# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/18/24** | **Daniel Vidmar** | **Initial Modification** |
| **1.1** | **9/21/24** | **Daniel Vidmar** | **Minor Grammar Fixes** |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Daniel Vidmar

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

**Client Needs and Potential Threats:**

Based on Artemis Financial's needs, they are focused on modernizing their operations and providing a RESTful API that handles sensitive information. The company focuses on preventing external threats, safeguarding data, and maintaining compliance with industry standards and regulations. Below is an interpretation of the critical client needs and potential threats:

1. **The Value of Secure Communications**: Secure communications are critical to Artemis Financial due to the nature of the data they handle, which includes financial plans, personal information, and investment data. Failure to properly secure communications could result in financial losses, lost customer trust, and potential regulatory and legal penalties. Securing data in transit and ensuring encrypted communication is paramount for the company's success.
2. **International Transactions**: If Artemis Financial is involved in international transactions, it is essential to consider the security of internation transactions. These transactions are subject to even further regulatory situations, such as the GDPR for Europe and other local data protection laws. This increased risk will make secure communications even more vital to operations.
3. **External Threats**: Artemis Financial's primary external threats include cyber-attacks such as phishing, Distributed Denial of Service (DDoS) attacks, malware, ransomware, and data breaches. Financial institutions are attractive targets for hackers exploiting sensitive customer financial data. For these reasons, developers must constantly stay updated on evolving attack methods and security vulnerabilities.

**Requirements:**

1. **Open-Source Libraries**: Artemis Financial's modernization efforts may involve integrating open-source libraries to enhance the functionality of its web application. However, while open-source libraries provide flexibility and scalability, they also present a security risk if not properly vetted and updated. Vulnerabilities in outdated libraries can serve as entry points for attackers. This fact is the reason why Artemis must adopt a dependency management protocol to ensure that all libraries remain up-to-date.
2. **Evolving Web Technologies**: Part of Artemis Financial's modernization is the fact that the underlying technology stack may need an overhaul. This could include container technology like Docker and switching to a microservices architecture. Each of these systems will introduce new security risks and mitigation plans that must be thought through before full implementation.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

Based on the identified needs and threats, the following critical areas of security are relevant to Artemis Financial's RESTful web application, each justified based on the functionality of the application:

1. **Authentication and Authorization**: The web application requires robust authentication mechanisms, such as multi-factor authentication (MFA), to ensure that only legitimate users can access sensitive customer data. Role-based access control (RBAC) must be enforced to restrict access based on user roles, ensuring that only authorized personnel can access critical financial data.
2. **Data Encryption**: Encrypting data in transit using protocols like TLS 1.2 or higher is crucial for protecting customer data from interception during communications. Similarly, data encryption at rest is essential to protect against unauthorized access in case of a server breach. Given the financial nature of the data, end-to-end encryption must be employed.
3. **Vulnerability and Patch Management**: The company needs a consistent vulnerability and patch management process to address software security flaws. As part of its modernization, Artemis Financial must implement automated tools to scan for vulnerabilities in real-time, especially those stemming from open-source libraries. Regular patching is crucial to defending against external threats.
4. **Secure APIs**: Since Artemis Financial operates a RESTful API, securing API endpoints is critical. Common threats like API injection attacks (SQL/NoSQL injection) and cross-site scripting (XSS) must be mitigated through input validation, secure coding practices, and proper endpoint authentication mechanisms like OAuth 2.0.
5. **Security Monitoring and Incident Response**: A comprehensive logging and monitoring system must be in place to detect and respond to security incidents in real time. Artemis Financial should implement a Security Information and Event Management (SIEM) system to monitor API activity, detect anomalies, and alert security teams in case of any suspicious activity. This helps prevent or quickly mitigate data breaches or DDoS attacks.
6. **Backup and Recovery**: As part of the security framework, Artemis Financial must ensure regular backups of critical financial data to protect against data loss from ransomware attacks or system failures. A disaster recovery plan that enables quick service restoration without data corruption is essential.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

The review of the source code for Project One reveals several vulnerabilities. A few files have no input validation, which could lead to SQL Injection attacks in the future. There is also a lack of proper data validation in the customer and greeting classes, which could lead to a malicious actor injecting invalid data. Beyond these vulnerabilities, inconsistent error handling could expose internal errors to end-users and leave the system vulnerable. A more severe issue is that DocData stores sensitive information without any form of encryption, which could lead to massive data breaches in the future.

Additionally, GreetingController.java lacks authentication and access control checks, potentially allowing unauthorized access to services. Furthermore, while logging is present in RestServiceApplication.java, it is not used for critical security events, such as failed logins. Finally, the application.properties file contains hardcoded sensitive information, such as database credentials, which could be exposed if not adequately secured. Addressing the issues with the code above is vital to the application's security.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

**A screenshot of a computer

Description automatically generated**

1. bcprov-jdk15on-1.46.jar

* Vulnerability Code: cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-cryptography-api:1.46:*:*:*:*:*:*:\*
* Description: Bouncy Castle cryptography library contains high-severity vulnerabilities. Attackers could exploit cryptographic weaknesses, leading to data exposure or breaches.
* Recommended Solution: Upgrade to a newer, secure version of the Bouncy Castle library.

2. spring-boot-2.2.4.RELEASE.jar

* Vulnerability Code: cpe:2.3:a:vmware:spring\_boot:2.2.4:release:*:*:*:*:*:*:\*
* Description: Spring Boot contains critical vulnerabilities that could allow attackers to bypass authentication or execute arbitrary code.
* Recommended Solution: Update to a more recent, patched version of Spring Boot.

3. logback-core-1.2.3.jar

* Vulnerability Code: cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:\*
* Description: Logback has known high-severity vulnerabilities that can be exploited for code execution or to disclose sensitive information.
* Recommended Solution: Upgrade to a more secure version of Logback.

4. log4j-api-2.12.1.jar

* Vulnerability Code: cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:\*
* Description: The Log4j library has critical vulnerabilities allowing remote code execution (RCE) when untrusted data is processed.
* Recommended Solution: Upgrade to Log4j 2.17.1 or higher to address the RCE risks.

5. snakeyaml-1.25.jar

* Vulnerability Code: cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:*:*:*:*:*:*:\*
* Description: Snakeyaml has critical deserialization vulnerabilities that can lead to arbitrary code execution when processing untrusted input.
* Recommended Solution: Use a newer, secure version of Snakeyaml, addressing the deserialization flaws.

6. jackson-databind-2.10.2.jar

* Vulnerability Code: cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:\*
* Description: Jackson-databind contains high-severity vulnerabilities, including issues with deserialization of untrusted data that could lead to RCE.
* Recommended Solution: Update to a version of Jackson-databind that patches these vulnerabilities.

7. tomcat-embed-core-9.0.30.jar

* Vulnerability Code: cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:\*
* Description: Tomcat has critical vulnerabilities related to request handling, allowing for the possibility of RCE or bypassing security mechanisms.
* Recommended Solution: Upgrade to a newer, secure version of Tomcat to mitigate these risks.

8. hibernate-validator-6.0.18.Final.jar

* Vulnerability Code: cpe:2.3:a:redhat:hibernate\_validator:6.0.18:*:*:*:*:*:*:\*
* Description: Hibernate Validator contains medium-severity vulnerabilities that could lead to validation bypass or improper input handling.
* Recommended Solution: Upgrade to a more recent version of Hibernate Validator to ensure safer validation handling.

9. spring-web-5.2.3.RELEASE.jar

* Vulnerability Code: cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:*:*:*:*:*:*:\*
* Description: Spring Web has high-severity vulnerabilities that may allow attackers to bypass authentication or perform cross-site scripting (XSS) attacks.
* Recommended Solution: Upgrade to a patched version of Spring Web to address these security concerns.

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

The results from the manual review and static testing report revealed several significant security vulnerabilities in Artemis Financial’s software application, requiring immediate attention to mitigate potential risks. The manual review identified a lack of input validation in files like CRUD, leaving the system vulnerable to SQL injection attacks. In order to address the input validation vulnerability, the application must implement input validation and sanitization by using parameterized queries and prepared statements. In addition to the lact of input validation, incomplete error handling in files such as CRUDController and myDateTime could expose sensitive system details to users. This can be corrected by ensuring that error messages are handled properly, while maintaining detailed internal logs. Sensitive data exposure is another critical issue, particularly with the unencrypted data storage in DocDataand the use of hardcoded credentials in application.properties. Encrypting data at rest and in transit and moving sensitive information like credentials to secure vaults or environment variables will reduce the risk of data breaches.

Another major issue is the lack of proper authentication and authentication checks. In the class GreetingController, there is a risk of unauthorized access, which could be addressed by implementing proper role access control. The static testing report also identified several external dependencies, such as Log4j and Spring Boot, with known critical vulnerabilities that could lead to remote code execution or deserialization flaws. Regularly updating these libraries to the latest patched versions is essential to maintain security. Additionally, improving security logging and monitoring mechanisms will allow for better detection of suspicious activities, enabling quicker response to potential threats. By addressing these vulnerabilities through proper input validation, improved error handling, encryption, authentication, and regular updates, Artemis Financial can significantly strengthen the security of its application.