

Batch: C3 _____ Roll No.: 110

Experiment No. 5

Title: XSS using dvwa and burp suite

Objective: To hack dummy websites like dvwa using XSS

Expected Outcome of Experiment:

CO	Outcome
CO3	Identify and analyze web attacks

Books/ Journals/ Websites referred:

<https://portswigger.net/web-security/all-labs>

<https://pentest-tools.com/blog/xss-attacks-practical-scenarios#xss-attack-1-hijacking-the-users-session>

Abstract:-

Cross-site scripting (XSS) is a security vulnerability commonly found in web applications. It occurs when an attacker injects malicious scripts into web pages viewed by other users. These scripts can execute in the context of the user's browser, allowing the attacker to steal sensitive information, hijack sessions, or deface websites. XSS attacks can be categorized as stored, reflected, or DOM-based, depending on how the malicious payload is delivered to the victim. Preventive measures such as input validation, output encoding, and using security mechanisms like Content Security Policy (CSP) are essential for mitigating XSS risks and ensuring the security of web applications.

Related Theory: -

XSS Overview: XSS occurs when an attacker injects malicious scripts, usually in the form of JavaScript, into web pages viewed by other users. This can happen through input fields, URLs, or other vulnerable areas of a website.

Types of XSS: There are three main types of XSS: Stored XSS, where the malicious script is permanently stored on the server and executed every time the page is loaded; Reflected XSS, where the malicious script is reflected off the web server, such as in error messages or search results; and DOM-based XSS, where the vulnerability exists in the client-side code rather than the server-side code.

Impact of XSS: XSS can have serious consequences, including stealing user cookies, which can lead to session hijacking; redirecting users to malicious websites; defacing websites; and stealing sensitive information such as usernames, passwords, and credit card details.

Prevention Techniques: To prevent XSS attacks, developers should sanitize user input, encode output, use Content Security Policy (CSP), and validate and sanitize all data before displaying it on a web page.

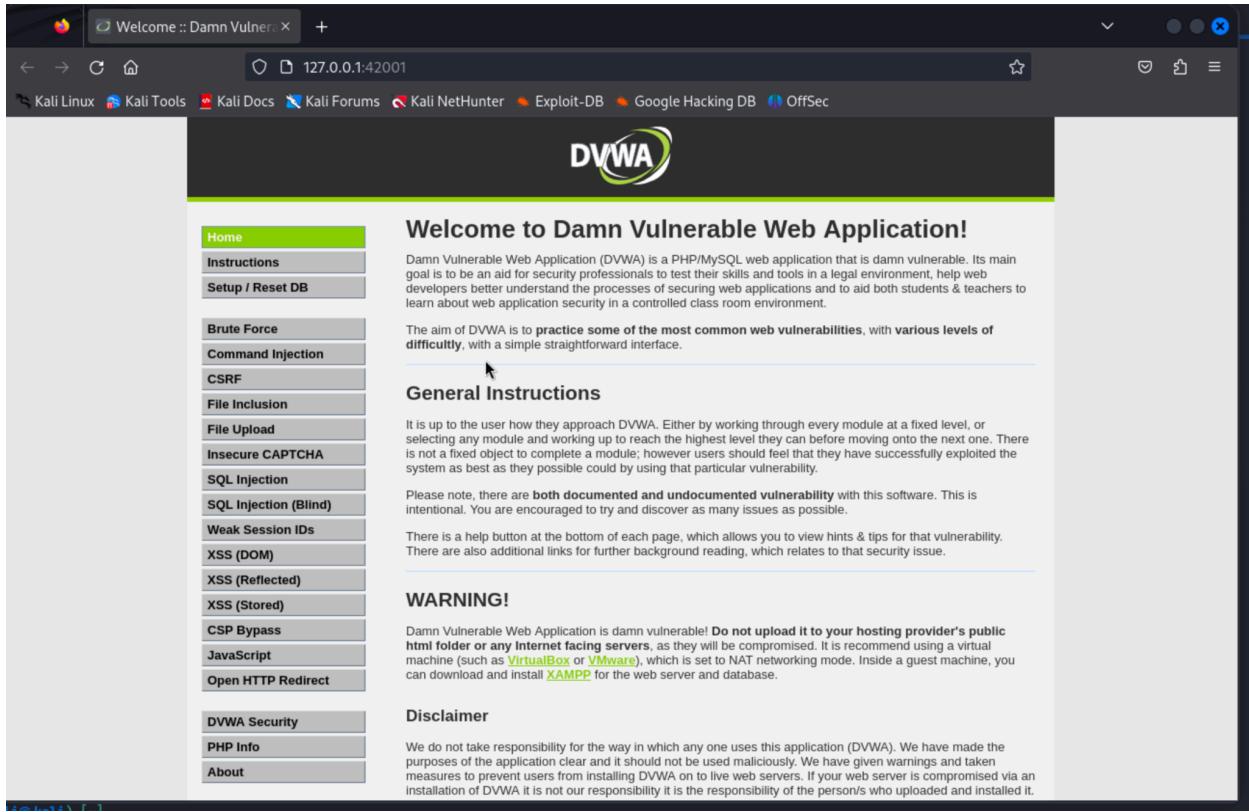
Implementation Details:

In this experiment, we are going to use an alert javascript query as a malicious query. Whenever the page alerts anything it means that it has been hacked. In actual practise, it will be changed from alert to something more malicious code.

Step 1 - Installation of DVWA in KALI

```
(kali㉿kali)-[~]
└─$ sudo apt install dvwa
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
libapache2-mod-php8.2 php8.2 php8.2-cli php8.2-common php8.2-fpm php8.2-gd
php8.2-mysql php8.2-opcache php8.2-readline
Suggested packages:
php-pear
The following NEW packages will be installed:
dvwa php8.2-fpm php8.2-gd
The following packages will be upgraded:
libapache2-mod-php8.2 php8.2 php8.2-cli php8.2-common php8.2-mysql
php8.2-opcache php8.2-readline
7 upgraded, 3 newly installed, 0 to remove and 1838 not upgraded.
Need to get 6498 kB of archives.
After this operation, 7087 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://http.kali.org/kali kali-rolling/main arm64 php8.2-readline arm64 8.2.12-1+b1 [12.1 kB]
Get:3 http://http.kali.org/kali kali-rolling/main arm64 php8.2-mysql arm64 8.2.12-1+b1 [108 kB]
Get:4 http://http.kali.org/kali kali-rolling/main arm64 libapache2-mod-php8.2 arm64 8.2.12-1+b1 [1503 kB]
Get:5 http://http.kali.org/kali kali-rolling/main arm64 php8.2-cli arm64 8.2.12-1+b1 [1562 kB]
Get:6 http://http.kali.org/kali kali-rolling/main arm64 php8.2-common arm64 8.2.12-1+b1 [651 kB]
Get:7 http://http.kali.org/kali kali-rolling/main arm64 php8.2-fpm arm64 8.2.12-1+b1 [1574 kB]
Get:8 http://kali.download/kali kali-rolling/main arm64 php8.2 all 8.2.12-1 [32.6 kB]
Get:10 http://kali.download/kali kali-rolling/main arm64 dvwa all 2.2.2-0kali1 [532 kB]
Get:2 http://http.kali.org/kali kali-rolling/main arm64 php8.2-opcache arm64 8.2.12-1+b1 [496 kB]
98% [Connecting to mirrors.neusoft.edu.cn (219.216.128.25)]
```

Step 2 open dvwa using dvwa-start



Step 3: perform xss injection

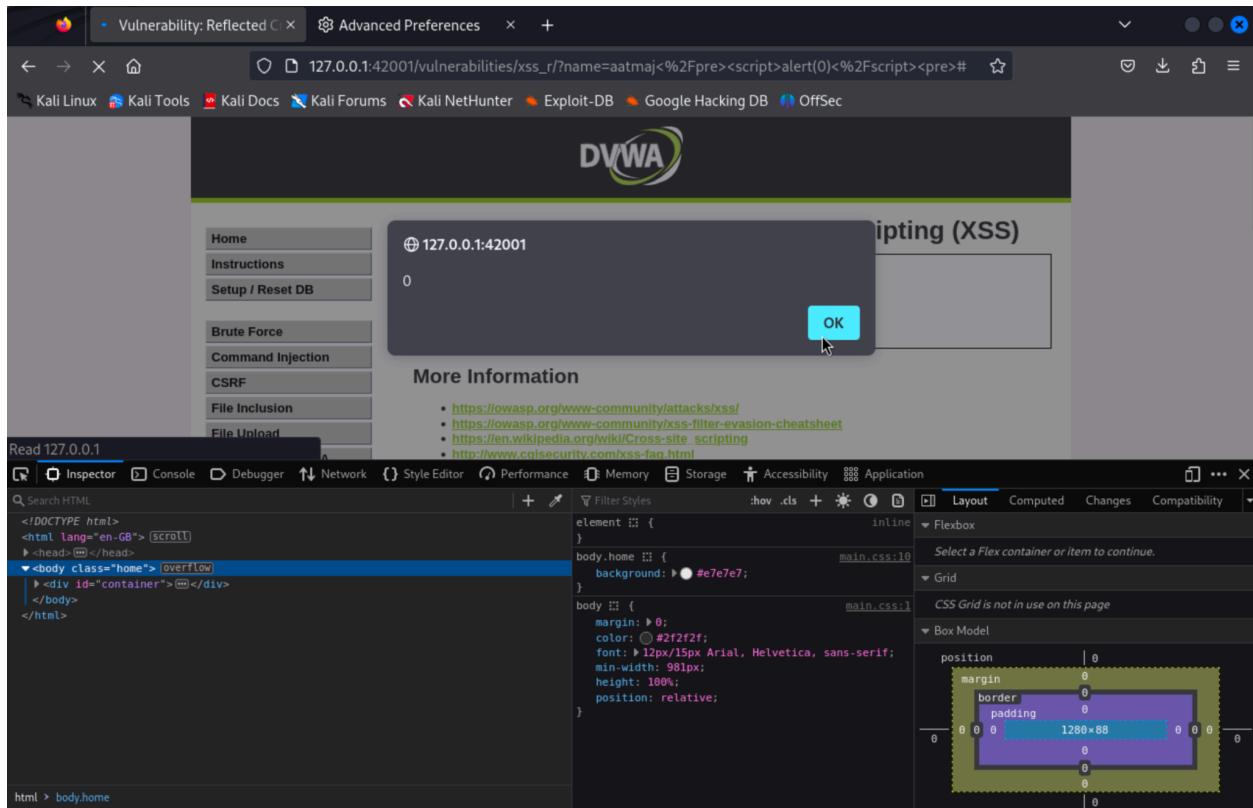
In the dvwa search bar, we can find that what we input comes in the output without any formatting. We can exploit this by entering queries like this

```
aatmaj</pre><script>alert(0)</script><pre>
```

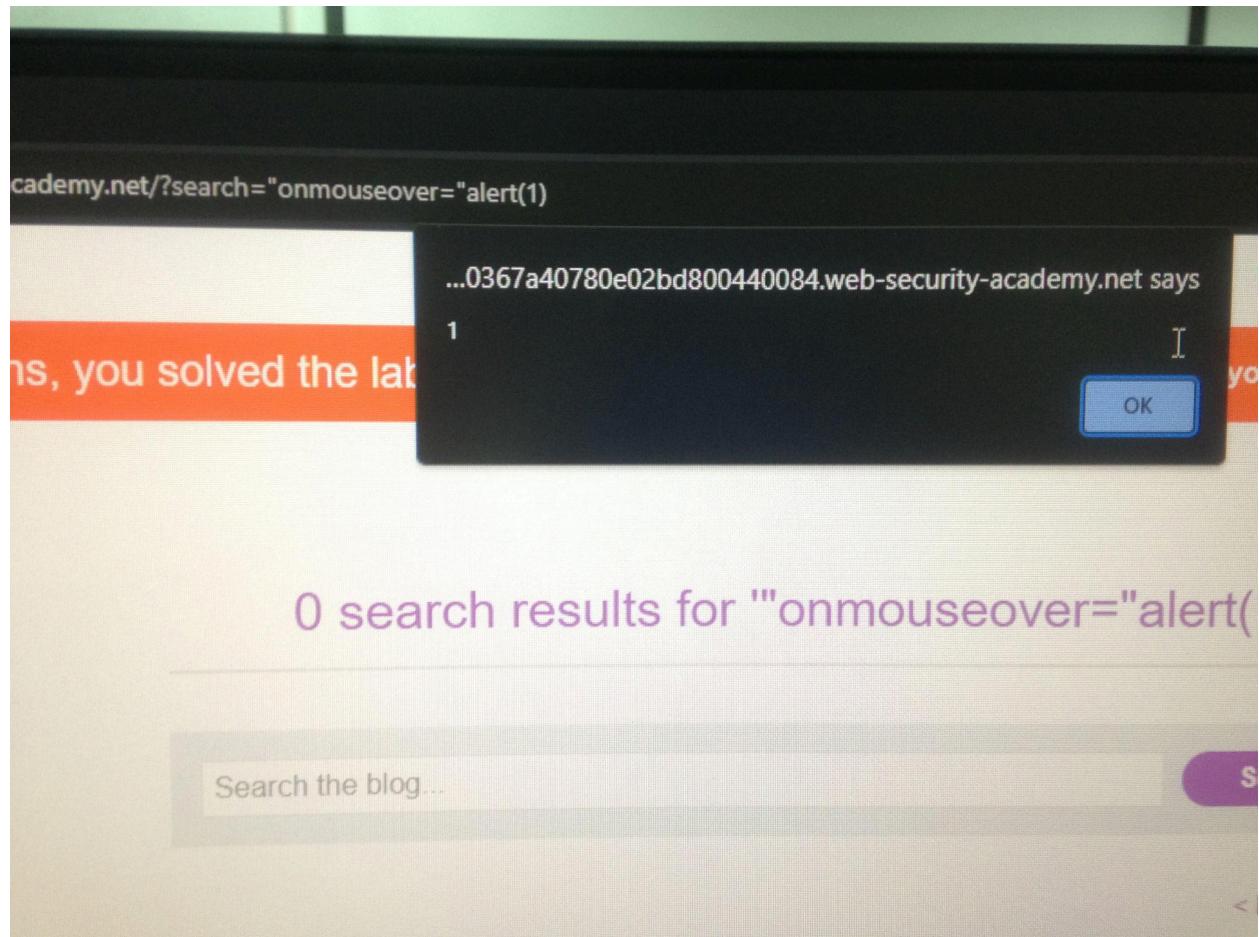
This will execute an malicious script.

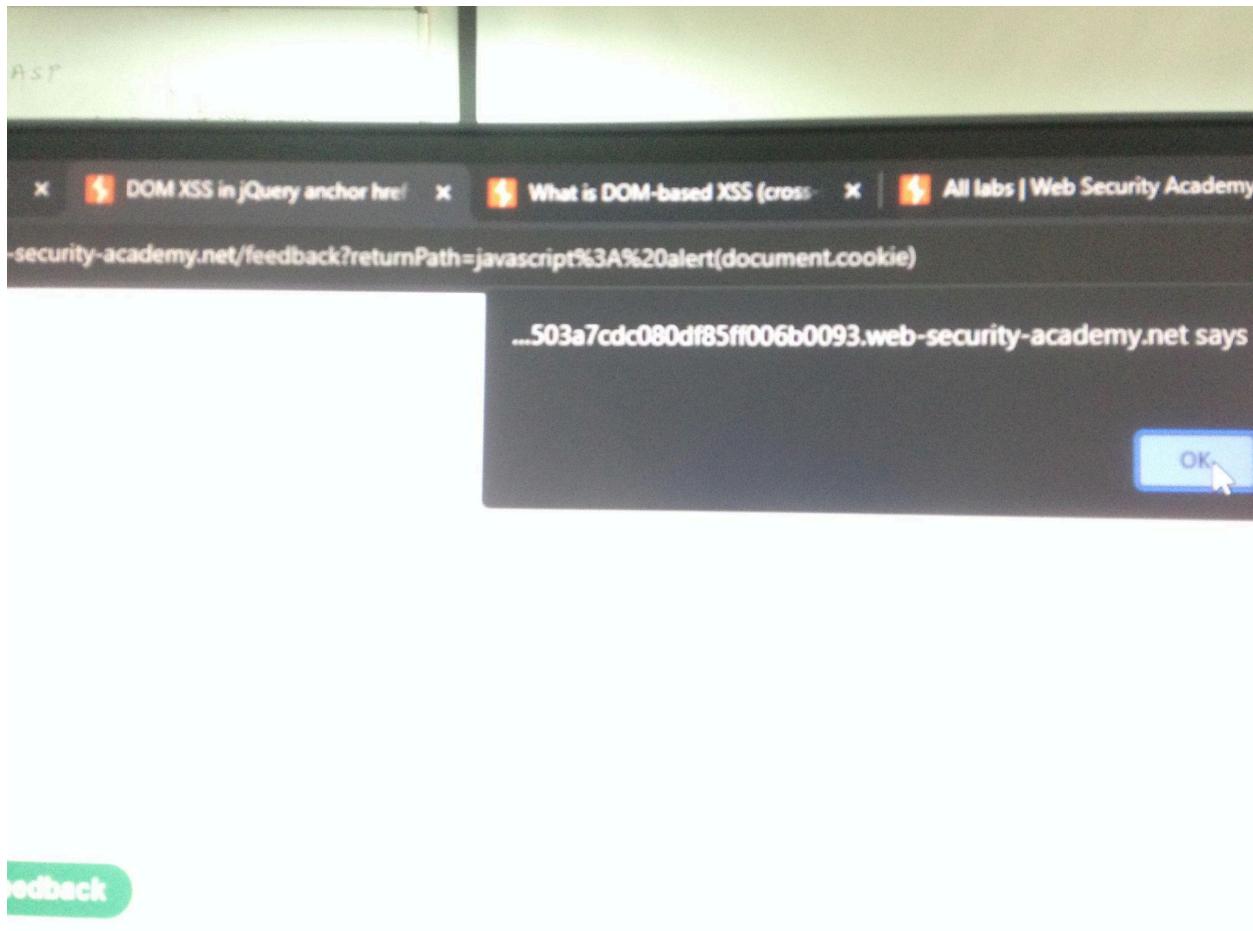
The screenshot shows a browser window on a Kali Linux system (indicated by the taskbar icons) displaying the DVWA (Damn Vulnerable Web Application) 'Reflected Cross Site Scripting (XSS)' page. The URL in the address bar is 127.0.0.1:42001/vulnerabilities/xss_r/?name=aatmaj<%2Fpre><script>alert(0)<%2Fscript><pre>#. The DVWA logo is at the top right. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. The main content area has a heading 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below it is a form with the placeholder 'What's your name?' containing the value '<script>alert(0)</script><pre>'. A 'Submit' button is next to it. The output of the script is displayed below the form, showing 'Hello aatmaj'. To the right of the browser window, a developer tools window is open, specifically the 'Layout' tab of the CSS inspector. It shows the DOM structure of the page and the computed styles for the body.home element, which has a background color of #e7e7e7. The layout panel displays a visual representation of the page's structure with dimensions and margins.

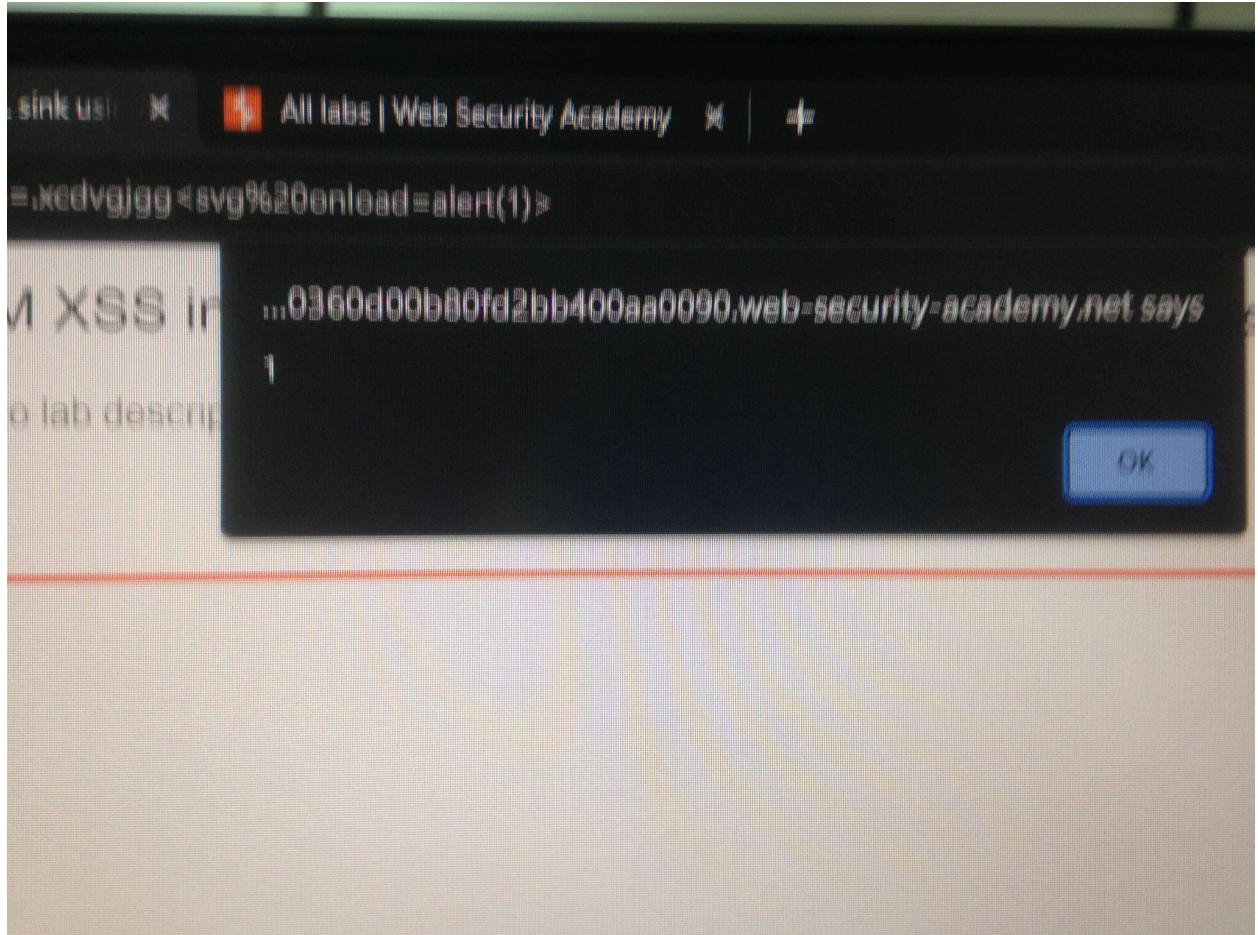
And alert is launched

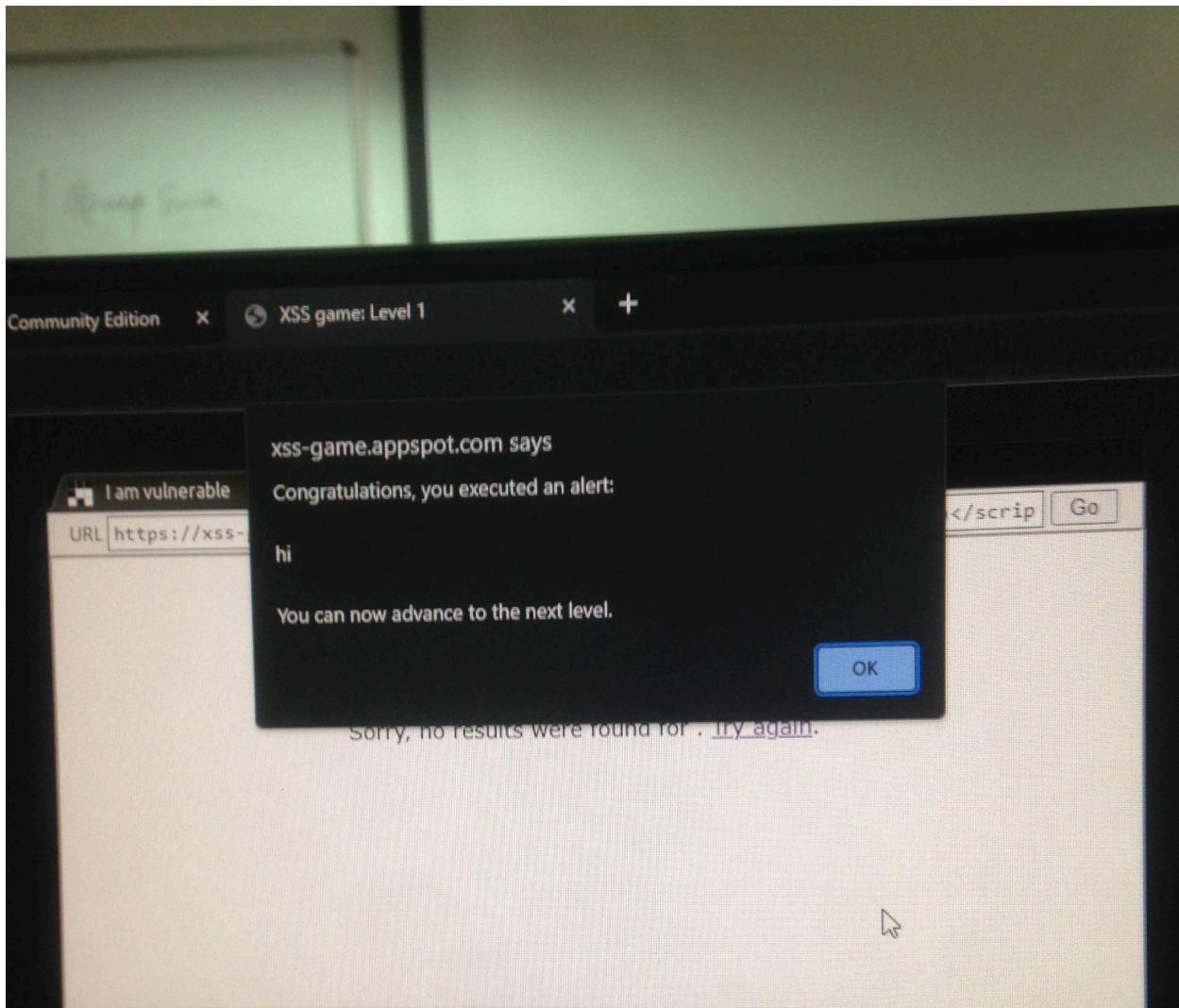


Step 5 Practising various forms of XSS injection in Burp Suite practise websites.



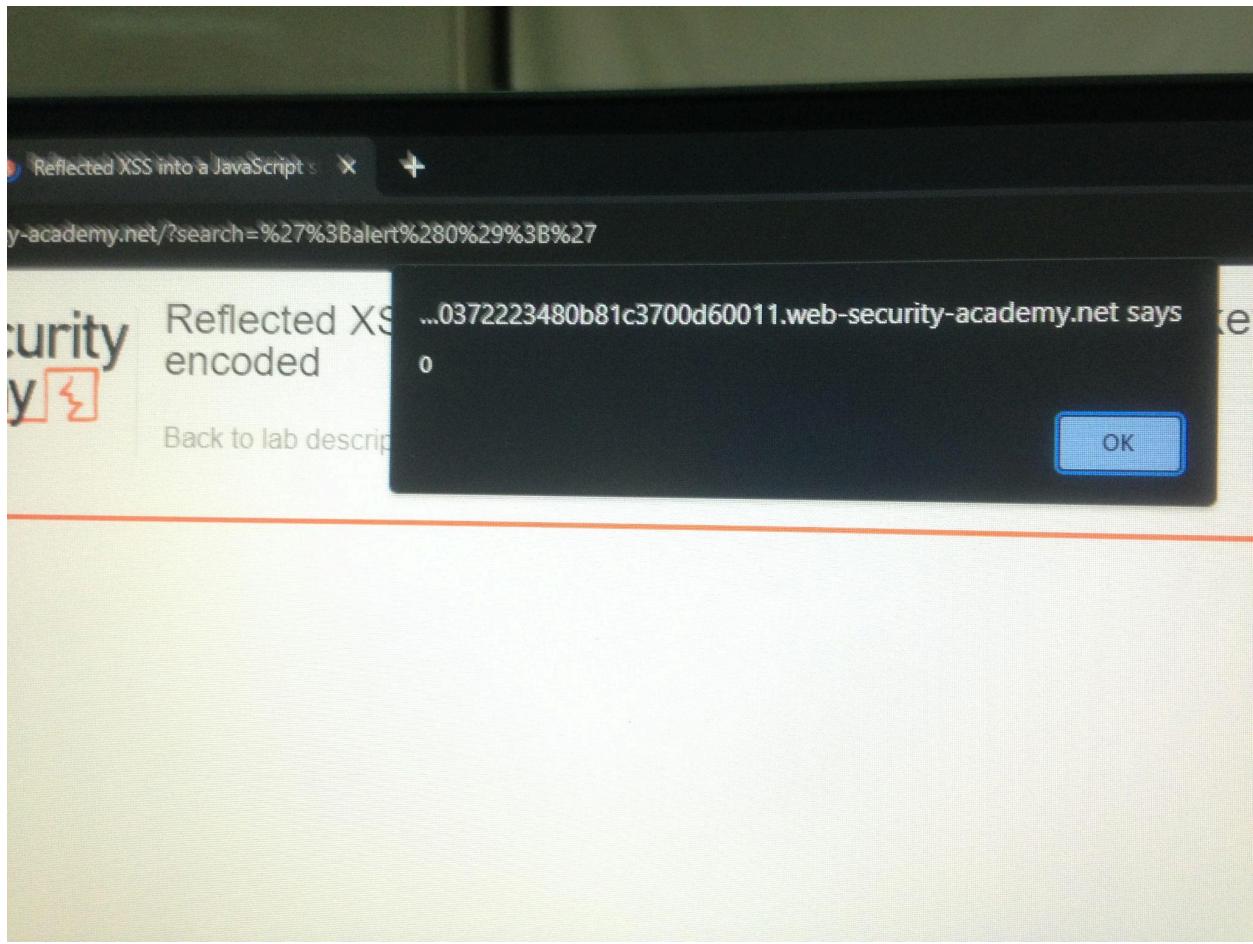




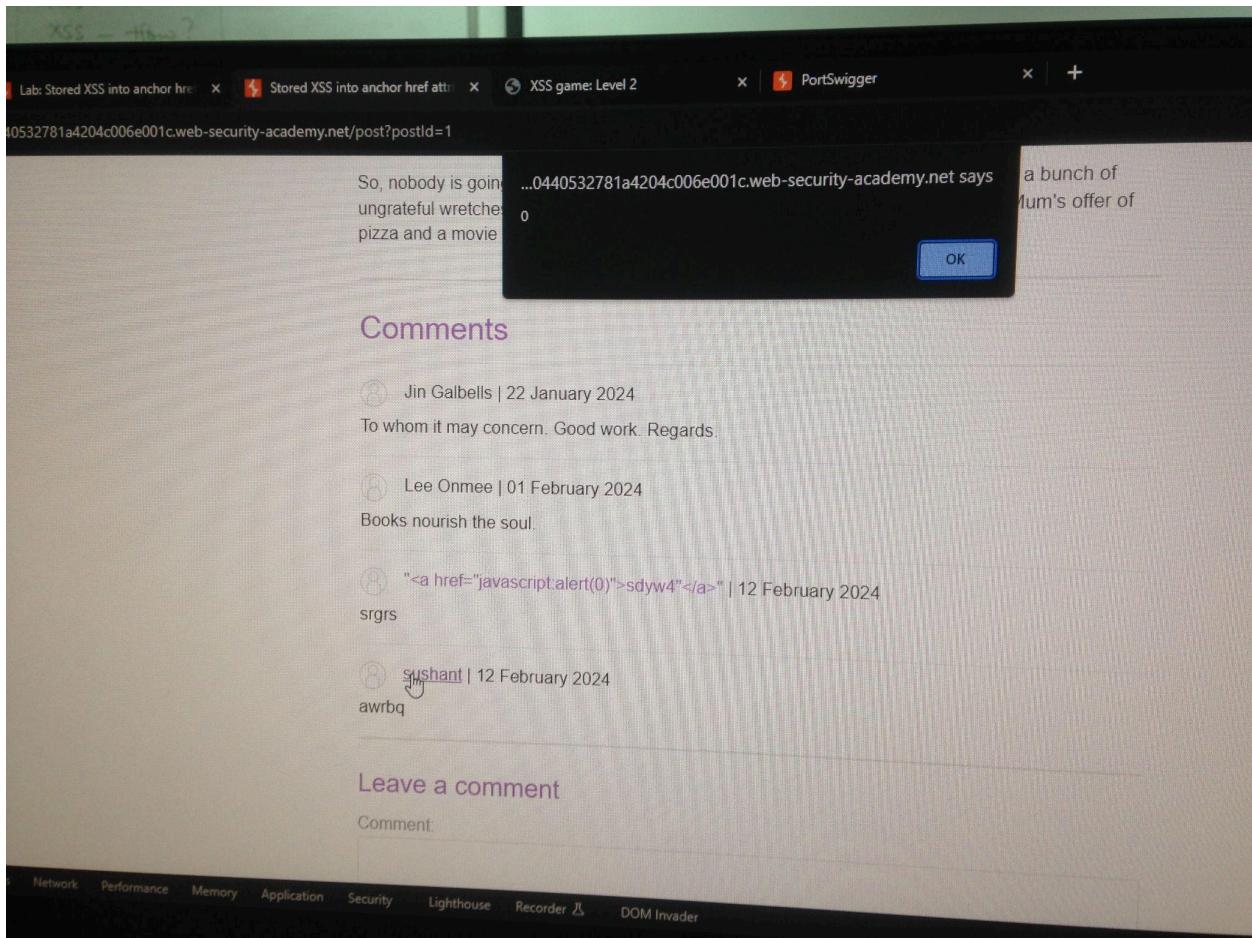


Target code ([toggle](#))

```
34 def get(self):
35     # Disable the reflected XSS filter for demonstration purposes
36     self.response.headers.add_header("X-XSS-Protection", "0")
37
38     if not self.request.get('query'):
39         # Show main search page
40         self.render_string(page_header + main_page_markup + page_footer)
41     else:
42         query = self.request.get('query', '[empty]')
43
44         # Our search engine broke, we found no results :-( 
45         message = "Sorry, no results were found for <b>" + query + "</b>."
46         message += " <a href='?>Try again</a>."
47
48         # Display the results page
49         self.render_string(page_header + message + page_footer)
50
51     return
52
53 application = webapp.WSGIApplication([ ('.*', MainPage), ], debug=False)
```



Step 6: Stored XSS



Cross-site scripting



APPRENTICE

Reflected XSS into HTML context with nothing encoded →

✓ Solved



APPRENTICE

Stored XSS into HTML context with nothing encoded →

✓ Solved



APPRENTICE

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

✓ Solved



APPRENTICE

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

✓ Solved



APPRENTICE

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

✓ Solved



APPRENTICE

DOM XSS in jQuery selector sink using a hashchange event →

✓ Solved



APPRENTICE

Reflected XSS into attribute with angle brackets HTML-encoded →

✓ Solved



APPRENTICE

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

✓ Solved



APPRENTICE

Reflected XSS into a JavaScript string with angle brackets HTML
encoded →

✓ Solved



PRACTITIONER

DOM XSS in `document.write` sink using source `location.search`
inside a select element →

Not solved

Conclusion:- Cross-site scripting (XSS) is a common security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. This can lead to various attacks, such as stealing sensitive information, session hijacking, and defacing websites. In this document, we have explored the concept of XSS, its types and its impact on web security. Thus we have performed XSS attacks on dummy websites. We have performed stored and reflected XSS attacks. Various forms of XSS attacks have been analyzed.