|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Information Security (116U01L602)** | **Semester:** | **VI** |
| **Date of Performance:** | **02 / 04 / 2025** | **DIV/ Batch No:** | **A-3** |
| **Student Name:** | **Kashish Mamania** | **Roll No:** | **16010122104** |

|  |
| --- |
| **Title: Introduction to Open Web Application Security Project and implementation of Cross-site scripting (XSS) DVWA/ Burp Suite** |

|  |
| --- |
| **Objectives:** |
| To study Open Web Application Security Project and implement XSS. |

|  |
| --- |
| **Expected Outcome of Experiment:** |
| **CO1** |

|  |
| --- |
| **Books/ Journals/ Websites referred:** |
|  |

|  |
| --- |
| **Pre Lab/ Prior Concepts:** |
|  |

|  |
| --- |
| **New Concepts to be learned:** |
|  |

|  |
| --- |
| **Abstract:** |
| Damn Vulnerable Web Application (DVWA) is a deliberately insecure web application designed for security professionals and students to practice web vulnerability exploitation. It contains multiple security flaws, allowing users to test various attack techniques in a controlled environment. |

|  |
| --- |
| **Related Theory:** |
| Key Features of DVWA:  1. Security Levels: DVWA offers different levels of security—Low, Medium, High, and Impossible—to help users  learn how vulnerabilities change with security improvements.  2. Web-Based Vulnerabilities: Includes XSS (Cross-Site Scripting), SQL Injection, CSRF (Cross-Site Request Forgery),  File Inclusion, Command Injection, and more.  3. Learning Platform: Used by security researchers and ethical hackers to improve their penetration testing skills.  4. PHP and MySQL-Based: Runs on a LAMP/WAMP/XAMPP server and requires database setup for full functionality.  Why Use DVWA?  1. It provides a safe and legal environment to learn hacking techniques.  2. Helps understand how attackers exploit vulnerabilities and how to secure applications.  3. Used for penetration testing training and cybersecurity education.  What is XSS?  Cross-Site Scripting (XSS) is a web security vulnerability that allows an attacker to inject malicious scripts  into web pages viewed by users. These scripts run in the victim&#39;s browser, leading to data theft, session  hijacking, or malware injection.  Types of XSS Attacks:  1. Stored XSS – The script is permanently stored in the website’s database and executed when the page is  loaded.  2. Reflected XSS – The script is embedded in a URL or request and executed when clicked by the victim.  3. DOM-Based XSS – The malicious script manipulates the DOM (Document Object Model) instead of  the server response.  Impact of XSS Attacks:  1. Stealing cookies and session tokens → Session hijacking  2. Keylogging → Capturing user keystrokes  3. Defacing websites → Modifying site content  4. Phishing attacks → Redirecting users to fake login pages |

|  |
| --- |
| **Implementation Details:** |
| 1. XSS Attack 1: Hijacking the user’s session          1. XSS Attack 2: Perform unauthorized activities.          1. XSS Attack 3: Phishing to steal user credentials.        1. XSS Attack 4: Capture the keystrokes by injecting a keylogger.            1. XSS Attack 5: Stealing sensitive information. |

|  |
| --- |
| **Post Lab Questions:** |
| 5.1 What is OWASP? List the Latest Web Security Application Risks by OWASP  **What is OWASP?** The Open Web Application Security Project (OWASP) is a nonprofit foundation focused on improving software security. It provides tools, frameworks, and guidelines to help organizations secure web applications. OWASP is best known for its "OWASP Top 10," a widely recognized list of the most critical web application security risks. This initiative raises awareness about vulnerabilities and promotes best practices among developers and security professionals  **Latest OWASP Web Security Application Risks (2025 Predictions):**   1. Broken Access Control 2. Injection 3. Insecure Design 4. Identification and Authentication Failures 5. Cryptographic Failures 6. Security Misconfiguration 7. Vulnerable and Outdated Components 8. Software and Data Integrity Failures   5.2 Explain Countermeasures for Injection Attacks  Injection attacks, such as SQL injection, occur when an attacker sends malicious input to manipulate the execution of commands or queries in an application.  **Countermeasures:**   * **Input Validation and Sanitization:** Ensure all user inputs are validated against expected formats and sanitized to remove harmful characters * **Parameterized Statements (Prepared Statements):** Use parameterized queries to separate SQL code from user input, preventing malicious code execution * **Escaping User Input:** Escape special characters in user inputs to neutralize their potential harm * **Stored Procedures:** Use stored procedures to execute SQL queries securely without exposing raw SQL commands * **Least Privilege Principle:** Limit database permissions to the minimum required for the application to function, reducing potential damage * **Web Application Firewalls (WAFs):** Deploy WAFs to monitor and block malicious traffic patterns, including injection attempts   5.3 List the Types of XSS Attacks  Cross-Site Scripting (XSS) attacks exploit vulnerabilities in web applications to inject malicious scripts into web pages viewed by users.  **Types of XSS Attacks:**   1. **Stored XSS (Persistent XSS):** Malicious scripts are stored on the server (e.g., in a database) and executed when users access the affected page 2. **Reflected XSS (Non-Persistent XSS):** Malicious scripts are reflected off a web server in responses, often delivered through malicious links or phishing emails 3. **DOM-Based XSS:** The attack occurs entirely on the client side by manipulating the Document Object Model (DOM), with no involvement of the server |

|  |
| --- |
| **Conclusion:** |
| DVWA is a powerful tool for learning web application security and penetration testing. It helps students and security researchers understand common vulnerabilities like XSS, SQL Injection, and CSRF. |

|  |
| --- |
| **Post-Lab Questions:** |
| **4.1 Major Types of Web Application Attacks**   1. **SQL Injection (SQLi)**    * Exploits vulnerabilities in database queries    * Allows attackers to manipulate or retrieve unauthorized data 2. **Cross-Site Scripting (XSS)**    * Injects malicious scripts into web pages viewed by other users    * Types: Stored XSS, Reflected XSS, DOM-based XSS 3. **Cross-Site Request Forgery (CSRF)**    * Tricks users into performing unintended actions on authenticated web applications 4. **Broken Authentication and Session Management**    * Exploits weaknesses in login systems or session handling    * Can lead to account takeover or identity theft 5. **Insecure Direct Object References**    * Allows attackers to access unauthorized resources by manipulating references 6. **Security Misconfiguration**    * Exploits improperly configured web servers, databases, or application frameworks 7. **XML External Entity (XXE) Attacks**    * Targets poorly configured XML parsers to access internal files or execute remote code 8. **Unvalidated Redirects and Forwards**    * Abuses trust in a domain to redirect users to malicious sites 9. **Remote Code Execution (RCE)**    * Allows attackers to execute arbitrary code on the target server 10. **File Inclusion Vulnerabilities**     * Exploits improper handling of file inclusions to execute malicious code   **4.2 Mitigating SQL Injection Attacks**   1. **Use Parameterized Queries**    * Separate SQL logic from user input    * Example (PHP with PDO):   php  $stmt = $pdo->prepare('SELECT \* FROM users WHERE username = :username');  $stmt->execute(['username' => $user\_input]);   1. **Input Validation and Sanitization**    * Validate input type, length, format, and range    * Use whitelisting for allowed characters 2. **Least Privilege Principle**    * Use database accounts with minimal required permissions 3. **Stored Procedures**    * Use carefully implemented stored procedures to abstract SQL logic 4. **Web Application Firewalls (WAF)**    * Implement WAFs to filter malicious SQL patterns 5. **Escape Special Characters**    * Use language-specific escaping functions for user inputs 6. **Error Handling**    * Avoid exposing detailed error messages to users 7. **Regular Security Audits**    * Conduct code reviews and penetration testing 8. **Keep Systems Updated**    * Apply security patches promptly to all components 9. **Use ORM Frameworks**    * Utilize Object-Relational Mapping frameworks that inherently protect against SQLi   **4.3 Man-in-the-Middle (MITM) Attack**  **Definition**: A MITM attack occurs when an attacker secretly intercepts and possibly alters the communication between two parties who believe they are directly communicating with each other.  **Key Aspects**:   1. **Interception**: Attacker positions themselves between the victim and the legitimate service. 2. **Eavesdropping**: Captures sensitive information like login credentials or financial data. 3. **Modification**: Can alter the intercepted data before passing it on.   **Common MITM Techniques**:   * **ARP Spoofing**: Manipulates Address Resolution Protocol to redirect traffic. * **DNS Spoofing**: Alters DNS responses to direct users to malicious sites. * **SSL Stripping**: Downgrades HTTPS connections to unencrypted HTTP. * **Evil Twin**: Sets up a rogue Wi-Fi access point mimicking a legitimate one.   **Prevention Measures**:   1. Use HTTPS with proper certificate validation. 2. Implement strong encryption protocols (e.g., TLS 1.3). 3. Use Virtual Private Networks (VPNs) on public networks. 4. Enable HSTS (HTTP Strict Transport Security) on web servers. 5. Educate users about the risks of unsecured Wi-Fi networks.   **Example Scenario**: Alice wants to log in to her bank account. Eve, the attacker, intercepts the connection:  text  Alice <---> Eve <---> Bank  Eve can now see Alice's login credentials and potentially modify transactions. |