****

# Artemis Financial Vulnerability Assessment Report

**Table of Contents**

[Document Revision History 3](#_heading=h.gjdgxs)

[Client 3](#_heading=h.30j0zll)

[Instructions 3](#_heading=h.1fob9te)

[Developer 4](#_heading=h.3znysh7)

[1. Interpreting Client Needs 4](#_heading=h.2et92p0)

[2. Areas of Security 4](#_heading=h.tyjcwt)

[3. Manual Review 4](#_heading=h.3dy6vkm)

[4. Static Testing 4](#_heading=h.1t3h5sf)

[5. Mitigation Plan 4](#_heading=h.4d34og8)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/23/2024** | **Vy Huynh** | **Vulnerability Assessment** |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer - Vy Huynh

## Interpreting Client Needs

The client, Artemis Financial, is hoping to seek advice from Global Rain, to find ways to modernize and improve the organization's security and protect their data and information from outside threats.

For a finance company like Artemis Financial, the importance of secure communication cannot be disregarded. The confidentiality of customer data is extremely important, with important information like savings, and investment plans as part of their databases it’s important to protect the information to prevent breaches of trust from customers and avoid potential lawsuits. Secure communications make sure that client information and personal details are well protected will help with gaining customers’ trust and preventing lawsuits from other customers for data breaches, furthermore, secure communication could bring many potential customers as the customer understands how much Artemis Financial values their customer and their security privacy.

As a financial firm, they likely are involved in international transactions, which means they need to comply with international law and make sure to adhere to data procession laws for the country they are interacting with. Furthermore, since they are working with an international firm, it’s important to make sure the international firm takes high priority in data protection and security. A financial company needs to prioritize client security and if only one side is prioritizing the client's security, then it’s not an ideal situation. Another thing Artemis needs to do is focus on regularly performing audits to make sure their security measure is up to standard.

When working internationally it’s extremely important to follow government law, an example is if Artemis is working with a country that is a part of the European Union, they will have to follow the General Data Protection Regulation, GDPR, which is a set of laws in the EU which regulate handling of personal data. Artemis will have to follow the GDPR to stay within EU regulations and prevent potential lawsuits.

For any company the external potential threat is always there, to protect customer’s information/data it’s important to stay vigilant and create a migration plan for potential threats. Some of the most common threats are ransomware or phishing attacks. Ransomware is a hacker that uses malicious software to compromise the system, they will then request payment before the data is released. A phishing attack is a type of social engineering where hackers trick employees into releasing sensitive information or allowing the hacker to gain unauthorized access to the database. Another external threat that Artemis needs to be aware of is the potential vulnerabilities introduced by new technology like cloud computing. Since Artemis wants to modernize its system, it needs to be aware that when modernizing something, there will be new potential risks introduced as part of the modernizing process associated with things like cloud computing.

The most important part of modernizing a system is the usage of cloud computing, with the increase in cloud platforms like Azure or AWS. The cloud platform is a must-have for modernizing, cloud computing is considered scalability, and affordable which is perfect for Artemis to use as data storage and hosting. Using an open-source library is extremely useful since it can help reduce development time and enhance the system's quality. Another important requirement that needs to be considered is security, it’s important to integrate security measures the mitigate any vulnerability during the development lifecycle. These features help make sure that Artemis is modernizing efficiently and effectively while also prioritizing customer security.

## Areas of Security

The four areas of security that I would focus on are APIs, Cryptography, Client/Server, and Code Error.

APIs - Artemis needs to have secure APIs because RESTful APIs will interact with various things, therefore ensuring that the API is secure will protect against unauthorized data access and breaches.

Cryptography - Cryptography, is essential for a company handling customer’s finances, having cryptography like encryption will ensure that customer data like their investment is confidential, this help prevent data breaches.

Client/Server - Implementing client and server interaction helps maintain the security and confidentiality of sensitive data during transmission. This means having secure communication protocols, two-step authentication mechanisms, and data protection.

Code Error - Secure code is extremely important as it helps maintain the security of the API, this could include handling unexpected requests without exposing any sensitive information that could be exploited by hackers.

## Manual Review

*GreetingsController.java*

The value ‘name’ parameter (line 16) is an input without an input validation, this means it could be vulnerable to SQL injection. There appears to be a lack of error handling which is needed.

*CRUDController.java*

The RequestParam ‘business\_name’ (line 13) does not have an input validation, which means that it could be vulnerable to an injection attack, there is also no error handling which is needed for an explicit error.

*DocData.java*

The first thing that is immediately a problem for the class is that the database credential location is hard-coded, which is a vulnerability and a security risk. The connection con uses ‘try’ but it’s missing a closing ‘finally’ this could lead to leaked information since it’s not closed properly. The ‘read\_document’ method (line 21) has a comment saying, “implement read method” However, there’s nothing implemented, and the code is incomplete. Furthermore, the con details are hard coded (line 27), which is not a good or secure method to store credentials.

customer.java

The ‘account\_balance’ (line 5) should be a private modifier to protect the customer’s balance and prevent other classes from being able to access the information. The deposit method (line 12) should validate the amount and make sure it’s greater than 0, since there is no validation there will be an unexpected error if a negative number is passed into the method. The ‘account\_balance’ should be more regulated and there should be controlled access to the ‘acount\_balance.’ The class name should be Customer.java instead of customer.java to follow naming conventions.

CRUD.java

Since the class has a getter method it might be useful to also have a setter method, the first constructor (line 7) is a bit redundant since initializes both with the same value. The naming convention is bad, calling content1 and gets confusing, consider renaming it to a different name.

myDateTime.java

myDateTime is not Java naming convention, it should be named MyDateTime instead. Both the retreiveDateTime(), (line 9), and setMyDateTime(), (line 14), methods are incomplete, which means it’s not doing what it’s supposed to be doing or the method is just empty. The modifier access int mySecond, myMinute, and myHour should be private instead of public (line 5-7).

## Static Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Vulnerability Code:** | **Description** | **Attribution/ Mitigation** |
| **bcprov-jdk15on-1.46.jar** | cpe:2.3:a:bouncycastle:  legion-of-the-bouncy-castle-java-  crytography-api:1.46:\*:\*:\*:\*:\*:\*:\* | “The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.” | The Bouncy Castle is vulnerable to injection error, timing attack, and potential file check vulnerability. The best way to mitigate the dependency is to update to a newer version. |
| **spring-boot-2.2.4.RELEASE.jar** | cpe:2.3:a:vmware:spring\_boot:  2.2.4:release:\*:\*:\*:\*:\*:\* | “Spring Boot” | The spring boot is vulnerable to temporary directory hijacking. However, it will only affect the version that is no longer supported by Spring-Boot so updating will prevent the vulnerability. |
| **snakeyaml-1.25.jar** | cpe:2.3:a:snakeyaml\_project:  snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:yaml\_project:  yaml:1.25:\*:\*:\*:\*:\*:\*:\* | "YAML 1.1 parser and emitter for Java” | : “Allows entity expansion during load operations.” This can be mitigated by updating it. |
| **tomcat-embed-core-9.0.30.jar** | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:  apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | “Core Tomcat implementation” | HTTP smuggling vulnerability that is also a potential DDoS attack. Update Tomcat to the latest version. |
| **spring-web-5.2.3.RELEASE.jar** | cpe:2.3:a:pivotal\_software:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | “Spring Web” | “Suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data.” Update Spring Framework. |
| **spring-expression-5.2.3.RELEASE.jar** | cpe:2.3:a:pivotal\_software:  spring\_framework:5.2.3:  release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:  spring\_framework:5.2.3:  release:\*:\*:\*:\*:\*:\* | **:** “Spring Expression Language (SpEL)” | SpeEL expression that causes a Ddos attack. Mitigate this by upgrading the latest dependency version. |
| **spring-context-5.2.3.RELEASE.jar**  . | cpe:2.3:a:pivotal\_software:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | “Spring Context” | “Field is not effectively protected unless it is listed with both upper and lower case for the first character of the field.” Upgrade the two latest version. |
| **spring-beans-5.2.3.RELEASE.jar** | cpe:2.3:a:pivotal\_software:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | : “Spring Beans” | “remote code execution (RCE) via data binding. The specific exploit requires Tomcat as WAR development.” This dependency should be upgraded to latest release. |
| **logback-core-1.2.3.jar** | cpe:2.3:a:qos:logback:  1.2.3:\*:\*:\*:\*:\*:\*:\* | “logback-core module” | Allow for a hacker to “ craft malicious configuration allowing to execute arbitrary code from LDAP servers.” Update the log to address the issue. |
| **log4j-api-2.12.1.jar** | cpe:2.3:a:apache:log4j:  2.12.1:\*:\*:\*:\*:\*:\*:\* | “The Apache Log4j API” | “Allow for control log message or log message parameters.” Update to version 2.16.0 or later where is feature is removed. |
| **jackson-databind-2.10.2.jar** | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* | “General data-binding functionality for Jackson: works on core streaming API” | There is a vulnerability in the dependency where the hackers is allowing the perform a DDoS attack. To prevent this, update to the latest dependent version. |
| **hibernate-validator-6.0.18.Final.jar** | cpe:2.3:a:redhat:  hibernate\_validator:6.0.18:\*:\*:\*:\*: | “Hibernate's Bean Validation (JSR-380) reference implementation.” | There is a bug in the system in which “invalid EL is evaluated as they were valid.” To mitigate this input validation issue, update the dependency to the latest version. |
| **spring-webmvc-5.2.3.RELEASE.jar** | cpe:2.3:a:pivotal\_software:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:  spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | “Spring Web MVC” | Is vulnerable to log injection, update spring framework. |

## Mitigation Plan

For the dependency that was found in the Maven Check, a lot of the vulnerabilities and issues can be solved by updating the dependency to a newer version. After updating all the dependencies, it's important to set up a system that can help report unusual activities, this can help alert if anything unusual is occurring, and potentially help with damage control / quick responses.

Breaking down the classes, in GreetingsController and CRUDController, due to a lack of input validation both classes are vulnerable to injection attack therefore both classes should implement input validation to prevent the attack. In CRUDController there also is an absence of error handling, implement error handling to make sure it’s able to handle any unexpected errors.

In DocData, instead of hard coding the database credentials, which is a security vulnerability. It’s better to instead build a method and use variables to store and get credentials. The “finally” block is missing in the try method so it’s important to add that to prevent any potential leaked data. Finally, the “read\_document” method is incomplete so it’s important to complete the method to do what is asked.

In customer class, it’s important to rename the class to follow the Java naming convention, instead of customer.java it should be Customer.java. The “account\_balance” modifier should be private instead of public to protect customer’s information. For the “deposit” method add a validation to make sure it’s over zero since it’s not possible to deposit a negative amount. While it’s technically not a full error, the potential for the “showInfo” method, maybe show more than just the customer’s account number with no context, return customer’s account number and their balance too.

In CRUD.java class, the first constructor should be removed because it’s redundant, it’s initializing both content with the same value. Another thing is that the naming for the class is not that good calling content1 and content2 is confusing and gives no context as to what is content1 and content2, consider renaming them to a better name for better interpretation.

Finally, in the class, myDateTime.java should be named MyDateTime.java to follow the naming convention. Both the method retrieveDateTime and setMyDateTime is empty and incomplete so it’s important to complete the method to implement the expected function. Lastly, all the modifiers like mySecond, myMinute, and myHour should be private instead of public.