****

# 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 5](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan 6](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **03/22/2024** | **Mitchel Dauk** | **Security 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

Mitchel Dauk

## Interpreting Client Needs

Artemis Financial is a company that works with financial plans for their customers. Since there is not a current security plan in place, Artemis Financial has reached out for a vulnerability report and a security plan to keep their information secure. The value of secure communications is high for Artemis Financial because they are taking the initiative to reach out and make their company better. Since the company deals with financial plans, it is crucial that the confidential information stays confidential.

It is safe to say that Artemis Financial deals with international transactions, because it is not stated that they only deal in the United States. With investment and insurance, secure connections must be made to all clients. As with all forms of vulnerabilities, there are laws that cause governmental restrictions. For example, the “Gramm-Leach-Bliley Act” requires financial institutions to design, apply, and maintain a security program that safeguards the information (IT Governance 2024). External threats that may be present and in the immediate future are subject to DDoS attacks, man-in-the-middle attacks, brute force attacks, malware, ransomware, code injection, crypto jacking, and exploits.

The modernization requirements that must be considered are correct and secure code that works with an API. There must also be a team to secure and test the program regularly to search for any vulnerabilities that may pop up. With new technology comes new types of attacks, which is why updates to the system are extremely important.

## Areas of Security

Input Validation – User input can allow an attacker to enter harmful code that can cause issues with the program. Since Artemis Financial uses an API, they will need secure input validation. Without secure input, attackers can also find more vulnerabilities by testing the application with different types of code.

API – The data that is transferred to the interface is also liable to a man-in-the-middle attack and can potentially disrupt the service or cause a breach. The data transferred must be secure and from a legitimate source.

Cryptography – Cryptography is widely used in financial institutions, because of the confidential information. The messages are hidden and can only be seen or unlocked by the person who is supposed to have access to it. Artemis Financial should be using cryptography when sending and receiving information.

Code Error – Code error should be taken seriously when shown up in the code. Usually, the program will react differently and show there is a warning or error. This could be a potential opening for an attack.

Code Quality – It isn’t the quantity of the code, but the quality that makes code secure. Mistakes in the code can cause vulnerabilities, which can lead to an opening for an attacker. Code quality is extremely important when dealing with confidential information.

## Manual Review

Input Validation – No scanner for user input found. This could allow malicious code to be inputted and tested for vulnerabilities by the attacker.

API – There is a wrong version in the dependency check, which does not give the most up to date report. Also, the compiler compliance is specified wrong.

Cryptography – “xmlns” uses “http” and not “https” as a reference. This can cause an insecure connection. Classes are all public, when some or all should be private or closed.

Code Error – variable “con” is not used correctly in DocData. It needs to be referenced in a different or same class.

Code Quality – There are no JREs installed that are compatible. All necessary installments should be included.

## Static Testing



[bcprov-jdk15on-1.46.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) – 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.

Doesn’t fully validate encoding of signature on verification, generates weak private key, earlier versions unsafe, timing attack, padding issues, uncontrolled resource consumption, easier to obtain private keys, risky cryptographic algorithm, and can cause invalid keys.

[hibernate-validator-6.0.18.Final.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492) – Hibernate's Bean Validation (JSR-380) reference implementation.

A bug in message processor enables invalid expressions to be treated as valid.

[jackson-databind-2.10.2.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) – General data-binding functionality for Jackson: works on core streaming API.

Vulnerable to data integrity, allows DoS, and resource exhaustion.

[log4j-api-2.12.1.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb) – The Apache Log4j API.

Possible intercepted messages by the SMTPS connection.

[logback-core-1.2.3.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c) – logback-core module.

DoS attack and allowing malicious configurations.

[snakeyaml-1.25.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) – YAML 1.1 parser and emitter for Java.

Improper input validation, improper restrictions, and DoS attacks.

[spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714) – Spring Boot.

Susceptible to security bypass, temporary directory hijacking, and DoS attacks.

[spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l16_ec75d01d212b5229c16d872fb127744c0ed46ed8) – Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.

Susceptible to security bypass, temporary directory hijacking, and DoS attacks.

[spring-core-5.2.3.RELEASE.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b) – Spring Core.

Vulnerable to RCE, privilege escalation, bypassed RFD attacks, DoS attacks, case sensitivity handling, and insertion of additional log entries.

[spring-web-5.2.3.RELEASE.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l18_dd386a02e40b915ab400a3bf9f586d2dc4c0852c) – Spring Web.

Vulnerable to RCE, privilege escalation, bypassed RFD attacks, DoS attacks, case sensitivity handling, and insertion of additional log entries.

[spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l19_745a62502023d2496b565b7fe102bb1ee229d6b7) – Spring Web MVC.

Vulnerable to RCE, privilege escalation, bypassed RFD attacks, DoS attacks, case sensitivity handling, and insertion of additional log entries.

[tomcat-embed-core-9.0.30.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l20_ad32909314fe2ba02cec036434c0addd19bcc580) – Core Tomcat implementation.

AJP connection trust, high CPU usage, DoS attack, unauthorized actor, endless loop, inconsistent interpretations, improper encoding, cross-site scripting, and reverse proxy.

[tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\slipk\ProjectOneSS\rest-service\target\dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d) – Core Tomcat implementation.

AJP connection trust, high CPU usage, DoS attack, unauthorized actor, endless loop, inconsistent interpretations, improper encoding, cross-site scripting, and reverse proxy.

## Mitigation Plan

Input Validation – Causes a man-in-the-middle attack and could leak log messages.

Only communicate confidential information if you have to. Also, use a secure connection.

DoS – Denial of Service. Causes disruption in the service.

Set limits and trackers on what uses data. If there is an abnormality, allow it to send an alert.

Security Bypass – Security features disabled to allow an attacker to access content or settings. Send alerts when changes are made to the system.

Temporary Hijacking – Allows short periods of hijacking that may not be caught right away.

Always keep and check logs. Send alerts for suspicious logs or sign-ins.

Privilege Escalation – Lower-level privilege users can access content they are not supposed to.

Only allow least privilege. Do not allow any users to have “too” much access.

Extra Log Insertions – Causes chaos and improper logs. Can bog the system up.

Only allow a certain number of logs. If there is a large amount at once, allow alerts to be sent.

Man-in-the-Middle – Someone infiltrating a network to see what information is passed back and forth.

Always use a secure connection and check who is using the connection.

Remote Code Execution – An attacker can control the system remotely.

Multi-Factor Authentication is a great method to allow remote users to verify who they are. Also, send alerts to logins from new places.

Entity Expansion – Exhausts system resources to cause DoS. Set alerts as mentioned above.

Malicious Input – Allows malicious programs to be inputted into the system and behave a certain way. Do not allow users to input anything they want. Set boundaries.

Secure HTTPS – Use secure methods for web applications. Stops attackers from intercepting data.

Cryptography – Ensure that secure connections between two sources are using a private key and causes no interruption.

Updates – Ensure that current versions and updates are happening to the program. This will keep up with new and improved attacks.

Reverse Proxy – IP address is never revealed and keeps the location safe, similar to a VPN.

Case Sensitive Handling – Allow only certain inputs and reject anything that does not follow the plan. Don’t look for bad code, but block out anything that doesn’t fit the criteria.

References

*Federal Cybersecurity and Data Privacy Laws Directory*. IT Governance. (n.d.). https://www.itgovernanceusa.com/federal-cybersecurity-and-privacy-laws