****

# 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** | **[Date]** | **Alexis Scott** |  |

## 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

Alexis Scott

## 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?
  + The company Artemis Financial works in the area of personal finance. The clients that they have are individuals, these individuals trust the company with their personal and financial information. Their clients will expect their information to be protected at all costs, any breach that the company may have will cost Artemis the trust of the current clients they have, as well as getting new clients in the future.
  + Now Artemis may have clients with money from other countries, so they will need to have the ability to manage and exchange foreign currency safely. This may take additional steps to make sure that all the information is safely transferred and secured.
  + The government has plenty of restrictions when it comes to data and communication over the internet. For me to be sure of the company’s current system, I will need to know the government restrictions really well. Those restrictions may increase if you take international communication into consideration.
  + Artemis’ external threats are data that is stored in their systems and any communication/data being transferred over the internet. For the data that is stored on the systems it is best to be worried about injection attacks where SQL or other code can be injected into the user input boxes to extract data. We also need to make sure that we properly limit data behind credentials and have protocols to prevent any credentials from being wrongly accessed.
  + To modernize the company, I would make sure that any third-party libraries being used are kept up to date. I would also update to the newest versions of libraries to get the security patches that are available. Also, I would research specific versions so that I am aware of certain issues and fix them if I can.

## 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.

* **Input Validation:** After reviewing the Artemis codebase for security I believe that Input validation is one of the most important items to investigate. Input validation is one of the simpler issues to identify and resolve. I will make sure to look for parameterized SQL.
* **APIs:**  Artemis' codebase contains s RESTful API, checking to ensure that the API's are properly restricted is a definite must. First, I will validate API input similar to input validation, I will make sure that passing parameters or options with malicious intent does not result in data being sent unexpectedly. Also, I will ensure that access to the APIs are properly restricted by user levels. If a user does not have permission to access certain data, they should not be able to access it from the APIs.
* **Client/Server:** Since Artemis supplies APIs, I will need to ensure that Artemis is properly protecting data as it is being communicated from client to server. I will also need to ensure that permissions are properly enforced between client and server.
* **Code Error:** I will check to make sure that restricted data is not logged to the user in any error cases. These could be intentionally trigger errors and if the error logs contain sensitive data, this could be a security breach.

## 3. Manual Review

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

* **Update to latest Spring-boot-starter-parent:** The parent version in pom.xml:8 should be updated from 2.2.4 to 2.3.0. This ensures that we have the most recent dependency checking in infrastructure.
* **Dependency Versions:** Most dependency versions in the application are not up to date. It is likely that they were up to date at the time of writing the application. It may be worth updating to latest dependency versions. This may improve the static testing results below.
* **CRUDController.java:13:** On this line, you accept a parameter of "business name" in as a string. There is no input validation on this input. We do not trust the source of this parameter, so you need to validate input in some way.
* **GreetingController.java:16**: On this line, you accept parameter input from the user as a string. There is no input validation on this string, so a user could potentially pass anything, including an attempted injection attack.
* **DocData.java:26-27**: On these lines, you create a database connection. In this instance there are 2 issues. Firstly, you hard code a username and password. This is not recommended. Also, there are no privilege checks. Any user will have access to this database connection regardless of who they are. You should ensure the user has the correct permissions to access the data before creating a DB connection.
* **Customer.java:4-5:** These class level variables are a customer’s account data and account balance. These are both extremely sensitive pieces of information and need to be protected. At the very least, they will need to be private variables to prevent unintended access. Also, consider abstracting the data and storing the values outside of the program rather than storing them in the customer class.

## 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

My testing has identified 6 vulnerable dependency with 35 vulnerabilities.

* bcprov-jdk15on-1.46.jar (Bouncy Castle Crypto package)
* CVE-2013-1624: Java and C# libraries do not properly consider timing side-channel attacks on noncompliant MAC check operations.
* CVE-2015-6644: Info disclosure vulnerability could enable local malicious applications to gain access to user private info.
* CVE-2015-7940: Java library does not validate a point is within the elliptic curve.
* CVE-2016-1000338: DSA does not fully validate ASN.1 encoding of signature on verification.
* CVE-2016-1000339: CPU can be monitored sufficiently to leak information on the AES key being used.
* CVE-2016-1000341: DSA signature generation is vulnerable to timing attack
* CVE-2016-1000342: ECDSA does not fully validate ASN.1 encoding of signature on verification.
* CVE-2016-1000343: DSA key pair generator generates a weak private key when default values are used.
* CVE-2016-1000344: ECB mode is allowed which is unsafe.
* CVE-2016-1000345: Vulnerable to padding oracle attack
* CVE-2016-1000346: invalid keys can be used to reveal details about other keys
* CVE-2016-1000352: ECB mode is allowed which is unsafe.
* CVE-2017-13098: Use of JCE can allow an attacker to recover private keys from the application
* CVE-2018-1000613: private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code.
* CVE-2018-5382 : keystore files use HMAC that is only 16 bits. Not long enough to be secure.

## 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.

Firstly, all library dependencies should be updated to latest versions. Once the libraries are all up to date, I would focus on input validation for the API's. Make sure input validation is applied to all the areas identified earlier in this document and any future API’s. Make sure that all account information is properly abstracted and restricted by permission checks. Finally, I would ensure that all database connections require certain authentication and are only available to those with the certain permissions and make sure that any database username/passwords are abstracted and not coded in as plain text.