****

# 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** | **7/17/2022** | **Dakota McDonough** | **Original** |

## 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

Dakota McDonough

## 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?

It is crucial for Artemis Financial to have secure communication. The company deals with money and finances. This is of value for any hacker. Secure communications specifically relates to make sure that the information being exchanged between the host server and the user is not being leaked, accessed, stolen, or otherwise tampered with (Secure Communcations, 2022). Hackers might also go after the personal information kept for the clients of Artemis Financial or a person that is disgruntled against the company may try to hack the system as a form of revenge. Another possible attack could come from competing firms wanting to learn how Artemis Financial does their individualized planning. All of these external threats could lead to lasting damage for the company, its image, and its clients. Hackers are also advancing every day, so what is safe today may become unsecured in a year’s time. That’s why we need to constantly be vigilant and improving the security of the system.

There is definitely the possibility for the company to produce international transactions. A user could access their account in a different country. Also, we do not know who the client base for Artemis Financial is. They could very easily have overseas clients.

As far as government restrictions go, we would most likely be falling under the Graham-Leach-Bliley Act (Kim Manico, 2014) specifically in regard to disclosing information-sharing practices and what steps we take to protect any of the clients’ data. We may also have to deal with the payment card industry data security standard, PCI-DSS, as it’s not 100% clear in the scope of the project whether Artemis Financial will be storing, processing, or using credit cards within the software. Lastly, if we want to allow access from the European Union, we would need to adhere to the EU General Data Protection Regulation in regards to how personal data is allowed to be transferred and processed within member countries (Union, 2018).

Using open-source libraries and other evolving application technologies has become critical for staying cutting edge in any industry, keeping costs down on system development, and keeping development times at a minimum. While amazing on the production side of things, they present a security issue, thus we must scrutinize ever third-party source we utilize to make sure it’s not giving a back door entrance to those we would rather keep out. Thus, we should utilize the OWASP dependency checker as well as utilizing other security testing tools.

## 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, APIs, Cryptography, Client/Server, Code Error, Code Quality, and Encapsulation are all areas of security that are applicable to Artemis Financial’s software application. Input validation is any for of data entering the system and checking to make sure it is valid. This will been seen a lot with user logins to access the account. They already have an API, so we are more concerned with making sure that the API interactions with the system are also secure. Cryptography is crucial towards making sure that the client data is encrypted. This is required by the government restriction requirements we are expecting to have to meet. Client/Server is related to the API. We would want to double check that proper certification is being used during data transfer to ensure it is being protected fully. Code Error is focusing on making sure that all of the input functions and access layer code are properly reviewed so no avenues are left open for potential attacks. Code quality is ensure that we stop problematic problems from being used in the various input types we have, like not putting maximum lengths on usernames or not putting character type restrictions wherever possible. Ideally, we would have a Quality Assurance Engineer perform this testing. Encapsulation is necessary for us to make sure that data is being stored properly and not being leaked, as well as making sure that our input functions are not messing up the data structures themselves when accessing the system.

## 3. Manual Review

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

In the DocData.java portion of the code, specifically the public read\_document function, the location of where the data is being obtained from has the username and password as part of its name. This is extremely dangerous as someone could gain access to the system using a brute force attack. It would be as simple as running through the possible combinations of where to break up the last two portions of the location to get the username and password. This needs to be done a different way.

In the CRUDController.java file is an issue as it exposes internal data. It is accessing and then changing an internal data structure. It also passes the business\_name through the function, exposing that specific DocData document to access vulnerability.

## 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

The following are the names of the vulnerabilities found:

* Bouncy Castle, CVE-2016-1000338
* Apache Log4j SMTP Appender, CVE-2020-9488
* Core Tomcat, CVE-2020-1938

All of the falling information was obtained through the National Vulnerability Database (Commerce, 2022). This database was also used to screen out vulnerabilities that are false positives.

The Bouncy Castle vulnerability enables a malicious program to access confidential data that is saved within the system database. It is suggested that regular updates of this program across all of the operating systems be done to keep the application security from being compromised. (Hat, 2018)

The Apache Log4j SMTP appender vulnerability allows malicious software to reveal/access log messages if they are sent through the appender. It is recommended that we update to the newest version, 2.13.2, which has a built-in verification system. (Sicker, 2021)

The Core Tomcat vulnerability causes memory to be used rapidly. This can cause the system to crash. It is recommended that an updated version is used as the vulnerability and the dependency would be addressed. (BSRT-2020-001 Local File Inclusion Vulnerability in Apache Tomcat Impacts BlackBerry Workspaces Server and BlackBerry Good Control, 2022)

## 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.

From the manual code review, our mitigation plan needs to include changing the DocData.java file so that the read\_document function no longer reveals the username and password in the location. We should also update the requirements for our usernames and passwords to mitigate a brute force attack. The other area of improvement is in the CRUDController.java file. When building this file, we should employ proper code security techniques. The functions in this file should not be directly changing the data within the data structures directly. It needs to be run through proper authentication and error catching safety nets first.

Regarding our dependencies, we need to update the Apache Server and the Core Tomcat to the newest version. The Bouncy Castle dependency should also be updated to the latest version and a procedure should be put in place for maintaining regular updates of this program across all of the companies operating systems. This procedure should include charging a member of the team with running the update after a specific interval of time and documentation that this update was run successfully.

# References

*BSRT-2020-001 Local File Inclusion Vulnerability in Apache Tomcat Impacts BlackBerry Workspaces Server and BlackBerry Good Control*. (2022, 5 30). Retrieved from BlackBerry: https://support.blackberry.com/kb/articleDetail?articleNumber=000062739

Commerce, U. D. (2022, 7 17). *National Vulnerability Database*. Retrieved from National Institute of Standards and Technology: https://nvd.nist.gov/

Hat, R. (2018, 9 11). *RHSA-2018:2669 - Security Advisory*. Retrieved from Red Hat Customer Portal: https://access.redhat.com/errata/RHSA-2018:2669

Kim Manico, A. D. (2014). *Iron-Clad Java.* McGraw Hill Computing.

Oracle. (2022, 1). *Secure Coding Guidelines for Java SE*. Retrieved from Oracle: https://www.oracle.com/java/technologies/javase/seccodeguide.html

OWASP. (2022, 06 12). *Dependency-Check Maven*. Retrieved from JerenyLong GitHub: https://jeremylong.github.io/DependencyCheck/dependency-check-maven/index.html

*Secure Communcations*. (2022, 7 17). Retrieved from Defintions.net: https://www.definitions.net/definition/secure+communication

Sicker, M. (2021, 12 27). *Add support for specifying an SSL configuration for SmtpAppender*. Retrieved from Apache: https://issues.apache.org/jira/browse/LOG4J2-2819

SNHU. (2022, 7 17). *CS 305 Flowchart.* Retrieved from CS-305-T6614 Software Secutiry 22EW6: https://learn.snhu.edu/content/enforced/1116000-CS-305-T6614-OL-TRAD-UG.22EW6/course\_documents/CS%20305%20Vulnerability%20Assessment%20Process%20Flow%20Diagram.pdf?\_&d2lSessionVal=ICMihLhdQqYNXcuGAG266Mcy0&ou=1116000

SNHU. (2022, 7 17). *CS 305 Integrating the Maven Depenency Check Plug-in Tutorial.* Retrieved from CS-305-T6614 Software Secutiry 22EW6: https://learn.snhu.edu/content/enforced/1116000-CS-305-T6614-OL-TRAD-UG.22EW6/course\_documents/CS%20305%20Integrating%20the%20Maven%20Dependency%20Check%20Plug-in%20Tutorial.pdf?\_&d2lSessionVal=ICMihLhdQqYNXcuGAG266Mcy0&ou=1116000

SNHU. (2022, 7 17). Java Software Application. *CS 305 Porject One Code Base. Zip*. SNHU.

Turpin, K. (2010, November). *OWASP Secure Coding Practices-Quick Reference Guide.* Retrieved from OWASP: https://www.oracle.com/java/technologies/javase/seccodeguide.html

Union, E. (2018, 5 25). *European Union - Data Privacy and Protection*. Retrieved from Privacy Shield Framework: https://www.privacyshield.gov/article?id=European-Union-Data-Privatization-and-Protection#:~:text=The%20EU%20General%20Data%20Protection,to%20companies%20of%20all%20sizes.