Internet Security – COS80013 Lab - 7 Report

Internet Security – COS80013 Lab - 7 Report
Student ID: 104837257
Student Name: Arun Ragavendhar Arunachalam Palaniyappan
Lab Name: COS80013 Lab 7 – Cross-Site Scripting (XSS) Attacks
Lab Date: 24 /04/2025
Tutor: Yasas Akurudda Liyanage Don

Title and Introduction

This lab was about learning how attackers use Cross-Site Scripting (XSS) to run malicious code inside websites. It showed how vulnerable sites can allow attackers to steal session cookies, change how pages look, redirect users to fake websites, or trick users into giving up passwords. The lab used the JITXSS platform, with both Unsecure and Secure forums, to demonstrate what happens when input is not properly sanitised.

Methodology

Basic Script Injection: Logged in to the Unsecure Forum as hacker. Added a basic script using <script>alert()</script> to test for XSS. The popup confirmed the site was vulnerable.

Cookie and URL Extraction: Added another script to display document.cookie and document.Location, showing how attackers can view a user's session ID and current page.

UI Manipulation: Injected a colour-changing script using a dropdown and JavaScript to modify table backgrounds dynamically.

Browser Fingerprinting: Posted a script that extracted properties from the browser's navigator object, revealing system details.

Secure Forum Testing: All scripts were re-posted in the Secure Forum but were displayed as plain text due to sanitisation using htmlspecialchars() and htmlentities().

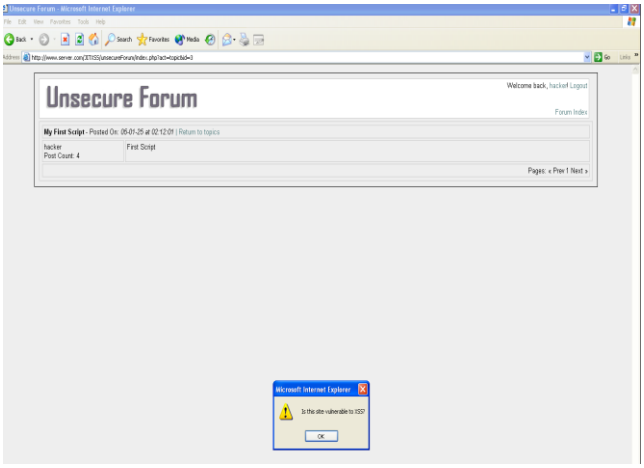
Redirect Attacks: Used two XSS scripts to redirect users—one on link click, one auto-redirect. Both worked on the Unsecure Forum but were blocked in the Secure Forum.

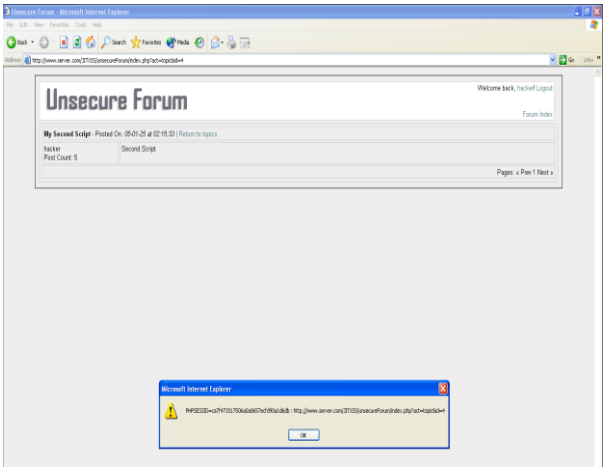
Session Stealing: A script was posted to capture a victim's cookie and send it to the Hacker's Console. The attacker copied the session ID and used a cookie editor to log in as the victim successfully.

Phishing Simulation: Injected a script that redirected victims to a fake login page. Once the victim entered their details, the credentials appeared in the Hacker's Console.

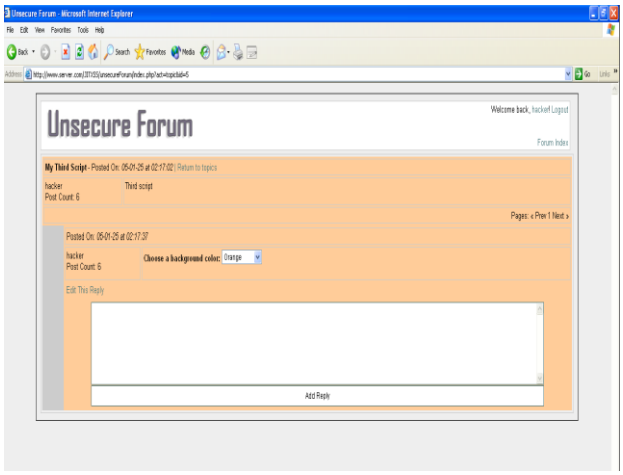
Data Recording and Screenshots

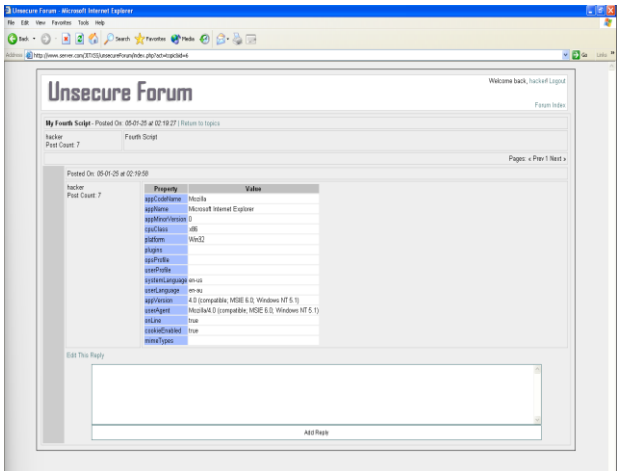
- Script alert, Cookies and URL displayed through alert popup, confirming XSS vulnerability.



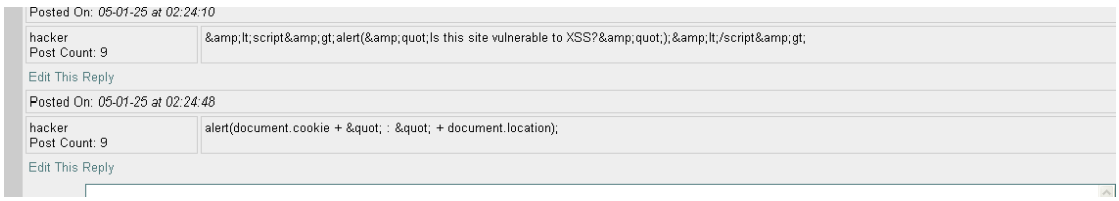


- Forum background changed using injected dropdown and browser details shown using navigator object.

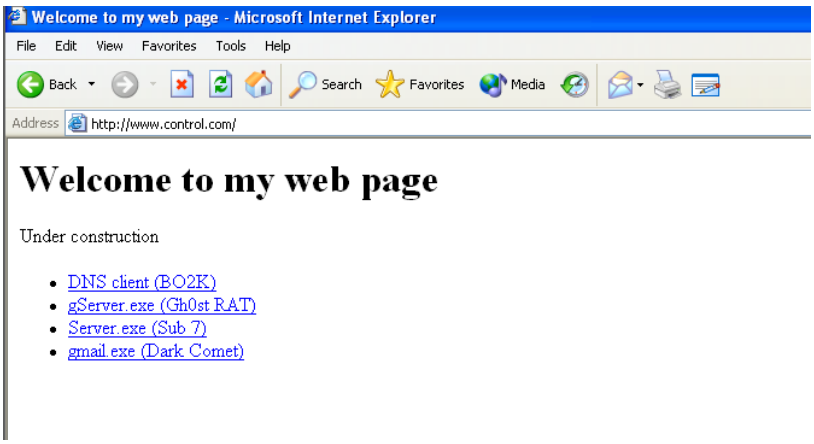




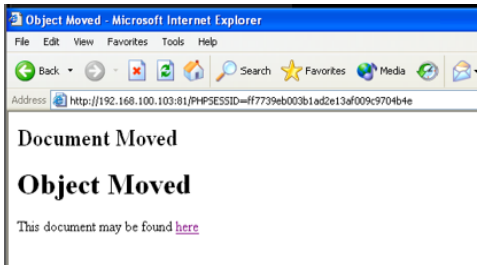
- Secure Forum displayed all code as harmless text after sanitising it.

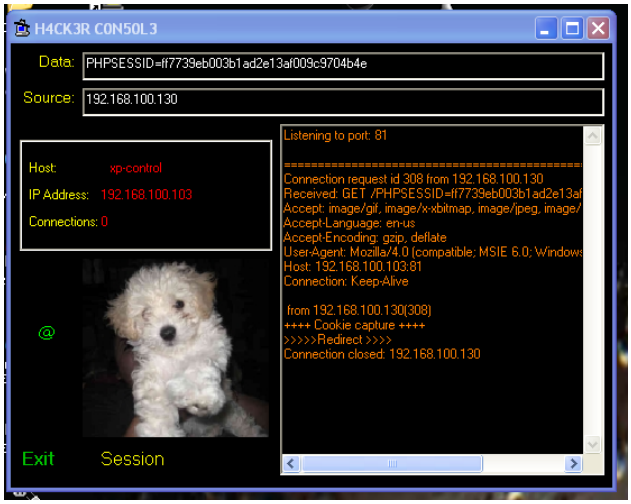


- Redirection to control.com confirmed on unsecure side

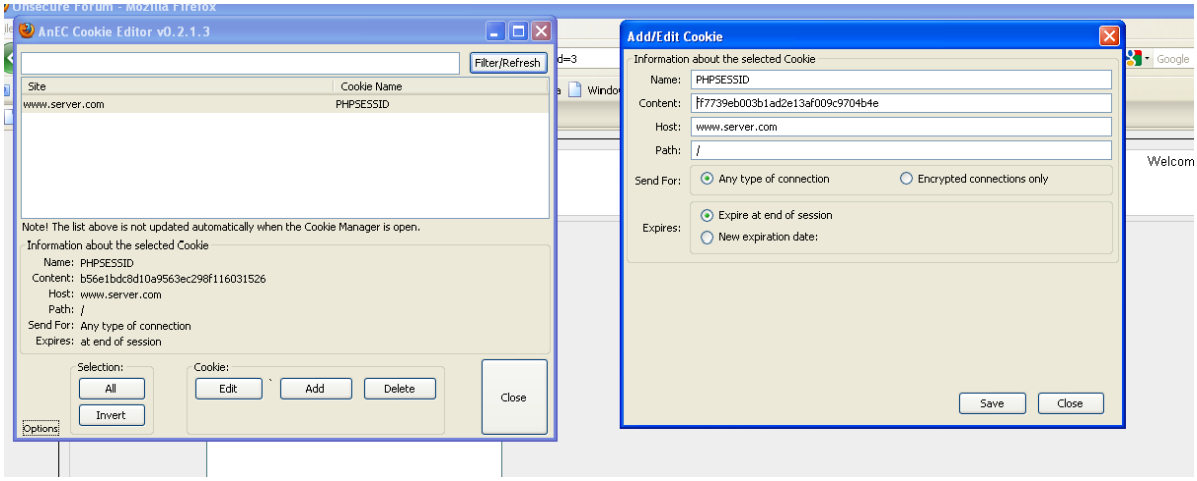


- Session Stealing – Stealing session ID from Victim system and hacker console getting the session ID from the transferred cookie on the hacker side.



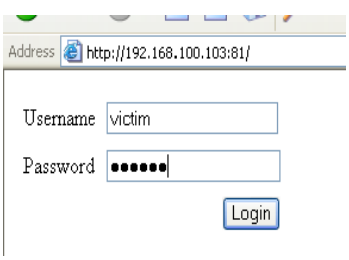


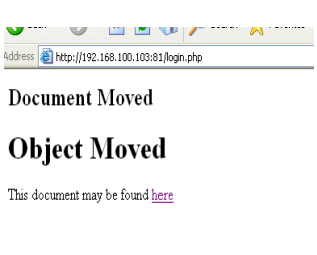
- Using the stolen session ID to login as the “victim” on the hacker’s VM

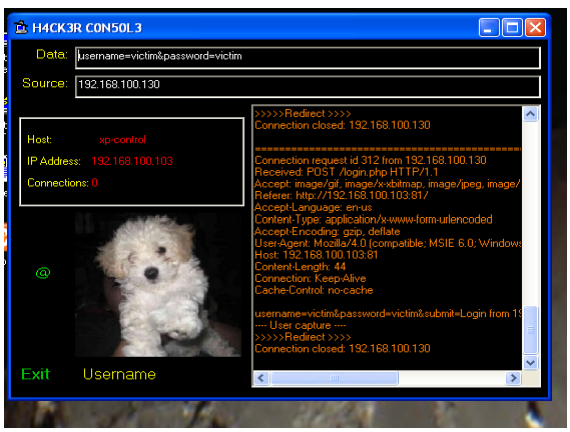


- Successful login as victim on hacker system

- Phished credentials collected from victim VM and visible on the hacker console.







Discussion and Learnings

Learning 1
Posting scripts directly into a forum reply made them execute instantly. This showed how vulnerable sites can be exploited by just injecting a small script.
Real-World Link: Many old forums or comment sections remain vulnerable to this. Attackers can easily exploit these to steal user data or run malicious code.

Learning 2
Sanitising user input using htmlentities() and stripping tags can stop scripts from running, even if they're submitted.
Real-World Link: Developers can prevent most XSS attacks by escaping or rejecting unsafe input. Frameworks should do this by default.

Learning 3
Modern browsers allow changing hidden fields, cookies, or scripts using developer tools. This helped trick the forum into doing things it should not do normally.
Real-World Link: If a system checks only on the browser side, attackers can bypass it easily. Server-side validation is a must.

Learning 4
The stolen cookie allowed logging in as another user without their password. No alerts or warnings were triggered.
Real-World Link: If session IDs are not protected using flags like HttpOnly and Secure, they can be stolen and reused by attackers.

Learning 5
Redirects and fake login pages worked without any issues. It was hard for the victim to notice they were being tricked.
Real-World Link: Phishing often starts with a simple redirect or popup that looks like the original site. Users can easily fall for it without proper awareness or browser protection.

Limitations

The lab was done in a sandboxed environment using pre-built VMs. In real life, most browsers have XSS filters, and websites implement stricter content policies (like CSP). Also, attacks involving redirection or session stealing may get blocked by modern browsers, web application firewalls, or intrusion detection systems.