****

# 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** | **3/20/21** | **Alexander Mock** | **Version 1 revision for Artemis Financial.** |

## 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

Alexander Mock

## 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**

In Artemis Financial’s situation Secure Communication is very important as it is dealing with software that holds personal information. In cases that there isn’t personal information secure communication is still important as it can help prevent attacks. In the presence of personal information it is that much more important to have this security. In our particular case Artemis is dealing with transactions, financial information, and personal information. All of this data is sensitive and people would not be likely to share their sensitive data with a company that cannot keep their data secure and safe from potential threats.

**International Transactions**

Apon initial review I do not see anything that indicates the presence of international transactions. In conjunction with this there is nothing indicating that there might not be international transactions. Ideally we should treat this as a potential possibility even if it is not stated for or against the possibility. Artemis will need a safe and secure method to secure communications for both local and foreign.

**Governmental Restrictions**

At this time I do not believe there are any regulations on encrypting or securing the communications that need to be addressed. There is however always a potential that something could get passed such as a bill that could require attention. In the case that Artemis does do business internationally, the secure communications should defiantly be accessible by law enforcement if need by in the countries where it is required by law.

**External Threats**

In our modern world there are always risks to external threats. Artemis will be working with sensitive data both personal and financial, this could pose as a higher risk for attacks. Threats can come from many different methods such as: invalid input, leaking data(browser data), SQL injection, DDos/dos, MITM (Man in the middle/ packet sniffing).. ect.

**“Modernization” Requirements**

There can be quite a few challenges that come with modernization. In our case Artemis deals with financial planning. If they were to go the route where they use compiled bond/stock info from another company we are only as strong as our weakest link. Meaning if their infrastructure isn’t built strong/secure we will have the same issues as them. Depending on how we are reliant on other companies and systems we could potentially adopt whatever security issues they have. This could cause a certain cascade effect.

## 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.]

**Input validation -** This is a critical component in the event we are dealing with input. In our case we do allow input. Within CRUDController.java there is a part in line 12/13 where it allows inputs to be passed where it expects a string. The validation of that string will need to be secured to avoid failures and or injections(like SQL). If there is a user with Mal-intent, they could do damage to structure or gain access to sensitive data that they would normally be unauthorized to have.

**API’s –** In our case Artemis application would be designed to run in an outside environment, like a web browser or application. Having a secure API is imperative. This will define how third party software will interact within our own program. API will decide which methods will be acceptable or not as well as which data is acceptable or not. If we are reliant or dependent on third part software we must mitigate the risks associated with someone elses software. Securing our API is crucial.

**Cryptography –** This has been included as we do not know the status of using a form of international transfers. If we are to export data and transfer we need to do so in a secure, encrypted communication. If we do have these type of transfers the appropriate steps must be taken to secure data and it will need to comply with the laws of the places we are transferring to, both local and foreign.

**Code Error -** Similar to input validation, this goes along with the same idealology of how we would handle input validation. Any errors within the code like those that related to input validation will need to be handled appropriately. This will help mitigate issues within software and those with mal-intent that wish to abuse the system for their own advantage

**Code Quality -** When API and input are present Code quality should be an area of concern as well. With proper Code quality we can ensure that data is not unintentionally exposed. Code quality will also make sure methods that are restricted will be so from end users who are not supposed to have access to them. Ideally only authorized users will get the data/methods that match up with their authorization.

## 3. Manual Review

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

To start the very first thing I decided to look for was input validation. The Pom.xml file was where I started. I made sure to check for any type of validators. After checking there I was looking for places where there was a user input. The first file I found where this occurred was in the GreetingController.java lines 15-18. There was an input placed there that was assuming to be a string and it was not passed through a validator. This would potentially allow to enter various things. Unfortunately I could not determine if it was being passed as there was no output. Under The CRUDController.java lines 12-13 was another area that requested information, expecting a string. Again you could potentially pass various things to take and there was no validator or output.

After Input validation I checked for API and whether or not it might be present. The interesting thing is even though nothing turned up the program still took data. In this instance the program was taking data/commands in an unsecure way.

Following API, I checked for Cryptography. In our case there was no encryption from what I reviewed. In in the future artemis needs this for international communication we will need to develop a system for this.

After checking for encryption/cryptography I looked for error handling. There was an instance within DocData.java. In its class there was a method to read a document when it was passed a key/value. In this there was a try catch block but no particular expected error handling. I could not find anything else to evaluate within any other classes/files.

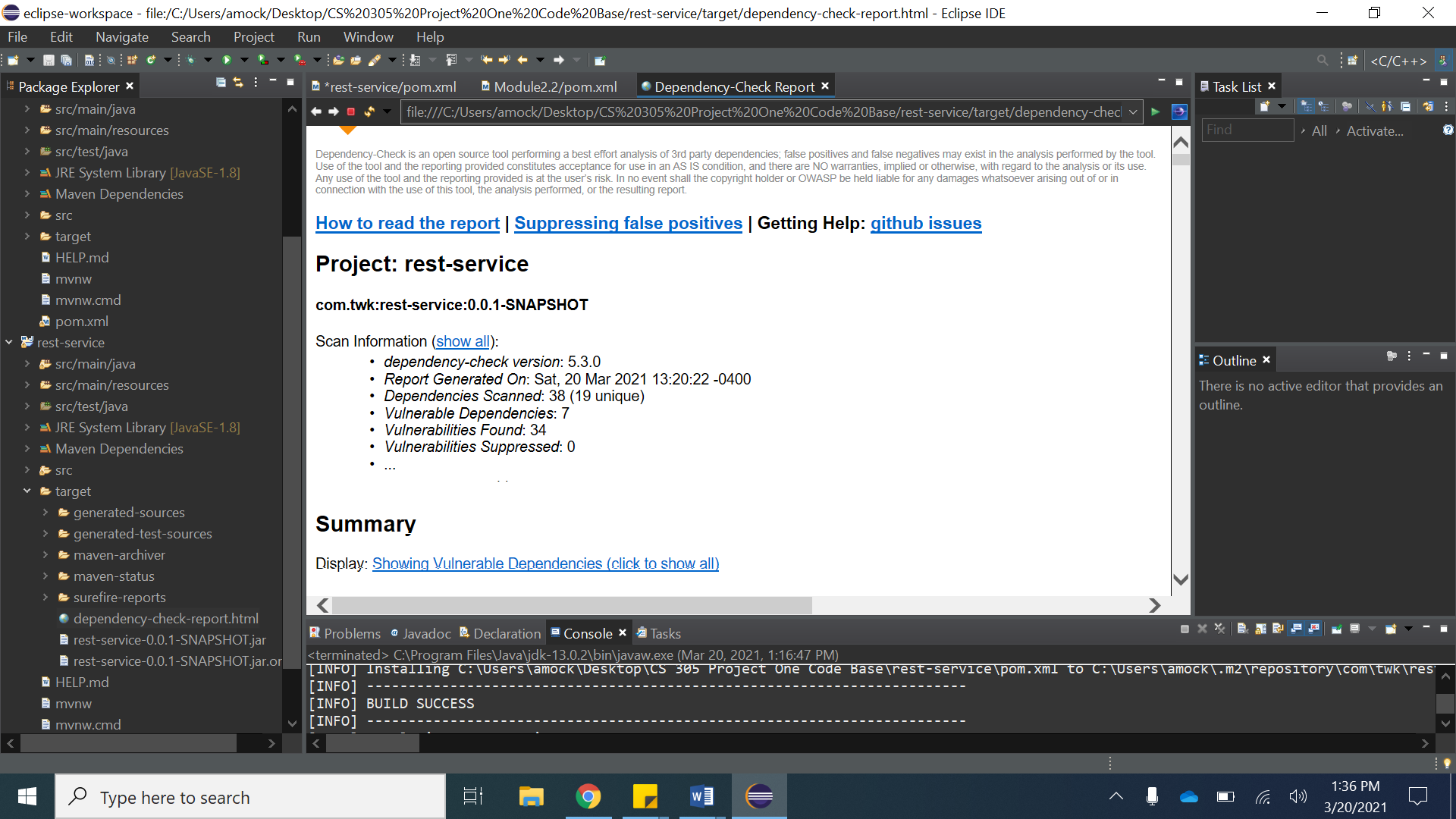
In the Final stage I looked into the code quality. It seems to be apparent that the code will need to be expanded apon in order to get the program fully functional and ready. Input validation will need to be implemented. API may need to be addressed. Code quality in general goes hand and hand with API and input validation.

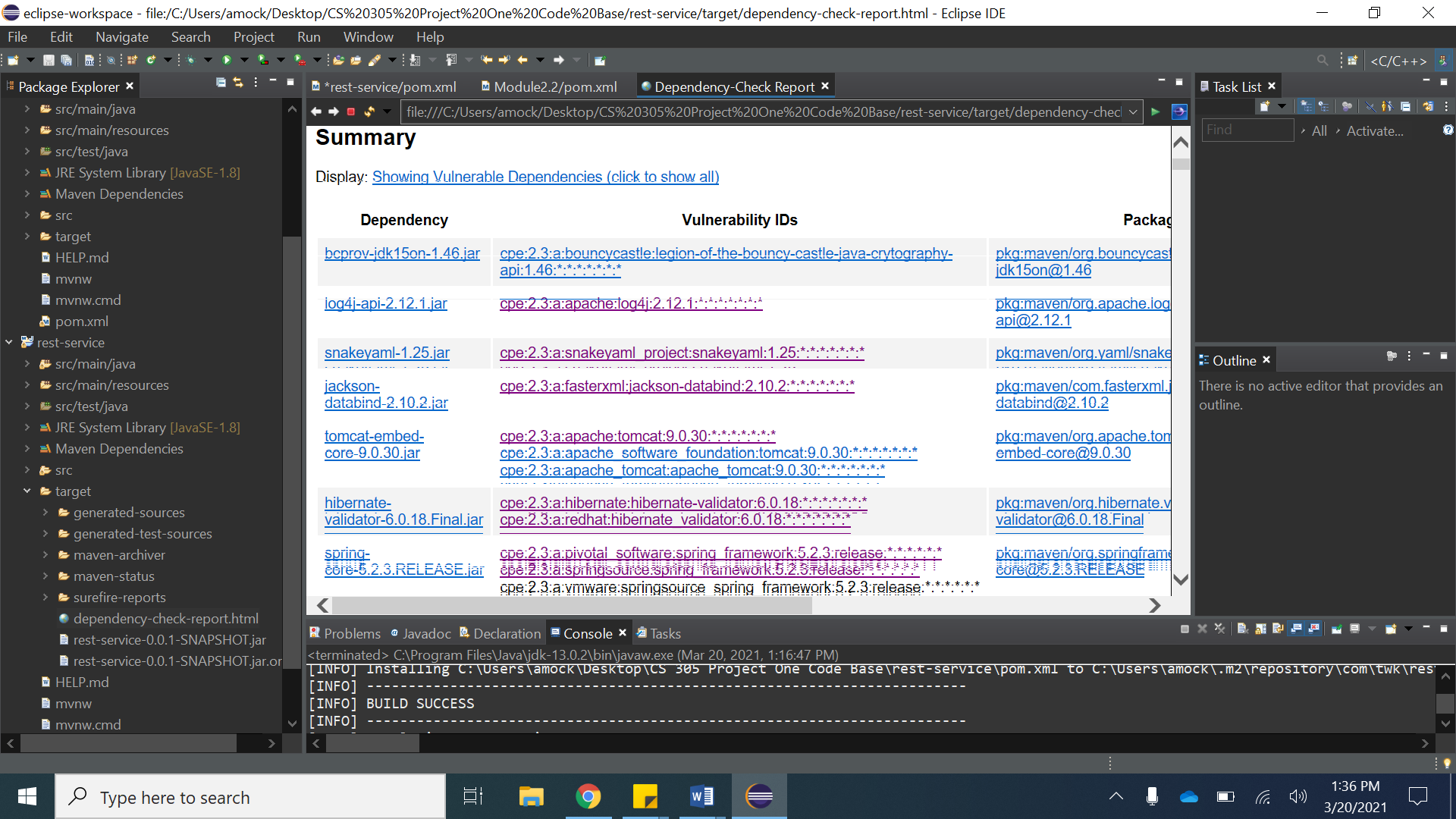
## 4. Static Testing

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

[Include your findings here.]





### 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.

#### Identifiers

* [pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46](https://ossindex.sonatype.org/component/pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress

**Note:** This could potentially have several issues. One really important one revolves around cryptography algorithms. This can impact secure communications and also expose sensitive data. This version(1.46) currently has a vulnerability where the second party DH public key is not fully validated and could potentially lead to other invalid keys to be used to show details about other party’s private key. In this case if another private key is found it renders encryption to be useless. Generally even exposure of the algorithm used can lead to finding keys and then those secure communications are not longer strictly private.

### log4j-api-2.12.1.jar

**Description:**

The Apache Log4j API

#### Identifiers

* [pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1](https://ossindex.sonatype.org/component/pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress

**Note:** Within this vulnerability, it is capable of impeding on SSL. The current certificate checks that are performed during a session could potentially not be validated at all or incorrectly. This in turn can allow software to connect to a potential malicious host while it renders/thinks its trusted source/host. If there is a lack of validation of the SSL certificate, this leaves this level of protection useless.

### snakeyaml-1.25.jar

**Description:**

YAML 1.1 parser and emitter for Java

#### Identifiers

* [pkg:maven/org.yaml/snakeyaml@1.25](https://ossindex.sonatype.org/component/pkg:maven/org.yaml/snakeyaml@1.25?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress

**Note:** Within this vulnerability there is a potential to lead to a Dos( denial of service) attack. The reason this can happen is because of the document definition type does not control the number of recursive definitions inside of it. This can lead to and cause exponential growth when the data itself is parsed. In this case Dos tends to happen when there is too much traffic coming in that the server cannot handle. After that it shuts down or doesn’t allow others access. This can be a critical issue and we must defend ourselves against these types of attacks. DDos and Dos attacks can be common so it is important we take precaution against this.

### jackson-databind-2.10.2.jar

**Description:**

General data-binding functionality for Jackson: works on core streaming API

#### Identifiers

* [pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2](https://ossindex.sonatype.org/component/pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress

**Note:** Within this vulnerability attackers can submit an XML file that defines some object outside of the defined scope within the app and proceed to have data sent back as an error. If abused this allows an attacker to view sensitive data if they submit something that accesses the files in the right way. Error message can be helpful and informative but in this case it might share too much information.

### tomcat-embed-core-9.0.30.jar

**Description:**

Core Tomcat implementation

#### Identifiers

* [pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30](https://ossindex.sonatype.org/component/pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress
* [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)  (*Confidence*:Highest)  suppress
* [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)  (*Confidence*:Highest)  suppress

**Note:** Within this Vulnerability, one part can be exploited to give unauthorized access to people that gives access to sensitive information. This happens when code inserts info directly, indirectly, or that the code manages resources that contains sensitive information and resources become available to unauthorized users. Also if a user can access a method that contains sensitive information they could potentially find this information. In this case they could potentially abuse this and gain more information on sensitive data that is not protected.

### hibernate-validator-6.0.18.Final.jar

**Description:**

Hibernate's Bean Validation (JSR-380) reference implementation.

#### Identifiers

* [pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final](https://ossindex.sonatype.org/component/pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [cpe:2.3:a:hibernate: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%3Ahibernate&cpe_product=cpe%3A%2F%3Ahibernate%3Ahibernate-validator&cpe_version=cpe%3A%2F%3Ahibernate%3Ahibernate-validator%3A6.0.18)  (*Confidence*:Highest)  suppress
* [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)  (*Confidence*:Highest)  suppress

**Note:** 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.

### spring-core-5.2.3.RELEASE.jar

**Description:**

Spring Core

#### Identifiers

* [pkg:maven/org.springframework/spring-core@5.2.3.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework/spring-core@5.2.3.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)  (*Confidence*:High)
* [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)  (*Confidence*:Highest)  suppress
* [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)  (*Confidence*:Highest)  suppress
* cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  (*Confidence*:Low)  suppress

**Note:** Within this vulnerability versions of spring up to version 5.2.8 are susceptible to RFD(reflected flie download) attacks. They can be bypassed by using the jssessionid path. This can be used to completely control the targeted system

## 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.]

Bcprov-jdk15-1.46.jar – In order to mitigate this threat we will need to update to a later version( 1.56 or later). Keeping all of our keys secure is the whole idea behind cryptography/encryption. This proves to be of great importance.

Jackson-databind-2.10.2.jar – in order to mitigate this threat we will need to configure the XML parser/validator. Ideally we should disable the expansion of the eternal entity. We don’t want the attackers to submit malicious DTD files within the XML documents. If done properly the parser should catch and ignore the malicious code.

Log4j-api-2.12.1.jar – In order to mitigate this threat we should not use certificate pinning and make sure all properties of the certificate are validated before pinning with focus on the hostname. Browsers will trust the SSL certificate and typically the end users of those browsers will too. We do not want to give a false sense of security when using our program.

Snakeyaml-1.25.jar – In order to mitigate this threat we should not allow the uses of DTDs or use a parser that limits expansion of recursive definitions. The issue with Recursion is it can generate a large loop. When using this method in code it should always be checked as it can cause issues. If we had a parser limit the expansion, this can stop these types of attacks. Ideally we want to mitigate the potential of DoS( denial of service)

Spring-Core-5.2.3.RELEASE.jar – In this current version there are no measures for mitigation listed. Ideally we could push to the latest version(5.3.1). Typically it is a good idea to use the most up to dated version, but this can also be a draw back as testing isn’t always there and there may be other flaws not yet known.

Tomcat-embed-core-9.0.30.jar/ tomcat-embed-websocket-9.0.30.jar – in order to mitigate these threats we should use the principle of least privilege along with safe zones. This will ideally allow sensitive information to remain in a safe area. We want to mitigate the possibility of data leaking, if we follow this the possibility will become less likely.