# CS 305 Module Two Written Assignment Template

## Instructions

Replace the bracketed text with the relevant information in your own words. If you choose to include images or supporting materials, make certain to insert them in all the relevant locations in the document.

## Areas of Security

1. Authorization:
   * Review the application's access control mechanisms to ensure users can only access resources and perform actions they are authorized to.
   * Check for any misconfigurations that might allow unauthorized access to sensitive data or functionality.
2. Data Security:
   * Assess data encryption mechanisms, especially for sensitive data stored in the application's database or transmitted over the network.
   * Verify that sensitive data is adequately protected from unauthorized access or disclosure.
3. Input Validation and Sanitization:
   * Ensure that user inputs are properly validated and sanitized to prevent common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and remote code execution.
   * Check for any security flaws related to command input and SpEL usage, as they can potentially lead to code injection vulnerabilities.
4. Session Management:
   * Review how user sessions are managed and secured.
   * Check for issues like session fixation, session hijacking, or insufficient session timeouts.
5. Error Handling and Logging:
   * Verify that error messages do not reveal sensitive information and are handled securely.
   * Ensure that logs are appropriately generated and reviewed for potential security incidents.
6. Configuration Management:
   * Check for secure configuration practices, especially for secrets and sensitive information.
   * Assess the application's resistance to common configuration-related security risks.

## Areas of Security Justification

1. Authentication:
   * Justification: Authentication is crucial to ensure that only authorized users can access the application and its functionalities. In a complex web application, sensitive data and critical operations are often involved. Without proper authentication, unauthorized users could gain access, leading to data breaches or misuse of the application's features.
2. Authorization:
   * Justification: Authorization complements authentication by specifying what actions or resources authenticated users are allowed to access. In a complex application, different user roles and permissions are typically defined. Proper authorization ensures that users can't overstep their privileges, which is vital to maintain data confidentiality and system integrity.
3. Data Security:
   * Justification: Data security is critical because web applications frequently deal with sensitive user information, such as personal details, financial data, or proprietary business information. Ensuring data encryption and access control mechanisms are in place safeguards this data from unauthorized access or breaches.
4. Input Validation and Sanitization:
   * Justification: Input validation and sanitization are essential to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS). In the context of your scenario, where a command input function is being implemented, improper handling of user inputs can lead to code injection vulnerabilities, potentially compromising the entire application.
5. Session Management:
   * Justification: Session management is important to protect user sessions from being compromised. In a complex web application, session hijacking or fixation could lead to unauthorized access and actions on behalf of the user. Proper session management helps maintain user privacy and application security.
6. Error Handling and Logging:
   * Justification: Error handling and logging are important for security incident detection and response. Secure error handling ensures that error messages don't reveal sensitive information, which could be exploited by attackers. Effective logging helps in monitoring and identifying potential security incidents or suspicious activities.
7. Configuration Management:
   * Justification: Secure configuration management is essential to protect sensitive configuration data and secrets. In a complex application, configurations often include database connection strings, API keys, and other critical information. Any misconfiguration could expose these secrets to attackers, leading to security breaches or unauthorized access.

## Code Review Summary

[Insert text.]

## Mitigation Plan

[Insert text.]