****

# 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** | **19 September 2020** | **Natalie Hurt** | **Initial findings are documented and discussed.** |

## 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

Natalie Hurt

## 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?

As a company that deals with the finances of an individual, the importance of secure conversation is obvious. Finances are a private matter and it is important to secure any conversations that would arise around that conversation. The company also deals with insurance. Insurance protects private assets, and when a customer needs to discuss those assets with a representative, it is important to have secure communications over that matter. This also applies to any health or personal insurance that the company may carry or provide.

As a financial institution that is operating on par with its peers, it is safe to assume there will be some international transactions. As an online platform, we will assume and secure this web application to handle international transactions. They will also need to be able to handle the transactions of their customers should they travel internationally as well.

There are no governmental restrictions that I could find or that I specifically know of that pertain to secure communications in the United States. If we are considering the in the European Union, I found that there is the General Data Protection Regulation (GDPR), which went into effect May 25th, 2018 and stands today (gdpr-info.edu, 2019).

External refers to threats that come from outside of Artemis Financial. There is a threat of cyber-attack and IT security issues related to internet banking. Potential threats for the immediate future would be information breach. API threats will be an immediate concern as well as error handling. In the long run, we will have to explore the possibility of a serve breach or the server being misappropriated. The company will need to consider authentication security, internal security, and denial of service security.

It is important to understand that with any web-based programming, we will need to be proactive with modernization. It will have to be done moderately and over time. Incremental modernization is important when concerning legacy systems. It reduces operational risks. We should also consider access to the application technologies and source libraries. This access will also allow for more developers to work on the code (Wennegren et al., 2019).

## 2. Areas of Security

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

Validation Input:

Programs that involve a log in should also include some sort of input validation. If the system allows for online check deposit, there will need to be numerical checks in the forms that allow for number entry.

API:

This web application has a call to REST API. We need to secure this configuration to ensure secure communication between the system and the API.

Cryptography:

In a financial institution that stores and transmits private user information, encryption is key. Lack of encryption could compromise the data when it is transmitted across the internet.

Client/Server:

Each computer that access the website is considered a front-end device. These devices communicate with the server. This client and server communication is important because there is information being transmitted across so the connection must be secure to ensure the confidentiality of the information.

Code Error:

It will be important for the code to handle and process error checks to apply the source code of the application.

Code Quality:

We need to maintain the consistency of the code for the business logic of the application.

Code Quality and Handling go hand in hand because there is a need for clean code in an application that is processing private information like what is being considered in the financial institution.

## 3. Manual Review

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

DocData.java

Data Access Vulnerability:

The username and password access method has some vulnerabilities. The password is a root password which is easily guessed and compromised. If you pair this with the root username it is easy to crack and could provide opportunity for unauthorized access to the system through a simple guess and check attack.

CRUDController.java

Controller Vulnerability:

There is the opportunity for the controller to expose an internal database. By passing “business\_name” there is an opportunity to expose DocData object database access vulnerability.

## 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. 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

A screenshot of a social media post

Description automatically generated

CVE-2013-1624:

The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.

Solution—Proper timing checks should be done and considerations on side channel attacks to be implemented to prevent timing data attacks. (Source, 2013)

CVE-2015-6644:

An information disclosure vulnerability in Bouncy Castle could enable a local malicious application to gain access to user’s private information

Solution— Regular updates of the application should be done as well as the underlying operating system to prevent malicious applications to compromise the security of this application.

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.

Solution—Upgrade to 2.13.2 which supports this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections. (Sicker, 2020)

CVE-2017-18640:

The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564

Solution— If the YAML is not coming from untrusted source (it is merely a configuration file) then it is a false positive. Just ignore it. The quality of NVD database is very low and contains tons of issues which appear to be false positives. Migrate to SnakeYAML Engine. It has a configuration option to restrict aliases for collections (the aliases for scalars cannot grow and they are not restricted) SnakeYAML has now also possibility to fail early. (Wiki, 2020)

CVE-2020-7712:

This affects the package json before 10.0.0. It is possible to inject arbritary commands using the parseLookup function.

CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Solution—Upgrade json to version 10.0.0 or higher. (Source, 2020)

CVE-2019-17569:

The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

Solution—Update the tomcat packages.

## 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.

DATA ACCESS (username and password issue):

The data access username and password will need to have input validation. We will need to require a strong password with a set of alphanumeric characters, special characters, non-identifiable words, and cannot include the username. The user should be created outside of the root in order to be a user accessing the database.

CODE REVIEW (modification):

We will review the code to implement secure coding practices. Code quality will be improved to implement proper code error catching behavior. This will prevent future attacks and allow for the catching and remedying of authentication vulnerabilities.

UPDATING (upgrading):

We will update the Apache server and the tomcat packages to remedy the CVE-2019-17569 vulnerability. This update will be used to fix the vulnerabilities in the out of date versions. We will also upgrade the JSON to version 10.0.0 or higher. This will prevent the ability to inject commands using the parseLookup function.

CERTIFICATE VALIDATION (CVE-2020-9488):

The code needs to be cleaned to allow for proper validation and verification of the digital certificates. We will update the Apache to 2.13.2 which supports this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections. (Sicker, 2020).

TLS CERTIFICATE (mutual checking):

There needs to be a mutual check involving the client/server communications. Proper timing checks should be done and considerations on side channel attacks to be implemented to prevent timing data attacks. (Source, 2013). This will prevent the exploitation of the CVE-2013-1624 vulnerability. It will prevent the compromise of client API requests and the authentication processes.

Citations:

Command Injection in json. (2020, August 30). Retrieved September, 2020, from https://snyk.io/vuln/SNYK-JS-JSON-597481

Dependency Check Maven. (2020, September 13). Retrieved 2020, from https://jeremylong.github.io/DependencyCheck/dependency-check-maven/index.html

Official Legal Text. (2019, September 02). Retrieved 2020, from https://gdpr-info.eu/

Oss-security - Re: CVE request: TLS CBC padding timing flaw in various SSL / TLS implementations. (2013, February 15). Retrieved 2020, from https://www.openwall.com/lists/oss-security/2013/02/05/24

OWASP Secure Coding Practices Quick Reference Guide. (2010). Retrieved 2020, from https://owasp.org/www-pdf-archive/OWASP\_SCP\_Quick\_Reference\_Guide\_v1.pdf

Secure Coding Guidelines for Java SE. (2020, June 3). Retrieved 2020, from https://www.oracle.com/java/technologies/javase/seccodeguide.html

Sicker, M. (2020, April 21). CVE-2020-9488. Retrieved September, 2020, from https://issues.apache.org/jira/browse/LOG4J2-2819

Wennergren, D., Buchholz, S., Cruley, K., & Siegel, B. (2019, December 16). Accelerating IT and Cybersecurity Modernization. Retrieved 2020, from https://deloitte.wsj.com/riskandcompliance/2018/09/24/accelerating-it-and-cybersecurity-modernization/

Wiki. (2020, April 21). Retrieved September, 2020, from https://bitbucket.org/asomov/snakeyaml/wiki/Billion laughs attack