****

# 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** | **1/22/2021** | **Taro Serigano** |  |

## 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

Taro Serigano

## 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 Communication:** Secure communication is highly regarded since the security violation on sensitive information could directly affect the company’s reputation and its operations. Today, a large number of applications handle financial transactions and transfer. Leak of such highly valuable information could cause huge financial loss and damage the company’s reputation as well as heavy fines.

**International Transactions:** Today, a lot of US companies expand its operations to the global market and handle international transactions. Also some customers may travel to foreign countries and need to access their accounts abroad.

**Government Restrictions:** The governments’ restrictions are getting tighter and tighter recently. For instance, the EU has their own guidelines and restrictions called the General Data Protection Regulation, which lately issues heavy fines on Google and Facebook in 2020. All companies that produce operations overseas must ensure that there is are security flaws that may put customer’s information at risk.

**External Threats**: There are various kinds of external threats today. A Denial-of-Service (DoS) attack that utilizes malicious malwares and group attack the applications to cause system down, Auth threats; basically the un authorized access to user accounts. Some examples of immediate future threats include: API threats and Error handling.

**Modernization:** When implementing new technology in the legacy applications, we would have to take incremental approach in order to minimize the risks of its operations and compatibility. We would also have to pay great attention to flexibility and accessibility of the applications when making modernization updates. By enabling the applications open source and accessible, we can benefit from more developers having access to them and contribute to the quality enhancement.

## 2. Areas of Security

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

**Cryptography**: In order to avoid compromisation of the confidential information of users when the information gets transferred over the internet, we would have to ensure that there is no lack of appropriate encryption in the transferring information.

**API**: When making API calls, we would have to ensure that there is great security between the application and API connection and secured configuration is properly set up.

**Secure Coding**: Secure coding plays a crucial role in ensuring the great security over the application system. It also provides consistency across different platforms and exception handling and error checks are very important for debugging purposes.

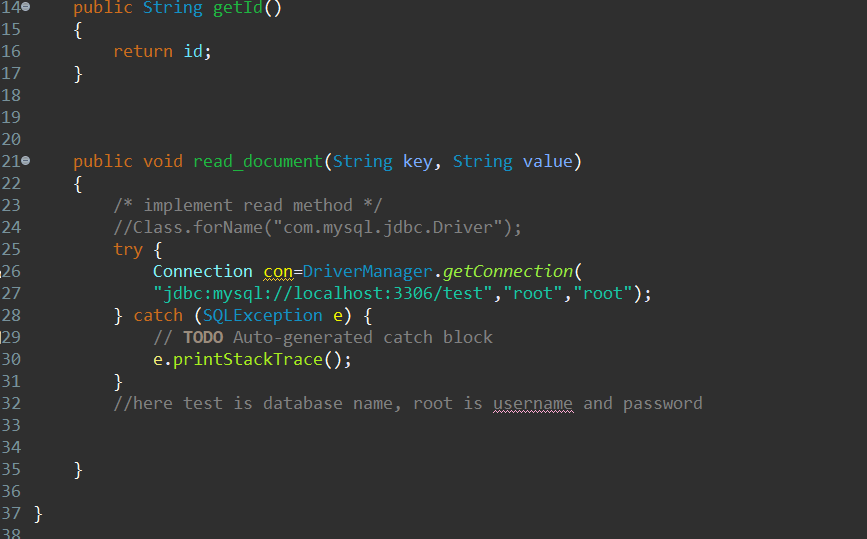
**Client and Server:** It is inevitable for Artemis to send over the highly sensitive information over the internet from the client to the server. The transaction has to go through this pathway to either send and retrieve the sensitive data. The connection needs to be highly secured and ensures that there is no risk for compromising the confidentiality of such information.

## 3. Manual Review

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

**Exposure of highly sensitive data**

There is exposure of highly sensitive data in the DocData.jave class. The method takes parameters as the database location, username and password. Here, the root username and password are utilized. It is highly recommended never to use the root user account, also the password is “root” which is the same as the “root username. Hackers can easily make unauthorized login and could attack the company’s system entirely in a matter of seconds.

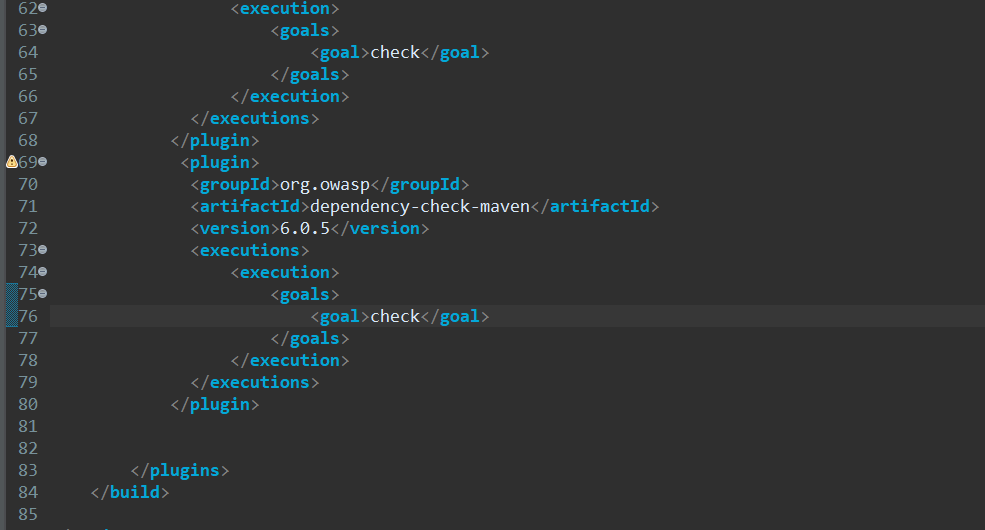


## 4. Static Testing

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:

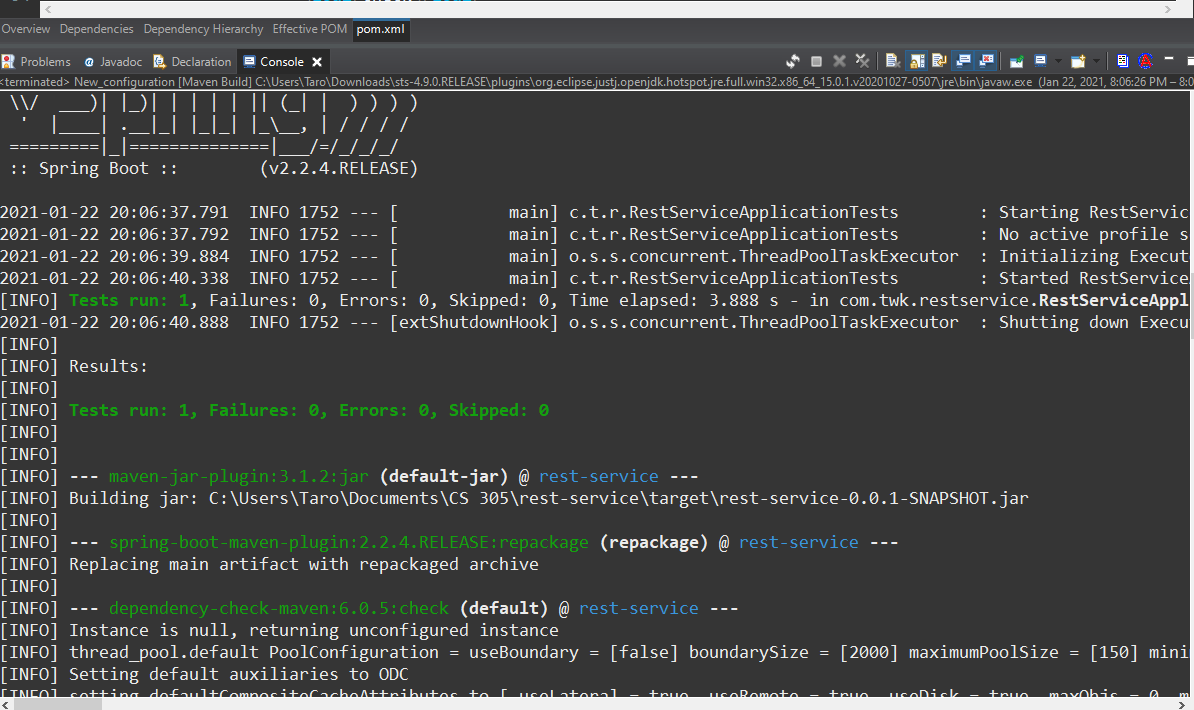
1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

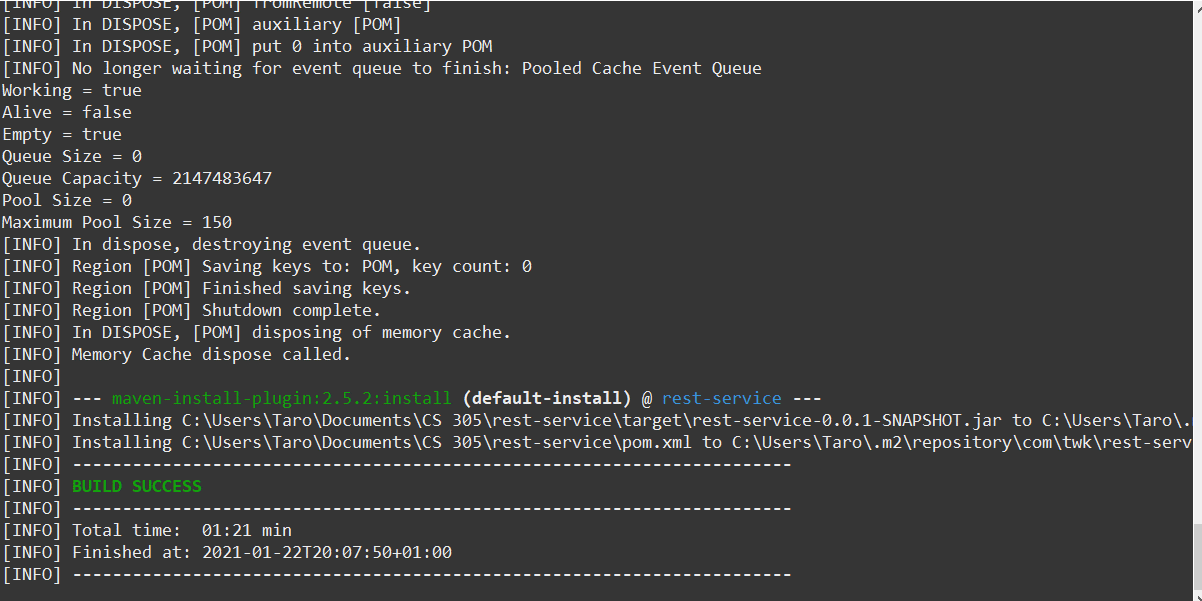
Installed the plugin:



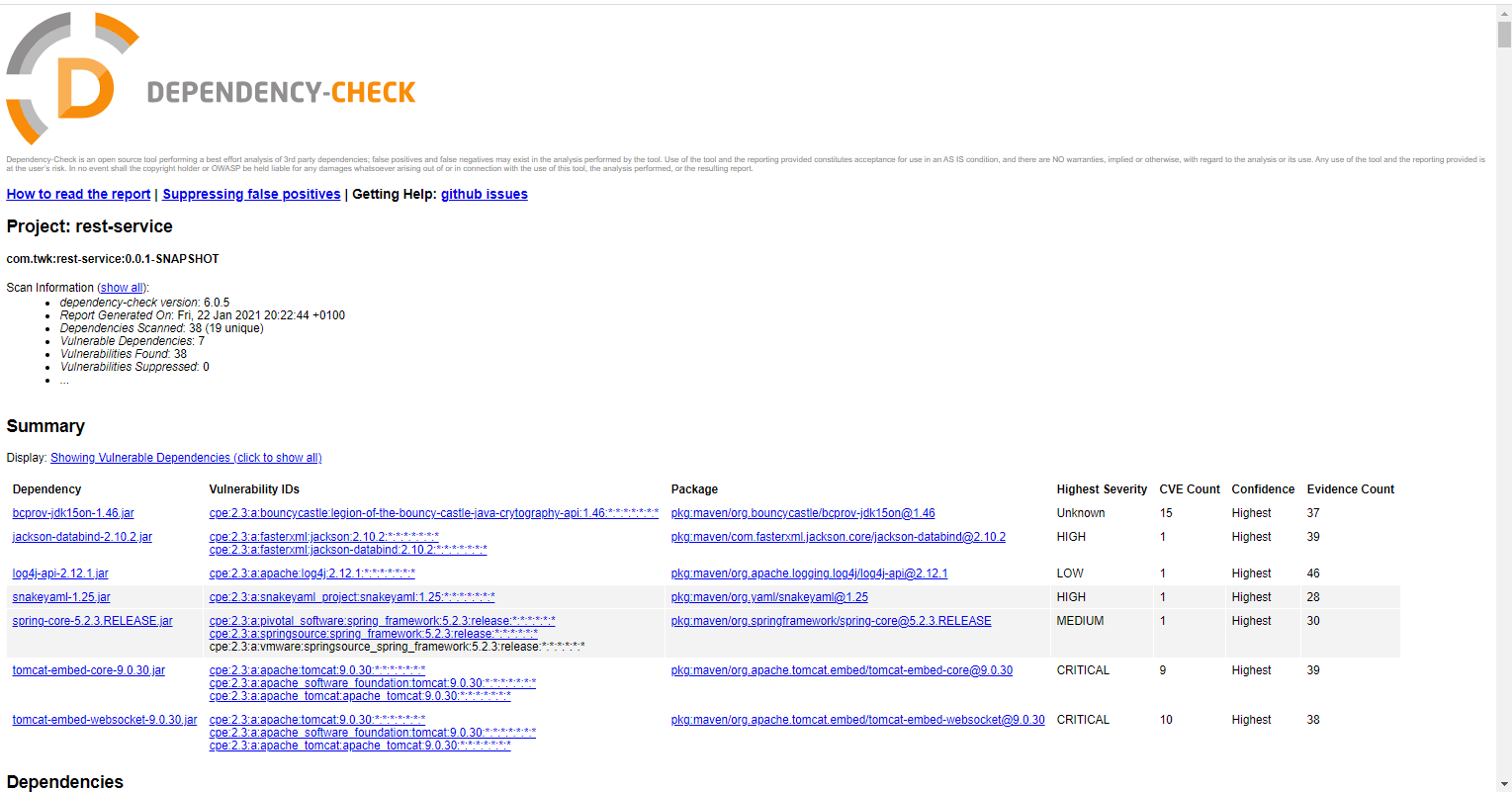
Ran the dependency check.

dependencies check plug-in integration into Maven.





Getting the dependency report:



**Vulnerabilities**

|  |  |  |
| --- | --- | --- |
| Code | Desc | Mitigation |
| [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624) | The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169. | The fix for CVE-2013-6468 enables the Java Security Manager (JSM) to sandbox the evaluation of MVEL expressions. This introduces performance degradation in high load environments. The following ways of running Red Hat JBoss BPM Suite are considered secure while mitigating performance degradation:  1. In high load environments where performance is critical, it is recommended to only deploy applications that have been developed on other systems and properly reviewed. It is also recommended not to create any users with the Analyst role on such systems. If these safeguards are followed, it is safe to leave JSM disabled on these systems so it does not introduce any performance degradation.  2. In testing and development environments without high loads, or in environments where rule authoring is exposed to external networks, it is recommended to have JSM enabled in order to achieve security benefits of properly sandboxed evaluation of MVEL expressions.  Allowing users with the Analyst role to log in to the Business Central console when JSM is disabled is not secure and not recommended. |
| 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. | Many XML parsers and validators can be configured to disable external entity expansion. |
| CVE-2017-18640 | The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564. | 1. If the YAML is not coming from untrusted source (it is merely a configuration file) then it is a false positive. Just ignore it. The quality of NVD database is very low and contains tons of issues which appear to be false positives. 2. Read the YAML and check its quality before giving the document to SnakeYAML (count \* and & for instance) 3. Migrate to [SnakeYAML Engine](https://bitbucket.org/asomov/snakeyaml-engine/src/). It has a configuration option to restrict aliases for collections (the aliases for scalars cannot grow and they are not restricted) 4. SnakeYAML has now also possibility to fail early. |
| CVE-2019-17569 | 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. | Several vulnerabilities were discovered in the Tomcat servlet and JSP engine, which could result in HTTP request smuggling and code execution in the AJP connector (disabled by default in Debian).  For the oldstable distribution (stretch), these problems have been fixed in version 8.5.54-0+deb9u1.  We recommend that you upgrade your tomcat8 packages. |
|  |  |  |

## 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.

**TLS Certificate**: Certification should be established between the client and server. It can prevent some of the risks such as authenticate process and API requests transfer risks.

**Access Username and Passwords**: Proper usernames and passwords that go through validations should be made. Validations could be such as requiring special characters, certain length of characters, sholdn’t be the account holder’s name or birthday, etc.

**Code Review:** The code needs to be reviewed in order to provide secure code discipline. The overall code structures need to be re-build or modified to prevent authentication vulnerabilities attacks helps to manage error handling.