|  |  |  |  |
| --- | --- | --- | --- |
| **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: SQL Injection using DVWA** |

|  |
| --- |
| **Objectives:** |
|  |

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

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

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

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

|  |
| --- |
| **Abstract:** |
|  |

|  |
| --- |
| **Related Theory:** |
|  |

|  |
| --- |
| **Implementation Details:** |
|  |

|  |
| --- |
| **Results/Output:** |
|  |

|  |
| --- |
| **Conclusion:** |
| The SQL injection analysis using DVWA demonstrated how unfiltered inputs allow attackers to manipulate database queries across various security levels. Our findings confirm that proper input validation, parameterized queries, and defense-in-depth strategies are essential for preventing SQL injection vulnerabilities in web applications. |

|  |
| --- |
| **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. |