****

# 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** | **5/22/22** | **Zachary Meisner** |  |

## 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

Zachary Meisner

## 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?

The value of secure communications to the company is high. With the creation of a security culture throughout the entirety of the corporation, in addition to those involved in the development of the product, we can see a forward focus on security specifically, as they have said that through their desire to modernize operations that a crucial part of its success is implementing and applying the most current and effective types of security pertinent to their application. This is quite apparent through the decision to hire me as a developer, as my job specifically is to assure that the security is up to par, and to create this hazard and mitigation security report to aid in educating the company and helping aid the customers in being the safest they possibly can be while utilizing this application.

* Are there any international transactions that the company produces?

Though there is seemingly no direct example of an international transaction that the company produces, there is enough information here to believe that international transactions are something to be considered within the security and development of this application. The first indicator of this is that the company I work for, Global Rain, is an engineering company that specializes in custom software for businesses and individuals around the world. There would be no reason to hire a company that specializes in this sort of development if there had not been the thought that global or international transactions would be involved. Secondly, Artemis Financial is a consulting company that develops individualized financial plans. There is no note of any specific target audience or customer in this instance, and the relayed audience that we do have of “individuals” is a broad spectrum of potential customers, making me believe that they are open to anyone interested, which could include corporations as they are individual entities in the United States as well. Lastly, the company specifically wants us to help as they are concerned about external threats. This is less “proof” of any type of international transaction, as every company and business would be concerned about external threats, but again we specifically were hired to aid with this issue, and if we were to know how best to do something, Global Rain would be best at handling issues on a global level.

* Are there governmental restrictions about secure communications to consider?

Yes, some governmental restrictions about secure communications that we need to consider pertain to what is called OWASP Secure Coding Practices. The implementation of these guidelines mitigates the most common, or general types of vulnerabilities. A general overview of this involves Data Validation, Authentication and Password Management, Authorization and Access Management, Session Management, Sensitive Information Storage or Transmission, System Configuration Management, General Coding Practices, Database Security, File Management, and Memory Management. Though this seems like a lot, software development for this type of product is incredibly important, as we are asking clients to trust us with their sensitive data, and if leaked, could destroy their lives, and lose a lifetime of their savings. Each part of the OWASP guidelines is incredibly important to follow, which means it is best to strive for this security culture that Global Rain and Artemis Financial are pushing for, in addition to stressing education and available resources to help teach everyone on the development teams so we can all watch out for one another, in addition to understanding the greater picture of what needs to be done to assure these guidelines are properly implemented.

* What external threats might be present now and in the immediate future?

There is no definite way to say exactly what might pose to be a threat in the immediate future as software development is an ever-changing topic that is amorphous when it comes to the transformative properties it can display when in reference to security. The only instance in which we can hope to secure software applications is through consistent education on up-to-date issues, in addition to continuous education in the software development lifecycle. This starts as early as gathering requirements, because it is best to set yourself up for success by building secure software initially than to respond to the problems at a later date, if we are not following these basic coding guidelines and practices set out by the OWASP in addition to newer available resources online as applications may become out of date or required to be updated with relevant technologies. This may also require a potential redesign of the software or system in the case that any of the libraries or dependencies or coding practices that we use become less safe, or out of date with the application, meaning it is important to be always vigilant throughout a products deployment. Lastly,

*“According to the Sophos 2022 Threat Report, three of the biggest threats businesses can expect to see this year are ransomware, malware on mobile devices and attacks on internet infrastructure.”* (“Three Cyber Attacks Likely to Hit 2022”).

Considerably these are the three top threats that we can consider as the most prevalent issues present now. Knowing this information, we can use what we know about these vulnerabilities to best mitigate these issues, in addition to preparing ourselves with resources in the unfortunate case there is a breach of some kind that may put the company employees and customers at risk. This will help assure that we are doing everything that we can to assure the safety and security of the company to provide the best type of assets to promote and grow a healthy security culture.

* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

One example of modernization requirements that must be considered is the use-case of access control for our application. There is no true simple answer but to take the proper steps and to assure that the design and requirements of our project match up with the type of access control that is needed due to the heavy integration into our system and how it effects security in our software and how our application acts in deployment. One key point to note is that simplicity is key, as the less complex our project must be, the easier it is to use design patterns that help secure our design even further. This is due to the overall architecture of our design, and how the software interacts with itself during runtime, as each different part of this system, i.e., objects, users, and functions must be properly called at the correct time, and act accordingly with one another to avoid any potential oversights. Therefore, the earlier we take this into consideration the better. When it comes to open source libraries, this can be both a blessing and a curse, as we know that when considering access control, or the security layer of our application, each part has to interact with one another at some point, and one positive thing about open source libraries is that there is a lot of eyes on these libraries, meaning that there are more people looking and testing for vulnerabilities, making it easier to find or create mitigation techniques in the case of a problem or vulnerability, making our applications potentially more secure. This though, is a double-edged sword, as the use of open-source libraries may also not follow proper coding procedures that align with the project that we are looking to develop, in addition to the lack of control we have of the library itself, making it potentially more dangerous in the long term. Evolving web application technologies are not always made private, and a lot of the most common tools are utilized to accomplish tasks such as creating or deploying an application with a framework. The positive highlighted tends to take precedence as the more eyes on this means the more likely these technologies are to update and become fixed, which means one of the only issues we must worry about is the framework becoming outdated, or a potential vulnerability that is being exploited that there is not a fix for just yet, in addition to a potential replacement in the long term if something does become outdated.

## 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.

I think that the main areas of the VAPFD that are applicable to Artemis Financial’s software application are:

1. Cryptography
2. Code Error
3. Code Quality
4. Encapsulation

To start off with why I do not think that Input Validation, APIs, or Client/Server are pressing security issues, is mainly due to the pre-automated and open source libraries or frameworks at play. As I described prior, a lot of these tools are updated by the open source community, in addition to having many people searching for potential vulnerabilities, we also have many people looking to fix problems and vulnerabilities as well. Despite the importance of this regardless of that, this may mean that we have less to focus on in these areas past the initial decision-making process of our architecture, and the update or maintenance of the application during deployment making these two parts more manageable.

For the rest of the chosen areas of concern, I think that a lot of what I chose are considerably choices made by people that can vastly change the outcome of the application, meaning that we must account for our own developer’s education, and the implementation of the methods they choose to use, in addition to assuring they are following proper security practices. This is due to our software not being open source, making it harder to find or take care of vulnerabilities if there are some, in addition to making it harder to find and catch mistakes before they become an actual issue later down the road.

Cryptography is a pressing issue because of the various types of cryptography that are used, and the methods used to break these encryptions potentially being easily broken. I think that this is a constant concern because if we do not have a protected type of method used to encrypt our data during transportation in the network or framework, it won’t matter how strong our security is if we end up leaking data through an unknown vulnerability in the framework, or even if someone happens to hack into our system to gather sensitive information.

Code Error & Code Quality are concerns due to the finely tuned importance of Quality Assurance, and how to best create a culture that promotes Code Quality, in addition to running proper tests on this quality to assure there are no code errors that are made, in addition to no sloppy code being accepted and added to the created code base. This can come with loads of documentation, and code reviews as the proper steps and documentation need to be taken, and then reviewed again making these issues pressing as it is one of the most difficult and potentially tedious parts of creating a software application. Assuring that we have proper and knowledgeable people assigned to this task, in addition to getting developers on the same page with those that test or manage quality in an application is key.

Lastly, encapsulation is important due to the databases utilized to assure that the content sent or received is filtered properly, in addition to making sure that it is safe. This goes hand in hand with Cryptography, Input Validation, meaning that there may already be established support for commonly utilized methods such as pre-made databases such as SQL. These databases also have vulnerabilities that we need to account for, making it a difficult part of software development as we need to stay vigilant when it comes to the initial design, and implementation of these systems, how data is received, sent, and then delivered to the user, or potentially even updated in the system. This is an incredibly important part because of how integrated encapsulation and proper database implementation can be overall for an application.

## 3. Manual Review

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

myDateTime.java: the variables are not private, in addition to no getter or setter methods used to safely retrieve or store data in addition to no singleton class.

Greeting.java: There are no filters in the get functions meaning that the stored data could be anything that fits the long or String variables which may break the program if the input is not validated correctly.

DocData.java: The getId method is different than the established method in greeting, making it possible for a collision due to the redundancy in the code. The id; variable in this class also is just private, and the one in Greeting.java is private final, which is an inconsistency that reduces overall code quality. Lastly, if we were to assume that read\_document is supposed to use the constructor in Greeting.java, again, the two variables do not match, as the String key, in the function is implemented as a long in Greeting, and then converted to string using the prior getId function within DocData, which doesn’t make any sense and is not consistent with the constructor.

Customer.java: The account number is private int when it should be private string. As an int maybe bad to use if any functions are implemented on this account number, making it harder to keep track of the ID method. In addition to this, account\_balance is not private, and neither variable are finalized after any transaction potentially making them unprotected, as the finalization of these variables in addition to any potential encryption and storage would be absolutely necessary for a proper filter to be implemented. Lastly, it looks as if there is a variable that is created within a function of deposit (int a) which is not great, as all variables should be created together at the top so everything can be accounted for, in addition to this variable not being private as well, we are using an int to just add the numbers to account balance, which is not properly coded to account for non-whole numbers, making it easy to break the program as int is not great for any financial use case. There should be some type of filter implemented here to assure the proper methods are used to the retrieved data.

CRUDController.java: In CRUDController, the only complaint I have is the call of DocData, which had been an empty function created in DocData that does not do anything. I did not see that there was any established constructor for DocData, and to return doc.toString to this function that does absolutely nothing and is not protected in any way is not proper code quality and may result in an error.

CRUD.java: content and content2 need to be more specific and further distinguished from one another to highlight the actual information that is being sent through this application. The names being this close together make it easy for errors to occur in the code in the case a developer mistypes one word, in addition to the same CRUD method being overloaded with one of the content identifiers, content1, not even being initialized in the beginning of the program with the other variables, and then having invocation with the wrong content, as it’s invoked as content and content2, when it should have been invoked as content1 content 2. Lastly, there are still no setter methods meaning that there would be no secure way to establish the retrieval of data within the data structure.

RestServiceApplicationTests.java: There are no tests which is not good, this needs to be done or we have no way of knowing if the application will work the way it is supposed to work.

Documentation: There was little to no documentation provided in the application explaining any of what was happening or why it was decided to be coded as such. This is sloppy and reduced the code quality of the application and makes it harder to maintain in the long term if different developers are put on the application.

## 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:

Graphical user interface, text, application, email

Description automatically generated

**Bouncy Castle** V.1.46 has the following 12 vulnerabilities and the solution is to update to V.1.55 or higher.  
CVE-2016-1000352, 346, 345, 344, 343, 342, 341, 339, 338  
CVE-2018-5382, CVE-2017-13098, CVE-2013-1624

**Hibernate Validator** V.6.0.18 has the following vulnerability with the solution being to update to V.6.0.20  
CVE-2020-10693  
  
**Jackson Databind** V.2.10.2 has the following vulnerabilities with the solution being to update to V.2.13.2.2  
CVE-2020-36518, 25649  
  
**log4j-api** V.2.12.12 has the following vulnerabilities with the solution being to limit JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2  
CVE-2021-44832, 45105, 45046, 44228  
CVE-2020-9488  
  
**logback-core** V.1.2.32 has the following vulnerability with the solution being to upgrade to 1.3.0-alpha12 and higher  
CVE-2021-42550

**Snakeyam**l V.1.252 has the following vulnerability with the solution being to upgrade to 1.28-1  
CVE-2017-18640  
**spring-aop** V.5.2.32, **spring-boot** V.2.2.42, and **spring-core** V.5.2.32 have the following vulnerabilities with the solutions being  
CVE-2022-22950 – to upgrade to 5.3.17+ and 5.2.x users to upgrade to 5.2.20+  
For the following: 5.3.x users upgrade to 5.3.19+ or 5.2.x users upgrade 5.2.21+  
CVE-2022-22968, 22965,   
CVE-2021-22060, 22096, 22118,   
CVE-2020-5421  
CVE-2016-1000027  
CVE-2022-27772 upgrade to 2.2.11+ and only affects products or versions no longer supported by the maintainer.  
  
**tomcat-embed-core** V.9.0.302 **tomcat-embed-websocket** V.9.0.302 and have the following vulnerabilities with the solution being to switch to an alternate solution such as running clustering communication over a VPN, or to upgrade to a version above 10.1.0-M1 or 10.1.0-M14  
CVE-2022-29885  
CVE-2021-41079, 33037, 30640, 25329, 25122, 24122  
CVE-2020-17527, 13943, 13935, 13934, 8022, 11996, 9484, 1938, 1935  
CVE-2019-17569

## 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.

After interpreting my results from the manual review, the suggested steps to remedy these vulnerabilities is to update the code referencing what I have outlined earlier in the report, and to implement the methodologies needed in addition to cleaning up the code and writing proper documentation to allow future developers to understand exactly what is happening when the application is running. This type of documentation will help the developers outline what is happening during the application which will help them code in a better way because they will have to go through the step by step of the architecture. In addition to this, committing to creating and running proper tests will assure that the methodologies are in fact implemented in the correct way.  
  
As for the static testing, it seems the most pertinent actions that need to happen are the updating of the dependencies within the application, as most of these vulnerabilities can be easily mitigated through upgrading the application and assuring that the proper dependencies are taken care of throughout the maintenance part of the development lifecycle. The one vulnerability that maybe an issue is tomcat-embed as one of the only mitigation solutions was running clustering communication over a VPN, other than upgrading in the case these upgrades are or become available.

References

‌“Three Cyber Attacks Likely to Hit 2022.” *Washington Post*, 8 Mar. 2022, www.washingtonpost.com/creativegroup/sophos/three-cyber-attacks-likely-to-hit-2022/.

‌