****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)-5

[3. Manual Review 5](#_Toc32574613)

[4. Static Testing 5-6](#_Toc32574614)

[5. Mitigation Plan 6](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/17/2020** | **Garrett Stubblefield** |  |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Garrett Stubblefield

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following 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 about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Here at Global Rain we take our clients safety and security seriously and have interpreted your needs and how an attack or vulnerability that we need to prepare for. As a financial institution your users are trusting you to keep their money and information safe and private. This also means that any communications with users or within the company will need to be secure and private. As the world today is more connected and travel is easier Artemis Financial will need to connect globally incase anyone needs access to their account while traveling abroad. Here in the US we have protocols on security and the level that financial institutions must take in order to keep the licensing. For credit card management we have compliance standards such as PCI-DSS. This standard defines what type of data can be stored with card numbers and how they will need to be encrypted when stored or transferring. There are also various international privacy laws in different parts of the world that we must prepare for. (Manico, 2014, Ch.10) Since this application will have such high value data external threats will be many now and in the future. When it comes to applications that use money they are typically the most vulnerable to an attack, because the attackers are highly motivated to get some. Since the attackers will motivated to attack financial institutions we will need to keep updating our security as new attacks present themselves. The best way we do this is through Query Parameterization to help against SQL injections.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financials’ software application. Justify your reasoning for why each area is relevant to the software application.

The Areas of Security we will need to access for this software application will be most of the seven areas of security. We will need to validate user input and make sure that when a possible SQL injection is being tried that we are not taking the input parameters as request and not a command. This can be done my output encoding that will be changed on the database. Since this software application will also be using an API and we will have to make sure they are secure and not vulnerable to SQL injection or attacks. With Cryptography we will have to incorporate some type of encryption when sending or receiving such sensitive data so this will be a major area for vulnerabilities. Not only will this application be sending or receiving that data we will have to store it and that means we have possible vulnerabilities for the client/server. This is an important vulnerability since our servers will be containing a lot of private and important data. Code errors will be another vulnerability to look out for since any possible errors could create a vulnerability that could hinder our application. Also, if we have a code error that could make it so some protections do not work properly and could be an area of vulnerability. Going along with Code Error is Code Quality and we will need to make sure we have quality code that protects against attacks. Having Quality Code will also help with Encapsulation vulnerabilities making sure that our data and variables have the private protection so only the class that it lives in can access that data. This will need to be careful with the accessors if other classes use it.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

While reviewing the code manually there are some vulnerabilities and aspects that need to be taken care of. In the CRUD Controller Class when it is called without any parameters it gives back a message about no business name being present. This is a vulnerability because it lets a potential attacker know that entering a business name that is a client may give some information they could use for possible injection. In the CRUD Class none of the methods have a private naming convention and they all are public. This is a vulnerability for any data stored in this class could be accessed by other classes and if an attack found way to use calls or commands for a certain class then it would be able to access the CRUD Class as well. The Greeting Class was good for input validation and when trying to use inputs on the URL like ‘--’ it just displayed that as the string and not a command. In the Customer Class while it set the account number variable to private the method that shows the information was set to public and this could lead to serious ramifications if there was an attack they could easily get the account numbers by calling that function. As for the DocData Class it was not complete and there were comments that the SQL part of the code needs to be complete. This Class will need serious oversight to prevent a possible SQL injection attack. The MyDateTime Class is also incomplete and has comments that it does need to be complete.

## 4. Static Testing

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

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

The first vulnerability name is bcprov-jdk15on-1.46.jar. This vulnerability has to do with the Bouncy Castle Crypto package for the JDK I have on my computer. This vulnerability has to do with using cryptographic algorithms in the application. While Artemis will need this at a later time right now the code base does not use any cryptography so the vulnerabilities from the CVE codes do not show any flaws in the code base. The next dependency is log4j-api-2.12.1.jar. This dependency deals with the API that the Spring framework is using and that it is vulnerable to a man in the middle attack through the SMTPS connection which is for the mail servers. Since this code base is not using any SMTPS to communicate this dependency also does not affect our code base for vulnerabilities as of now. The last dependency that the dependency check found was the tomcat-embed-core-9.0.30.jar. This dependency is an open source Java Servlet. Some of the vulnerabilities that the dependency check is showing for Tomcat is that the refactoring present in this version introduced a regression that could lead to invalid transfer-encoding and a possibility HTTP request smuggling. Specialty crafted HTTP sequence requests that could trigger high CPU usage for several seconds and the sever could become non-responsive. With this version of tomcat an attacker may be able to control the name and contents of a file.

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financials’ software application.

The mitigation plan for Artemis Financials’ will entail fixing the current code base and trying to make sure the software is prepared for all potential attacks that the dependency checks shed light on. The first part of the code base will be to clean up the CRUD controller and make sure that when the error message of no business\_name does not say that. This will ensure that any attackers will not know what is supposed to be called there. Next will be to ensure that all the variables and methods have the correct naming convention and that they have the proper modifier. Any data that is sensitive should have a private modifier. The DocData Class will need to be finished and the SQL code will need to be complete and have the correct monitors for any possible SQL injection. The final class that will need to be completed will be the MyDateTime Class and this is important because it will show when certain aspects of the software are accessed and should log users off if logged in for too long. As of now we do not use a cryptography, but in the future this will need to be implemented. The Bouncy Castle that is implemented on this application right now, but has a vulnerability that does not properly consider timing side-channel. As of now we are not using SMTPS to communicate through the mail servers. If we were to implement it into the code base then we must put in the proper security to defend against a man in the middle attack. The last vulnerability that the code base had from the dependency check was the Tomcat Java Servlet. To prevent any HTTP request smuggling or access to unauthorized users we will need to add in authorizations. As of now all the code is secure and there is no private data being stored, but as we implement the proper securities our defense will need to become more sophisticated with cryptography, encryption, and good quality code.

**Citations:**

MANICO, J. A. (2014). *IRON-CLAD JAVA building secure web applications;building secure web applications*. NEW YORK: MCGRAW-HILL EDUCATION.