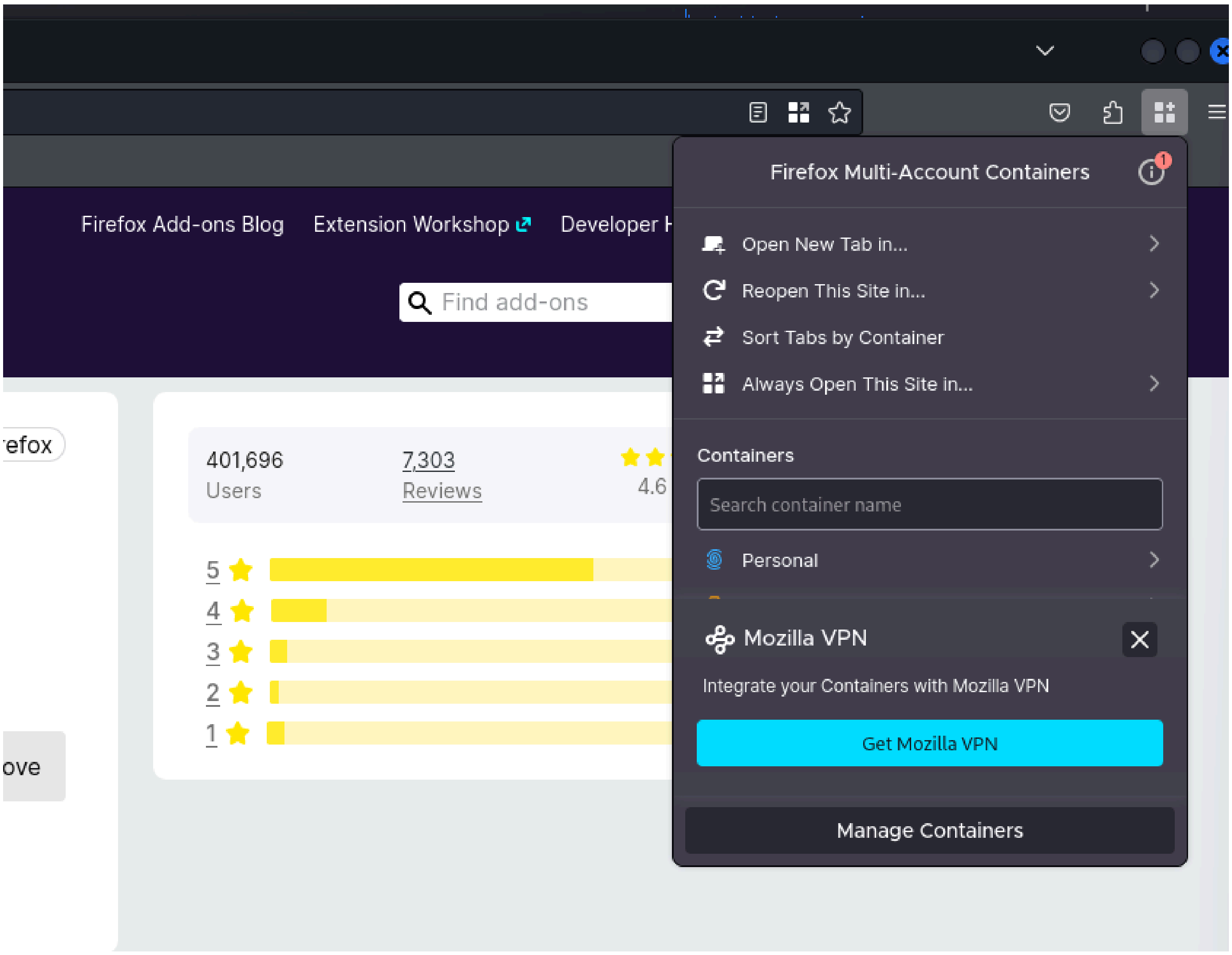
Firefox Multi-Account Containers
by Mozilla Firefox

Firefox Multi-Account Containers lets you keep parts of your online life separated into color-coded tabs. Cookies are separated by container, allowing you to use the web with multiple accounts and integrate Mozilla VPN for an extra layer of privacy.

Add to Firefox

401,696 Users

Next on all > Not now on all.


Firefox Add-ons Blog · Extension Workshop · Developer H

Find add-ons

401,696 Users · 7,303 Reviews · 4.6

Firefox Multi-Account Containers

Open New Tab in...
Reopen This Site in...
Sort Tabs by Container
Always Open This Site in...

Containers

Search container name

Personal

Mozilla VPN

Integrate your Containers with Mozilla VPN

Get Mozilla VPN

Manage Containers

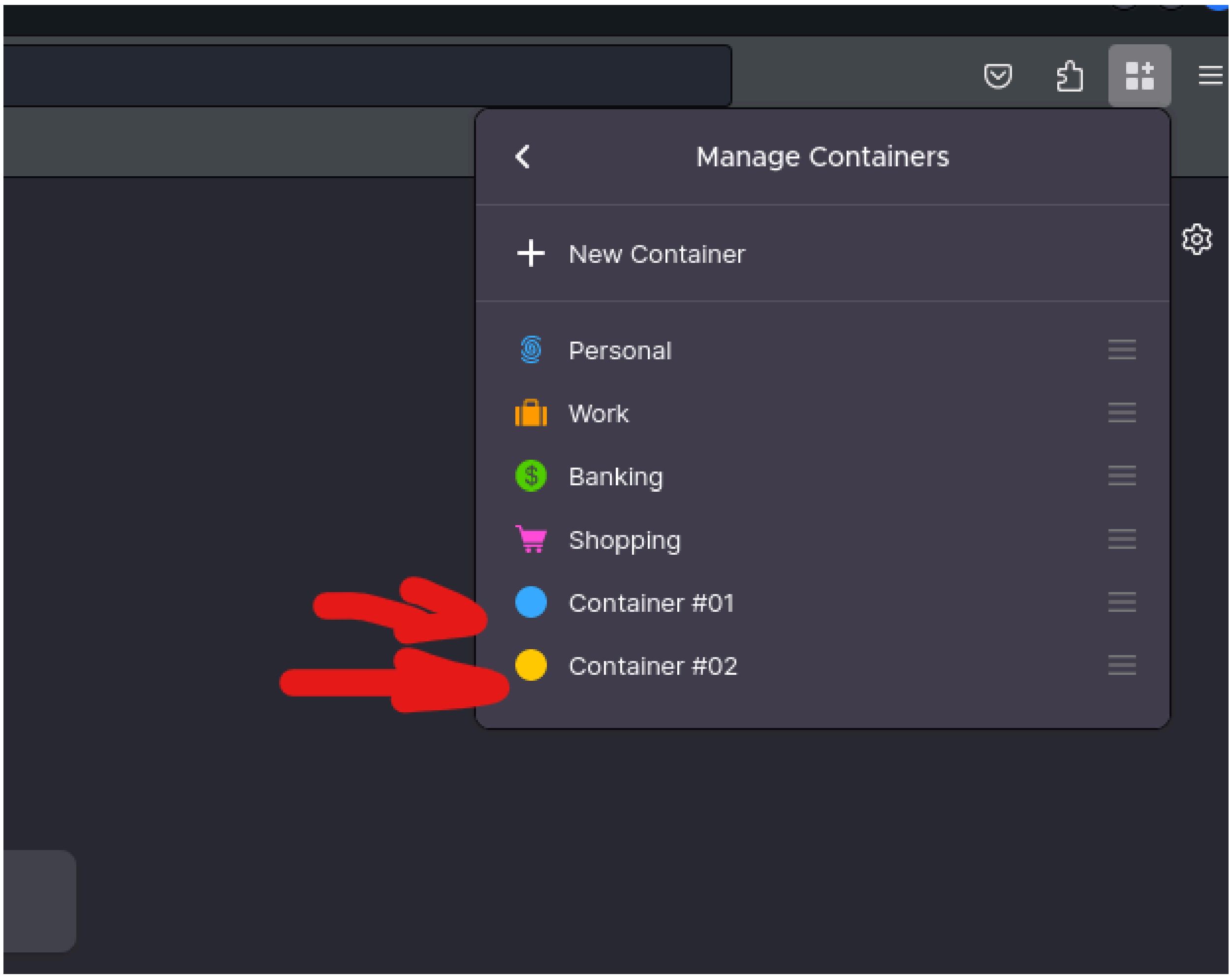
And. It is installed.

Alex mentions having on his Pentesting machine, two containers for low privileged users, and two for high privileged users.

Create two containers for exercise 2.

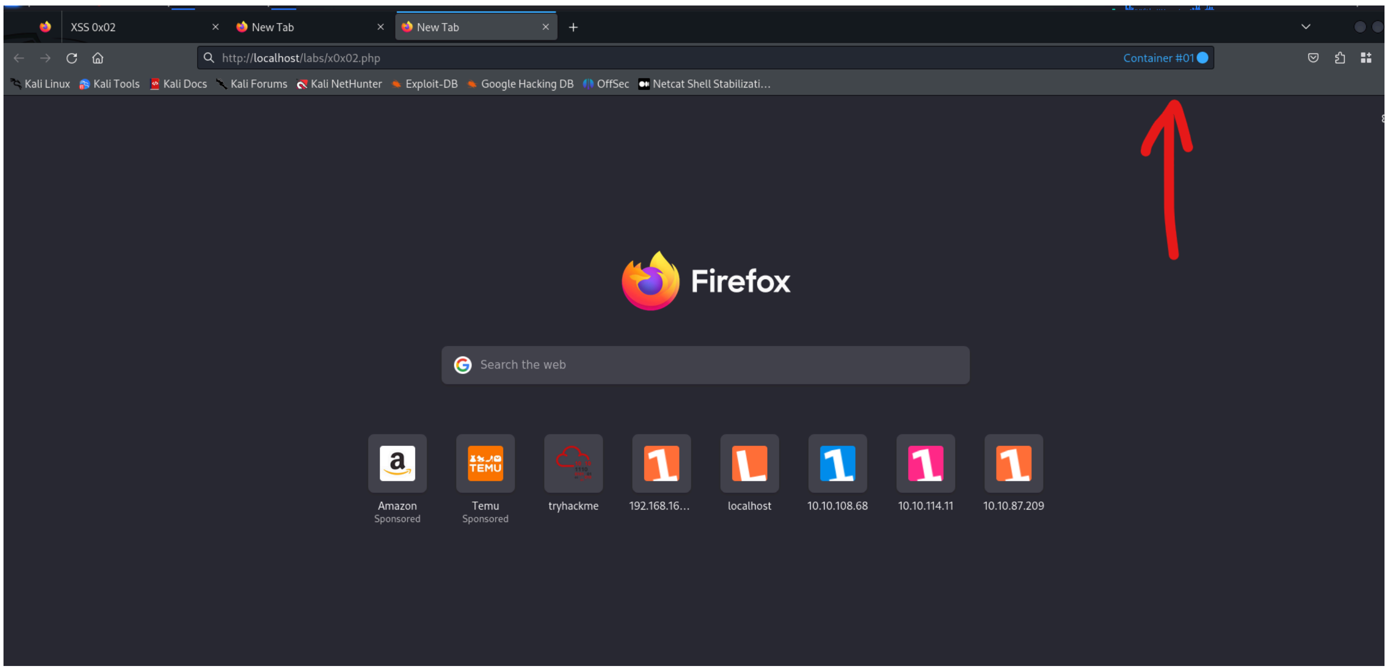
Open the extension > Manage Containers > New Container > Name you container.

I named mine Container#01 and Container#02.



Open container#01 > Past lab 2 URL.

Note that the new tab shows this is container#01



So, if we are trying different account, then we would have one account in one of these containers, and the other one, in another container. This is so the cookies, and data handled in the session do not get mixed up. If you try to login in one account, and then open a tab and login to another account in the same browser, the sessions cookies are going to get mixed up, and the last cookie updated to the browser will be always taking over the previous one.

So, using the containers allow us to not have those cookies, and data in general mixed up accross different sessions in the same web browser.

We can also use the extension to set cookies in these different containers.

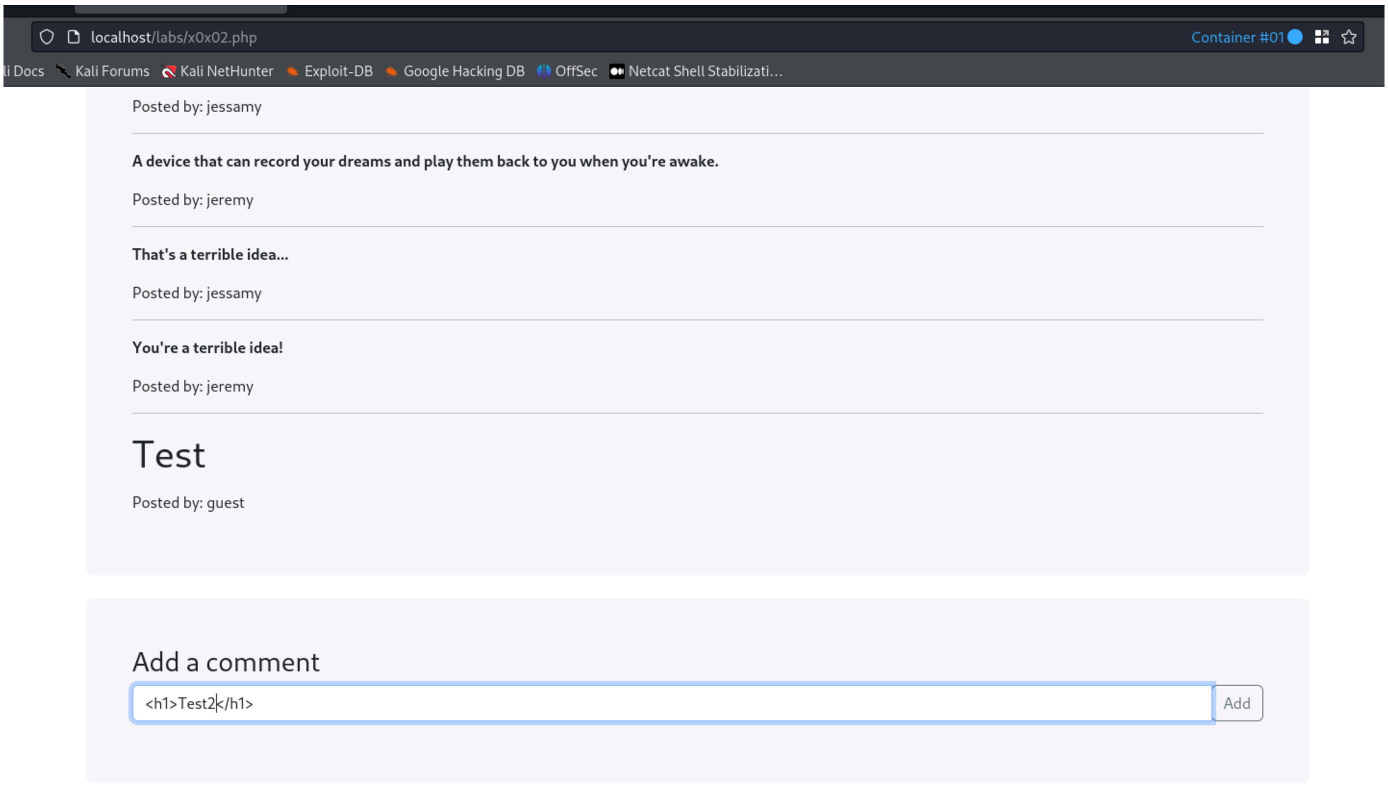
document.cookie()   - JavaScript command - Gets document cookie.

Quick note here: When testing for Cross Site Scripting, we can check for HTML injection vulnerabilities first. And, that is because HTLM injection is more likely to work, and once that works, we can figure out how to get our Cross Site Scripting payload to work. If there is some kind of filtering or some restricted characters, we can get around of that first when figuring out the HTML injection, and then adding Cross Site Scripting payload to that injection.

In this scenario, we can do:

<h1>Test</h1>

And to confirm this is indeed HTML injection, we can see our header written on the page:


localhost/labs/x0x02.php
Container #01
Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Netcat Shell Stabilizati…
Add

Posted by: jessamy

A device that can record your dreams and play them back to you when you're awake.

Posted by: jeremy

That's a terrible idea...

Posted by: jessamy

You're a terrible idea!

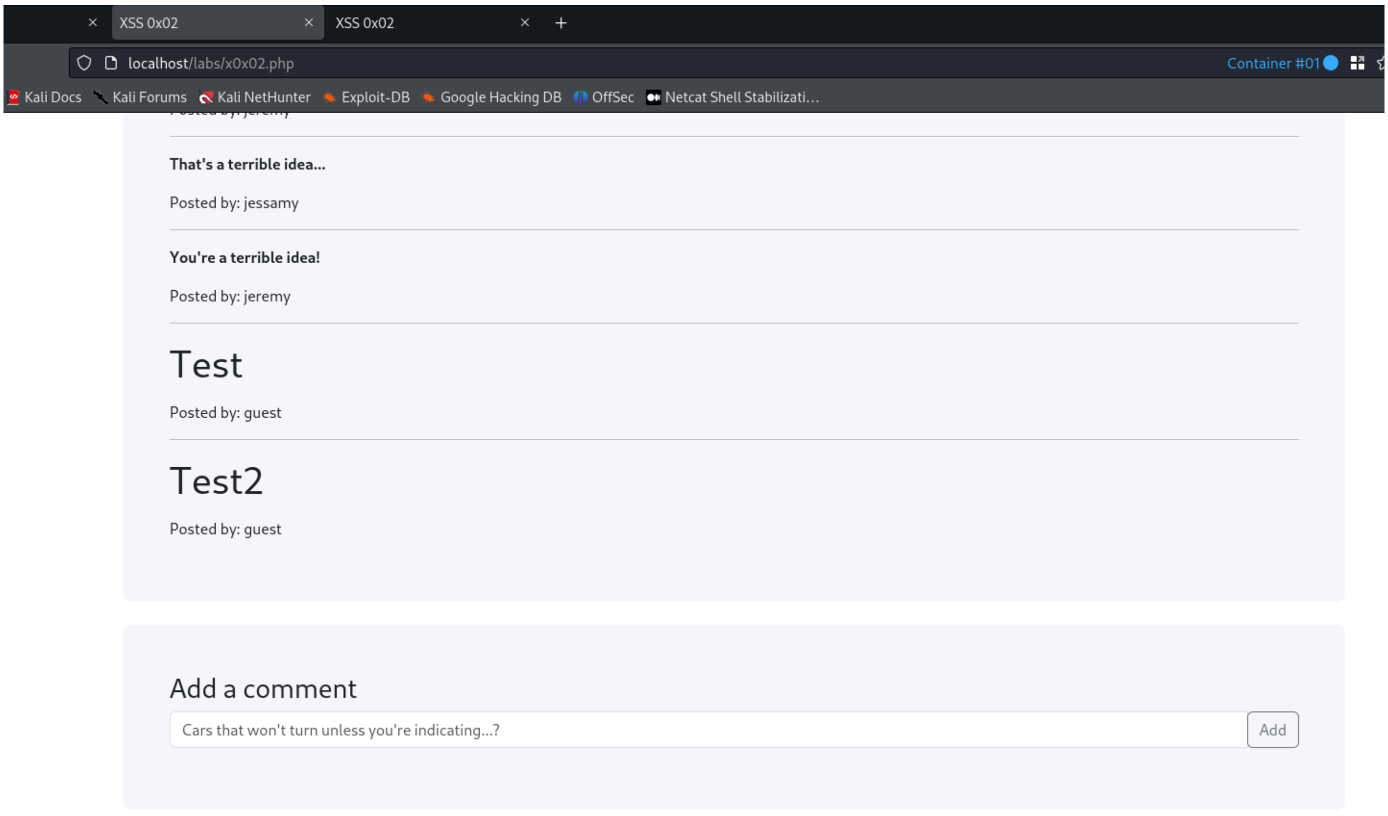
Posted by: jeremy

Test

Posted by: guest

Add a comment

```
<h1>Test2</h1>
```


XSS 0x02
XSS 0x02
localhost/labs/x0x02.php
Container #01
Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Netcat Shell Stabilizati…
Add

That's a terrible idea...

Posted by: jessamy

You're a terrible idea!

Posted by: jeremy

Test

Posted by: guest

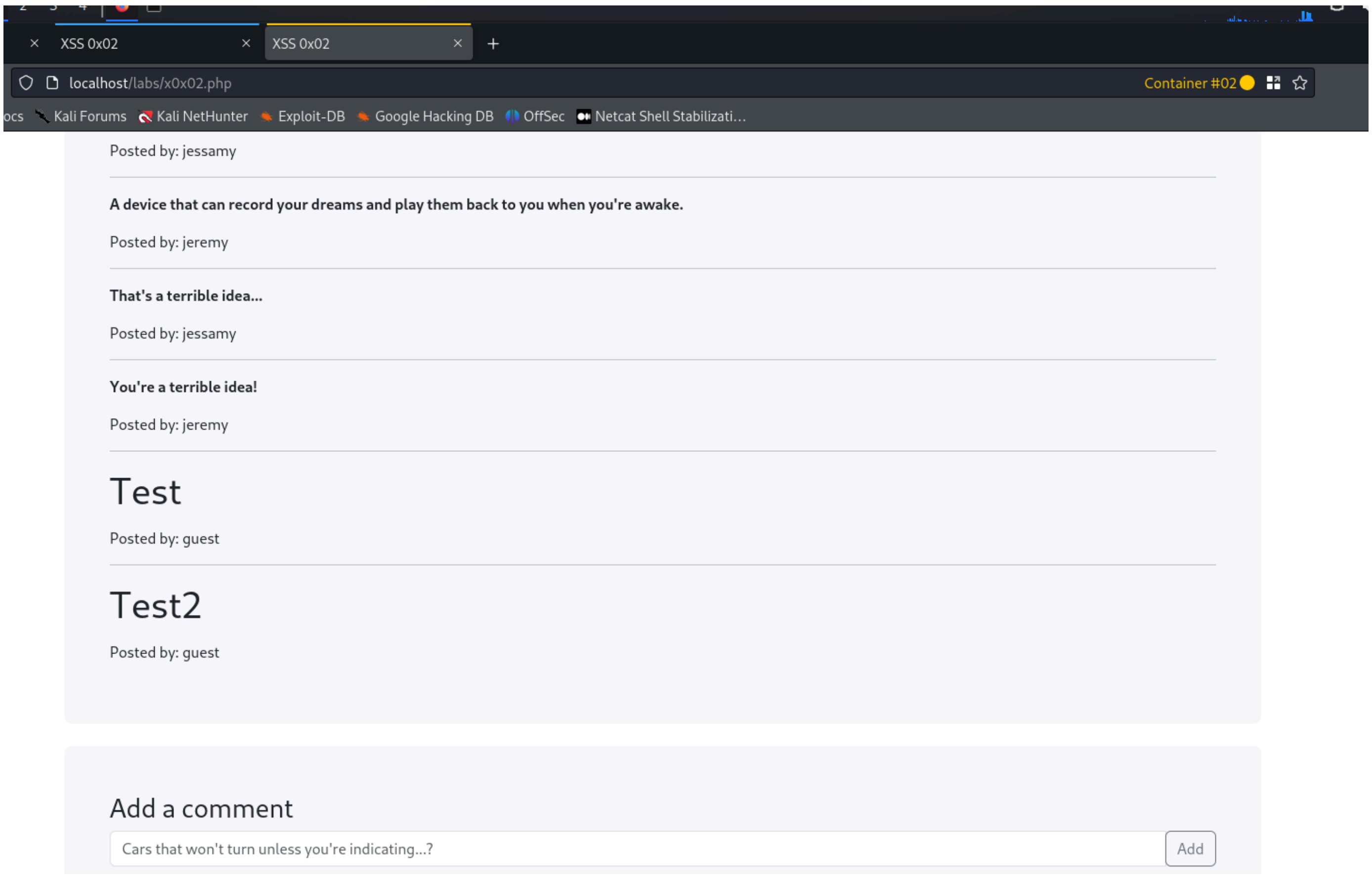
Test2

Posted by: guest

Add a comment

Cars that won't turn unless you're indicating...?

To check if this is indeed a Stored Cross Site Scripting vulnerability, we can go to Container#02, or if your current session is in Container#02 then go to Container#01, reload the page, and we should be able to see the Test2 header we Injected through HTML in container#01.


XSS 0x02 XSS 0x02

localhost/labs/x0x02.php Container #02

ocs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Netcat Shell Stabilizati...

Posted by: jessamy

A device that can record your dreams and play them back to you when you're awake.

Posted by: jeremy

That's a terrible idea...

Posted by: jessamy

You're a terrible idea!

Posted by: jeremy

Test

Posted by: guest

Test2

Posted by: guest

Add a comment

Cars that won't turn unless you're indicating...? Add

So, yes, we do indeed have a Stored Cross Site Scripting vulnerability.

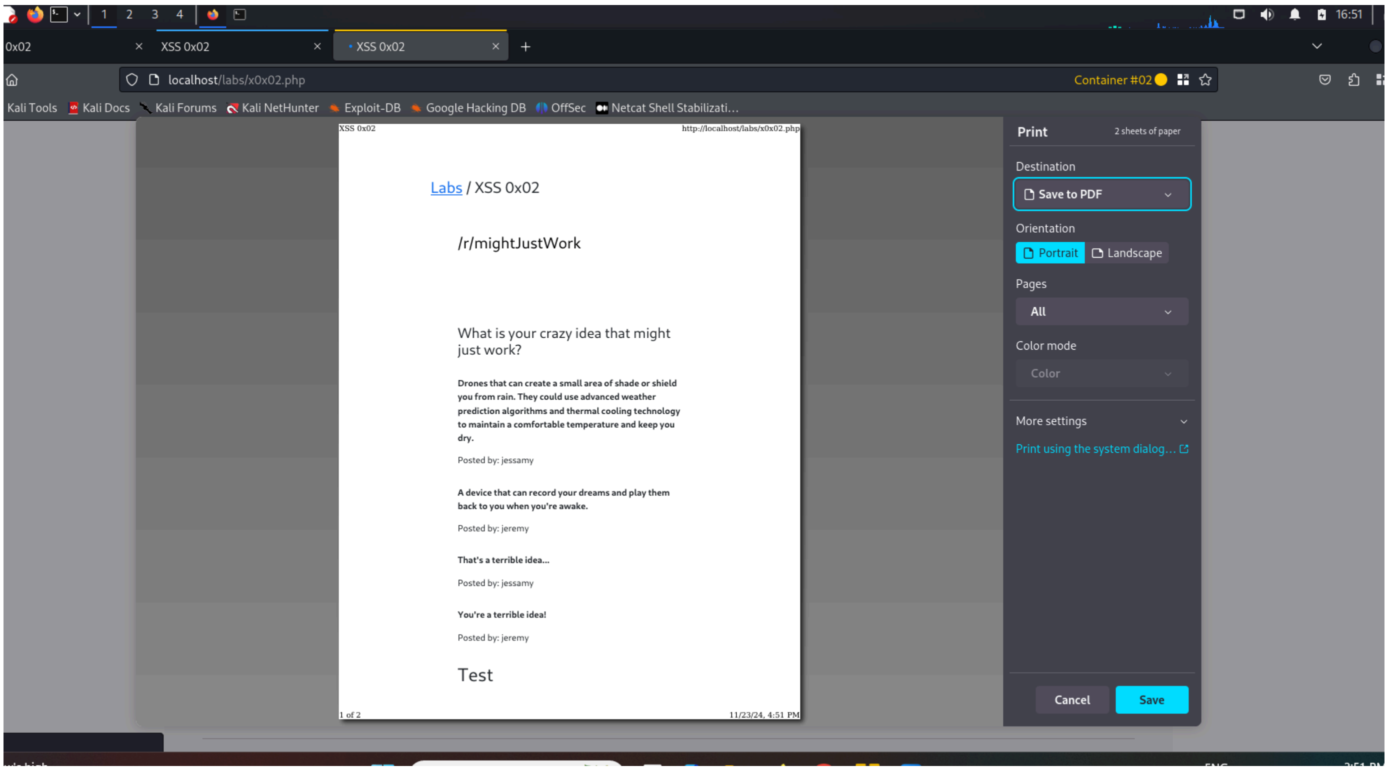
To make a proof of concept, we can use either print() or prompt(), in this case, I will demonstrate with print(). Our instructor already demonstrated with prompt().

Remember, we are dealing with JavaScript and sometimes HTLM.

We can use:

<script>prompt(1)</script> or <script>print()</script>

And, we get the print() screen in the current session because the page was loaded, and the script gets loaded in the page as well. If we were inputting here, to output in another page, this would not happen.


XSS 0x02 | localhost/labs/x0x02.php | Container #02

Labs / XSS 0x02

/r/mightJustWork

What is your crazy idea that might just work?

Drones that can create a small area of shade or shield you from rain. They could use advanced weather prediction algorithms and thermal cooling technology to maintain a comfortable temperature and keep you dry.

Posted by: jessamy

A device that can record your dreams and play them back to you when you're awake.

Posted by: jeremy

That's a terrible idea...

Posted by: jessamy

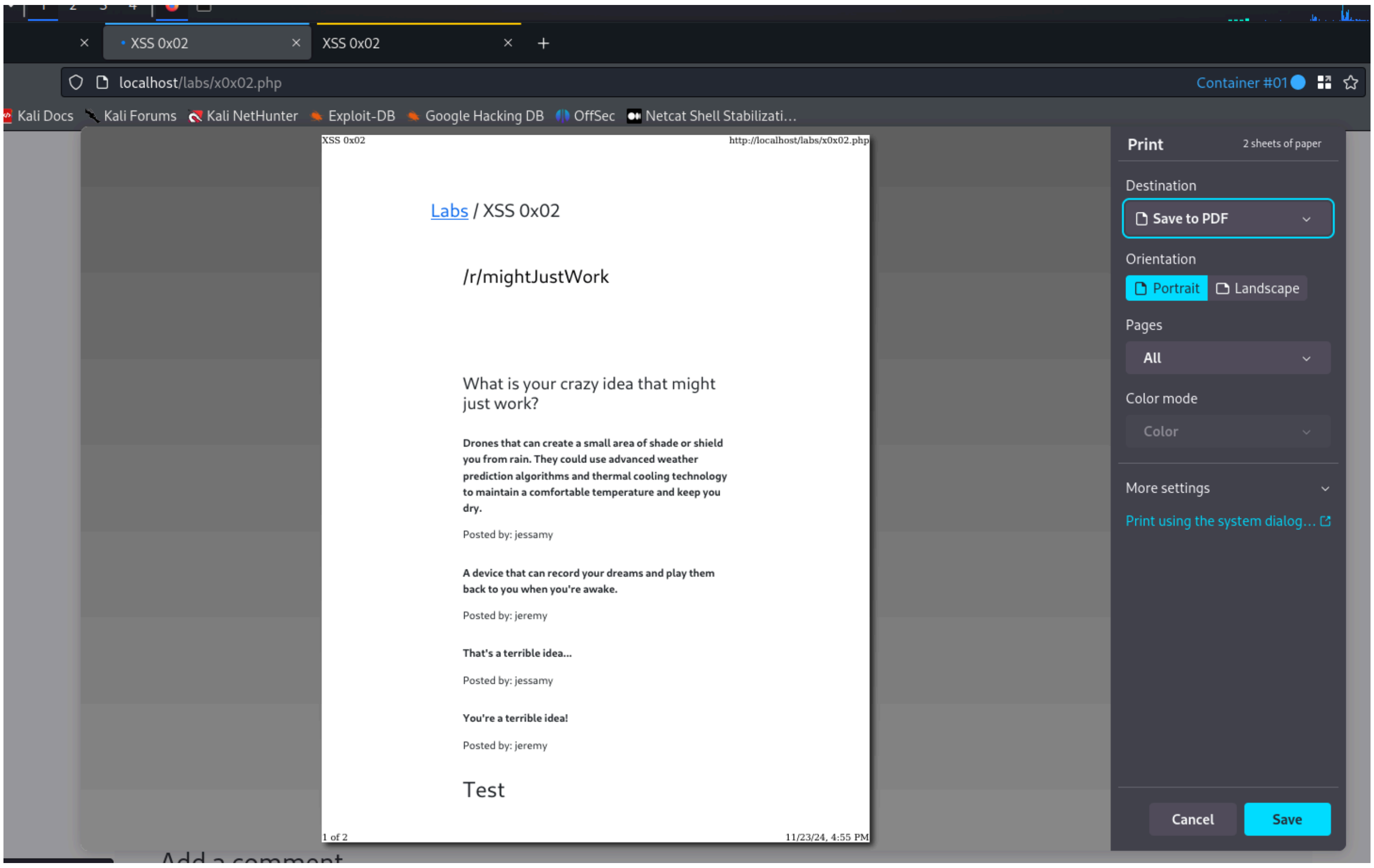
You're a terrible idea!

Posted by: jeremy

Test

Print — 2 sheets of paper. Destination: Save to PDF. Orientation: Portrait. Pages: All. Color mode: Color.

So, when we go back to container#01 now, and reload the page, we are going to get the same print() screen as we saw on Container#02.


XSS 0x02 | localhost/labs/x0x02.php | Container #01

Labs / XSS 0x02

/r/mightJustWork

What is your crazy idea that might just work?

Drones that can create a small area of shade or shield you from rain. They could use advanced weather prediction algorithms and thermal cooling technology to maintain a comfortable temperature and keep you dry.

Posted by: jessamy

A device that can record your dreams and play them back to you when you're awake.

Posted by: jeremy

That's a terrible idea...

Posted by: jessamy

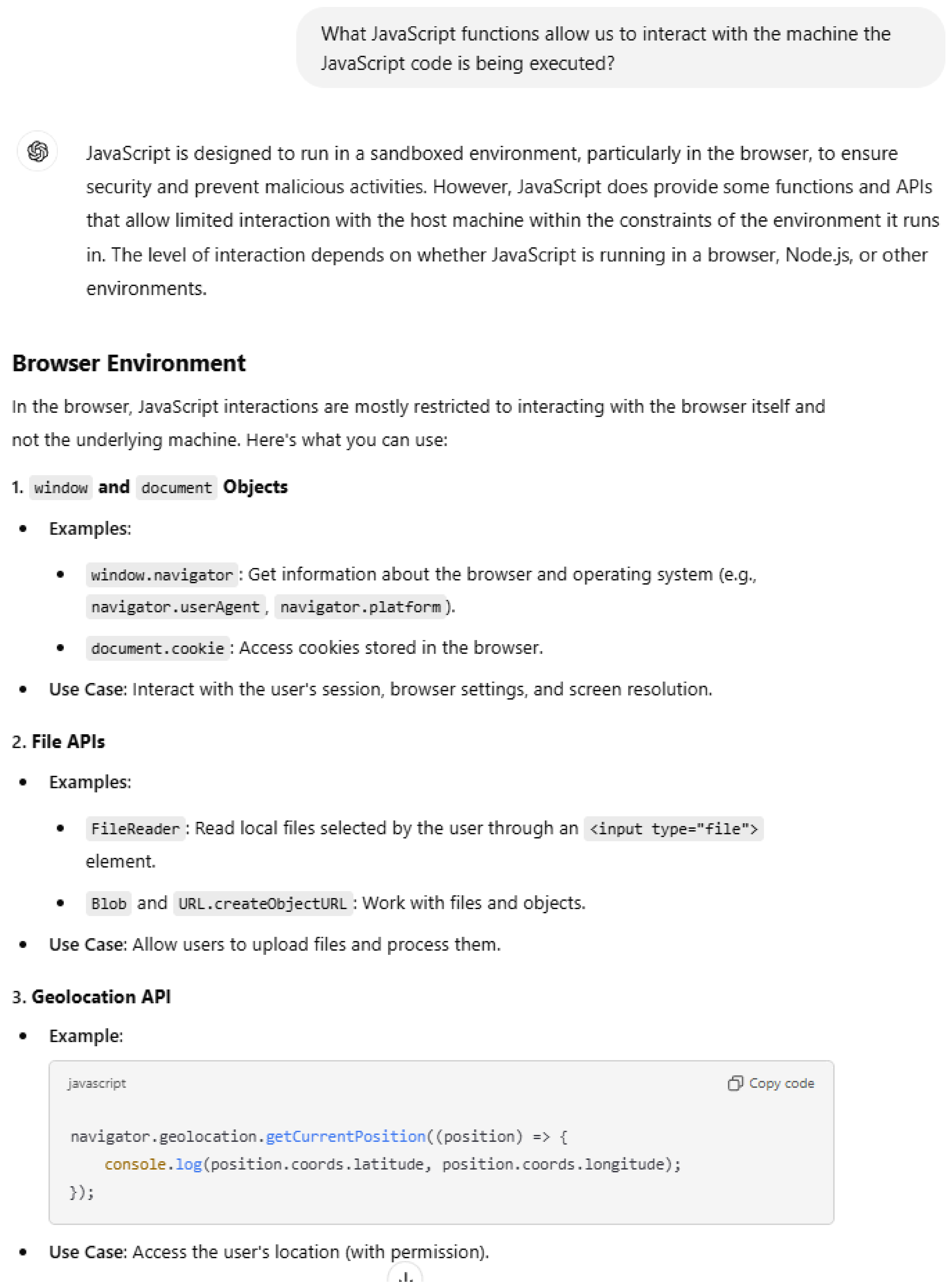
You're a terrible idea!

Posted by: jeremy

Test

Print — 2 sheets of paper. Destination: Save to PDF. Orientation: Portrait. Pages: All. Color mode: Color.

So, every user that comes to this page is going to be impacted by this payload. We could potentially get highjack cookies and use it in our own sessions,



What JavaScript functions allow us to interact with the machine the JavaScript code is being executed?

JavaScript is designed to run in a sandboxed environment, particularly in the browser, to ensure security and prevent malicious activities. However, JavaScript does provide some functions and APIs that allow limited interaction with the host machine within the constraints of the environment it runs in. The level of interaction depends on whether JavaScript is running in a browser, Node.js, or other environments.

Browser Environment

In the browser, JavaScript interactions are mostly restricted to interacting with the browser itself and not the underlying machine. Here's what you can use:

1. `window` and `document` Objects

- Examples:
  - `window.navigator`: Get information about the browser and operating system (e.g., `navigator.userAgent`, `navigator.platform`).
  - `document.cookie`: Access cookies stored in the browser.
- **Use Case**: Interact with the user's session, browser settings, and screen resolution.

2. File APIs

- Examples:
  - `FileReader`: Read local files selected by the user through an `<input type="file">` element.
  - `Blob` and `URL.createObjectURL`: Work with files and objects.
- **Use Case**: Allow users to upload files and process them.

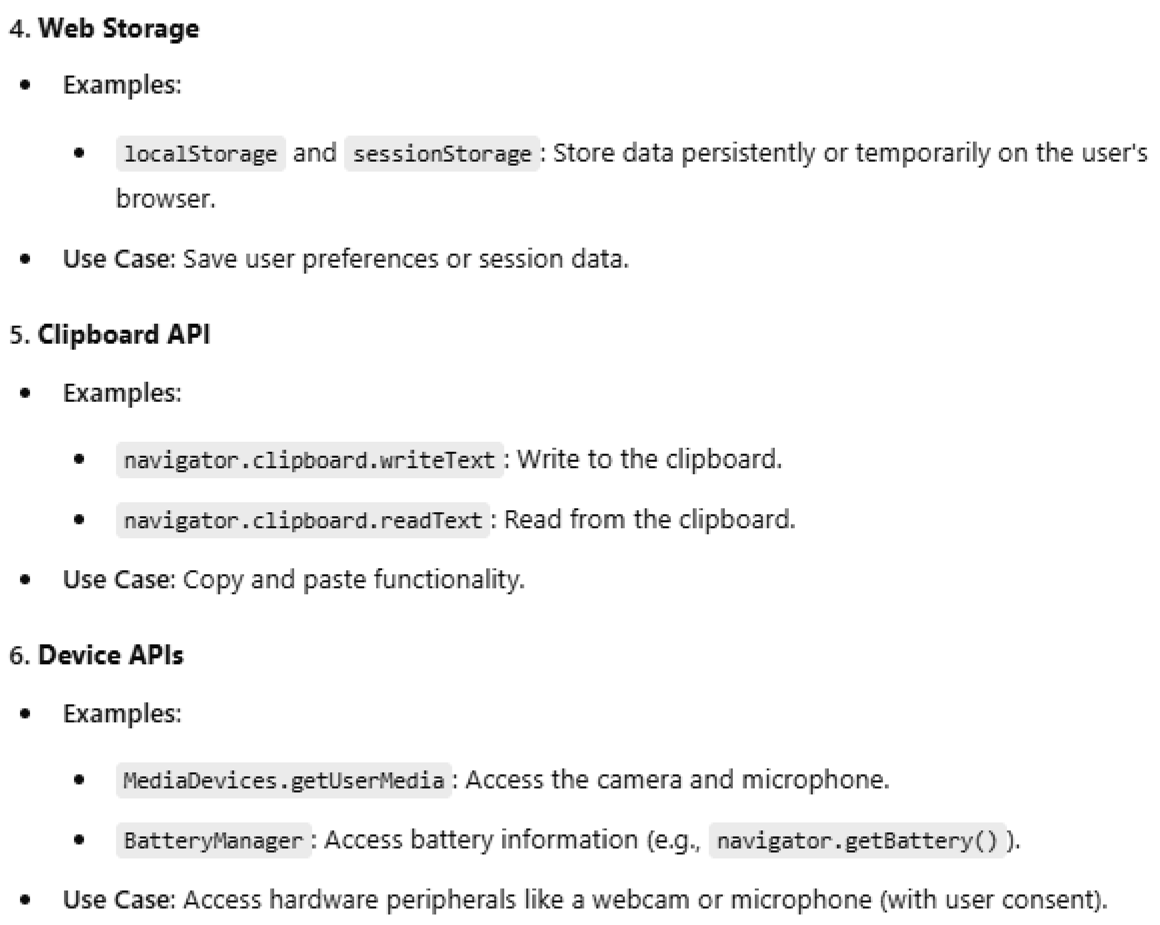
3. Geolocation API

- Example:

```javascript
navigator.geolocation.getCurrentPosition((position) => {
    console.log(position.coords.latitude, position.coords.longitude);
});
```

- **Use Case**: Access the user's location (with permission).

Wow, we can also check the users current location.



4. **Web Storage**

- Examples:
  - `localStorage` and `sessionStorage`: Store data persistently or temporarily on the user's browser.
- **Use Case**: Save user preferences or session data.

5. **Clipboard API**

- Examples:
  - `navigator.clipboard.writeText`: Write to the clipboard.
  - `navigator.clipboard.readText`: Read from the clipboard.
- **Use Case**: Copy and paste functionality.

6. **Device APIs**

- Examples:
  - `MediaDevices.getUserMedia`: Access the camera and microphone.
  - `BatteryManager`: Access battery information (e.g., `navigator.getBattery()`).
- **Use Case**: Access hardware peripherals like a webcam or microphone (with user consent).

If the website is utilizing Node.js, we can leverage more info on the underlying machine:

Node.js Environment

In Node.js, JavaScript has more access to the underlying operating system:

1. **File System**

- Examples:
  - `fs.readFile`: Read files from the file system.
  - `fs.writeFile`: Write to files on the file system.
- **Use Case**: Interact with the local file system for reading and writing files.

2. **Operating System Information**

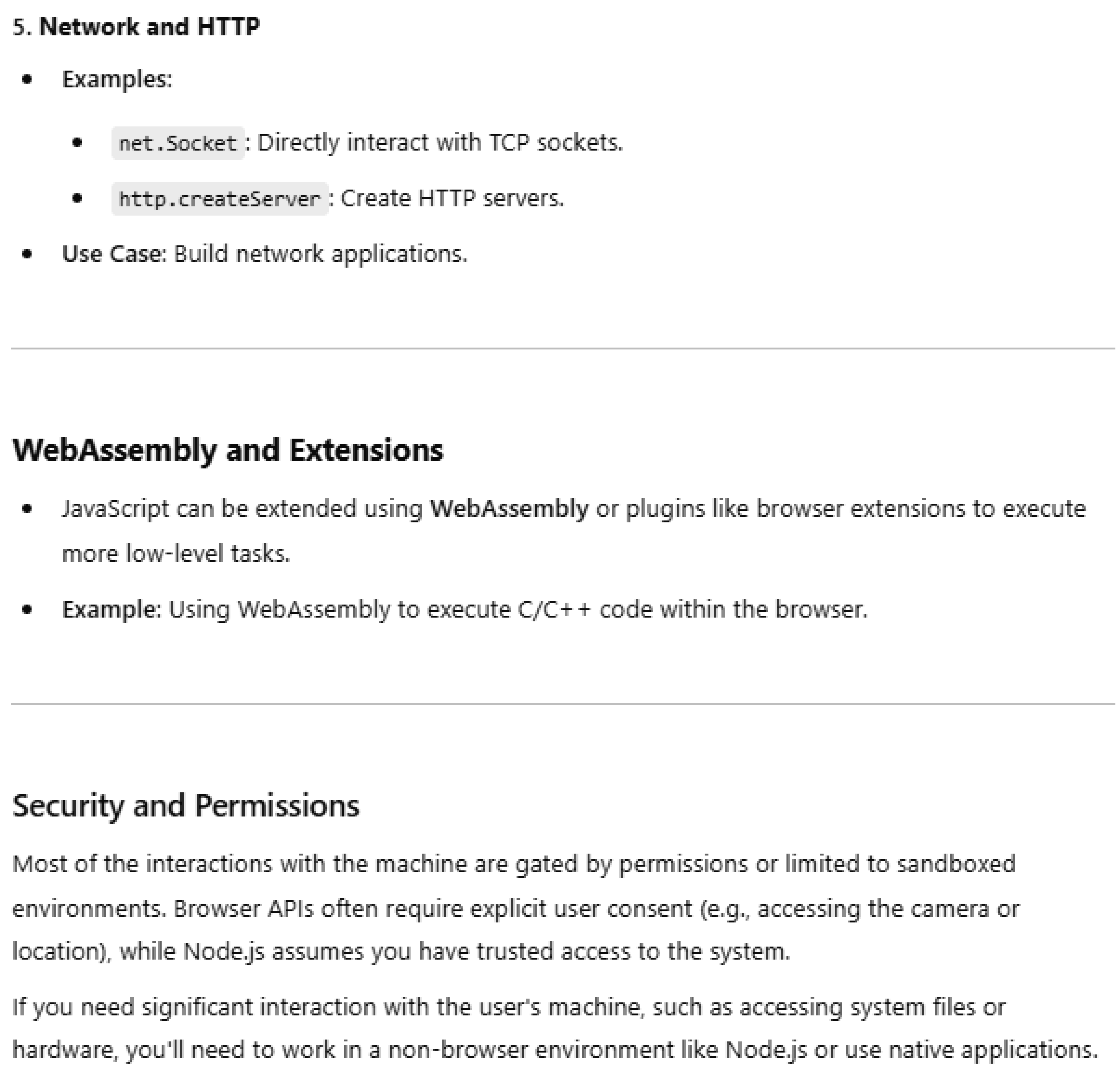
- Examples:
  - `os.platform()`: Get the operating system platform.
  - `os.cpus()`: Get CPU information.
- **Use Case**: Fetch system information.

3. **Process and Environment**

- Examples:
  - `process.env`: Access environment variables.
  - `process.memoryUsage()`: Get memory usage of the application.
- **Use Case**: Manage runtime settings and diagnostics.

4. **Child Processes**

- Examples:
  - `child_process.exec`: Execute system commands.
- **Use Case**: Interact with external applications or scripts.



5. **Network and HTTP**

- Examples:
  - `net.Socket`: Directly interact with TCP sockets.
  - `http.createServer`: Create HTTP servers.
- **Use Case:** Build network applications.

WebAssembly and Extensions

- JavaScript can be extended using **WebAssembly** or plugins like browser extensions to execute more low-level tasks.
- **Example:** Using WebAssembly to execute C/C++ code within the browser.

Security and Permissions

Most of the interactions with the machine are gated by permissions or limited to sandboxed environments. Browser APIs often require explicit user consent (e.g., accessing the camera or location), while Node.js assumes you have trusted access to the system.

If you need significant interaction with the user's machine, such as accessing system files or hardware, you'll need to work in a non-browser environment like Node.js or use native applications.

Good information all over.

Another payload our instructor shows is:

```
<script>alert(document.cookie)</script>
```

For some reason, the cookie value did not show up. Try it multiple time. Maybe JavaScript is not allowed to access my cookies.