****

# 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** | **[Date]** | **[Christopher Ivey]** |  |

## 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

[Christopher Ivey]

## 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?

Secure communications are valuable to Artemis Financials’ software application. Since the software is using RESTful APIs, the web application will need secure verification of input. Code quality is a superset of input validation and APIs. To provide secure coding for this web application, it is very important to not trust anything from the request and only to submit sensitive data over HTTPS POST in the body of the POST.(2022 O'REILLY MEDIA, INC) Web applications are prone to attacks from hackers by adding strings to legal input fields, cookies, and other parts of the HTTP request. SQL injection is among the highest attacks on a software application. Any type of banking payments or banking transaction type of software applications can be at risk to critical patches in the software.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financials’ software application. Justify your reasoning for why each area is relevant to the software application.

Input Validation: Secure Input validation and input representations should be the first line of defense for every web application. Making it a top priority to access for the Artemis Financial software application. The software application needs to be developed with authentication and authorization checks for each request.

APIs and secure API Interactions between the software application and the server is important. Since most communication between client and server should be private since we often send and receive private information. Therefore, using SSL/TLS for security is a must.

Cryptography, Encryption use and vulnerabilities are relevant to the software application because there is sensitive financial information that is being distributed between the server and the users of the account. This information needs to be encrypted to provide protection for the user against hackers.

Client / Server and Secure Distribution Composing this area is a relevant area of security for the software application because one of the most dangerous attacks against a web application is Cross Site Request Forgery. All sensitive transactions must be protected with a synchronizer token or similar.

Secure Code Error Handling Ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software. No matter how well written your web application is, sooner or later you will have to deal with an exception (2022 O'REILLY MEDIA, INC). When Java web applications error in unrecoverable ways, it is not uncommon to see an exception stack trace in your browser. Graphical user interface, text, application

Description automatically generated

Not only does the default server error message shown in the screenshot reveal the fact that you are using Java, the libraries in use, and the exact (vulnerable) version of Apache Tomcat the application is running on, it also lets an attacker know that you quite likely have an SQL injection vulnerability! On the bright side, there is a solution called Query parameterization that is the most powerful technique available, in any web language, to completely stop SQL injection—forever.

Code Quality are all areas of Security that applicable to Artemis Financials’ software application.

## 3. Manual Review

Graphical user interface, text, application, Word

Description automatically generated

For the customers information in the code, I would recommend using algorithms and data structures to ensure that the information is safe and secure. Graphical user interface, text, application

Description automatically generated

There is maybe a wrapper here that can be added for secure input validation of the customer information. Also, the read document should be private so no unauthorized access should be able to access sensitive information.

## 4. Static Testing

Run a dependency check on Artemis Financials’ 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

Graphical user interface, text, application

Description automatically generated

The picture above, displays the dependencies with their codes.

1. bcprov-jdk15on-1.46.jar : 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. Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.
2. CVE-2020-10693: A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.
3. CVE-2020-25649: A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
4. CVE-2020-9488 : Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.(Apache Software Foundation , 2020) CVE-2017-18640
5. CVE-2017-18640 : The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
6. CVE-2020-5421: In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. One solution for this vulnerability is by using a Critical Patch Update, which is a collection of patches for multiple security vulnerabilities.

## 5. Mitigation Plan

To encrypt the sensitive information from the Bouncy castle crypto package, the developer could use hashing tables. Use an adaptive hash function that can be configured to change the amount of computational effort needed to compute the hash, such as the number of iterations ("stretching") or the amount of memory required. Also, to mitigate against the input validation vulnerability, the software needs to enforce the principle of last privilege, by adding role checks either for a single role, or have more granular roles for each user. Also, creating the system to where a user will not be able to access another user’s account information. To mitigate the host mismatch to Apache Log4j SMTP appender, upgrade to 2.13.2 which supports this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections.

Input Validation: Validate inputs: Define wrappers around native methods.

•Cryptography: Secure data transfer; need to implement HTTPS protocol

•Client/Server: Denial of Service: Resource limit checks should not suffer from integer overflow.

•Code Error: Error handling and logging: Ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software.

•Code Quality: Memory Management: Double check that the buffer is as large as specified and check buffer boundaries if calling the function in a loop. Make sure there is no danger of writing past the allocated space.

References

Katrina Tsipenyuk, Brian Chess and Gary McGraw. “Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors”. NIST Workshop on Software Security Assurance Tools Techniques and Metrics. NIST. 2005-11-07.

[REF-18] Secure Software, Inc.. "The CLASP Application Security Process". 2005.<https://cwe.mitre.org/documents/sources/TheCLASPApplicationSecurityProcess.pdf>.

[REF-461] Matthias Kaiser. "Exploiting Deserialization Vulnerabilities in Java". 2015-10-28.<http://www.slideshare.net/codewhitesec/exploiting-deserialization-vulnerabilities-in-java-54707478>.