Web Application Penetration Testing with OWASP ZAP - Report

Task Overview:

This task involves setting up OWASP ZAP, configuring the browser, crawling the web application using the Spider tool, analyzing traffic, performing an active scan for common vulnerabilities, and conducting manual testing. The objective is to document all vulnerabilities and provide steps for reproduction and remediation.

Steps Performed:

1. Setup OWASP ZAP:

- **Download and Install OWASP ZAP:**
  - Downloaded OWASP ZAP from the official website (https://www.zaproxy.org/).
  - Installed OWASP ZAP on Kali Linux using the command:

```
bash
Copy code
sudo apt install zaproxy
```

- **Launch OWASP ZAP:**
  - Launched OWASP ZAP by typing `zaproxy` in the terminal.

2. Browser Configuration:

- **Configure Browser:**
  - Configured the browser (Firefox/Chrome) to intercept web traffic via the ZAP proxy.
  - Set the HTTP Proxy to `localhost` and Port to `8080` in the browser settings.

3. Use Spider Tool:

- **Start Spider Scan:**
  - Entered the target URL (`https://www.google.com`) in OWASP ZAP and clicked on the "Quick Start" tab.
  - Clicked on the "Attack" tab and selected the "Spider" option.
  - The Spider tool crawled the target web application to discover accessible pages and resources.

4. Perform Automated Scan:

- **Start Automated Scan:**

    - After the Spider scan completed, went to the "Attack" tab and selected "Automated Scan".
    - Confirmed the target URL (`https://www.google.com`) and clicked "Start Scan".
    - OWASP ZAP performed an automated scan to identify common vulnerabilities such as SQL injection, XSS, and CSRF.
- **Monitor Scan Progress:**
    - Monitored the scan progress in OWASP ZAP and checked the "Alerts" tab for identified vulnerabilities.

5. Analyze Scan Results:

- **Review Detected Vulnerabilities:**
    - After the automated scan completed, reviewed each detected vulnerability in detail in the "Alerts" tab.

6. Manual Testing:

- **SQL Injection Testing:**
    - Manually tested for SQL injection by inserting SQL code into input fields. Used tools like SQLMap when necessary:

      ```
      bash
      Copy code
      sqlmap -u "http://target-url.com/vulnerablepage.php?id=1" --batch
      --passwords
      ```

- **XSS Testing:**
    - Injected scripts into input fields and analyzed the application's response to test for XSS vulnerabilities.
    - Example payload: `<script>alert('XSS')</script>`
- **CSRF Testing:**
    - Crafted malicious requests and observed the application's response to test for CSRF vulnerabilities.

7. Advanced Techniques:

- **Fuzzing:**
    - Used OWASP ZAP's fuzzing capabilities to input various payloads into input fields to identify potential vulnerabilities.
    - Configured fuzzing settings in OWASP ZAP and initiated the fuzzing process.
- **Session Management:**
    - Analyzed how the application manages sessions and cookies to identify weaknesses in session management.

Findings and Documentation:

- **Document Vulnerabilities:**

- Documented all vulnerabilities identified during automated and manual testing.
- Provided detailed steps to reproduce each vulnerability and suggested remediation measures.
- Included screenshots of findings and the OWASP ZAP interface.

Tricky Task - Undetected Vulnerabilities:

- **Demonstrated Manual Testing Skills:**
  - Discovered and exploited a vulnerability that automated scans did not detect. This could include logical flaws, complex XSS payloads, or advanced SQL injection techniques.
  - Provided detailed documentation on how this vulnerability was discovered and exploited.