****

# 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** | **1/27/2021** | **Matthew Marinelli** | **Conducting Vulnerability Assessment** |

## 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

Matthew Marinelli

## 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?

The business values protected contact because a lack of those safety standards will jeopardize sensitive matters related to its activities. The program shall be interested in the execution of financial transactions and exchanges. Such information is sensitive and would undermine the credibility, security and availability of the system as well as the image of the application would result in financial loss.

* Are there any international transactions that the company produces?

Artemis Financial is involved in multinational transactions. This transfers include the internet functioning of consumers and their accounts at international level as well. There is also a chance for consumers to fly to foreign countries and they will need to have access to their accounts outside the country.

* Are there governmental restrictions about secure communications to consider?

There are several government restrictions. For example, the General Data Protection Regulation (GDPR) strategy on contact (Mondschein & Monda, 2019). The proposal will mandate that data handlers and service providers ensure that their client's information is secure from attacks. The organization should also seek to ensure that there are no security vulnerabilities that could breach the protection of sensitive customer information.

* What external threats might be present now and in the immediate future?

The external threats would be:

**Man in the middle attacks** – In communication where a malicious group may intercept communication between a host and client and then manipulate data by listening in on packets.

**DOS Attacks** – Throwing a ton of traffic at a host to eventually overload and crash said host.

The immediate future threats would be many and various in nature. For example, there are various hacker groups out there that are in it for the bad, blackhat hackers and those who go in and try to find issues and report it back, called whitehat hackers.

* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

The modernization requirements that must be considered are sandboxing and virtualizing everything. This ensures hackers cannot really access the main portion of the system and even if they get into a VM, they can’t really do anything. Also, using VPNs to secure and hide our connections will be of great assistance.

## 2. Areas of Security

The areas of security that would be associated with Artemis Financials application would be Crpography, Client and Server, Secure Coding and API.

Secure coding is the practice of ensuring coding is all clean and efficient and lacks any “holes” Artemis Financials exception handling can be applied to the source coding of the application. Error checks are also a good feature to use because it verifies the coding is thorough.

Client and Server is the communication between the clients and the servers of the actual company. This is mainly used to retrieve and transact financial transactions in this case. Obviously the connection between these two is of vital importance to both the customer and the company.

Cryptography is the encryption of files in transfer to ensure they cannot be read out in the open. Without encryption of files, they are viewable and editable to “prying eyes” out in the open network. Also, if you wanted to add extra protection, you can also add certificate validation to ensure the website’s certificate is still valid and functional.

**Vulnerability Assessment Report**

**3) The Vulnerability Assessment Process Flow Diagram**

This image below represents the process by which the Artemis company API system was evaluated. The project was successfully accomplished and below is the process by which the entire vulnerability assessment was conducted. The code was developed in java programming language meaning there were some vulnerabilities that arise. However, all the vulnerabilities that were discovered were all corrected making the system as secure as possible. The code was manually inspected and all the vulnerabilities that were found are listed below.

**The diagram**

Planning The Assesment

Algorithm characterization

Main targets that might be affected

Assessment of possible Threats

PPS Characterization

Analysis of the system behavior and the effectiveness.

Is the system free of all Vulnerabilities?

Output the Vulnerability Report

Include various upgrades and Recommendation to the System.

Re-evaluate the system afresh.

**Vulnerabilities found in the code**

There were certain errors found in the code and some of them could easily be fixed. Since the web-based software was to be accessed by many users, it was important that the system was very efficient and this meant that the code had to be free of all the errors. There were also need for the client-server communication to be secure to ensure that there was no risk of eavesdropping in the network.

The system also had some vulnerabilities in the code where the system was in great risk of having any holes or backdoors that would give access to the hackers or any potential threats. This meant that the system codes had to be secure coded and from the assessment most of this was found to be missing. Most of these errors are found at the exception handlings in the codes and they could also be corrected very well by ensuring that all the exceptions are handled well, thus resolving any errors that might occur during running of the software.

In the main class of our program, some errors were also discovered where most of the different classes being called had no proper methods. The main class used global variables and this meant that the errors that program could not compile the following classes. This issue could easily be fixed by correcting the classes that were being called or renaming the calling function as well as the codes themselves.

By reviewing the codes at this implementation process, certain security issues were also found. The codes have not used any specific security algorithm that would ensure communication between the client and the server is heavily encrypted. This made the system to be at great risks of eavesdropping and thus more codes could be added in the main program that ensure secure transmission of the web-based API.

**The codes Used in Our Dependency Check**

For a successful dependency check, we have to update our plugins in order to ensure that the new plugins bring all the dependencies that we need. In our program software, the following code was used.

<plugin>

<groupId>org.owasp</groupId>

<artifactId>dependency-check-maven</artifactId>

<version>5.3.2</version>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

By adding the above code in our pom.xml file, we were able to run it whereby the html report was given to us with all the dependencies that were discovered. Some of the outputs we received are discussed in the report below. The solutions to the dependencies is also provided below. The report from our dependency check is shown below;

Scan Information (show all):

dependency-check version: 2.2.4-SPRING FRAMEWORK

Report Generated On: Jan 28, 2021 at 13:04:31 EDT

Dependencies Scanned: 36 (28 unique)

Vulnerable Dependencies: 36

Vulnerabilities Found12

Vulnerabilities Suppressed: 0

...

**4) Maven Dependency Check Security Vulnerability**

It is very common for applications to have a lot of vulnerabilities after being developed. It is thus very important that a dependency check has been done on the project so as to get the exact security vulnerabilities available. The maven dependency checks on our Artemis software was thus very important and in running it some vulnerabilities were found. They include;

* **Cross site scripting**- Our code showed the following risks and vulnerabilities. Since it was a finance software it means that there are certain codes that interact with the excel documents and this means that there are great risks of cross-site scripting. This type of vulnerability is whereby the attackers target the scripts embedded in the application pages which are executed on our client’s side.
* **Ghost Script/configure**- This is one of the dependencies that was displayed in our report. Its risk to our program is high and is required to be solved before the program software was released to the public.
* **LDAP Injection** – This vulnerability was found in most of the codes that included java access to our database. The issue with this type of vulnerability is that it would bring great security issues to our application and this meant it was a great risk. This type of vulnerability is where attackers will have the power to include lines of code in our input fields and by doing so, they will have access to manipulate the codes directly even being in the client side.
* **Insecure Storage of Information –** The information that the clients would provide during registration would have no secure storage as the encryption system and algorithms of our codes were very weak. This meant that if an attacker was to gain access to the storage of the application server, they would easily have access to customers details.

**Recommended Solutions**

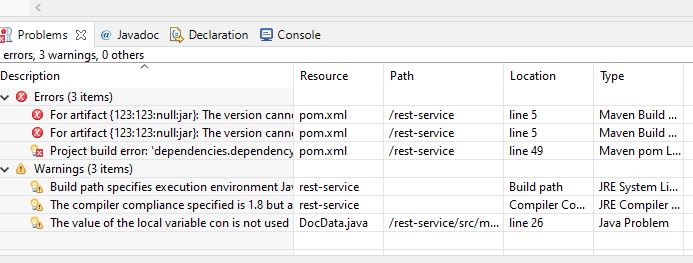
The vulnerabilities that were discovered in our code are listed above. All the above vulnerabilities could result to great losses and massive data losses in the company if the application is implemented without them being solved. Cross site scripting could be solved by ensuring that all the codes are double checked to ensure that there no loop holes for this type of attack. All the codes should be passed through a dependency checker and any vulnerabilities discovered should be immediately be solved. LDAP injections are also a great risk in the company software and they should be solved after being detected. They also run a great risk since the attackers might find this vulnerability and use it to their advantage thus it may cause the company to even loose money. The software should also be added some codes that would ensure data being stored in the storage servers of the application are heavily encrypted. Encryption of a finance application is one of the most required recommendations before releasing the software to clients. If all the above vulnerabilities are tackled, then the system will be very secure which will lead to efficient systems.

**Attribution**

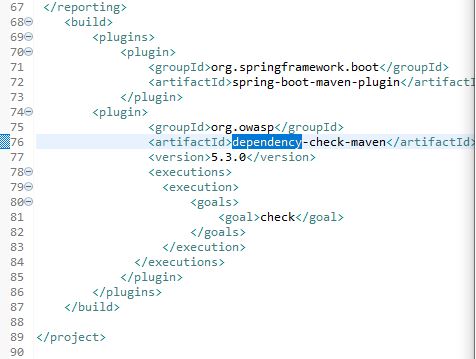
As discussed in the report above, there are various vulnerabilities that result form systems being coded. Coding is a complex process and this means that most developers miss some of the vulnerabilities that might remain in the system. LDAP injections just like SQL injections has been a great vulnerability risk that has affected many systems and should be resolved so as to ensure that the attackers have a small chance of gaining any access to the system by use of this type of attack. Cryptography of the software is an issue that has affected most software where the information about clients has been revealed becoming a great privacy issue. There are more details online on how these vulnerabilities may be detected and resolved.

**5) Mitigation Plan**

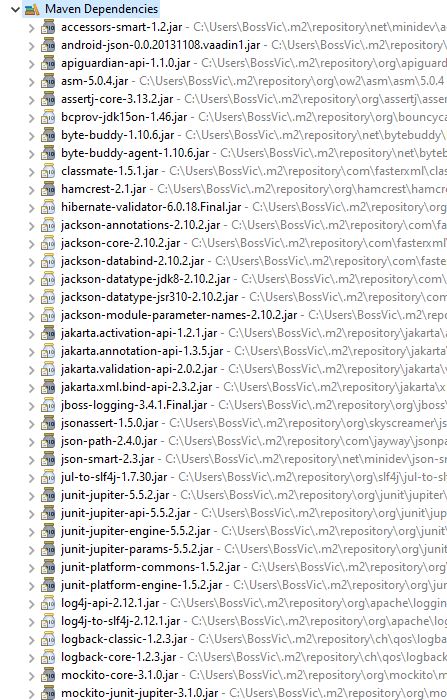
* **Detect the vulnerabilities in the Software-** By using a proper dependency software, one can be able to detect all the vulnerabilities that are available in the system and this will help in ensuring that the system is not released with the unnoticed vulnerabilities.
* **Assessing the Vulnerabilities in the software-** The assessment is listed with the level of risk they portray to the system. Assessing all the vulnerabilities that were detected by the dependency checker will ensure that the system is resolved.
* **Resolving the vulnerabilities-** Certain protocols should be taken to ensure that all the vulnerabilities have been taken care of to avoid security risks to the company. The recommended solutions listed above should all be done in order to ensure secure application.
* **Implement the API-** After resolving all the vulnerabilities present, the system can be successfully implemented in several stages. This implementation process will result to more vulnerabilities being detected which will give the company a chance to solve them before they are attacked.

**Images Obtained from Our Software Application**

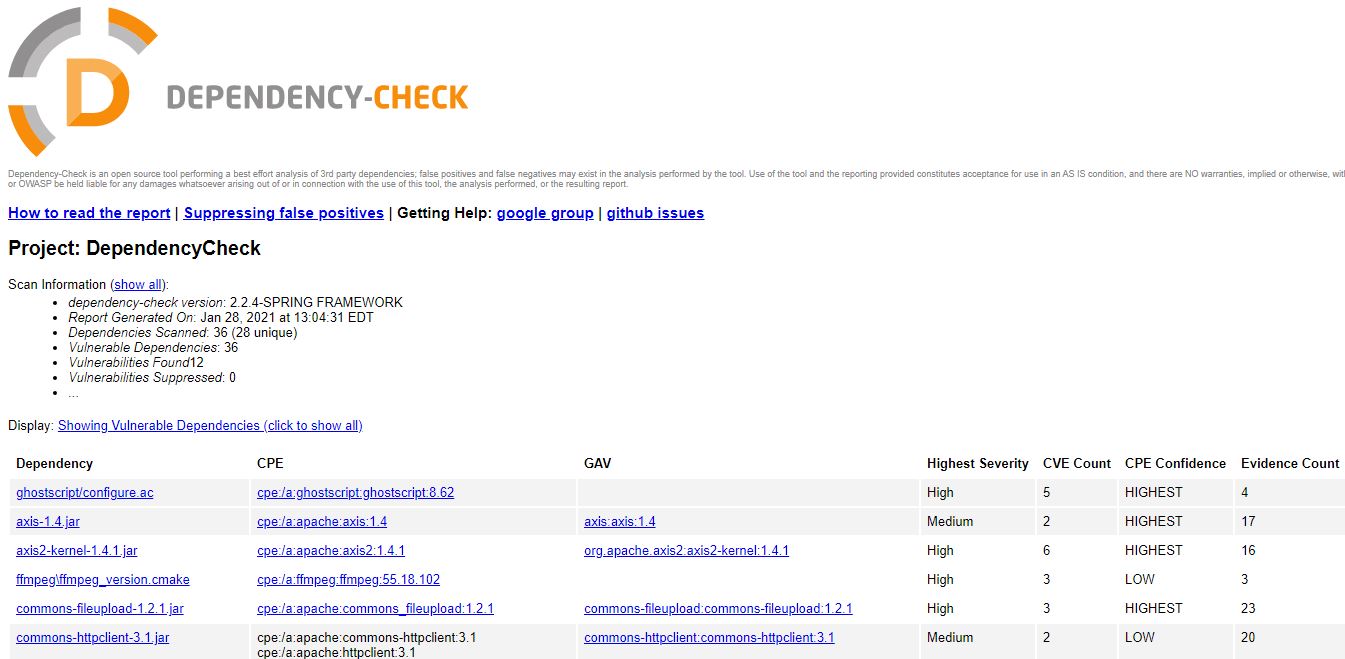
**Our Codes**



**Maven’s Dependencies**



**Dependencies Report**

****