****

# 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)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/19/2021** | **Eddie Helmick** | **Project 1** |

## 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

Eddie Helmick

## 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:

1. What is the value of secure communications to the company?
   1. Artemis financial can benefit greatly from secured communications. Having secured communications would provide greater peace of mind to their clients, protect company secrets and sensitive data, and it would also help them better comply with government restrictions (below). The most important piece to all of this is that is allows longevity of their business. Data leaks are subject to large lawsuits and fines – protecting this information is the best way to avoid such ordeals.
2. Are there any international transactions that the company produces?
   1. It is likely that this company produces international transactions, since it is a massive financial institution.
3. Are there governmental restrictions about secure communications to consider?
   1. Not surprisingly, the government has a lot of regulations around secure communications to protect citizens. The Gramm-Leach-Biley act, for example, requires organizations to develop and maintain a comprehensive security program which matches the size and complexity of the application. It is specific to banks and financial institutions. Another important regulation is the consumer privacy act of 2017, which requires organizations to mitigate user information from identity theft, fraudulent access, and so on (itgovernanceusa).
4. What external threats might be present now and in the immediate future?
   1. There are a lot of potential external threats, but generally speaking software applications which are web based are always at risk of DDoS attacks, and the clients are at risk of phishing attempts. It is entirely possible that security vulnerabilities that hackers are already aware of and utilizing for malicious purposes currently exists inside of the application, and this can always open up in the future.
5. What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies?
   1. The major requirements to be considered when modernizing an application is what individual requirements the stakeholders may have, what security flaws could open up when migrating to newer versions of external libraries, what budget is available to complete the task, and how to continue protecting sensitive information.

## 2. Areas of Security

The following are the areas of security to address, per the VAPFD:

1. Input validation: It is always important to validate input in any application that accepts input from outside sources. Without input validation, the application will be more vulnerable to hack attempts. Hackers may add strings to legal input fields, cookies, and other parts of the HTTP request (Oracle Press). A great way to validate input fields is by using regex to ensure the input is exactly what you expect, eliminating illegal characters.
2. APIs: This application is an API, and it takes in information for processing. It is important to assess the API interactions in the application to prevent issues such as (but not limited to) SQL injections, user authorization issues, and resource abuse.
3. Cryptography: This application connects to a SQL database and utilizes client account information such as the account number, name, and account balance. It is important for this data to be encrypted to prevent compromising of client data.
4. Client/server: This application is a REST API interface, and user clients will connect to it to access data relevant to their account. It is important to secure this connection to prevent malicious use.
5. Code error: It is important to have proper code error handling to ensure what happened, who did it, and how they did it. Without this logging, it will be difficult to identify any missed security vulnerabilities in the future. It will also be challenging to detect when there are security issues in the first place.
6. Code quality: Generally speaking, it is always important to ensure code quality is high. Planning for high quality, secure code, and following best practices, will allow for easier expandability in the future without the risk of adding unseen security vulnerabilities. Doing it the right way now will all for long term sustainability.
7. Encapsulation: Since this application takes in information from outside sources, it is important to be sure the data structures are encapsulated to protect sensitive data from being accessed. This is especially important when working with external APIs, as it can add an extra layer of security to prevent outside sources from accessing methods and data that they are not meant to access.

## 3. Manual Review

There are a few different code vulnerabilities that exist in the current code base, as seen below:

1. The application lacks sufficient error handling.
2. The GET requests in the CRUD controller and greeting controller lack input validation.
3. The code lacks informative commenting, which may lead to errors in the future when adding new features.
4. The Spring version being used, 5.2.3, is out of date. The most up to date version is 5.3.6.
5. The variable “account\_balance” in customer.java should be a private variable with an accessor method.
6. The CRUD controller may be returning potentially sensitive information back to the user.

## 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 dependency check report. Include the following:

1. bcprov-jdk15on-1.46.jar - The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.
   1. There are many CVE’s for the bouncy castle crypto package, with the bulk of the problems being centered around issues which allow invalid keys or requests to be processed through the dependency which would allow an attacker to recover sensitive information. The most recent is CVE-2020-26939, which allows attackers to obtain sensitive information about a private exponent because of Observable Differences in Behavior to Error Inputs. This occurs in the encodings.OAEPEncoding file. Updating this dependency to 1.61/BC-FJA 1.0.1.2 will allow for the prevention of this issue (ossindex).
2. hibernate-validator-6.0.18.Final.jar – Hibernate's Bean Validation (JSR-380) reference implementation.
   1. CVE-2020-10693 – the issue identified was a bug in the message interpolation processor that enables invalid EL expressions to be evaluated as if they were valid. The fix was patched and pushed in update 6.0.20.Final, and the solution to prevent this vulnerability in our application is to update to this version.
3. jackson-databind-2.10.2.jar - General data-binding functionality for Jackson; works on core streaming API
   1. CVE-2020-25649 – the issue identified was a flaw found in FasterXML, where it did not have entity expansion secured properly. It does not appear that we are using this in our application, therefore this can be disregarded.
4. log4j-api-2.12.1.jar – The Apache Log4j API
   1. CVE-2020-9488 – Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. We do not appear to be using any SMTP services in this application, so this can be disregarded.
5. snakeyaml-1.25.jar - YAML 1.1 parser and emitter for Java
   1. CVE-2017-18640 – the issue identified is a bug which allows entity expansion during a load operation. This issue was fixed in snakeyaml version 1.26, so updating to version 1.26 would fix this problem.
6. spring-aop-5.2.3.RELEASE.jar - Spring AOP
   1. CVE-2021-22118 – the issue identified was a bug causing webflux applications to be vulnerable to a privilege escalation. Upon reviewing the code base, Spring version 5.2.3 is being used. Because it affects versions 5.2.x to 5.2.14, this is a false positive and can be ignored. We do not need to be concerned, only informed, with the security flaws of past versions. It is important to understand what could possibly come up again and plan for it, but not necessary to make direct changes right now.
   2. CVE-2020-5421 – the issue identified was a bug allowing RFD attack protections to be bypassed. This vulnerability was patched in Sprint version 4.3.29, so it is a false positive and can be ignored. We do not need to be concerned, only informed, with the security flaws of past versions. It is important to understand what could possibly come up again and plan for it, but not necessary to make direct changes right now.
7. spring-core-5.2.3.RELEASE.jar - Spring core
   1. CVE-2021-22118 – the issue identified was a bug causing webflux applications to be vulnerable to a privilege escalation. Upon reviewing the code base, Spring version 5.2.3 is being used. Because it affects versions 5.2.x to 5.2.14, this is a false positive and can be ignored. We do not need to be concerned, only informed, with the security flaws of past versions. It is important to understand what could possibly come up again and plan for it, but not necessary to make direct changes right now.
   2. CVE-2020-5421 – the issue identified was a bug allowing RFD attack protections to be bypassed. This vulnerability was patched in Sprint version 4.3.29, so it is a false positive and can be ignored. We do not need to be concerned, only informed, with the security flaws of past versions. It is important to understand what could possibly come up again and plan for it, but not necessary to make direct changes right now.
8. tomcat-embed-core-9.0.30.jar - Core Tomcat implementation
   1. There are many CVE’s for the tomcat .jar files. The most recent is CVE-2021-33037. A large amount of the CVE’s exist due to missed pieces of fixes required for previous CVE’s. The issues are generally related to vulnerabilities in file structures and HTTP requests. To resolve all of these issues, we should ensure we are using tomcat version 9.0.48 .
9. tomcat-embed-websocket-9.0.30.jar - Core Tomcat implementation
   1. There are many CVE’s for the tomcat .jar files. The most recent is CVE-2021-33037. A large amount of the CVE’s exist due to missed pieces of fixes required for previous CVE’s. The issues are generally related to vulnerabilities in file structures and HTTP requests. To resolve all of these issues, we should ensure we are using tomcat version 9.0.48.

## 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 Financial’s software application.

1. Ensure all input fields have proper validation. Specifically, limit the size of accepted input fields to prevent massive data being passed through the controller and parse inputs to prevent or identify string patterns which may cause issues.
2. Upgrade the version of Spring being used in the application to the most recent version, to ensure all patched security vulnerabilities between the old and up to date versions are being included in the project.
3. Preform necessary dependency version changes to avoid/fix listed security vulnerabilities.
4. Add commenting for the code to allow for clean and easy to read code. This can help explain the purpose of specific blocks of code and allow for easier maintainability in the future.
5. Implement error handling and logging to understand what is being passed through the controllers. This can help identify other security vulnerabilities in the future.

References

*Federal cybersecurity and data privacy Laws directory*. IT Governance. (n.d.). Retrieved September 19, 2021, from https://www.itgovernanceusa.com/federal-cybersecurity-and-privacy-laws.

Sonatype, I. (n.d.). *[CVE-2020-26939] in Legion of the bouncy Castle BC Before 1.61 And BC-FJA before 1.0.1.2, Attacke...* Sonatype OSS Index. Retrieved September 19, 2021, from https://ossindex.sonatype.org/vulnerability/9e56f765-fe13-4d65-925a-241a8047f1b6?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm\_source=dependency-check&utm\_medium=integration&utm\_content=6.3.1.

Manico, J., Detlefsen, A., Kenan, K., & Smith, M. (2015). *Iron-clad java: Building secure web applications*. McGraw Hill Education.