****

# 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** | **10.30.2021** | **Peter Lilley** | **First Review** |

## 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

Peter Lilley

## 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 value of secure communications for Artemis Financial is huge. Artemis Financial is a company that develops individualized financial plans for savings, retirement, investments, and insurance for their patrons. Now because they are managing peoples money, they have access to a lot off different kinds of sensitive personal information. Some of this information includes, bank account numbers and routing numbers, addresses, and social security numbers. Because of this Artemis Financial has both a moral and ethical responsibility to make sure that their client’s information is secure and kept out of the reach of hackers.
* Are there any international transactions that the company produces? Given the scope of Artemis Financial business capacity, financial plans for savings, retirements, investments, and insurance for their patrons coupled with using a RESTful API I would assume that their business is not limited to local national customers. While the scenario does not specifically state weather or not they are and international company given that they are on the web I would plan on them producing international transactions because they use a REST API their clients can access their information from anywhere around the world.
* Are there governmental restrictions about secure communications to consider? Yes, there are governmental restrictions about secure communications to consider when dealing with individuals’ personal financial information. Not only are they privileged to personal financial information, but Artemis would have addresses, credit history, phone numbers and bank account information all personalized for each account and client. Because of this there are specific benchmarks for security that Atriums must hit to ensure it is following the most reascent laws and regulations.
* What external threats might be present now and in the immediate future? The Artemis web application or App will allow users to access their information anywhere in the world at any time. This presents a multitude of threats both short term and long term. For the short term the various threats would include trojan horses, phishing attempts, malware, or ransomware-based attacks. While we cannot predict the future, we can do our best to plan for what we already know exists in the best way we can. There could be system threats as well for example with Windows 7 and 8 both being at EOF, if Artemis is using those versions, they would stand more prone to threats than if they used Windows 10 or 11 because Microsoft is no longer updating or providing support for Windows 7 and 8.
* What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies? Some modernization requirements that must be considered such as open-source libraries like REST and Spring Frameworks. Because these are open source, they can allow more users to access and discover vulnerabilities within the latest versions of the code. Allowing new version to be made more frequently thus protecting Artemis. Software updates as I mention previously with Windows are extremely important as well. Updating your systems frequently with the latest patches available allow us to prevent unseen potential attack on our system as well as protect us from threats that are out already.

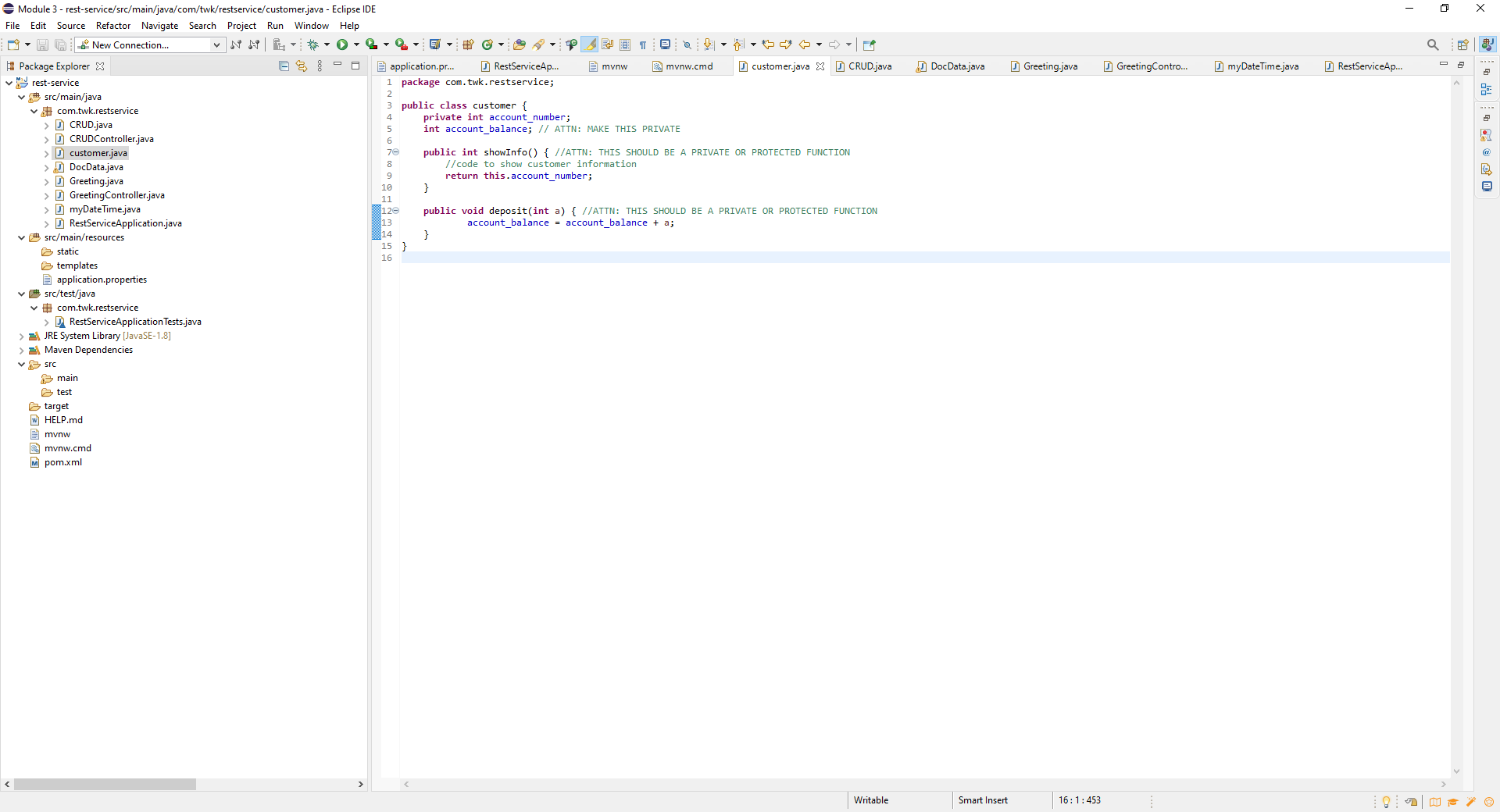
## 2. Areas of Security

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

* Input Validation: Every user of the application will have to validate their entry into the system. That will be true for clients trying to access their own information as well as secretaries updating customers information, IT managers changing infrastructure and whoever else decides they want to gain permission to use the application. Ensuring that the users’ credentials are creditable before they access the system will also make the customers feel like their information is protected. We can validate users in many ways from facial recognition, finger printer authorization, and two step authentication, just to name a few.
* APIs: The application programming interface or API has already been developed in this case and is a RESTful API. REST stands for representational state transfer, a RESTful API conforms to the constraints of REST formatting and allows for interaction with web services. Using the newest and most updated APIs will help ensure customers security and privacy.
* Cryptography: With almost all of the data being that is transmitted within the application containing sensitive information like social security numbers, address, financial statements, and banking records to name a few it is critical we do out best to maintain best coding practices. Data should be encrypted before ever being sent, for example if a new user creates their account through the app after they fill out all their information it should be encrypted before it is sent over the internet to ensure that the data not only stays safe but also falls within the legal parameters of acceptability. Doing this will help stop potential hackers from getting anyone’s information to easily.
* Client/Server: This is critical because the information can be uploaded from anywhere in the world there needs to be a very rigorous client server protection protocol in place to make sure that not only can I access my information from England to Mexico. But also, to make sure that no one can steal my information form Mexico to England. If we do not do the best, we can in this area we let our code be subject to hacking, hijacking, and SQL interference, just to name a few.
* I have left out code error, code quality and encapsulation, it is not because they are not applicable however I just feel as though they are not as applicable as the others I have listed. Inaccurate or incomplete login authentication will result in system crashes and would need to be check before they are entered into the system. I feel as though this would be common understanding for building a sign in or log in screen for the application. Encapsulation and code quality will apply to the data structures regarding how the data is kept and organized. Issues with these would be avoided because we use the best standard and practices for code development. Coding security systems is no easy task and always requires a great deal of detail from all aspects of coding.

## 3. Manual Review

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

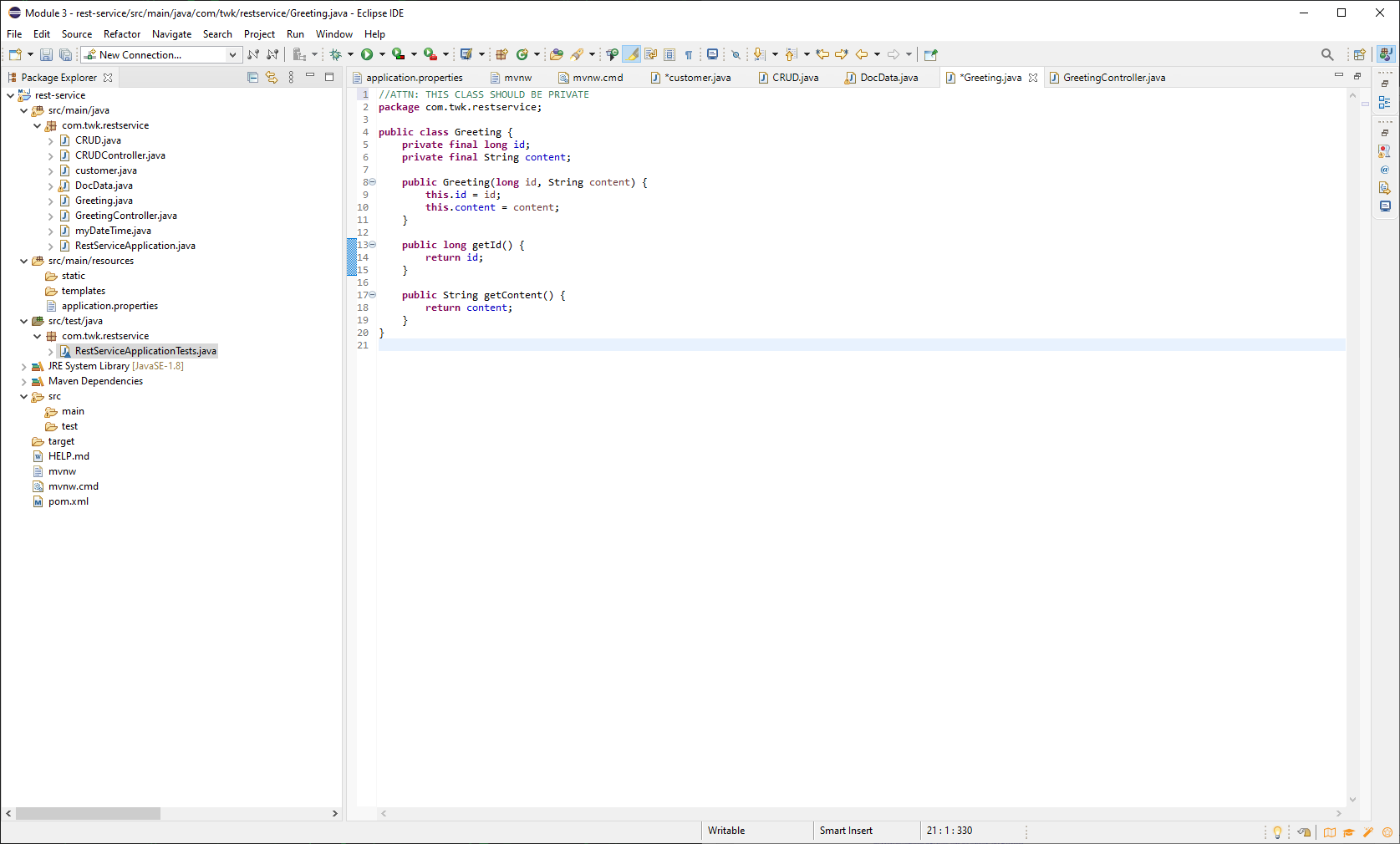


This is my first example of a couple of vulnerabilities withing the class “customer.java”. Here we have one private in “account\_number” where both “ints” should be private. Also both functions “public in showInfo and public void deposit” should be private.

Graphical user interface, text, application, email

Description automatically generated

This is my second example of a classes that needs to be addressed for security reasons. This “public class” should be private. This is because we are hiding return id variable with a private string. However, this does not mean that this information is protected because a hacker can still get into this class and return the id thus defeating the point of making the variable private anyway. To eliminate this, we must change the class to private.



This is another example of a class, “Greeting” that should be private as well. This also has the issue of public functions with private variables, so the hacker could still come into this class and steal valuable and useful data that would then allow them to gain access to other parts of the system.

## 4. Static Testing

Run a dependency check on Artemis Financials software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:

A screenshot of a computer

Description automatically generated

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

Vulnerability Codes:

1. Bouncy Castle Crypo Package

* Dependency: [bcprov-jdk15on-1.46.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)
* CVE Count: 17
* Severity: Unknown
* 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.

1. NO VULNERABLITY ID

* Dependency: [classmate-1.5.1.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l2_3fe0bed568c62df5e89f4f174c101eab25345b6c)
* CVE Count: 0
* Severity: BLANK
* Description: Library for introspecting types with full generic information including resolving of field and method types.

1. Redhat Hibernate Validator

* Dependency: [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)
* CVE Count: 1
* Severity: Medium
* Description: Hibernates Bean Validation (JSR-380) reference implementation. Pis for JDK 1.5 to JDK 1.7.

1. Faster XML Jackson Modules

* Dependency: [bcprov-jdk15on-1.46.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7), [jackson-databind-2.10.2.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)
* CVE Count: 1
* Severity: Unknown
* Descrition: 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. A flaw was found in FasterXML JKackson 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.

1. NO VULNERABLITY ID

* Dependency: [jakarta.annotation-api-1.3.5.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l6_59eb84ee0d616332ff44aba065f3888cf002cd2d), [jakarta.validation-api-2.0.2.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l7_5eacc6522521f7eacb081f95cee1e231648461e7)
* CVE Count: 0
* Severity: BLANK
* Description: This version of jakarta.validation-api has no known vulnerabilities!

1. NO VULNERABLITY ID

* Dependency: [jboss-logging-3.4.1.Final.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l8_40fd4d696c55793e996d1ff3c475833f836c2498)
* CVE Count: 0
* Severity: BLANK
* Description: This version of jboss-logging has no known vulnerabilities!

1. No VULNERABILITY ID

* Dependency: [jul-to-slf4j-1.7.30.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l9_d58bebff8cbf70ff52b59208586095f467656c30)
* CVE Count: 0
* Severity: BLANK
* Description: This version of jul-to-slf4j has no known vulnerabilities!

1. Apache Log 4j

* Dependency: [log4j-api-2.12.1.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)
* CVE Count: 1
* Severity: LOW
* Description: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender This could allow an SMTPS connection to be intercepted by a man in the middle attach which could leak any log messages sent through that appender.

1. Snakeyaml Project

* Dependency: [snakeyaml-1.25.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l13_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)
* CVE Count: 1
* Severity: HIGH
* Description: The Alias feature ion SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

1. Pivotal Software, Spring Source, Spring Framework

* Dependency: [spring-aop-5.2.3.RELEASE.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l14_9cdd9a1dd636331767fffcc7fe49a7bb00e7b34b), [spring-core-5.2.3.RELEASE.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l16_3734223040040e8c3fecd5faa3ae8a1ed6da146b)
* CVE Count: 2
* Severity: HIGH
* Description: In Spring Framework versions 5.2.0-5.2.8, 5.1.0-5.1.17,5.0.0-5.0.18,4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-52121 may be bypassed depending on the browser used through the use of a jsessionid path paramieter.

1. Apache Tomcat 9.0.30

* Dependency: [tomcat-embed-core-9.0.30.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l17_ad32909314fe2ba02cec036434c0addd19bcc580), [tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\user\Desktop\SNHU\SNHU%20Oct%20-%20Dec%202021\CS-305%20Software%20Security\Module%203\rest-service\target\dependency-check-report.html#l19_33157f6bc5bfd03380ebb5ac476db0600a04168d)
* CVE Count: 16, 17
* Severity: CRITICAL
* Description: Apache Tomcat 8.5.0 to 8.5.63, 9.0.0-M1 to 9.0.43 and 10.0.0-M1 to 10.0.2 did not properly validate incoming TLS packets. When Tomcat was configured to use NIO+OpenSSL or NIO2+OpenSSL for TLS, a specially crafted packet could be used to trigger an infinite loop resulting in a denial of service. A vulnerability in the JNDI Realm of Apache Tomcat allows an attacker to authenticate using variations of a valid user name and/or to bypass some of the protection provided by the LockOut Realm. This issue affects Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65. When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 could duplicate request headers and a limited amount of request body from one request to another meaning user A and user B could both see the results of user A's request. While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests. The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service. A Incorrect Default Permissions vulnerability in the packaging of tomcat on SUSE Enterprise Storage 5, SUSE Linux Enterprise Server 12-SP2-BCL, SUSE Linux Enterprise Server 12-SP2-LTSS, SUSE Linux Enterprise Server 12-SP3-BCL, SUSE Linux Enterprise Server 12-SP3-LTSS, SUSE Linux Enterprise Server 12-SP4, SUSE Linux Enterprise Server 12-SP5, SUSE Linux Enterprise Server 15-LTSS, SUSE Linux Enterprise Server for SAP 12-SP2, SUSE Linux Enterprise Server for SAP 12-SP3, SUSE Linux Enterprise Server for SAP 15, SUSE OpenStack Cloud 7, SUSE OpenStack Cloud 8, SUSE OpenStack Cloud Crowbar 8 allows local attackers to escalate from group tomcat to root. This issue affects: SUSE Enterprise Storage 5 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP4 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 12-SP5 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 15-LTSS tomcat versions prior to 9.0.35-3.57.3. SUSE Linux Enterprise Server for SAP 12-SP2 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 12-SP3 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 15 tomcat versions prior to 9.0.35-3.57.3. SUSE OpenStack Cloud 7 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud 8 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud Crowbar 8 tomcat versions prior to 8.0.53-29.32.1. When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103 if a) an attacker is able to control the contents and name of a file on the server; and b) the server is configured to use the PersistenceManager with a FileStore; and c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter="null" (the default unless a SecurityManager is used) or a sufficiently lax filter to allow the attacker provided object to be deserialized; and d) the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over; then, using a specifically crafted request, the attacker will be able to trigger remote code execution via deserialization of the file under their control. Note that all of conditions a) to d) must be true for the attack to succeed. In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led 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.

## 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 Financials software application.

After analyzing the Artemis Financial application, we can see that there are a few vulnerabilities that we should address. Some of these vulnerabilities can be fixed simply by updating the firmware and software of different API infrastructure like Spring.

Mitigation Plan

Input Validation:

* Hibernate Validation (CVE-2020-10693)
* Maven Dependcy, Hibernate Validator 6.0.18
* Private int for account\_balance
* Private Class DocData
* Private Class Greeting

APIs:

* Multiple APIs
* Tomcat embed core 9.0.3
* Tomcat websocket core 9.0.3
* spring-aop-5.2.3.Release
* spring-core-5.2.3 Release
* log4j-api-2.12.1
* jackson-databind-2.10.2.
* Make sure that when we are using an SPI that we are using the most up to date API version available, this will help insure our application is secure

Cryptography:

* Bouncy Castle Crypto Package 1.46
* Update API to ensure application is secure and an older version isn’t running the hackers could penetrate

Client/Server:

* Spring Boot Release 5.2.3
* Dependency Check maven using 6.4.1

Code Quality:

* Best industry Standard and Practices when writing and executing code
* Customer.java file – private variable, private Functions
* DocData.java file – private class to secure “return id”
* Greeting.java file- class should be private Functions should be protected.