

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

￼

￼

￼

￼

￼

￼

￼

￼

￼

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/19/21** | **Payton Mitchell** |  |

## 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

Payton Mitchell

## 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 value of secure communications to this company is invaluable because of the nature of business that this company is in. Artemis financial is a financial company that helps its clients come up with financial plans for savings, investing, retirement and insurance. This kind of information is some of the most sensitive kind that can be stolen. Within their web applications they have to keep data of their client’s addresses, banking information and possible even their social security numbers. This being the case, it’s no wonder that they came to Global Rain to make sure their security was up to date. This company has to keep track of money all over the world, which makes securing their international transactions even more of reason to be concerned about their software security. As far as governmental restrictions go, I do not believe that here are any that we have to consider here in terms of secure communication. The number of external threats that are present now and will certainly be around on the future are numerous. Some of these include input that can overload the system, injection attacks that can give a hacker access to the database, sensitive data exposure and unauthorized logins. These are only a few of the numerous external security vulnerabilities that Artemis financial has to worry about. When looking at modernization, we have to start at making sure that all dependencies are up to date. Doing this helps us make sure that we are not trying to solve any security vulnerabilities that have already been fixed. We also need to make sure that the application has been developed using the most updated best security practices. This means making sure that highly sensitive information is not logged, sensitive information does not show up in exceptions and generating valid formatting. The security vulnerabilities that have mentioned here only scratch the surface of the amount of security vulnerabilities this company has to deli with. These and many more are the reasons that they have called Global Rain for their security needs.

## 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.

[Include your findings here.]

When looking at the Vulnerability Assessment Diagram we can clearly see that every single one of the top row items apply to Artemis Financial. To begin, nowadays when it comes to applications of this nature, we will rarely see any that don’t take input. Any application that takes input has to do some sort of input validation. There are many reasons to do input validation which include to protect against hackers, to make sure that the user has entered the correct information and formatting. If input formatting is not used correctly it can lead to XXS and SQL Injection attacks. Next, we will look at APIs and how this affects this application. Within the project guidelines we are told that Artemis Financial has a RESTful web application programming interface. Since we know that we are using an API we know that this is an area of security that we will want to consider. If an API is implemented without using best practices it could leave an application open to numerous vulnerabilities. These include but are not limited to unauthorized access to objects in the software, authentication using illegitimate tokens, excessive data exposure and injection. This being said, having secure use of the API here is crucial. Next, we will look at the importance of Cryptography in this application. Here we know that we are storing and managing data within this application. When handling sensitive information encrypting the data helps manage who can see this data. This is huge because if a hacker gets into the system, but all of the data is encrypted, they will have a much harder time reading what is there. Next, we will be looking at the value of security in the client-server portion of the application. Most applications use the client-server relationship, and this application is no anomaly. Some vulnerabilities of this aspect include cross-side scripting, SQL injection and sensitive data exposure. Next, we will look at error handling which is a very big deal with almost any complex program. Handling errors incorrectly can leave an application open to a large number of attacks including fail-open security check and can even display detailed internal stack traces to the hacker. Next, we have code quality which is important for obvious reasons. Every application that will be deployed in commercial use should have good code quality or else security will be a big issue. A lack of code quality can leave software open to all of the vulnerabilities that I have discussed thus far and many more. Lastly, we will touch on encapsulation which is one of the main reasons that object-oriented way of developing came about. Encapsulation ensures that data cannot be accessed by any unauthorized user. Containing data to specific objects and not allowing any other object to access variables goes a very long way in terms of security. Using best encapsulation practices makes a hacker's job very hard. This is important to this specific application because of the sensitive information of the clients that will be in the system.

## 3. Manual Review

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

[Include your findings here.]

When walking through the vulnerability assessment diagram, the first branch that we are expected to check is the views. Looking at the code I can see that there are no html documents within the application but there is one working view. Within the GreetingController class there is mapping to a page that shows the id of a user and a hello world message. This is a security concern, and we just want to make sure as the application develops that we don’t show any sensitive data to the wrong users. Next, we are set to check the models for the database. The only class that is set up to be a part of the model is the Doc Data class that has a single attribute which is id. The class also has a method that pulls from a test database which I’m sure will be the actual database when the code is in production. This means that this class is a very important class when it comes to security. If a hacker was to get to this class, it would mean serious trouble for the privacy of Artemis Financial’s customers. Next, we want to look at the controller classes within the program. Here we can see two controller classes within the application named CRUDController and GreetingController. Within the CRUDController class there is access to the DocData class which makes this a very big security concern. The page that this controller maps to does not work, but I am sure that this will change in production and we should be mindful of the security implications that go along with that. In terms of data access, DocData is the only object that has any connection with the SQL database. This object is accesses in the CRUDController class which is the only object in the code that does this. These at the classes at the moment that have to be taken into consideration when looking at how a hacker will be able to access our database. In terms of services and API’s I do not see any within the application. There are no Interfaces nor are there any service classes that use any API’s here to my knowledge. We also are not using any added-on plug-ins within the application that I am aware of.

## 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

[Include your findings here.]

## After ruining the dependency check I got a few security vulnerabilities that should be looked at. First, [CVE-2013-1624](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624) which basically allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets. This has been fixed in a later release of the Bouncy Castle Crypto dependency. Next, we will be looking at the Apache Log4j AP. The code that I got was CVE-2020-9488 which could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through the appender. This can be solved by updating to version 2.13.2 or older versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections. Next, we will look at the YAML 1.1 parser dependency which I got a vulnerability code of [CVE-2017-18640](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) for. This allows for entity expansion during loading and can be fixed by updating to version 1.26 of the dependency. Next, we will look at Hibernate's Bean Validation that had a vulnerability code of [CVE-2020-10693](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693). This basically enables invalid EL expressions to be evaluated as if they were valid. One solution stated that You can pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext.

## 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.

[Include your findings here.]

Now that we have identified all of the security concerns within the application it’s time to come up with a plan as to how we will mitigate them. As we saw in the previous section, a lot of the security vulnerabilities that we saw in the report can be solved by using a later version of the dependency. So, the first thing that I would suggest that Artemis Financial do is make sure that all of the technologies that they are using are up to date. Next, we should make sure that all of the vulnerabilities that I identified when manually examining the code are handled. This means making sure that the program was made with best OOP programming practices. We also want to make sure that any class that has access to the database is secure and does not allow for access to the wrong eyes. This means making sure that all error messages do not show sensitive information like the stack trace. We also want to make sure that if the user is giving input at any time that it is processed and validated correctly. We also want to make sure that we validate anything that we will save to the database. Taking these steps should ensure that the application is secure, and their clients’ information is protected.