****

# 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** |
| --- | --- | --- | --- |
| **3.0** | **5/30/2021** | **Steven Reid** |  |

## 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

Steven Reid

## 1. Interpreting Client Needs

As a financial firm, Artemis conducts financial transactions, creates relationships, and aims to help customers get the most out of their money as well as help manage their financial position. The firm will no doubt deal with information that needs to be secure as a breach of this kind could result in customers losing money. In addition to servicing customers, Artemis needs to be able to conduct business of their own as well. This means accessing relevant financial information, watching trends, and being able to action on anything needed based on financial information or the will of their customers.

Along with keeping customer information secure, communications should be secure as well. Each conversation between anyone associated with the firm could be considered confidential. These conversations also lead to action that has monetary implications, which makes it extremely valuable.

Some of the most common security threats facing financial institutions today range from malware being introduced into the companies software via e-mail to monitoring company employee internet traffic and infecting machines via other sites not related to Artemis Financial. The threats are vast.

John Hock lists the top 5 common security threats to financial institutions on DorenMayhew’s website.

1. Malware
   1. Malicious code introduced unknowingly via outside sources.
2. Phishing
   1. Obtaining user credentials via a fake email which tricks people into giving up their passwords, often in an attempt to reset them.
3. Ransomware
   1. An attack that involves infiltrating a software system and then holding it hostage for money.
   2. Sometimes, after the ransom has been paid, the hacker may take the time to inform then company they got in.
4. Denial of Service
   1. Flood the servers with internet traffic via requests that effectively shut down the service to customers. This could be especially be detrimental to allowing customers to conduct business on our platform. Trading is a time sensitive activity.
5. Watering Hole
   1. Lurking on other sites known to be visited by employees of the target firm and infects a less secure site with malware.

**Modernization of Open Source Libraries and Evolving Web Technologies**

Some of the most common security threats can be solved by adding in additional controls. 2 factor verification is common when logging into systems containing sensitive data. Additionally – we can mitigate security threats by ensuring we are using the latest version of a particular dependency. In the case of our tomcat dependencies, which are ranked as a critical threat according to our dependency check would be to follow CVE-2021-25329’s instructions and update the version of the dependency to 9.0.43 or even 10.0.2, where the risk has been mitigated.

## 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.

The following areas below represent the most critical threats to our application in terms of resources, information protection, and productivity. The below have been called out due to their evidence count, which tells us that the vulnerability is valid and not a false positive. Deserialization of serial keys can mean that passwords may not be secure. Input validation is also a very important thing for us to mitigate as it is a primary means for logging in.

**Client / Server**

* Tomcat
  1. Information leakage

**Cryptography & API**

* BouncyCastle Legion
  1. Key deserialization
  2. BKS Keystore Compromised

**Input Validation**

* Hibernate Bean
  1. Input sanitation.

## 3. Manual Review

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

**Hibernate Bean Validator Review**

Package Name: hibernate-valdidator-6.0.18.jar

Upon inspecting the package in our development environment we found that our current version of hibernate bean 6.0.18 is impacted by the CVE-2020-10693. It does not have good measures in place for input sanitation and stripping.

**Bouncy Castle Review**

Package Name: Bcprov-jdk15on-1.46.jar

The vulnerabilities mentioned in CVE-2018-5382 impacts versions of this package up to 1.49. We can verify in the class file ISO9796D1Encoding.class within package org.bouncycastle.crypto.encodings that it sets the max integer size to 16 for the HMAC. This allows attackers to compromise the BKS Keystore (*CVE-2018-5382 Detail*, 2018).

**Tomcat Review**

Package Name: tomcat-embed-core-9.0.3.jar

Within this code base is where our authenticators are. This is where we authenticate client requests. Without proper handling of checking for uniqueness, request headers could be duplicated and passed through, causing information to be seen or can allow someone to watch another across the internet by obtaining a copy of their requests.

While we did find some vulnerabilities we need to further understand and mitigate, it is of note that the application itself is free of error. Additionally, the quality of the code is respectable. It makes use of several common best practices in cryptography such as Shamir’s Trick, which allows us to have public key encryption. This helps with password security (Green, 2018).

## 4. Static Testing

**Client / Server**

* Tomcat Dependency Vulnerabilities:
  + 1. CVE-2021-25122 exposes us to information leakage. The dependency has unsecure code that enables the input of duplicate request headers and the ability to duplicate request headers and can track which websites are visited. (*National Vulnerability Database*, 2020). We consider this a true vulnerability because the issues CVE count is one of the highest on the report. Additionally, its evidence sufficient to determine its validity (Evidence Count: 32).

**Cryptography**

* BouncyCastle Legion Vulnerabilities:
  + 1. CVE-2018-5382 refers to vulnerabilities in our cryptography. This can expose our system to the execution of unexpected code . Our current version (1.46) of Bouncy Castle Legion Java Cryptography allows for the deserialization of public keys which are a crucial aspect of how information securely travels across the web. We consider this a legitimate threat to our information security as the evidence count is significantly high (37).

**Input Validation**

* Hibernate Bean Vulnerabilities:
  + 1. A medium vulnerability per our dependency report, however, it exposes our security by allowing attackers to bypass basic input sanitation measures such as stripping and escaping (*CVE-2020-10693 Detail*, 2020).

## 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.

**Client / Server**

* Tomcat Dependency Solution:
  1. The primary mitigation for CVE-2021-25122 is to upgrade our tomcat library to 9.0.43 or later.

**Cryptography**

* BouncyCastle Legion Vulnerabilities:
  + 1. The vulnerabilities found within the review according to CVE-2018-5382 dealing with encryption has been mitigated in bouncy castle version 1.68. We need to upgrade to this version to fix the issue.

**Input Validation**

* Hibernate Bean Vulnerabilities:
  + 1. One possible solution for mitigating this risk is simply upgrading to Hibernate Validator 6.0.20.Final in which the bug has been corrected vs our current version 6.0.18 (Smet, 2020).

***Sources:***

Hock, J. H. (2020, October 8). *5 Common Cyber Threats to The Financial Institution Industry*. Doeren Mayhew. <https://doeren.com>

*CVE-2018-1000613 Detail*. (2018, July 9). National Vulnerability Database. <https://nvd.nist.gov>

*CVE-2021-25122 Detail*. (2021, March 1). National Vulnerability Database. https://nvd.nist.gov

*CVE-2020-10693 Detail*. (2020, May 6). National Vulnerability Database. <https://nvd.nist.gov>

Smet, G. S. (2020, May 7). *Hibernate Validator 6.1.5 and 6.0.20 Released*. HiberNate. <https://in.relation.to>

Green, M. G. (2018, July 2). *A Few Thoughts on Cryptographic Engineering*. Blog.Cryptographyengineering.Com. <https://blog.cryptographyengineering.com>

*CVE-2018-5382 Detail*. (2018, April 16). National Vulnerabilities Database. <https://nvd.nist.gov>

Apache, & Thomas, M. (2021, March 1). *[SECURITY] CVE-2021-25122 Apache Tomcat h2c request mix-up*. PonyMail! <https://lists.apache.org>

The Legion of Bouncy Castle. (2020, December 1). *Release Notes*. https://www.bouncycastle.org