

**CS 305 Project One**

**Artemis Financial Vulnerability Assessment Report**

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **11/11/2021** | **Anh Tran** |  |

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

Anh Tran

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?

[Include your findings here.]

The company that I'm working with is Artemis Financial. This is a financial firm where they will develop an individualized financial plans for savings, retirement, investment, and insurance for their patrons. With that being said, this company hold many valuable information including clients' financial and personal information. The secure communication play a huge role in the success of this company. Without secure communication, hackers can exploit clients' data and company's strategic plan to help client grow. This can negatively impact the company and damage its reputation. Since it is a financial firm, company is working with people all over the world so there will be international transactions. There is currently no governmental restrictions about secure communications to take into consideration.

Since it's the financial firm, there are too many threats that can hugely damage the company now and in the future. The company will need to stay up to date with their open-source libraries and evolve their technologies in order to stay secure and prevent any exploits.

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.

[Include your findings here.]

After reviwing the Vulnerability Assessment Process Flow Diagram, there are a few things that I think we can improve on. First will be the input validation. This is first line of defense in every system. We need to validate users' input to prevent any attack like an SQL injection. Second will be the APIS. Since most system now run mostly on the cloud and rely heavily on APIs to transfer and recieve data. The API will need to be secure in order to safely protect clients' data. The third one would be cryptography. This is always important. Everytime, data is being transfered from one place to another, messages are needed to be encrypted. This way we can safely transfer clients' internationally. Client and server are also important in building a secured system. We need to make sure that datas are safely stored in both clients and servers. Additionally, code errors and code quality also play an important role in secure communication. We will need to constantly fixing and debuging any errors and continuosly improve the quality of our codes in order to prevent any current and future exploit. This way we can prevent any unauthorized access and prevent any data leak.

3. Manual Review

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

[Include your findings here.]

After looking through all the java file, most of the input are in string type. However, I do not see any input validation. Since it's a string type, we should validate the user input to prevent any exploit. We can easily prevent any SQL injection by simply implement the input validation. This is the first defense and it can easily prevent user to gain access to our system. I also look at the APIs to see how the system interact with it. Look like they are still using the GET method and this can easily leak clients' data into the browser history. Hackers can easily get access to these datas. Looking through the code, I do not see any encryption in the system. I'm not an expert in how data should be encrypted but I don't see anything that the system do to encrypt clients' data in order to prevent any attacks. The system should have a better way to handle errors as well. I only see the the one time they try to catch error is in the docdata.java. However, not much is happening here, it doesn't seem like they do much when they found an errors.

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:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency check report
* Attribution (if any) that documents how this vulnerability has been identified or documented previously

[Include your findings here.]

Dependecies:bcprov-jdk15on-1.46.jar

Descriptioin and recommened solution: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. This is related to a java implentatin of cryptographic algorithms. I think we should upgrade to the latest version.

Dependecies:hibernate-validator-6.0.18.Final.jar

Descriptioin and recommened solution: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. For this one, we will need to upgrade to the newest version as well.

Dependecies:jackson-databind-2.10.2.jar

Descriptioin and recommened solution: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. This will need to upgrade to current version

Dependecies:log4j-api-2.12.1.jar

Descriptioin and recommened solution: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. Upgrade to the current version

Dependecies:snakeyaml-1.25.jar

Descriptioin and recommened solution:The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

CWE-776 Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

Dependecies:spring-aop-5.2.3.RELEASE.jar

Descriptioin and recommened solution: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.

Dependecies:tomcat-embed-core-9.0.30.jar

Descriptioin and recommened solution:The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

Dependecies:tomcat-embed-websocket-9.0.30.jar

Descriptioin and recommened solution:The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

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.

[Include your findings here.]

Most of these dependencies will need to be reviewed and upgared to the most current version.