****

# 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** | **5.16.2022** | **Denis Dzenyuy** |  |

## 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

Denis Dzenyuy

## 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?
* The value of secure communications to the company is of extreme importance, because for one it is a company that deals with client financial information, which is often a ‘high value’ target for cyber-attacks. The company has to ensure the confidentiality, integrity, accessibility of data in motion as this in important to the company’s reputation, profitability and survival. Their web-based software application (despite having an architectural style, and communication strategy for web services development (RESTful API) can still fall victim to (Injection, DoS, Broken Authentication, Parameter Tampering, Broken Access Control etc.) vulnerabilities and exploits.
* There are international transactions that the company produces due to the type of sector and clients the company deals with. People retire, spend, invest and/or need insurance everywhere/anywhere in the world; therefore, the company is bound to conduct multinational transactions.
* Governmental restrictions about secure communications such as privacy laws, financial services acts, credit protection, unfair and deceptive practices, disclosure, Children’s Protection, abuse, fraud, intelligence sharing, privacy, needs to be considered. Considering these along with taking implementing robust information security programs will help mitigate cybersecurity threats and/or communication breaches.
* External threats that might be present now and in the immediate future will include :
* Outdated Software: Software that is not current or up-to-date with recent patches/fixes or upgrades.
* Exploits such as Injection attacks: Malicious code could be sent to the web app disrupt it’s normal operation.
* Security Misconfiguration: Malware delivered by posing as a trust worthy organization or entity.
* Broken Authentication: Compromise of (weak) passwords, keys or session tokens, improper application session timeouts, poorly hashed passwords mostly due to lack of proper or stronger authentication methods such a MFA/2FA, Password complexity/rotation policy, server-side, secure, built-in session manager, or limit/delay failed log-in attempts.
* Banking Trojans: Spoofed website of the company to redirect client data to the attacker.
* Broken Access: Web application components that become accessible instead of being protected.
* DDoS attacks: Overwhelming of a target system or network with requests (traffic) by bots (machines).
* Session hijacking: Clients IP address is replaced by an attacker (man in the middle) so that communication/traffic can be directed through the attacker.
* Considering the wide availability of modernization tools on the market such as Kubernetes, GoogleCloud, CloudFoundry, and OpenShift, Software modernization could be an extreme but beneficial approach towards the client’s web application update. Which would include code refactoring, updating certain application modules, adding new components, configurations and automation, cloud migration or micro services, etc. In one form or another, any software needs modernization at some point. It is necessary to either correct some errors (performance issues, for example), make it operate more effectively, or bring the solution up to date with newly available technologies. It is simply integral to keeping a piece of software relevant. Modernization and security advancements, web applications that work in sand-boxed mode can also make use of open source library in order to add a layer of functionality to their program.

## 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.

* Input Validation: Since we are implementing a command input function, it will be important for us to strictly validate all input in to the function. Successfully validating input will promote a secure system and prevent any injection attacks.
* APIs: If our command input function needs to be accessible outside of our system, we may need to implement a RESTful API to allow access to the functionality. In this case, we will need to ensure that the API is created in a way that does not allow unexpected access to our systems. We could also do some input validation at the API level.
* Cryptography: In the case we are going to use an API, we should ensure proper cryptography is used to encrypt and protect our data and customer data.
* Client/Server: If we are to implement API access to our function, we need to ensure the proper certificates are used to ensure data is safe during transfer via https requests.
* Code Error: We will need to code review all command input functions as well as any API access layer code.
* Encapsulation: If our command input function needs to access data stored in our system, we need to ensure we are using it properly are not messing up our data structures.

## 3. Manual Review

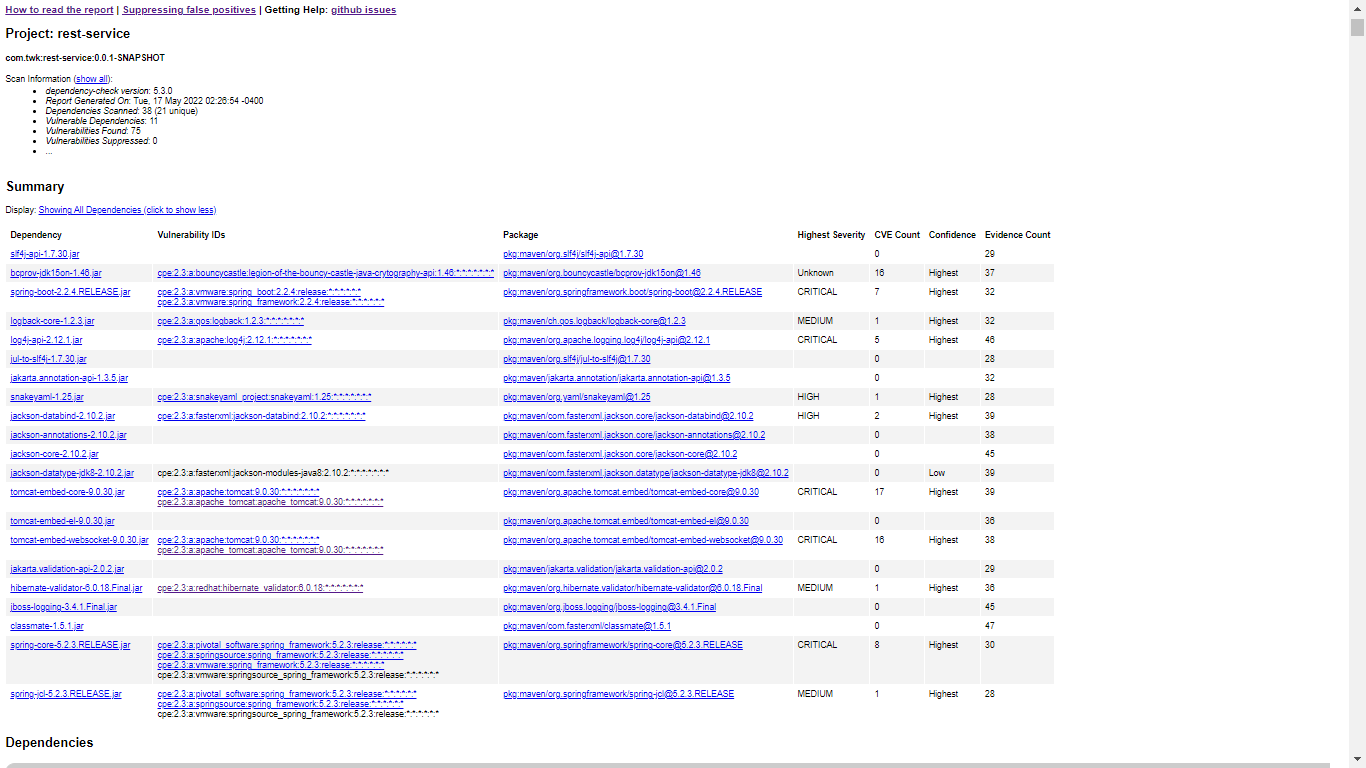
Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

* spring-data-rest-webmvc version: In our pom.xml, we have a dependency on spring data rest webmvc. The version we are dependant on is out of date. (2.6.5.RELEASE). The current live version is 3.3.0. We will need to upgrade to the latest version since the latest version of the dependency could have many important security patches/updates since 2.6.5.
* In GreetingController.java line 27-30 we take in data from parser, and put it directly in to a string value that is then added to our template as plain text. This is risky as someone might be able to format a string that can cause a problem in our system down the line. We should at least limit the size (perhaps length of the string) that can be taken in.
* spring-break-data-webmvc: On our pom.xml, we will rely on webmvc rest data. The kind I trust is scandalous. (2.6.5. Additional) The current direct version is 3.3.0. We have to update to the latest version as the last version had a lot of security updates up to 2.6.5.
* In 'GreetingController.java' lines 27-30 we take the scan data from parser, which we attach directly to the string and then add a clear text format. This is especially risky as an actor could design or format a string that would cause problems for our system below. We must at least limit the size (perhaps the length of the string) that can be taken.

## 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
   * 1. Published Vulnerabilities:[**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624) **-** [[CVE-2015-6644] (OSSINDEX)](https://ossindex.sonatype.org/vulnerability/3a59870b-28b3-4b6b-86b0-9629ebe9de40?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0) - [[CVE-2015-7940]](https://ossindex.sonatype.org/vulnerability/58377ddb-36ee-4586-9d29-c11f5c1e78ba?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0) - [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338) - [**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339) - [**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341) - [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342) **-** [**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343) **-** [**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344) **-** [**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345) **-** [**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346) **-** [**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352) **-** [**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098) **-** [**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382) **-** [**[CVE-2020-26939]**](https://ossindex.sonatype.org/vulnerability/9e56f765-fe13-4d65-925a-241a8047f1b6?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0) **-** [**CVE-2013-4152**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-4152) **-** [**CVE-2013-7315**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-7315) **-** [**CVE-2014-0054**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0054) **-** [**CVE-2016-1000027**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027) **-** [**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) **-** [**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968) **-** [**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772) **-** [**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550) **-** [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) **-** [**CVE-2021-44228**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-44228) **-** [**CVE-2021-44832**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-44832) **-** [**CVE-2021-45046**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-45046) **-** [**CVE-2021-45105**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-45105) **-** [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) **-** [**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649) **-** [**CVE-2020-36518**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518) **-** [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569) **-** [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996) **-** [**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934) **-** [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935) **-** [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943) **-** [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527) **-** [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935) **-** [**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938) **-** [**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022) **-** [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484) **-** [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122) **-** [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122) **-** [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329) **-** [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640) **-** [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037) **-** [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079) **-**[**[CVE-2021-42340]**](https://ossindex.sonatype.org/vulnerability/d67e84cc-a7e5-4088-ace7-7bd7a0a5e010?component-type=maven&component-name=org.apache.tomcat.embed.tomcat-embed-core&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0) **-** [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569) **-** [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996) **-** [**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934) **-** [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935) **-** [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943) **-** [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527) **-** [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935) **-** [**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938) **-** [**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022) **-** [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484) **-** [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122) **-** [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122) **-** [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329) **-** [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640) **-** [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037) **-** [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079) **-** [**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693) **-** [**CVE-2016-1000027**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027) **-** [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421) **-** [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060) **-** [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) **-** [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118) **-** [**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950) **-** [**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) **-** [**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968) **-** [**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950)**.**



* + 1. 38 Dependencies were scanned of which 21 were unique, and a total of 75 CVE’s amongst which are five (5) critical, two (2) high, three (3) medium, one (1) unknown, and ten (10) zeros, CVE Severities. Before we make any determinations as to the overall accuracy of the dependency check, we first have to ensure correctness of the identified CPE as the report may contain false positives as well as false negatives (CPE identified incorrectly) due to how Dependency-check works (checks for ‘known vulnerabilities’). Additionally, there may be vulnerabilities within the project dependencies that have yet to be known publicly.

After any obvious false positives and/or negatives have been weeded out all remaining entries should be reviewed in order to determine if any of the identified CVE entries are actually exploitable in our environment. Determining if a CVE is exploitable in our environment or many environments can be tricky hence, the best option is to upgrade the library if possible to be safe. Some CVE entries can be fixed by either upgrading the library or changing configuration options. The false positives will be contained in the CPE values (highest on our report is 46, lowest is 28). A wrong CPE value is candidate for the use of the suppression feature in the report (approximately 75 identified in our report) to generate a suppression XML file that can be used on future scans. In addition to looking at the CPE values in comparison to the name of the dependency, it is important to consider the confidence of the CPE. Each identified vulnerability includes a link to the the [NIST](https://www.nist.gov/) where specific detail and recommended solutions can be found.

* + 1. The report contains data retrieved from the [National Vulnerability Database](https://nvd.nist.gov), [NPM Public Advisories](https://www.npmjs.com/advisories),  [RetireJS](https://retirejs.github.io/retire.js/), and [Sonatype OSS Index](https://ossindex.sonatype.org).

## 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.

**Code review and modification:**

To help in mitigating authentication vulnerabilities, code review and modification by way of integrating secure coding practices will help the quality of the code in terms of having proper authentication, error trapping, and exception handling when it occurs.

**Updating Apache server:**

Updating the apache server in using or benefiting from vulnerability CVE-2020-9488 by updating the Apache Server to the newer version. This will help fix prior version issues related to vulnerabilities within the older versions of the Apache Server.

**TLS Certificate mutual authentication:**

TLS Certificate mutual checking and related issues will apply to both the client and server-side through pinned certificates during mutual authentication. This will help avoid compromising the requests of client API and help mitigate the potential vulnerabilities found in the Bouncy Castle dependency

**Certificate Validation:**

Proper certificate validation will refer to code being properly sanitized during correct validation and verification of all digital certificates for the application and webserver. This should help avoid the exploitation of vulnerability CVE-2020-9488 and CVE-2020-13935.

**Data access username and password:**

The use of controlled access to data username and password will mean enforcing the creation of strong password-based authentication guidelines that will include a combination of alphanumeric characters for both the username and password. Multi factor authentication (MFA) or Token-based authentication.Using one of these methods or as a combination of two methods will improve the risk from a brute force attacks or breach into the system.