****

# 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 5](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan 6](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/18/2021** | **Charles French** |  |

## 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

Charles French

## 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 is a high priority. While this may not entail things like voice over internet communications, the handling of sensitive banking information securely is vital. A leak of client’s banking information can not only be devastating for the customer, but it is also very destructive to the institutions reputation.

The prompt does not explicitly state that Artemis Financial handles international transactions. That said, with the company’s desire to modernize plus the fact that this is a web application, it can be assumed that international transactions take place, as anyone that knows the address for the website would be able to access the program, regardless of what country they are in.

As the company offers financial and investment advice, it would fall under jurisdiction of the Gramm-Leach-Bliley Act, and the web application would need to meet its data privacy requirements. Other countries will have similar regulations to review.

External threats will entail individuals or groups looking to gain access to customers accounts for monetary gain. Other possible, but less likely threats would include DDoS attacks to bring down the application temporarily.

Modernization can include a variety of options, such as a smooth running user interface tailored for a mobile device, but more importantly the interface should have a modular design so that it is easier to update it regularly. Use of open source libraries can increase the “wow factor” for a portion of the customer base, but other libraries shouldn’t be omitted just for the sake of a label – the best tool for a job should be used. Open source has the benefit of community support and often regular releases, but other licensed ones will as well. Even in-house built ones can have a planned update schedule. From a cost perspective, open source is an ideal choice, but a custom built library (even if it is just a fork of an open source library with some changes made) will give more versatility.

## 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 will be applicable due to the need to check against a database for matching information, as well as to prevent attacks that leverage user input, such as buffer overflow or SQL injection.

API interaction will be applicable as the program will need to be structured so that data doesn’t “leak” during utilization of the REST API.

Cryptography will be applicable as user information, both server and client side, will need to be encrypted throughout travel to both hide sensitive information and comply with government regulations.

Encapsulation will be applicable as the data structures of the database and the way the program interacts with the database will need to be able to handle data without “knowing” what the data is in order to prevent the leak of sensitive information.

## 3. Manual Review

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

CRUDController.java, line 13: passing a string as a parameter without validating input. This can lead to buffer overflow or SQL injection attacks.

DocData.java, line 27: root username and password are not only left as “default” values (“root” for both username and password), but are in plain text in the code (even written out and clarified in line 32!).

GreetingController.java, line 16: passing a string as a parameter without validating input, leading to vulnerabilities seen above in CRUDController.java.

## 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 – CVE-2013-1624: This is a vulnerability to timing attacks, where an attacker can essentially decode encrypted information one character at a time based on the time it takes the database to validate the “guess” the attacker used (ie. If more characters are correct and in order, the database takes longer to respond due to reading and checking the correct characters before finding an incorrect one)(ropesec.com).

log4j-api2.12.1.jar – CVE-2020-9488: Vulnerable to incorrectly validating certificates, opening up the possibilities of man-in-the-middle attacks.

Snakeyaml-1.25.jar – CVE-2017-18640: Vulnerable to XML bomb type attacks due to not limiting recursive references (i.e. Billion laughs attack)(Wikipedia, 2021).

Jackson-databind-2.10.2.jar – CVE-2020-25649: Vulnerable to XML external entity attacks, which can lead to data leaks or denial of service. (OWASP)

Tomcat-embed-core-9.0.30.jar – CVE-2019-17569: Vulnerable to HTTP request smuggling, allowing the bypass of typical security measures and leaking of data. (portswigger.net)

hibernate-validator-6.0.18.Final.jar – CVE-2020-10693: Vulnerability involving the ability to bypass input sanitation, opening the program to SQL injection style attacks.

spring-core-5.2.3.RELEASE.jar and spring-jcl-5.2.3.RELEASE.jar – CVE-2020-5421: safeguards against reflected file download attacks have been found to be able to be bypassed, allowing an attacker to gain control over the 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.

To mitigate the found vulnerabilities, first, CRUDController.java and GreetingController.java should be revised to validate and sanitize input before using it as a parameter. A function should be made to do this, and the function call used as the parameter in the previously mentioned lines in section 3. Next, in DocData.java, objects should be made to encapsulate the username and password of the database, and the file revised to not have the two be hard coded into the system. The comment advising of the username and password should especially be removed as well.

CVE-2013-1624: To combat against a timing attack vulnerability, the authentication function should have utilize a constant time procedure, such as adding a delay or implementing rate limiting into the system to safeguard from this brute-force style of attack. (ropesec.com)

CVE-2020-9488: Setting the system property mail.stmp.ssl.checkserveridentity to true will enable verification across the system, mitigating this vulnerability. (Sicker, 2020)

CVE-2017-18640: Reading the YAML before using it, such as counting the number of “\*” or “&” characters and rejecting the file if an excessive number of them are found, will mitigate the issue. (bitbucket.org, 2020)

CVE-2020-25649: Blocking declaration of DOCTYPE through use of the factory.setFeature() function will mitigate this issue.(github.com, 2021)

CVE-2019-17569: Updating tomcat to version 9.0.31 patches this vulnerability, mitigating the issue. (opensuse.org, 2020)

CVE-2020-10693: If implementing ConstraintValidator, do not disable the default, but rather utilize context.unwrap() when validating input. Alternatively, and update to version 6.0.20.Final mitigates the issue as well. (Smet, 2020)

CVE-2020-5421: Updating Spring to version 5.2.9 will mitigate the issue. (vmware.com, 2020)

**Citations**

*CVE-2020-5421: RFD Protection bypass via jsessionid: Security*. VMware Tanzu. (2020, September 17). Retrieved September 19, 2021, from https://tanzu.vmware.com/security/cve-2020-5421.

FasterXML. (n.d.). *`DOMDeserializer`: SetExpandEntityReferences(false) may not prevent external entity expansion in all cases [cve-2020-25649] · Issue #2589 · FASTERXML/JACKSON-DATABIND*. GitHub. Retrieved September 19, 2021, from https://github.com/FasterXML/jackson-databind/issues/2589.

*Oss-Security - Re: CVE REQUEST: TLS CBC padding timing flaw in various SSL / TLS implementations*. Openwall. (2013, February 5). Retrieved September 19, 2021, from https://www.openwall.com/lists/oss-security/2013/02/05/24.

Sicker, M. (2020, April 13). *Add support for specifying an SSL configuration for SmtpAppender*. [LOG4J2-2819] add support for specifying an SSL configuration for SMTPAPPENDER - Asf jira. Retrieved September 19, 2021, from https://issues.apache.org/jira/browse/LOG4J2-2819.

Smet, G. (2020, May 7). *Hibernate validator 6.1.5.Final And 6.0.20.Final released*. In Relation To. Retrieved September 19, 2021, from https://in.relation.to/2020/05/07/hibernate-validator-615-6020-released/.

*What is a Timing Attack Vulnerability?* What is a timing attack vulnerability? (n.d.). Retrieved September 19, 2021, from https://ropesec.com/articles/timing-attacks/.

*What is HTTP REQUEST smuggling? Tutorial & Examples: Web SECURITY ACADEMY*. What is HTTP request smuggling? Tutorial & Examples | Web Security Academy. (n.d.). Retrieved September 19, 2021, from https://portswigger.net/web-security/request-smuggling.

*Wiki*. Bitbucket. (2020, April 21). Retrieved September 19, 2021, from https://bitbucket.org/asomov/snakeyaml/wiki/Billion%20laughs%20attack.

Wikimedia Foundation. (2021, May 23). *Billion laughs attack*. Wikipedia. Retrieved September 19, 2021, from https://en.wikipedia.org/wiki/Billion\_laughs\_attack.

*XML external Entity (XXE) PROCESSING*. OWASP. (n.d.). Retrieved September 19, 2021, from https://owasp.org/www-community/vulnerabilities/XML\_External\_Entity\_(XXE)\_Processing.

*[Security-Announce] opensuse-su-2020:0345-1: Important: Security update for tomcat*. openSUSE Mailing Lists. (2020, March 15). Retrieved September 19, 2021, from https://lists.opensuse.org/archives/list/security-announce@lists.opensuse.org/message/F3FOVKJAK2YR7UVBYBATR7JKLD5IA6WI/.