# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/20/2024** | **Cody Adams** |  |

## 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

Cody Adams

**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?

The value is tremendous. This is because Artemis Financial handles sensitive financial information. Communication would include being able to communicate between clients, and the application. The need for secure data transmission is great, as well as making sure that data stays private.

* Are there any international transactions that the company produces?

Since the company has software for entrepreneurs, business, and government agencies around the world, this means they will be making international transactions. With that being said we will need to ensure that Artemis Financial is secure In making these transactions, as well as making sure that we are compliant with data protection regulations in any country that we do business in.

* Are there governmental restrictions on secure communications to consider?

Yes, governmental restrictions will vary from place to place around the world. This means that we will need to be compliant with those government regulations. Some examples would be from the SEC, FINRA, and other financial governing bodies. We will need to ensure secure communication to satisfy these regulations.

* What external threats might be present now and in the immediate future?

There are many types of threats that we could experience. Some of those could be data breaches, DDos attacks, phishing attacks, SQL injections, and cross site scripting attacks. These will target weak code within the application. By targeting the company with these attacks, hackers can obtain vital personal information.

* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

We need to make sure that all open source libraries that are used are up to date. They must also have no known vulnerabilities. For web application technologies we should make sure that they have greater security features. Some examples of this could be utilizing encryption, security testing, firewalls, DDos protection, and API security.

**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.

**Architecture Review:**

This allows us to see and identify security risks in the overall design.

**API:**

We need to ensure that different APIs are secure, and will prevent access from users that are not authorized. This secures the way the applications will talk with each other. Since Artemis Financial will be dealing with peoples finances people will need to be able to transfer money in and out of their accounts, which will require for secure APIs to communicate with other institutions.

**Cryptography:**

Cryptography will be important because when we transmit information back and forth, we do not want anyone being able to intercept or at least blatantly see that information, by using cryptography we can ensure that the important information we are transmitting, is protected. This also applies for the information that we are storing in our databases.

**Client/Server:**

Utilization of protocols like HTTPS will allow for even securer communication between the client and server.

**Code Quality:**

Creating robust, and modular code that is built with security in mind will limit vulnerabilities, and protect against improper error handling. We will stick to secure coding practices.

**Encapsulation:**

Building our data and methods into a single unit, helps to hide what is important, and show the unimportant. This practice is something we will use to combat against unauthorized access, and attempts to access sensitive information.

**3. Manual Review**

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

CRUDController.java: Injection Vulnerability

Customer.java: Visibility of Variables for account balance, and lack of synchronization for the methods that modify state.

DocData.java: No input validation, lack of prepared statements, and hardcoded credentials

GreetingController.java : No input validation

All issues found: Injection vulnerability, hardcoded credentials, lack of input validation, lack of prepared statements, visibility of variables, lack of synchronization, and no input validation.

**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

Here are the vulnerability codes found:

bcprov-jdk15on-1.46.jar

* CVE-2013-1624
* CVE-2015-6644
* CVE-2015-7940
* CVE-2016-1000338
* CVE-2016-1000339
* CVE-2016-1000341
* CVE-2016-1000342
* CVE-2016-1000343
* CVE-2016-1000344
* CVE-2016-1000345
* CVE-2016-1000346
* CVE-2016-1000352
* CVE-2017-13098
* CVE-2018-5382
* CVE-2020-0187
* CVE-2020-26939
* CVE-2023-33201
* CVE-2024-29857
* CVE-2024-30171
* CVE-2024-34447

It looks as though Bouncy Castle library contains multiple vulnerabilities. One of the recommendations for a solution would be to update to the newest version of Bouncy Castle. It also looks like these vulnerabilities have been documented. Looks like they are documented on OSSINDEX, and Red Hat advisories.

**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.

Recommended steps:

Update the dependencies for Bouncy Castle to start. This will be updated in the pom.xml files.

Use prepared statements: Defend against SQL injections. This is all for database queries in the DocData.java file.

Protect against injection vulnerabilities: We noticed business\_name in the CRUDController.java file. Validate this to protect against injection vulnerabilities.

Visibility of variables: Make sure that the variables used in the customer.java file are private and not available globally.

Encode User Inputs: This prevents data exposure.

Use Access Control: This will prevent unauthorized access. Properly define user roles.

Improve error handling: This protects sensitive information from being seen by the wrong eyes. For example, account\_balance is too visible. Make the access to that private.

Remove any redundant code: This will protect against people taking advantage of the redundant code.

Complete Date/Time Handling: The inputs should be validated. Implementation of missing functionalities. This can be changed in the myDateTime.java file.