****

# 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)-5

[3. Manual Review 5-6](#_Toc32574613)

[4. Static Testing 6-8](#_Toc32574614)

[5. Mitigation Plan 9](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/8/2021** | **Elizabeth Hodgman** | **First Assesment** |

## 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

Elizabeth Hodgman

## 1. Interpreting Client Needs

Artemis Financial is a financial consulting company that puts together financial plans to help their clients store their money wisely. Since Artemis Financial is working with their client’s livelihoods, security is the up-most priority to them. As a financial institution, it is best to assume that they may have clients that will need to make transactions abroad. Therefore, Artemis’ application should include capabilities to produce international transactions. Since the application will work with communications abroad, communication security will be a high priority. Though, there are laws within the US, and other countries, that will restrict some communication securities. For example, the Electronic Communications Privacy Act (ECPA) allows the U.S. government to access all digital communications (Norton Life Lock, 2018). Also, if international interactions are made, those countries’ governments may also have similar laws in place. So, it is essential that our software is compliant with these regulations. Though, it is still important to keep communications as secure as possible to prevent sensitive information from being stolen. Unfortunately, since money is one of the main motivations for attackers, the possibility of external threats to Artemis is high. Within Artemis’ database, costumer information such as bank account information, email, phone number, address, and social security number may be present. So, attackers could try to run DDoS, SQL injection, password, phishing, or other attacks to steal that information. Finally, as technology advances, so does the knowledge of the attackers. Therefore, it is important to developing the application with the most up-to-date technology and dependencies. That way, there is a smaller chance of the software having security bugs from using outdated dependencies.

## 2. Areas of Security

**Input Validation:** One way that attackers can hack into the system is when they use malicious input. To prevent this, it is best to implement input validation for certain areas, such as user sign in. This type of security would allow only numbers, letters, and certain characters to be used within the input fields. This prevents malicious character input that would allow SQL injection and false sign-ins, causing sensitive information to be accessed.

**API:** Since this software is going to be used with a RESTful API web-based application, keeping that API secure is essential. To access their account through the HTTP, the clients’ information will be connected to this API. If the API is not secure, an attacker could retrieve client information.

**Cryptography:** For this project, cryptography is an area of security for a couple of reasons. Those reasons are because the application will work with international transactions, and it will route information between the client and server. It is very important to keep the information within transactions safe. Encryption would only allow Artemis Financial and its recipient to view the information within that transaction. Thus, if it gets intercepted during the transfer, the contents would not be accessed by untrusted entities. Though, different countries have their own encryption laws and policies (Global Partners Digital, n.d.). To allow Artemis to provide the best service to their clients, we would want to make sure that our software encryption complies with most countries’ regulations.

**Client/Server:** As a web-based application, there will be a server that multiple clients will be on at any given time, such as end-users, employees, IT, and maintenance. So, securing the traffic between the client and server is highly needed. If the route between the server and client is not secure, an attacker could easily see the information that is routing back and forth between the two. One option to secure this architecture is to have the data encrypted while in route. Hence, the use of cryptography security.

**Code Error:** For a web application, it is essential to have proper error handling. Especially if an error pop-up displays implementation details to a hacker. Improper error handling can give an attacker information about flaws and could leak data within the site.

**Code Quality:** Code quality can impact the amount of code errors. Having poor code quality can lead to having more code errors, which risks data leakage. Also, when using an API, since there is going to be multiple types of users, the application should be developed to only allow authorized users to access certain methods. It is also essential to have a high-quality code to reduce vulnerabilities within the application. Otherwise, for example, an attacker can get full access to the application by exploiting buffer overflow issues.

**Encapsulation:** Since Artemis Financial will have a data base full of information that is highly sought after, it would be very beneficial to have encapsulated security. We should encapsulate the data structures that hold the information and the functions that use it. Encapsulation hides the data from public functions and makes it harder for outside users to get to. Since encapsulation is an OOP best practice, this security area also coincides with code quality.

## 3. Manual Review

When doing the manual review, I checked for any vulnerabilities in the areas of security that this application should focus on. For the input validation, I do not see any code that validates what the client is inputting. For example, in Greeting.java it seems like the user can input any character they want as their ID, resulting in possible SQL injection. The CRUDController.java file is another important area of vulnerability to SQL injection. Within that file, there is no input validation of the business name, which allows access to documents. So, an attacker could easily use malicious input to access the database. Next, I ran the code to check the API, but got an error page. The application does not seem to have any mapping to an API. Furthermore, it is uncertain if this application is currently doing anything with data. So, we would need to develop an API connection for the application to run properly and securely. Moving onto encryption, I do not see anything that involves encrypting data. Encryption is especially important for the account and deposit information in customer.java. This issue will have to be mitigated to secure that sensitive information. Again, encryption is needed if Artemis is involved with international transactions. While mitigating, we would want to keep in mind of cryptography laws. Next, there doesn’t seem to be anything in place that secures the data in route between the server and client. So, as stated, we would need to implement encryption for when data is in transit or use an encrypted HTTPS. Then, looking through the code, I didn’t find any advanced error handling throughout the files. If there is no secure error handling, this could result in having sensitive data or application information being leaked. After looking for error handling, I looked for encapsulation. The customer.java file uses some encapsulation, but not very securely. The account number is private, but the account balance is not. Having a public customer’s account balance is not very smart, as it would let attackers know how much money is involved with the application and business. Lastly, I looked at the code quality. For early development, the quality of the current code looks okay. Though, there are a lot of things that need to be added for the application to be functional and fully secure.

## 4. Static Testing

**bcprov-jdk15on-1.46.jar**

* Description: 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.
  + It has an invalid curve attack, which allows an attacker to recover a private key from the vulnerable application. Using a private key, the attacker can cause the execution of unexpected code and gain access to vulnerable data. Bouncy Castle Java Cryptography APIs 1.58 up to, but not including, 1.60 uses externally controlled input to select classes or code. Again, resulting in deserializing a private key. So, it seems that the vulnerabilities have been patching in versions 1.60 and later.
* Attribution Identifier: [cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46)

**hibernate-validator-6.0.18.Final.jar**

* Description: Hibernate’s Bean Validation reference implementation.
  + A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This vulnerability allows attackers to bypass input sanitation that the developers may have put in place when handling user-controlled data in error messages. This bug was found in version 6.1.2.Final. Though, it was fixed in hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, and hibernate-validator 6.0.20.Final
* Attribution Identifier: [cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18)

**jackson-databind-2.10.2.jar**

* Description: General data-binding functionality for Jackson: works on core streaming API
  + A flaw was found in FastXML Jackson Databind, where it did not have entity expansion secured properly. This flaw causes vulnerability to XML external entity attacks, threatening data integrity. The issue has been fixed in versions jackson-databind-2.11.0 and jackson-databind-2.10.5.1
* Attribution Identifier: [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2)

**log4j-api-2.12.1.jar**

* Description: The Apache Log4j API
  + Improper certificate validation with host mismatch in Apache Log4j Simple Mail Transfer Protocol appended. Since the appender does not verify the hostname, the Simple Mail Transfer Protocol Secure connection could be intercepted by a man-in-the-middle attack. This type of attack could leak any log messaged sent through that appender. This issue has been fixed in version 2.13.2, or the system property mail.smtp.ssl.checkserveridentity can be set to true to globally enable hostname verification in older versions.
* Attribution Identifier: [cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1)

**snakeyaml-1.25.jar**

* Description: YAML 1.1 parser and emitter for Java
  + The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation. During a load operation, entities can reference other entities. So, an attacker could use this by providing a YAML document with many entities that reference each other, causing a Denial of Service. Though, this issue seems to be fixed in snakeyaml versions 1.26 and up.
* Attribution Identifier: [cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25)

**spring-aop-5.2.3.RELEASE.jar**

* Description: Spring AOP
  + In this version, the protection against Reflected File Download attacks may be bypassed depending on the browser used through the jsessionid path parameter. This issue can lead to an attacker gaining complete access over a machine. Also, in this version, the software does not neutralize or incorrectly neutralizes output that is written to logs. So, it is possible for a user to provide malicious input to forge or insert additional log entries. Then, a WebFlux application is vulnerable to a privilege escalation. It recreates a temporary storage directory. So, a malicious user that was locally authenticated can read or modify files that have been uploaded to the WebFlux application. There has been fixes to these vulnerabilities in versions 5.3.12+ and 5.2.18+.
* Attribution Identifier:

[cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)

[cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3)

[cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)

**spring-core-5.2.3.RELEASE.jar**

* Description: Spring Core
  + The vulnerabilities and fixes are the same as spring-aop-5.2.3.RELEASE.jar. First, the protection against Reflected File Download attacks may be bypassed depending on the browser used through the jsessionid path parameter. This issue can lead to an attacker gaining complete access over a machine. Also, in this version, the software does not neutralize or incorrectly neutralizes output that is written to logs. So, it is possible for a user to provide malicious input to forge or insert additional log entries. Then, a WebFlux application is vulnerable to a privilege escalation. It recreates a temporary storage directory. So, a malicious user that was locally authenticated can read or modify files that have been uploaded to the WebFlux application. There has been fixes to these vulnerabilities in versions 5.3.12+ and 5.2.18+.
* Attribution Identifier:

[cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)

[cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3)

[cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)

**tomcat-embed-core-9.0.30.jar**

* Description: Core Tomcat implementation
  + There are a few vulnerabilities with this dependency. First, the refactoring of this version introduced regression that resulted in invalid Transfer-Encoding headers incorrectly processing. This issue would allow HTTP Request Smuggling. It is also located behind a reverse proxy that handled invalid Transfer-Encoding headers incorrectly. Another issue with this version is that there is a parsing issue, which allowed some invalid HTTP header to be parsed as invalid. This bug could also lead to possible HTTP Request Smuggling. Then, 9.0.30 has a file contents discloser vulnerability that can cause an attacker to have access to arbitrary files from anywhere in the application. Another problem with this version is that it could trigger high CPU usage and could possibly crash the server. Also, there are many vulnerabilities that could lead to a denial of service and memory leaks. These issues have been fixed in versions 9.0.31, 8.5.51, and 7.0.100.
* Attribution Identifier:

[cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30)

[cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_software_foundation&cpe_product=cpe%3A%2F%3Aapache_software_foundation%3Atomcat&cpe_version=cpe%3A%2F%3Aapache_software_foundation%3Atomcat%3A9.0.30)

[cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30)

**tomcat-embed-websocket-9.0.30.jar**

* Description: Core Tomcat implementation
  + This has similar vulnerabilities to the tomcat-embed-core-9.0.30.jar. Such as, the refactoring of this version introduced regression that resulted in invalid Transfer-Encoding headers incorrectly processing. This issue would allow HTTP Request Smuggling. It is also located behind a reverse proxy that handled invalid Transfer-Encoding headers incorrectly. Another issue with this version is that there is a parsing issue, which allowed some invalid HTTP header to be parsed as invalid. This bug could also lead to possible HTTP Request Smuggling. Then, 9.0.30 has a file contents discloser vulnerability that can cause an attacker to have access to arbitrary files from anywhere in the application. Another problem with this version is that it could trigger high CPU usage and could possibly crash the server. Also, there are many vulnerabilities that could lead to a denial of service and memory leaks. These can be fixed by upgrading to version 9.0.31, 8.5.51, or 7.0.100. Though, it has another vulnerability that involves an Incorrect Default Permissions issue in the packaging of tomcat, which causes a local root exploit. This issue has to do with openSUSE, not tomcat. So, we would need to install openSUSE Security Update YaST online\_update or “zypper patch”. Alternatively, we could run the command - openSUSE Leap 15.1: zypper in -t patch openSUSE-2020-911=1 .
* Attribution Identifier:

[cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30)

[cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_software_foundation&cpe_product=cpe%3A%2F%3Aapache_software_foundation%3Atomcat&cpe_version=cpe%3A%2F%3Aapache_software_foundation%3Atomcat%3A9.0.30)

[cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30)

## 5. Mitigation Plan

* Manual
  + Since this project is in the very early stages of development, a lot of mitigation must take place for the application to be secure. First, as the code is being developed, an input validation method must be added to lessen the chance of SQL injection through the username, password, and business name inputs. Also, it is important to follow OOP best standards to improve code quality, including the encapsulation of sensitive data. Sensitive information needs to be private and should not be accessible in other parts of the system. Then, implementing error handling throughout the application is a must to stop data from being leaked to malicious users. Secure error handling can be developed with the use of try-catch blocks that show proper error messages to the user. Next, adding a technique to encrypt data is needed to secure the clients’ sensitive information. To encrypt this data, we could utilize one of many encryption APIs that java has to offer or use the “hashing with salt” technique. Adding cryptography would also help the client/server security. Another option the secure the client/server connection is to implement an encrypted HTTPS. Lastly, to allow the web application to be functional, it needs to be properly mapped to an API.
* Static
  + bcprov-jdk15on-1.46.jar
    - Upgrade to version 1.60 or later
  + hibernate-validator-6.0.18.Final.jar
    - Upgrade to7.0.0.Alpha2, hibernate-validator 6.1.5.Final, or hibernate-validator 6.0.20.Final
  + jackson-databind-2.10.2.jar
    - Upgrade to versionjackson-databind-2.11.0 and jackson-databind-2.10.5.1
  + log4j-api-2.12.1.jar
    - Upgrade to version 2.13.2 or set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification.
  + snakeyaml-1.25.jar
    - Upgrade to at least version 1.26
  + spring-aop-5.2.3.RELEASE.jar **|** spring-core-5.2.3.RELEASE.jar
    - Upgrade to version 5.3.12 or 5.2.18.
  + tomcat-embed-core-9.0.30.jar
    - Upgrade to version9.0.31, 8.5.51, or 7.0.100
  + tomcat-embed-websocket-9.0.30.jar
    - Upgrading tomcat to one of the above versions andinstall openSUSE Security Update YaST online\_update or “zypper patch”. Alternatively, instead of installing the update, we could run the command - openSUSE Leap 15.1: zypper in -t patch openSUSE-2020-911=1 .

**References:**

World map of encryption laws and policies. (n.d.). *Global Partners Digital.* Retrieved from <https://www.gp-digital.org/world-map-of-encryption/>

What are some of the laws regarding internet and data security? (2018). *Norton Life Lock.* Retrieved from <https://us.norton.com/internetsecurity-privacy-laws-regarding-internet-data-security.html>

Securing traffic between clients and your server. (n.d.). *Tableau.* Retrieved from [Securing Traffic Between Clients and Your Server - Tableau](https://help.tableau.com/current/guides/everybody-install/en-us/everybody_admin_config_ssl.htm)

Stappe44.(2020.) *The alias feature in snakeyaml 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.* Retrieved from [The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564. · Issue #458 · prometheus/jmx\_exporter (github.com)](https://github.com/prometheus/jmx_exporter/issues/458)