****

# 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** | **1/20/22** | **Andrew McPherson** | **Version 1.0 is the initial entry of vulnerability assessment report.** |

## 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

Andrew McPherson

## 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 Communications:  
Since this company is working with financial plans, savings, retirement, investments, and insurance for their customers secure communications is paramount. These sorts of things are potentially very sensitive and important personal information, which must be secure and protected from attack as to not have this information leaked or exposed and potentially abused by outside sources.

International Transactions:  
It does not appear to be specified by this company if international transactions are going to occur or not, the company may be working internationally as no description of where they are geographically based is present, so it is better to assume that this can and will be performing internationally and prepare for this possibility accordingly.

Governmental Restrictions:  
If all transactions happen to be in one country, we simply must adhere to the restrictions of that country, however if this company does work internationally there may be a great many of restrictions that need to be considered when operating internationally, each from whichever country is included.  
When it comes to restrictions from the US Government, I found out about the Electronic Communications Privacy Act (ECPA), which basically “allows the US Government to access digital communications such as email, social media messages, and information on public cloud databases” (NortonOnline, 2018). This isn’t a restriction exactly, but it does mean that the government would be able to access their data if they so desired, so this company would need to be amenable to such laws.

External Threats:  
Due to the very nature of the internet, there will always be threats to consider, there may always be an attacker trying to steal sensitive or personal information for a myriad of reasons. Threats that are present now, may very well surface again even if protected against in the moment, so these sorts of things should be predicted and prepared for in the event of an inevitable attack. Problems that may be fixed now but could return would be like dependency vulnerabilities, which may be patched out and fixed, but could potentially have a resurfacing issue.  
Things like SQL Injection or attacks through improper input validation are the first to come to mind. Overall, threats are near constant, an ebb and flow of attacks and countermeasures and protection. Since this company is especially working with financial information, it may be an even larger target for attackers.

Modernization:  
Primarily the modernization requirements is simply keeping up to date with the current releases of software. Since Artemis wishes to implement the most current and effective software security, some of that security will simply be met by staying up to date. Open source, and open source software, comes with risks and problems of its own. It isn’t guaranteed to be the most reliable or safe method of use. If something were to go wrong with your software that is open source, there may not be any suitable fix any time soon, it isn’t monitored or maintained as closely and securely as something you might get with closed source. You lose that accountability when using open source. The pros and cons of using open source must be measured and considered, especially for something like this.

## 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.

Primarily the areas of security that are most applicable are Input Validation, APIs, and additionally Cryptography in the case of international transactions, but this is potentially unknown at this time.  
I imagine that Code Quality and Code Error is also important, but I also feel that goes without saying.  
The importance of input validation, or secure input, I find is crucial. The sensitive nature of the information being sent through this application demands a strong level of protection. There is also the possibility of misuse of input by attackers, so there is even more reason for secure input validation.  
As for APIs, this is meant to be a web application, probably accessed via a web browser, so it should have a good and reliable API for the application to use, I also believe this goes without saying.  
As said above, Cryptography is a potential route for security, especially through international transactions one would want data to be encrypted for secure transactions. If international transactions are not planned for this company, this may be omitted (though would not hurt to have anyways)

As for Code Error, since there is input validation, there could be error messages that print to the system output log. I also imagine it may be necessary as a part of input validation to confirm if input was accepted or not.

## 3. Manual Review

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

For Input validation, there does not seem to be any way for input to be validated in the code, and this may also go double for API as there does not seem to be any interface to work with in this application from what I have seen.

An issue with potential code quality found would be in line 12 and 16 of the Greeting.java class file.  
Text

Description automatically generated  
Here the return should be something like “return this.id;” and “return this.content;” as according to “Building REST services with Spring,” (n.d.).

For Cryptography, I do not see anything in this program that shows any form of encryption, though this may be intentional due to not being certain if the company wishes for international transactions, but if there were, there would need to be some form of encryption security.

For Code error, there was something easily noticeable in the DocData.java file, there is a function called read\_document on line 21 that has a try catch block, yet the catch portion (line 28) is incomplete and has a //TODO section.   
A screenshot of a computer

Description automatically generated

More incomplete code was found in myDateTime.java on line 9, but especially line 14 where there is an empty function  
Text

Description automatically generated with low confidence

## 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

* bcprov-jdk15on-1.46.jar:   
  cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46   
  cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46  
  cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46   
  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46
  + Description: Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for Bouncy Castle Cryptography APIs or JDK 1.5 to 1.7  
    The majority of problems and vulnerabilities for this dependency come from the current version of 1.46, but can be solved by updating to version 1.60 or higher
* hibernate-validator-6.0.18.Final.jar:  
  cpe:2.3:a:redhat:hibernate\_validator:6.0.18
  + Description: Hibernate’s Bean Validation (JSR-380) reference implementation.  
    A bug in the message interpolation processor enabled invalid EL expressions to be evaluated as if they were valid.  
    The problems with this vulnerability have been fixed in versions 7.0.0.Alpha2, hibernate-validator 6.1.5 Final, and hibernate 6.0.20.Final.
* jackson-databind-2.10.2.jar:   
  cpe:2.3:a:fasterxml:jackson-databind:2.10.2  
  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2
  + Description: General data-binding functionality for Jackson: works on core streaming API.  
    A flaw was found in the FasterXML Jackson Databind where it did not have entity expansion secured properly, allowing vulnerabilities to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.  
    The problems with this vulnerability have been fixed in the jackson-databind -2.11.0 and jackson-databind-2.10.5.1
* log4j-api-2.12.1.jar:  
  cpe:2.3:a:apache:log4j:2.12.1
  + Description: Apache Log4j API  
    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 attack which could leak any log messages sent through that appender  
    This vulnerability has been fixed by upgrading to version 2.13.2.
* logback-core-1.2.3.jar:  
  cpe:2.3:a:qos:logback:1.2.3
  + Description: logback-core module  
    In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configuration files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.  
    This vulnerability is fixed in versions 1.3.0 and 1.2. 9
* snakeyaml-1.25.jar:  
  cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25
  + Description: YAML 1.1 parser and emitter for Java  
    The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation.  
    This vulnerability is fixed in version 1.26
* spring-aop-5.2.3.RELEASE.jar:  
  cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  
  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  
  cpe:2.3:a:vmware:spring\_framework:5.2.3:release
  + Description: Spring AOP  
    In Spring Framework versions 5.2.0 – 5.2.8, 5.1.0 – 5.1.17, 5.0.0 – 5.0.18, and versions 4.3.0 – 4.3.28 and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used using a jsessionid path parameter.  
    For a solution, 5.3.x version users should upgrade to 5.3.14 or higher, and 5.2.x version users should upgrade to 5.2.19 or higher
* spring-core-5.2.3.RELEASE.jar:  
  cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release   
  cpe:2.3:a:springsource:spring\_framework:5.2.3:release   
  cpe:2.3:a:vmware:spring\_framework:5.2.3:release   
  cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release
  + Description: Spring Core  
    In Spring Framework versions 5.2.0 – 5.2.8, 5.1.0 – 5.1.17, 5.0.0 – 5.0.18, and versions 4.3.0 – 4.3.28 and older unsupported versions, the protections against RFD attacks from CVE-20155211 may be bypassed depending on the browser used using a jsessionid path parameter.  
    For a solution, the following Spring Framework versions should be used. 5.2.9, 5.1.18, 5.0.19, and 4.3.29.
* tomcat-embed-core-9.0.30.jar:  
  cpe:2.3:a:apache:tomcat:9.0.30   
  cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30   
  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30
  + Description: Core Tomcat Implementation  
    The refactoring present in Apache Tomcat versions 9.0.28 – 9.0.30, 8.5.48 - 8.5.50, and 7.0.98 – 7.0.99 introduced regression. 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.  
    This problem is fixed in the version higher versions of 9.0.31, or 9.0.36
* tomcat-embed-websocket-9.0.30.jar:  
  cpe:2.3:a:apache:tomcat:9.0.30  
  cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30   
  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30
  + Description: Core Tomcat implementation  
    The refactoring present in Apache Tomcat 9.0.28 – 9.0.30, 8.5.48 to 8.5.50, and 7.0.98 – 7.0.99 introduced a regression. 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.  
    This problem was fixed in version 9.0.36

## 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.

Implement some form of proper input validation, as there does not seem to be a way to confirm if input is valid or not.

As mentioned above under the manual review, some form of cryptography should be considered and implemented if international transactions are planned for this company, it could pose a serious problem if sensitive information was intercepted between overseas operations.  
  
Fixing and finishing incomplete code, as to not leave any empty functions or missing sections denoted with a //todo comment.

Based on findings from the static testing, all possible vulnerabilities from the dependency check have fixes to the problem simply by updating to and using higher and more specific versions. All versions that have these vulnerabilities identified have fixes already and have been listed above for each one. This is the easiest way to mitigate, by no longer using older versions of dependencies where issues have arisen or discovered workarounds to previous mitigation attempts.

References:

*Building Rest Services with spring*. Spring. (n.d.). Retrieved January 23, 2022, from https://spring.io/guides/tutorials/rest/

NortonOnline. (2018, January 18). What are some of the laws regarding internet and data security? Retrieved January 20, 2022, from https://us.norton.com/internetsecurity-privacy-laws-regarding-internet-data-security.html#:~:text=Electronic%20Communications%20Privacy%20Act%20(ECPA)&text=The%20law%20allows%20the%20U.S.,180%20days%20old%20or%20older.