

# 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** | **May 26, 2022** | **Elizabeth Danzberger** |  |

## 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

Elizabeth Danzberger

## 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?

Because Artemis Financial is a financial consulting company who helps patrons develop financial plans for retirement, investments, insurance, and more, they are likely dealing with a lot of very personal information about their patrons that needs to be protected. Without proper security, the identities of their patrons could and would be compromised and because there are finances involved, money would like be compromised and stolen as well. In addition, software security is of great importance these days, as even if an attacker was not interested in the personal information or money of the company’s patrons, they could still take down the service and disrupt the company’s infrastructure if proper security measures are not put in place.

I do not have enough information to determine whether or not international transactions will be handled by Artemis Financial, but it makes sense to assume that international transactions would be taking place. Therefore, it is indeed important to consider this when developing the software. International transactions are at even more risk to be hijacked, as many other parties are involved in the transaction who may benefit politically or financially from disrupting the transaction.

There are most certainly government restrictions in place to oversee the proper handling of financial transactions both nationally and internationally, and there are also software security requirements defined by government institutions which outline the necessary security measures that should be taken when developing such software. These will have to be closely understood in order to ensure that Artemis Financial and Global Rain are not breaking any laws and that the companies are operating on the same “frequency” as other national and international institutions in this field.

There are a variety of external threats that might be present now and in the future. Namely, because Artemis Financial is a company where finances are their primary focus, they will be the target of many attackers who wish to steal personal information about the company’s employees and patrons, and attackers who wish to gain something financially. This is inevitable, and it is required that attacks like these be mitigated as swiftly as possible, and that the proper security measures are put in place to prevent these attacks from happening at all.

Open-source technologies are great, and I am a strong supporter of open-source software. However, I think it is important to keep the codebase of all software to be used by Artemis Financial in-house only, as those with malicious intentions may be able to easily gain access to the source code for any open-source libraries we may use in order to discover vulnerabilities. Although the open-source nature of these libraries means that vulnerabilities will likely be mitigated very quickly, avoiding open-source in this situation will allow the vulnerabilities to be analyzed in-house and prevent them from being known to the public in the first place, giving attackers less time to implement their attack strategies.

## 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.

Secure API transactions are a must. I assume that such an application will indeed need to communicate with different services in order to complete a transaction, and that such services will provided APIs for this. Therefore, the interactions with these APIs must be secure. Not only is this a standard in the modern software development, but also taking into consideration that this application will handle transactions involving finances and personal information, both nationally and internationally, this is an absolute necessity.

Cryptography is one of the most important topics of study in modern cybersecurity, and an application like this will need to make use of it in order to process transactions securely, both client-side and server-side. This will be necessary in order to properly encrypt personal information, banking information, and any other information that is being transmitted.

Secure error handling is a must for any application, but for a financial application such as this especially. Failure to successfuly handle errors in the code both before and during runtime may result in the leakage of sensitive data. This goes hand-in-hand with secure coding practices/patterns.

Although these are, in my opinion, the most important areas of security that I have identified from the vulnerability assessment process flow, I believe the others are equally as applicable (even though there is specifically a note in the assignment guidelines that states otherwise). I’m sure I’m not the only student who has expressed this, but from a software development perspective, no matter what type of application you’re working on, are these not all important?

## 3. Manual Review

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

Overall, I don’t find much wrong, because there isn’t a lot of code to begin with. However, the first thing I notice is that, in the customer.java class, the showInfo() method shows the customer information seemingly without any authentication required. It is important to always verify the client who is trying to access this information, since it contains sensitive information. It shouldn’t blindly return the customer information, trusting that the client is who they say they are.

In the same class, the deposit() method seems okay, other than that it blindly deposits money into an account. Ideally, in a real-world situation, it would check to see if there is even enough money to transfer from the source into the user’s account, or this should be conveyed by the other service and verified by our service. Authentication is also a concern here, since there is no way to tell if a trustworthy user is trying to deposit money, or if a malicious user is trying to exploit the service.

It does not seem that any encryption is taking place, unless it’s included in the Spring framework being used. When sending and receiving information, like faniancial information or customer information, cryptography is important in order to encrypt the data and prevent it from being read if it’s intercepted.

## 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

|  |  |
| --- | --- |
| **Vulnerability ID:** | CVE-2013-1624 |
| **Description:** | The vulnerability in the Bouncy Castle Java library allows attackers to carry out “distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets.”  The suggested solution for this vulnerability is to update the Bouncy Castle library to the latest version. |
| **Attribution:** | https://nvd.nist.gov/vuln/detail/CVE-2013-1624#vulnCurrentDescriptionTitle  https://access.redhat.com/errata/RHSA-2014:0371.html |

|  |  |
| --- | --- |
| **Vulnerability ID:** | CVE-2016-1000027 |
| **Description:** | The vulnerability in the Pivotal Spring Framework is that there is a remote code execution issue. This occurs when it is “used for Java deserialization of untrusted data.”  The issue is fixed by “detailing HttpInvoker endpoints should not be exposed to untrusted clients.” |
| **Attribution:** | https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2016-1000027 |

|  |  |
| --- | --- |
| **Vulnerability ID:** | CVE-2021-42550 |
| **Description:** | In this version of logback, “an attacker with the required priveleges to edit configuration files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.”  It is highly suggested to update to the latest version in order to remediate the issue. |
| **Attribution:** | https://security.netapp.com/advisory/ntap-20211229-0001/ |

|  |  |
| --- | --- |
| **Vulnerability ID:** | CVE-2020-25649 |
| **Description:** | In FasterXML, a vulnerability was found “where it does not have entity expansion secured properly.” This could allow XML external entity attacks.  The flaw has been fixed in a recent version, and it is recomended to upgrade to the most recent version. |
| **Attribution:** | https://github.com/FasterXML/jackson-databind/issues/2589 |

|  |  |
| --- | --- |
| **Vulnerability ID:** | CVE-2020-10693 |
| **Description:** | In Hibernate Validator, there is a flaw that has been found which affects the interpolation processor that enables invalid EL expressions to be executated even though they’re invalid. It can allow attackers to bypass input sanitization.  A solution is to “pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext.” |
| **Attribution:** | https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693  https://in.relation.to/2020/05/07/hibernate-validator-615-6020-released/  https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html\_single/#\_the\_code\_constraintvalidatorcontext\_code |

These are some of the highest vulnerability concerns that were found by the dependency checker.

## 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.

To fix the possible vulnerabilities found by a manual review of the code, the first step is to ensure that best coding practices are being followed. Namely, proper error handling. Properly handling errors will prevent sensitive data from potentially being leaked when the program shuts down unexpectedly, or something else happens that causes data to be exposed or corrupted. It is impossible to say what might occur unless proper error handling is implemented to catch errors.

The vulnerabilities found from a review of the static testing results (from the dependency checker plugin) can mostly be fixed by upgrading to the latest versions of the dependencies, and their solutions have been implemented by the package maintainers and contributors and are outside the scope of this analysis. However, for some of them, there are solutions that you can implement for the time being, if it is not possible to update a dependency (another dependency might rely on a very specific version of said dependency, which is bad practice in my opinion, but could break your application nonetheless). The solutions can be found by visiting the links attributed above and reading the description of the vulnerability.