

# Experiment No.: 9 Title: Security Testing

**Batch:B1 Roll No.:16010421119 Experiment No.:8 Aim:** To perform Security testing using any open source tool.

**Resources needed:** Internet, Chrome Browser

## Theory:

**Security Testing** is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders. The purpose of Security Tests is to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue, repute at the hands of the employees or outsiders of the Organization.

## Why Security Testing is Important?

The main goal of Security Testing is to identify the threats in the system and measure its potential vulnerabilities, so the threats can be encountered and the system does not stop functioning or cannot be exploited. It also helps in detecting all possible security risks in the system and helps developers to fix the problems through coding.

## Types of Security Testing in Software Testing

There are seven main types of security testing as per Open Source Security Testing methodology manual. They are explained as follows:

* **Vulnerability Scanning:** This is done through automated software to scan a system against known vulnerability signatures.
* **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.
* **Penetration testing:** This kind of testing simulates an attack from a malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
* **Risk Assessment:** This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.
* **Security Auditing:** This is an internal inspection of Applications and Operating systems for security flaws. An audit can also be done via line by line inspection of code
* **Ethical hacking:** It’s hacking an Organization Software systems. Unlike malicious hackers, who steal for their own gains, the intent is to expose security flaws in the system.
* **Posture Assessment:** This combines Security scanning, [Ethical Hacking](https://www.guru99.com/ethical-hacking-tutorials.html) and Risk Assessments to show an overall security posture of an organization.

## Security Testing throughout SDLC

It is always agreed, that cost will be more if we postpone [security testing](https://www.guru99.com/security-testing-tools.html) after software implementation phase or after deployment. So, it is necessary to involve security testing in the SDLC life cycle in the earlier phases.

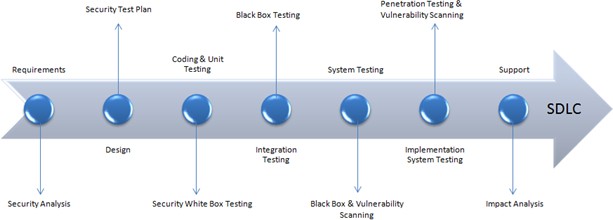
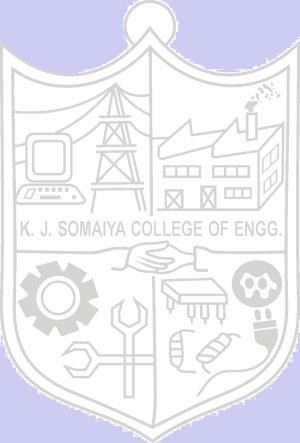


Fig 1: Security processes to be adopted for every phase in SDLC

Following are the open source tools which are useful in security testing:

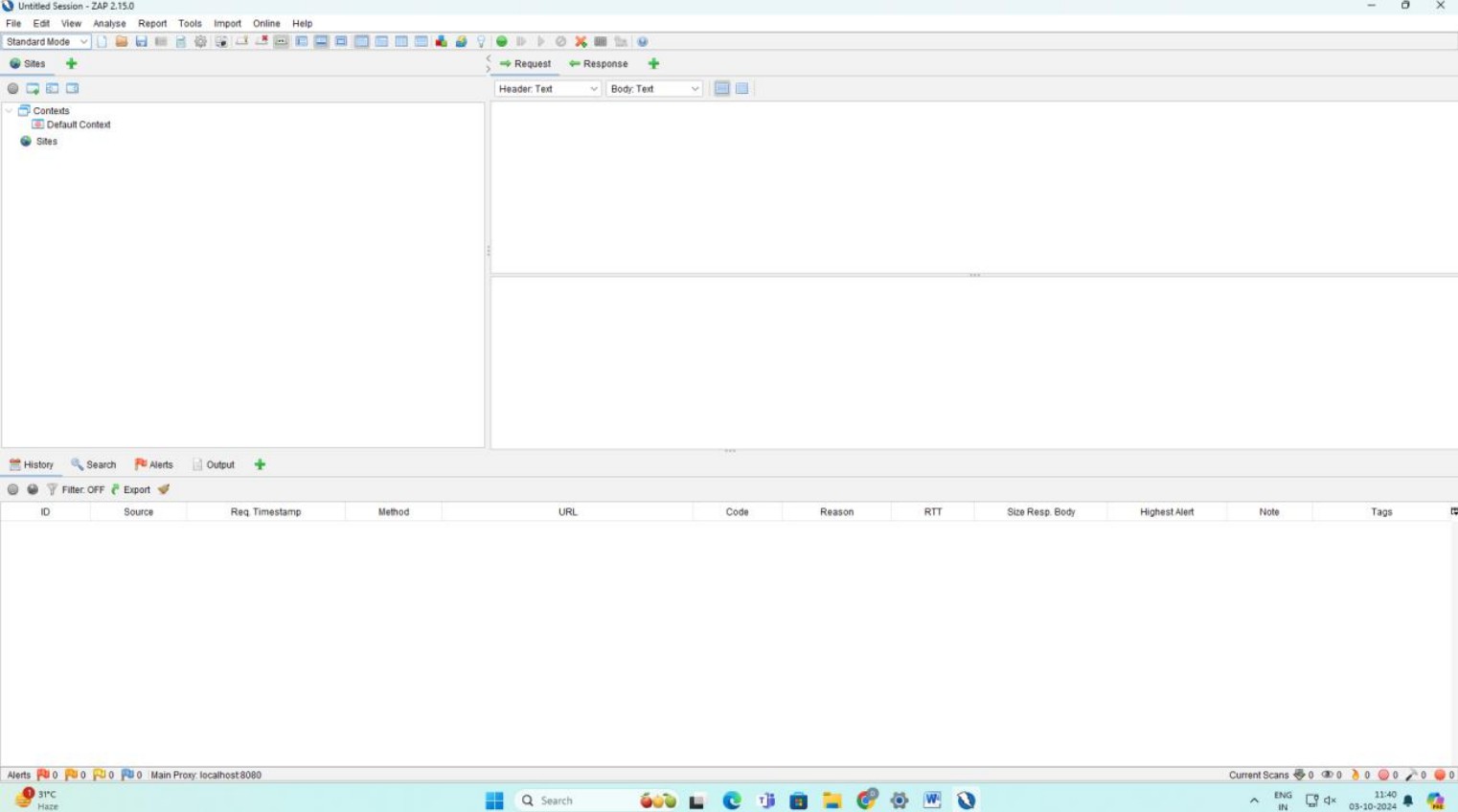
* Zed Attack Proxy (ZAP)
* Wfuzz
* Wapiti
* W3af
* SQLmap
* SonarQuobe
* Nogotofail
* IRONWASP
* Grabber
* Arachni



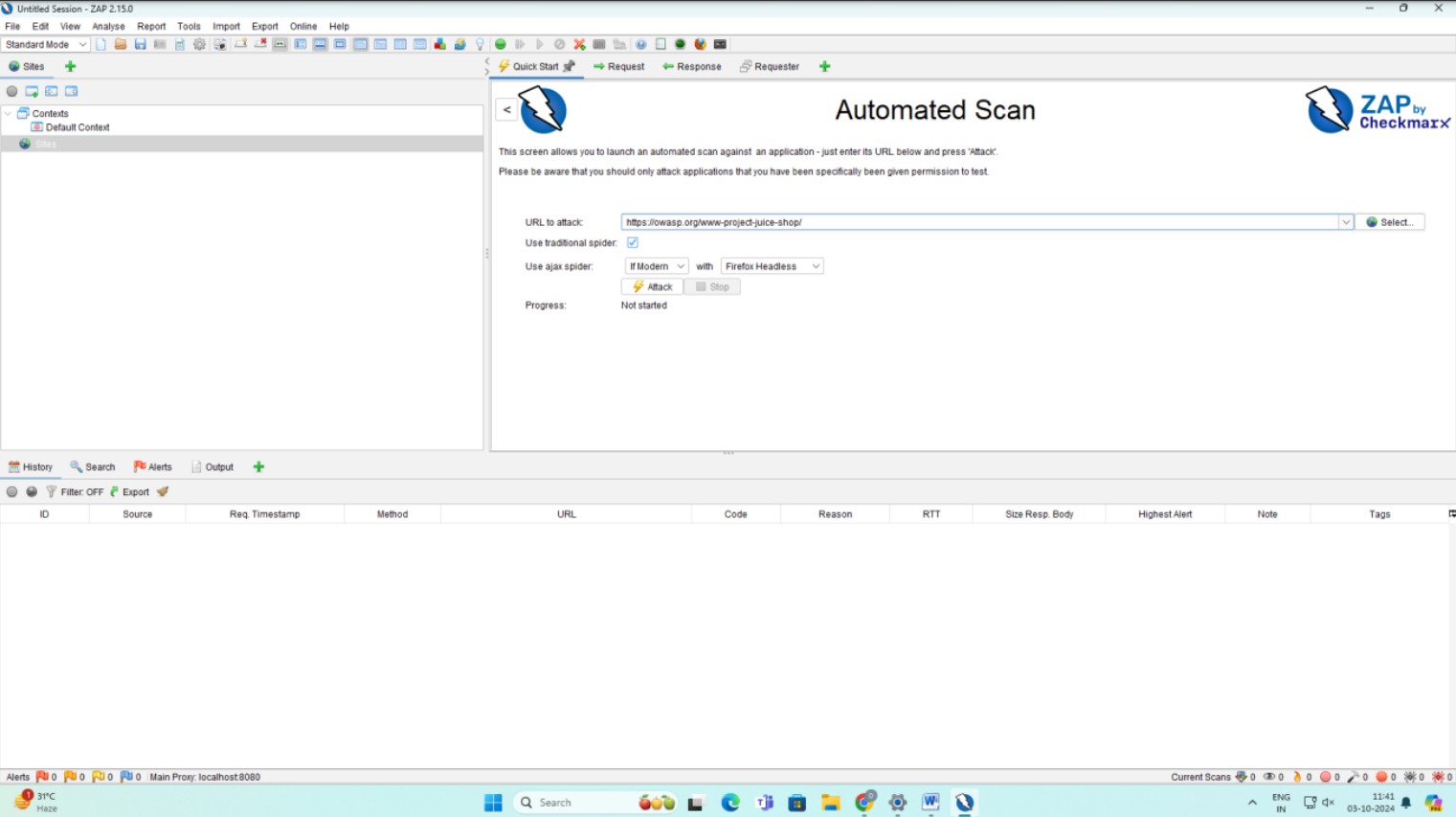
**Procedure:**

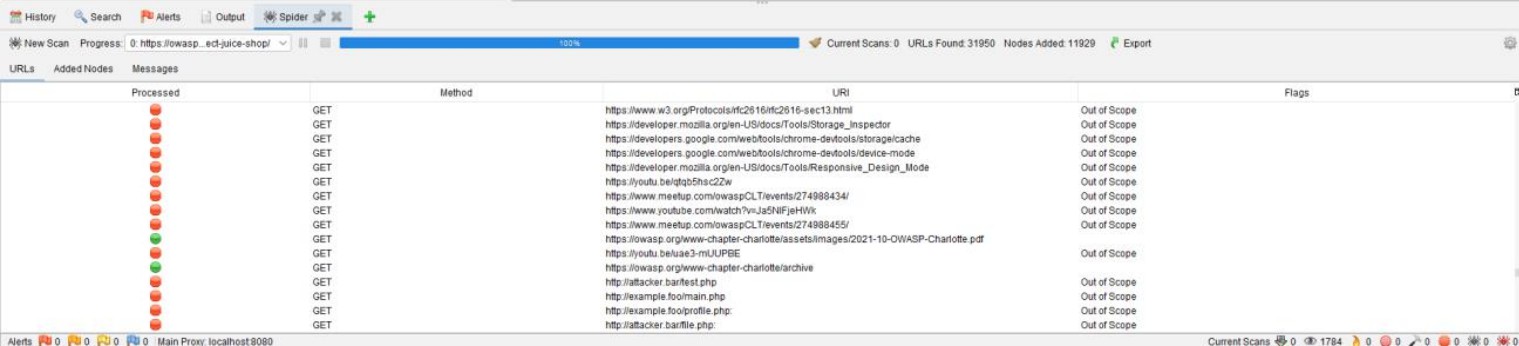
1. Explore any one tool for Security testing.
2. Perform Security testing of any website and generate the report for the same.
3. Analyze and summarize report for quality and performance of web pages.
4. Suggest improvements based on your analysis.

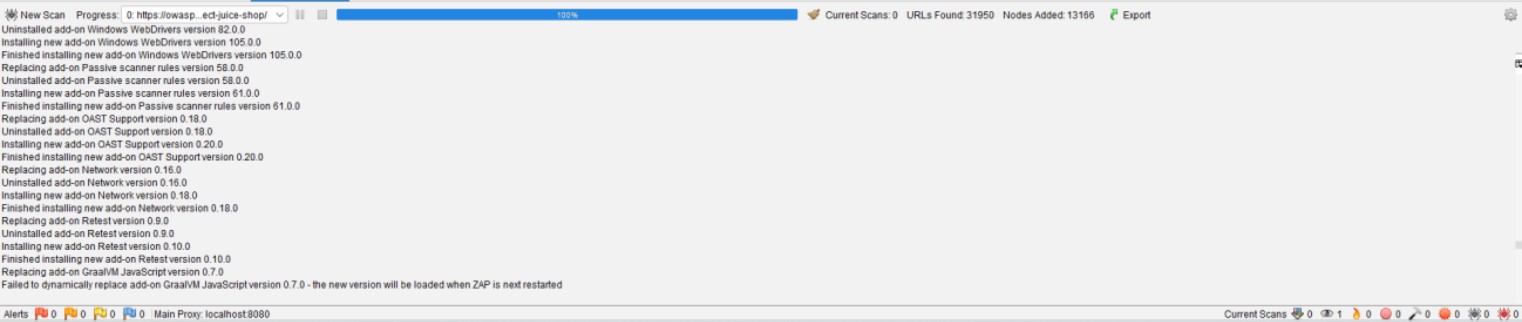
**Using ZAP tool**

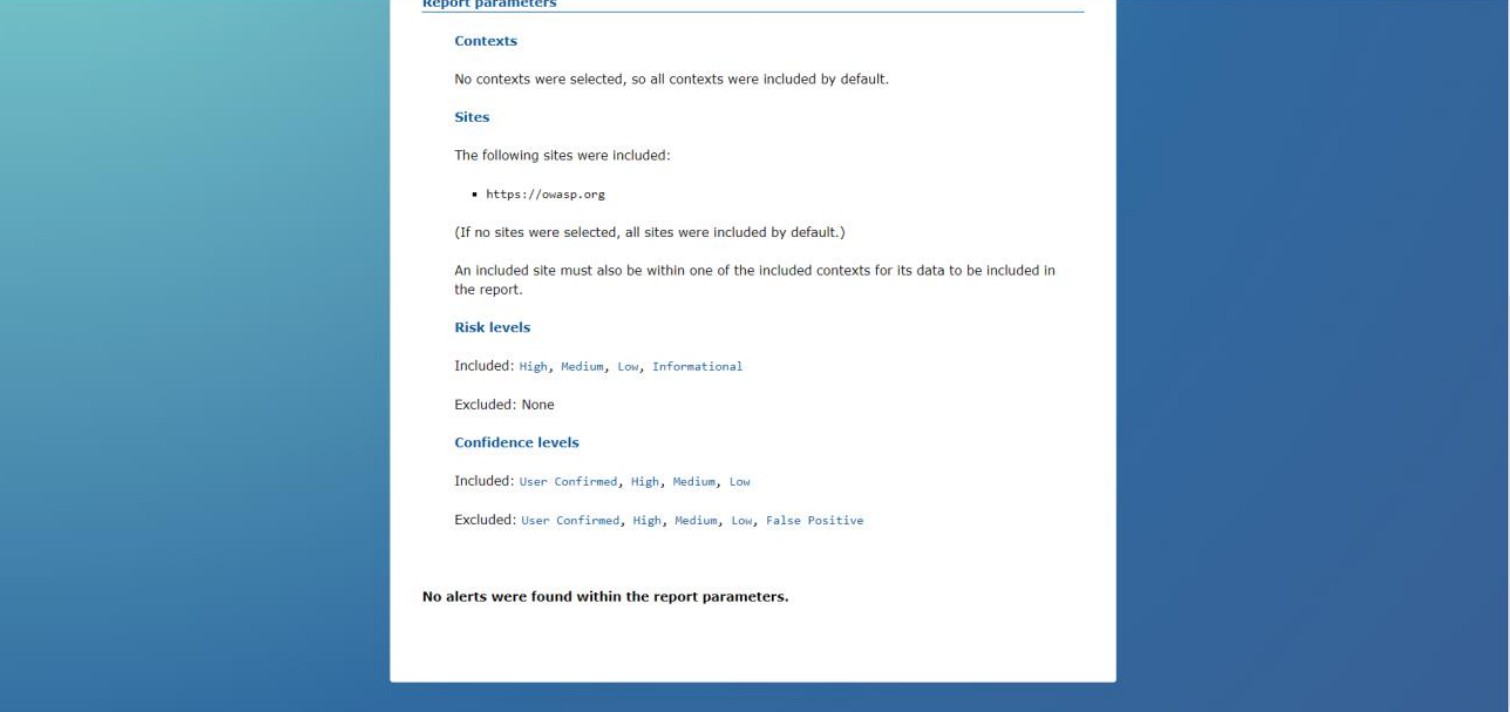


1. **Testing**







**Report:**

# Analysis

## Quality of Web Pages:

* + - **No Alerts**: This indicates that the website did not show any known vulnerabilities during the scan. This is a positive outcome, implying that the website has passed the security test for common vulnerabilities like Cross-Site Scripting (XSS), SQL Injection, and other attack vectors.

KJSCE/IT/B.Tech/SEMVII/STQA/2024-2025

* + - **Included Risk Levels**: The scan included all risk levels (High, Medium, Low, and Informational), meaning the test was thorough and covered a wide range of potential security issues.

## Performance of Security Testing:

* + - Since no alerts were found, the performance of the website from a security perspective seems satisfactory. However, this does not necessarily mean that the site is completely secure, as ZAP mainly focuses on known vulnerabilities and attack patterns.
    - If there are advanced, targeted attacks or undiscovered vulnerabilities, they might not have been identified by this scan.

# Improvements based on analysis

## Revalidate with Penetration Testing:

* Since no alerts were found, it is worth conducting **manual penetration**

**testing** to simulate more targeted attacks. Automated tools can sometimes miss subtle issues that only arise under certain conditions.

## Regular Audits and Updates:

* Continue regular **security audits** and ensure that the website software is

updated to the latest security patches. Even if no issues are found now, new vulnerabilities are constantly being discovered.

## Check for Business Logic Flaws:

* While ZAP focuses on technical vulnerabilities, it may not catch **business logic flaws**—errors that occur in the workflow of an application. It is

important to complement this with reviews focused on the application's logic

and user workflows. d. **Utilize Additional Tools**:

* + Consider using other tools alongside ZAP, such as **Burp Suite** or **Arachni**, for more comprehensive testing, as different tools might catch different

vulnerabilities.

## Questions:

1. Describe security testing. Explain any five test cases/scenarios for testing security of a web application.

Ans: Security testing is the process of assessing and validating the effectiveness of software applications in protecting against potential vulnerabilities and malicious threats. The primary objective of security testing is to identify potential security risks within a system and to ensure that data is safeguarded from unauthorized access, loss, or corruption. The testing focuses on verifying that the software application adheres to security standards, preventing unauthorized users from accessing confidential or sensitive data. This testing also ensures that the application maintains its availability, confidentiality, and integrity.

There are different types of security testing, including:

* + **Vulnerability Scanning**: Automated testing to detect known vulnerabilities.
  + **Penetration Testing**: Ethical hackers simulate attacks to exploit vulnerabilities.
  + **Risk Assessment**: Evaluates security risks based on asset value, threat level, and vulnerability.
  + **Security Auditing**: Review of code, processes, or system configurations for security weaknesses.

## Five Test Cases/Scenarios for Security Testing in a Web Application:

1. **SQL Injection Testing**:
   * **Scenario**: The attacker tries to insert malicious SQL code via input fields, which, if executed, manipulates the database, potentially accessing or altering sensitive data.
   * **Objective**: Verify that input fields properly sanitize user inputs to prevent unauthorized SQL queries.
   * **Test Case**: Test the application's forms, search bars, or login screens with inputs like ' OR '1'='1 or '; DROP TABLE users;-- to see if they are executed or rejected.

## Cross-Site Scripting (XSS) Testing:

* + **Scenario**: The attacker injects malicious scripts into trusted websites, which are then executed in the victim’s browser, often stealing cookies or session information.
  + **Objective**: Ensure that the application escapes or sanitizes user inputs to prevent the execution of malicious scripts.
  + **Test Case**: Insert code such as <script>alert('XSS')</script> into user input fields (e.g., comment boxes) to verify if the input is executed in the user's browser.

## Cross-Site Request Forgery (CSRF) Testing:

* + **Scenario**: The attacker tricks an authenticated user into unknowingly submitting a request to the web application (e.g., making a money transfer) without their consent.
  + **Objective**: Ensure that the application has CSRF protection, such as tokens or re-authentication for sensitive actions.
  + **Test Case**: Generate a malicious link that triggers an action on behalf of the user (e.g., transferring funds) and check if the application requires verification (like a unique token) to prevent the action.

## Session Management Testing:

* + **Scenario**: Session tokens or cookies are intercepted or manipulated by attackers to impersonate a legitimate user or

hijack an active session.

* **Objective**: Verify that sessions are handled securely by ensuring proper expiration of tokens, secure cookies (marked with HttpOnly and Secure flags), and appropriate session timeouts.
* **Test Case**: Capture a session cookie using tools like Burp Suite and replay it to attempt accessing another user's session. Ensure the system invalidates the session when cookies are modified or expired.

## Brute Force Attack Testing:

* + **Scenario**: The attacker attempts to guess a user’s password by trying multiple combinations in rapid succession.
    - **Objective**: Ensure that the application has rate-limiting or account lockout mechanisms to prevent infinite attempts at

guessing login credentials.

KJSCE/IT/B.Tech/SEMVII/STQA/2024-2025

* + **Test Case**: Use a tool like Hydra to simulate multiple failed login attempts and verify if the application temporarily blocks the user or

requires a

CAPTCHA after a certain number of failed attempts.

1. What are the benefits of Security testing for website owners?

Ans:

* + **Protecting Sensitive Information**: Security testing is crucial for protecting users' sensitive data, such as login credentials, payment information, or personal identifiers (e.g., Social Security numbers). Ensuring data encryption, secure storage, and

restricted access prevents attackers from obtaining critical information, which can lead to identity theft, financial fraud, or regulatory fines.

* + **Enhancing User Trust and Business Reputation**: A secure website fosters confidence among users and partners, enhancing the website owner's reputation. Frequent security incidents, on the other hand, can damage a company's image and erode customer trust. Conducting security tests demonstrates a commitment to data protection, fostering user loyalty and a strong brand reputation.
  + **Regulatory Compliance**: Many industries, especially those dealing with financial data, healthcare information, or personal data, must comply with regulations like the General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), or Payment Card Industry Data Security Standard (PCI DSS). Security testing helps ensure compliance with these legal requirements, protecting the business from potential fines and legal consequences.
  + **Preventing Financial Loss**: Security breaches can result in severe financial repercussions, from the direct costs of incident recovery and compensation to the indirect loss of customers and potential legal liabilities. Proactive security testing reduces the chances of a breach, saving businesses money in the long run by identifying and fixing vulnerabilities before they can be exploited.
  + **Improving Software Quality and Resilience**: Regular security testing improves not only the security of a web application but also its overall quality. By addressing security flaws early in the software development lifecycle (SDLC), developers can ensure that the application is resilient and stable, minimizing bugs or weak points that attackers could exploit. This leads to a more robust product that provides a safer experience for end-users.
  + **Protecting Against Business Disruption**: Cyberattacks can lead to system downtime, service outages, or even total loss of control over a website. For businesses relying on web applications for operations, security testing ensures that these platforms are resistant to attacks that could disrupt normal business activities, thus safeguarding continuity.

## Outcomes:CO3: Apply recent automation tools for testing software.

**Conclusion: (Conclusion to be based on outcomes)**Through this experiment we performed security testing using ZAP tool. Security testing is vital for ensuring the safety and integrity of web applications. It helps identify vulnerabilities like SQL injection, XSS, and session hijacking, protecting sensitive data, enhancing user trust, ensuring regulatory compliance.

**Grade: AA / AB / BB / BC / CC / CD /DD Signature of faculty in-charge with date**

**References:**

**Books/ Journals/ Websites:**

KJSCE/IT/B.Tech/SEMVII/STQA/2024-2025

1. [ZAP (zaproxy.org)](https://www.zaproxy.org/)
2. [Edge-security group - Wfuzz](http://www.edge-security.com/wfuzz.php)
3. [Wapiti : a Free and Open-Source web-application vulnerability scanner in Python](https://wapiti-scanner.github.io/) [(wapiti-scanner.github.io)](https://wapiti-scanner.github.io/)
4. [w3af - Open Source Web Application Security Scanner](https://w3af.org/)
5. [sqlmap: automatic SQL injection and database takeover tool](https://sqlmap.org/)
6. [Code Quality, Security & Static Analysis Tool with SonarQube | Sonar (sonarsource.com)](https://www.sonarsource.com/products/sonarqube/)
7. [Google Online Security Blog: Introducing nogotofail—a network traffic security testing tool](https://security.googleblog.com/2014/11/introducing-nogotofaila-network-traffic.html) [(googleblog.com)](https://security.googleblog.com/2014/11/introducing-nogotofaila-network-traffic.html)
8. [GitHub - amoldp/Grabber-Security-and-Vulnerability-Analysis-](https://github.com/amoldp/Grabber-Security-and-Vulnerability-Analysis-)
9. [Arachni - Web Application Security Scanner Framework – Ecsypno](https://ecsypno.com/pages/arachni-web-application-security-scanner-framework)