TASK 6 – Cyber Security

🔐 Web Application Security Audit Report

1️⃣ Target Website

URL: http://testphp.vulnweb.com
Type: Intentionally vulnerable test web application (legal lab)
Purpose: Educational security testing and vulnerability assessment

---

2️⃣ Tools Used

- **OWASP ZAP** – Automated vulnerability scanning
- **Manual Testing (Browser-based)** – Input validation & logic testing
- **Kali Linux** – Testing environment

3️⃣ Vulnerabilities Found

Vulnerability	Risk Level	Impact
SQL Injection	High	Database data exposure, authentication bypass
Cross-Site Scripting (XSS)	Medium	Session hijacking, malicious script execution
Missing Security Headers	Medium	Increased attack surface
Insecure Input Validation	High	Injection attacks
No HTTPS Enforcement	Low	Data interception risk

4️⃣ Exploitation Overview

🔴 SQL Injection (High Risk)

Some input fields (such as search and login parameters) did not properly sanitize user input. By inserting basic SQL payloads like ' OR '1'='1, the application responded abnormally, indicating possible SQL injection.

Impact:
Attackers can read, modify, or delete database records and bypass login authentication.

🟠 Cross-Site Scripting – XSS (Medium Risk)

User input fields reflected JavaScript payloads like <script>alert(1)</script> without proper encoding.

Impact:
An attacker can inject malicious scripts that run in a victim's browser, leading to session hijacking or phishing attacks.

🟠 Missing Security Headers (Medium Risk)

Important HTTP headers such as:

- Content-Security-Policy
- X-Frame-Options
- X-Content-Type-Options

were missing.

Impact:
Makes the application more vulnerable to clickjacking and content injection attacks.

🔴 Insecure Input Validation (High Risk)

Inputs were not properly validated or filtered.

Impact:
Allows multiple injection attacks including SQLi and XSS.

🟡 No HTTPS Usage (Low Risk)

The application runs over HTTP instead of HTTPS.

Impact:
Data sent between client and server can be intercepted using man-in-the-middle attacks.

5️⃣ Security Recommendations

✓ Implement **server-side input validation and sanitization**
✓ Use **prepared statements / parameterized queries** to prevent SQL Injection
✓ Encode output to prevent **XSS attacks**
✓ Enforce **HTTPS** using SSL/TLS certificates
✓ Add important **security headers**:

- Content-Security-Policy
- X-Frame-Options
- Strict-Transport-Security
  ✓ Implement secure authentication and session handling
  ✓ Regularly perform security testing and audits

🆕 📝 What I Learned From This Task

From this task, I learned how to conduct a practical web application security audit using both automated and manual techniques. I understood how common vulnerabilities like SQL Injection and XSS are identified, analyzed, and categorized based on risk level. This task improved my understanding of real-world web security flaws, exploitation methods, and how to document findings professionally. It also enhanced my skills in using OWASP ZAP and applying cybersecurity best practices for secure web development.