

**Artemis Financial Vulnerability Assessment Report**

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **09/24/2023** | **Ilir Tagani** |  |

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

Ilir Tagani

* Interpreting Client Needs

a. Value of secure communications: Artemis Financial places a really high value on secure communications. Given this information their business involves handling sensitive financial information and develops individualized financial plans. Secure communications are crucial to protect customer data, which will in turn ensure trust between clients and will also help with compliance in-regards to privacy regulations.

b. International transactions: Artemis Financial does engage in international transactions, which involves cross-border data transferring and also interactions with clients and various financial institution in various countries. The international transactions bring an entirely new level of security threats that are related to data protection and secure communications.

c. Government Restrictions: There will be governmental restrictions related to secure communications, especially when Artemis is conducting international financial transactions. Compliance with data protection laws, and encryption standards that are imposed by governments are essential in avoiding potential legal issues and will help ensure that the data is secure.

d. External Threats: Artemis does face a numerous amount of external threats such as:

* Cyberattacks: Such as ransomware, data breaches which can expose sensitive client data and financial information.
* Regulatory and Compliance Risk: Failure to comply by Artemis with the financial regulation body can lead to legal setbacks as well as financial windfalls.
* Financial Fraud: Attempts can manipulate financial data or transactions for potential fraudulent purposes.

e. Modernization Requirements: In order to address modernization requirements, Artemis should consider Open-Source Libraries. This will leverage the libraries to accelerate development, but will require non-stop monitoring for potential vulnerabilities and timely updates that will help avoid security risks. Also the using up to date web application technologies will be essential to ensure the application's security and compatibility will continue with modern practices as well as standards.

* Areas of Security

Relevant Areas of Security:

1. Input Validation: The reason I think input validation is crucial to ensure the prevention of malicious input from users and potential vulnerabilities such as SQL injection or cross-site scripting. Given that Artemis handles highly sensitive financial data, ensuring the integrity and security of user input is essential in protecting against potential data manipulation.

2. Architecture Review: Architecture review will help to ensure the overall structure as well as the design of the application. For Artemis Financial, a well designed architecture will help to enhance security by ensuring secure communication channels, and will isolate critical components.

3.Code Review: Code review is also an essential way to identify potential vulnerabilities in the software's source code. It will help to uncover potential issues related to security, such as poor coding practices and coding errors.

4. Secure Input and Representations: Secure input and data representations are extremely necessary to safeguard sensitive financial information. Encoding of data, which should include user inputs are critical in preventing security vulnerabilities like injection attacks and data leakage.

5. APIs: Artemis Financial's web app will be interacting with external APIs, other financial institutions, and data providers. Ensuring secure API interactions is critical to prevent potential data breaches, unauthorized access, and financial fraud.

6. Cryptography: Will play a significant role in protecting sensitive financial data, which includes encryption of data at rest and as well as in transit. Artemis should use strong encryption algorithm's as a detrrence.

* Manual Review

1. CRUD.java: What I found here was insecure initialization, the 'public CRUD(String content)', both of the 'content' and 'content2' are initialized with the same 'content' parameter, which means that they will have the same value. Also 'public CRUD(String content1, String content2)' are defined but not used anywhere in the code itself. This should be removed to help improve the code itself.

2. customer.java: I found there to be an incomplete method 'showInfo()' which is declared as an integer, but there is no actual implementation provided to show customer information. This method being incomplete and does not performing its intended action, should be completed to provide a meaningful functionality.

3.DocData.java: What I found here is a database connection issue. The code includes a database connection credentials directly from the source code. Storing credentials in source code is a major security vulnerability because it expose sensitive information which can be easily accessed. This must be addressed immediately!

* Static Testing

When I ran the dependency check, I was given 13 vulnerabilities:

1. Code: bcprov-jdk15on-1.46.jar

Risk: High

Description and Solution: This library has a high-severity vulnerability. Updating the library

to a version that will address the vulnerability.

Attribution: Support for this mode has been deemed unsafe and has been removed by the

provider.

2. Code: spring-boot-2.2.4.RELEASE.jar

Risk: Critical

Description and Solution: This library has a high-severity vulnerability. Updating the library

to a version that will address the vulnerability.

Attribution: None

3. Code: logback-core-1.2.3.jar

Risk: Medium

Description and Solution: This is library has a medium vulnerability. A attacker with

the required privileges to edit configurations files could craft a malicious

configuration allowing to execute arbitrary code loaded from LDAP servers. Update

the library to address the vulnerability. Also verifying that the updated version does

not cause compatibility issues.

4. log4j-api-2.12.1.jar

Risk: Critical

Description and Solution: Vulnerability to a remote code execution (RCE). To fix this issue

limiting JNDI data source names to the java protocol.

5. snakeyaml-1.25.jar

Risk: Critical

Description and Solution: Parsing malicious or large YAML documents can consume

excessive amounts of CPU or memory. Updating library to a version that addresses

the vulnerability.

6. jackson-databind-2.10.2.jar

Risk: High

Description and Solution: This flaw allows vulnerability to XML external entity (XXE)

attacks.

7. tomcat-embed-core-9.0.30.jar

Risk: Critical

Description and Solution: This would cause unwanted URL redirection to a none trusted

site.

8. hibernate-validator-6.0.18.Final.jar

Risk: Medium

Description and Solution: This is bug in the message interpolation processor which enables

invalid EL expressions.

9.spring-web-5.2.3.RELEASE.jar

Risk: High

Description and Solution: N/A

10. spring-beans-5.2.3.RELEASE.jar

Risk: High

Description and Solution: N/A

11. spring-webmvc-5.2.3.RELEASE.jar

Risk: Medium

Description and Solution: N/A

12. spring-context-5.2.3.RELEASE.jar

Risk: Medium

Description and Solution: N/A

13. spring-expression-5.2.3.RELEASE.jar

Risk: Medium

Description and Solution: N/A

* Mitigation Plan

When it comes to a plan to mitigate the vulnerabilities this is the course of action I would take:

Manuel Review

1. CRUD.java:

Insecure initialization of content and content2 parameters in the

constructors, leading to both having the same value. To mitigate remove the unused

constructor public CRUD(String content1, String content2) and ensure distinct

initialization of content and content2 in the existing constructor.

2. customer.java:

Incomplete implementation of the showInfo() method, which lacks functionality.

To mitgate a complete implemenatation of the 'showInfo()' method would provide

the intended customer information that displays functionaility.

3. DocData.java:

Here sensitive database connection credentials are hardcoded into the source code. To

mitigate credentials should be in a configuration file with restricted access. Also removing

hardcoded credentials from the source code would be a great step as well.

Static Testing:

Something that I found when it came to all the vulnerabtilites associsated with the dependency check was to update all the libraries to a version that would address all vulnerabilities first. After the update is done to the libraries verifiying that there are no compatibilitiy issues within the application itself.