# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/20/2024** | **Chantel Hoesman** | **Updated section 1** |
| **1.0** | **245/24** | **Chantel Hoesman** | **Finished remainder** |

## 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

Chantel Hoesman

**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?

Secure communications to Artimis would mean protecting customer data in all states. Since the project is for a RESTful web API. There are some special parameters that we must meet to make this happen. REST API uses HTTP request to interact with data. The data is manipulated with command methods. Then returns information to the user in JSON, XML, or HTML. Best way to defend against attacks is to implement validation, access controls, and encryption (Nair, 2023).

Since we are using an API data can be accessed from anywhere in the world. We can assume international transactions will need to occur. Since this is a financial application, we should be following FinTech guidelines. Per and Idea Soft article it would need to follow “the Gramm-Leach-Bliley Act (GLB), the Bank Secrecy Act (BSA), the US Patriot Act, the Electronic Signatures in Global and National Commerce Act (E-Sign Act), the Truth in Lending Act (TILA), the Truth in Savings Act (TISA), the Electronic Fund Transfer Act (EFTA), the federal Red Flags Rule (IdeaSoft, 2024)” and Payment Card Industry Data Security Standard (PCI DSS). PCI DSS requires companies to create digital keys to validate user identities. (IdeaSoft, 2024)

External threats the software is mostly due to the vast amount of data that is collected about the customers. Things such as malware attacks and cyber breaches cause irreparable harm to companies. (Casey, 2024)

The modernization of financial technology is ever growing links to 3rd party applications. It is important that data is secured and transferred without access to leaks or a threat actor. By implementing secure coding practices, you can avoid things such as SQL injection. You can follow OWASP standards of coding to reduce risk. An important factor of the code is authentication. This will reduce unauthorized access to the system. With the threat of AI, I would suggest avoiding using non bio marked 2fa. This will assist in maintaining integrity. (Williamson, 2024)

Open-source code can be used in the development of the project. The debate between open and commercial code use is only an issue if known vulnerabilities are accepted. According to TuxCare even commercial code can have issues that are not discovered. It may be easier to use open source and verify the code is created with proper coding procedures (TuxCare, 2023).

**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.

**Data Encryption is important in this field to protect customer financial data.**

**Authentication and Authorization – also important as it is required to protect theft of customer data and finances.**

**Input Validation – important to protect against malicious injections.**

**Session Management – it is a great idea to use session management to make sure that it is protected against hijacking.**

**Error Handling and Logging – this is important for catching inside as well as intruders who may have gained access we can track where they have been. Or see persons who have access reviewing files they should not have access to.**

**Dependency Management – Patching and verifying dependency issues is an important step to protecting your applications.**

**Infrastructure Security – Depending on how your program is housed there you may or may not be responsible for the infrastructure, however they will all have to meet government requirements.**

**3. Manual Review**

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

After cursory review of the code. There are some issues with the code that will require a secondary team to verify issues. Unable to locate input validation, data encryption, error handling, or understand what infrastructure security is in place.

**4. Static Testing Plan Dependency Check Report included in secondary documentation.**

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

Most of these dependencies can be mitigated by downloading the most recent releases. However, you can also avoid using these dependencies as well.

* bcprov-jdk15on-1.46.jar is a dependency that could allow for plain text recovery attacks as one vector however there are many CVE that would be addressed on this dependency. This item seems to have several issues and it would be best to avoid the use of this jar or any version 1.55 or earlier.
* spring-boot-2.2.4.RELEASE.jar has a dependency that could allow for temporary directory hijacking. avoid the use of 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older.
* logback-core-1.2.3.jar has a vulnerability that allows arbitrary code to be introduced and run. Use log back version after 1.4.11.
* log4j-api-2.12.1.jar has an improper validation certificate which allows SMTPS connection to be intercepted causing MIM access. Avoid Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1). Best to utilize 2.12.3.
* snakeyaml-1.25.jar the alias feature allows for entity expansion may allow for DOS attack. Use SnakYaml version 2.0 or higher.
* jackson-databind-2.10.2.jar flaw in the binding of entity expansion which allows for external entity attacks. We recommend upgrading to version 2.13.4 and beyond.
* tomcat-embed-core-9.0.30.jar refactoring introduces a regression in transfer headers. Use version 8.5.64 or higher.
* hibernate-validator-6.0.18.Final.jar message extrapolation led to invalid expressions being validated. Do not use version 6.1.2 to avoid this dependency.
* spring-web-5.2.3.RELEASE.jar remote code execution vulnerability. Avoid the use of 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions.
* spring-beans-5.2.3.RELEASE.jar remote code execution vulnerability via data binding. Avoid the use of this dependency as there may be other ways to exploit.
* spring-webmvc-5.2.3.RELEASE.jar allows for malicious input from users. Avoid the use of versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18 and older.
* spring-context-5.2.3.RELEASE.jar older versions are not properly protected. avoid the use of 5.3.0 - 5.3.18, 5.2.0 - 5.2.20 and older.
* spring-expression-5.2.3.RELEASE.jar specially crafted SpEL can cause denial of service. Avoid the use of versions 5.3.0 - 5.3.16 and older.

**5. Mitigation**

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.

For the most part the plan to mitigate the dependencies from the report is to upgrade to the recommended versions. If no recommended versions exist, we will avoid the use of those dependencies all together.

The main issue is the code is using older jar files that have known issues. There is no reason to utilize software we know has issues.

We should also take the time to review OWASP guidelines on secure code to verify that we have created a code that is sound infrastructure.

Citations

Nair, S. V. (2023, March 8). What are the common rest API security vulnerabilities? https://beaglesecurity.com/blog/article/rest-api-security-vulnerability.html

*Fintech software regulation in 2023 - across the Globe*. IdeaSoft. (2024, May 14). <https://ideasoft.io/blog/fintech_legal_regulatory_aspects/#2.1>

Casey, A. (2024, April 9). *The growing cyber risks in fintech and how to mitigate them*. Woodruff Sawyer. <https://woodruffsawyer.com/insights/fintech-cyber-risks>

Williamson, R. (2024, March 21). *Guide to secure Fintech application development using best practices*. Medium. https://ryanwilliamsonc.medium.com/guide-to-secure-fintech-application-development-using-best-practices-060a44172aa0

TuxCare PR Team. (2023, April 13). *Is it secure to use open-source code to develop Fintech apps?*. TuxCare. https://tuxcare.com/blog/is-it-secure-to-use-open-source-code-to-develop-fintech-apps/