**CS 305 Project One - Artemis Financial Vulnerability Assessment Report**

Charles M. Haines II

Department of Computer Science: Southern New Hampshire University

CS-305: Software Security

Dr. Vivian Lyon

July 9, 2022



# 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** | **07/16/2022** | **Charles Haines** | **Completed Document** |

## 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 M. Haines II

## 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?

As a financial corporation, secure communications are essential for Artemis’ day to day business operations and overall reputation. Ensuring that the web application uses secure communication protocols and API’s is considered mission critical. If Artemis plans to conduct international transactions via the new web-based software, it is important that Artemis follows all government regulations both foreign and domestic. It may not apply to Artemis unless they handle direct deposits, but my research has identified that there are strict regulations governing international direct deposits. Any US company trying to deposit money into a foreign account is required by law to report the transaction information to the Office of Foreign Assets Control (OFAC) (*Will your direct deposit leave the U.S.?,* n.d.).

The Gramm-Leach-Bliley act requires that all companies that sell financial services or products disclose their privacy practices in addition to taking steps to protect confidential information (*Gramm-Leach-Bliley Act,* 2022). Since this law applies to Artemis, it is important that the new web application takes care to ensure their financial data is stored and transmitted securely. Attackers may try to access data that is stored on the server as well as intercepting data that is being transmitted from the server to client software.

Denial of service and ransomware attacks are two other popular cyber-attacks that Artemis must consider when building the new web-based software application. It is important that Artemis considers security as a paramount concern throughout the development process and while the application is in production. While the development team can do everything they can to protect the application and make it as secure as possible, no application is immune to security vulnerabilities. The application developers will build security logging into the app to keep track of attacks, so it is important that Artemis has a security team that is constantly reviewing security logs and is prepared with a plan of action in the event of a security breach.

The use of open-source libraries that are trusted to be safe and secure can be a good option for companies, however, it is imperative that any open-source libraries used are reviewed by the development team and constantly monitored for any emerging security vulnerabilities that are discovered in the future. I would discourage Artemis from using any newly emerging web application technologies until they are matured enough to be considered secure by the development community and other major corporations who are knowledgeable about web and application security.

## 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.

Cryptography is the most important area of security that the development team should be concerned with. Since Artemis is a financial company, the data that the company stores and maintains is considered confidential and as discussed in the first section of this document, Artemis has a legal obligation to protect confidential information. Cryptography is the practice of encrypting data with a private key using a secure algorithm or cypher making the data unreadable to anyone who does not possess the key to decrypt the data (Manico & Detlefsen, 2015). It is important that Artemis encrypts all customer data and ensures that any data being transmitted is encrypted prior to sending the transmission.

Application Programming Interfaces (APIs) are the second most important security area that the new web application will need to focus on to ensure the privacy of sensitive data. If the web application makes use of any APIs, Artemis needs to make sure that the API’s are secured since there is a high likelihood that the data being transmitted via API is sensitive. The use of secure APIs in addition to encryption will give Artemis the best possible chances of keeping their customer’s data secure. If the new web application does not make use of APIs, then the development team should keep their focus on ensuring that all data is encrypted using a strong encryption technology.

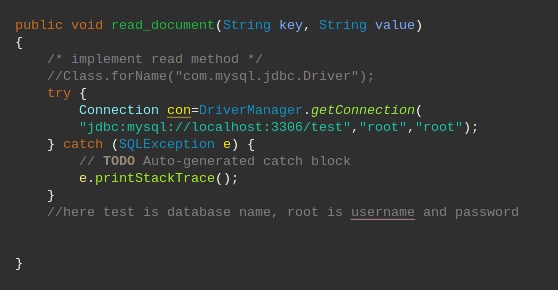
Another area of security is called input validation. This simply means that a program should validate any untrusted data as it is entering a system and defining rules concerning what types of input should be allowed in each situation (Manico & Detlefsen, 2015). Since the web application will accept input in various forms, it is important that the developers establish a way to check the validity of any input before it is transmitted or otherwise used in any way since entering malicious input is a tool that an attacker can use to compromise a piece of software.

Client / Server is another area of security that should be taken into consideration. If the web application is to be hosted on a server, the server should be secured and kept up to date to ensure that important security patches are installed as they are developed by the operating system manufacturer.

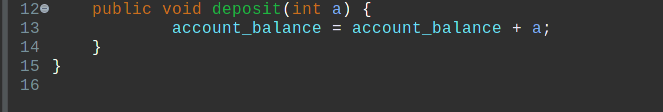
Lastly, the security areas of code quality and encapsulation are important for the new web application because code quality and encapsulation are the keys to creating any reliable piece of software. Since not writing quality code can lead to bugs and bugs can lead to security vulnerabilities as well as causing the development team to need longer periods of time to review and fix any identified vulnerabilities, it will save the company a lot of trouble and money overall to ensure that all code written is high quality. I included encapsulation because encapsulation is a straightforward way to keep code simple and generally considered a best practice among developers.

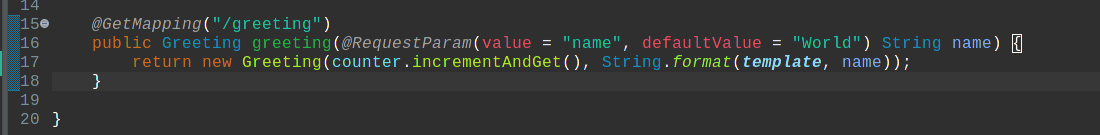
## 3. Manual Review

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



## There is a serious security issue with the code located in the DocData class. In the function read\_documents() located on lines 21 – 35, a database connection string is hard-coded into the function. Database credentials for the root account are stored in plain text in the connection string. Even an untrained eye would find this considering there is a comment advertising that the root account credentials are in the connection string. While it may seem safe to store these credentials in the source code, there are many reasons this is not a safe practice. If a developer decides to email some of the source code to another developer, some of the code gets pushed to a public source control repository, or an attacker decompiles the binary, an attacker could use these credentials to gain full control of the database (*How to keep secrets secret, n.d*). Prompting the user for the database credentials after removing the hard-coded credentials will solve this issue and make the application more secure.

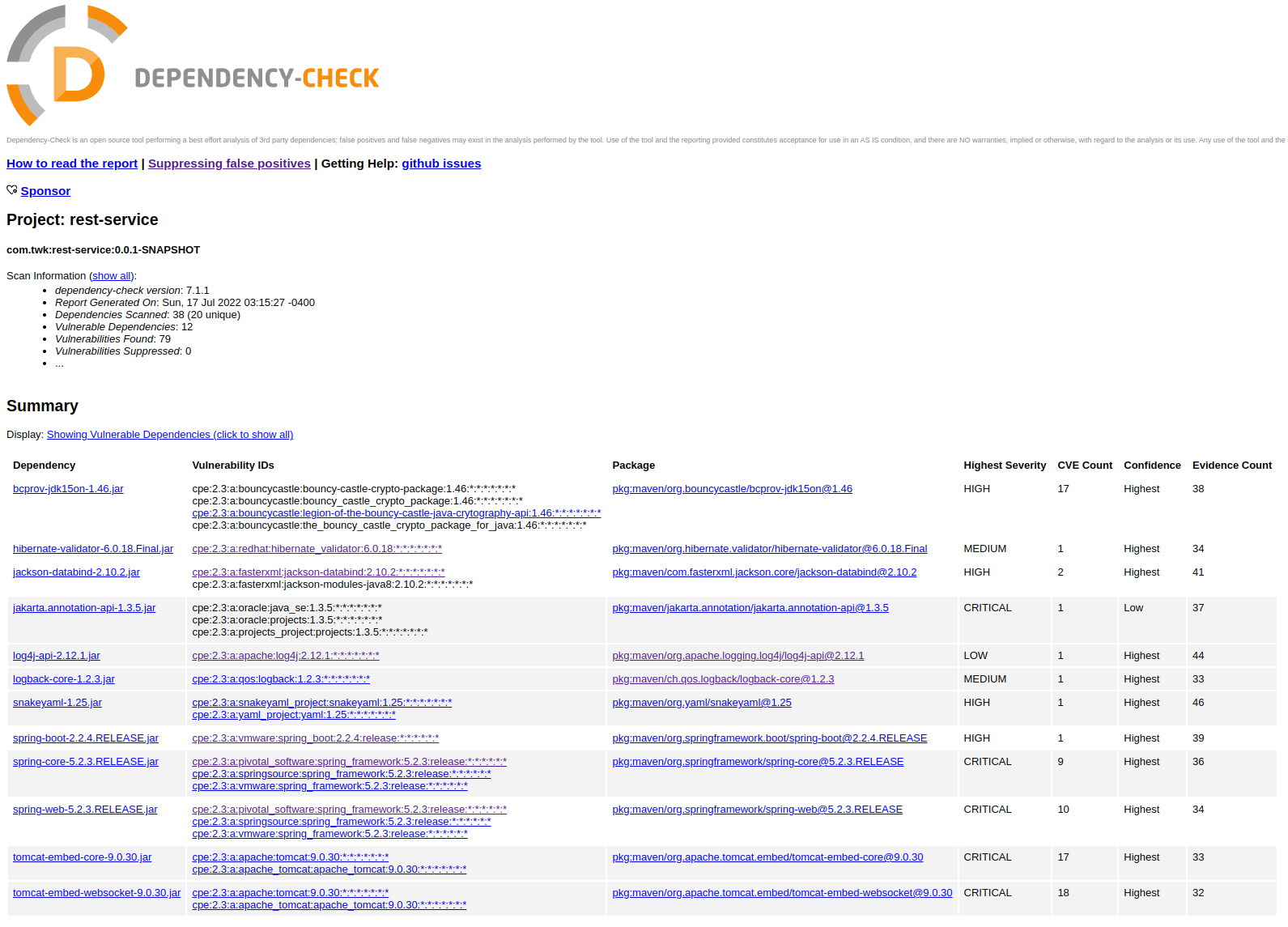
In the function deposit(int a) located on lines 12 – 16 of the customer class, the function adds the value of the integer a to another integer account\_balance. The function does not check the value of account\_balance before adding the deposit amount. The value of the integer variable ‘a’ could potentially overflow the integer variable account\_balance and cause it to roll over to a negative number (Nidecki, 2020). Simply checking that the value will not exceed the maximum value an integer can hold prior to performing the operation would prevent the integer from overflowing.

In the GreetingController class, a function mapped to /greeting on the web application accepts a name and then adds it to a greeting without doing any form of input validation. Concatenating strings without performing any sort of input validation is dangerous and is a security concern (Manico & Detlefsen, 2015). The string should be checked to ensure it is not empty or too long as well as not containing any symbols that could be misinterpreted by the server or client software.

## 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:

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



In the dependency bouncy castle crypto package, the dependency scanner identified several security vulnerabilities. CVE-2016-1000338, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000352, CVE-2016-1000341, CVE-2016-1000345, CVE-2017-13098, CVE-2020-15522, CVE-2020-0187, CVE-2016-100339, CVE-2020-26939, CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, and CVE-2015-6644 are known vulnerabilities of the library.

Updating the library to the latest version, 1.7.0, will resolve all the above listed vulnerabilities.

In the dependency hibernate-validator-6.0.18.Final.jar, the vulnerability CVE-2020-10693 was identified. This vulnerability could allow invalid expressions to be evaluated and could allow attackers to bypass input validation.

Updating to the latest version, 8.0.0.CR1 will resolve the security issue identified in the dependency check report.

The dependency jackson-databind-2.10.2.jar has two identified vulnerabilities, CVE-2020-25649 and CVE-2020-36518. Updating to the latest version of the library, 2.13.3 will resolve the issue identified by the report. More information about the discovery of this vulnerability can be located in this bug report: <https://bugzilla.redhat.com/show_bug.cgi?id=1887664>

In the dependency jakarta.annotation-apo-1.3.5.jar, CVE-2022-31569 was identified. A function for sending files is used in an unsafe manor. Updating to a version released on the author’s source repositor after 04-03-2022 will resolve the issue. Information about the discovery of the vulnerability can be found at the following GitHub discussion:

[https://github.com/github/securitylab/issues/669#issuecomment-1117265726](https://github.com/github/securitylab/issues/669" \l "issuecomment-1117265726)

In the dependency log4j-api-2.12.1.jar, CVE-2020-9488 was identified. Improper validation of a certificate with a host mismatch could lead to a man-in-the middle attack. This vulnerability was fixed in versions 2.12.3 and 2.13.1. More information about the origins of this vulnerability can be found in this JIRA ticket on Apache’s website: <https://issues.apache.org/jira/browse/LOG4J2-2819>

In the dependency logback-core-1.2.3.jar, CVE-2021-42550 has been identified. This is a medium risk vulnerability due to the attacker needing privileges to modify configuration files to exploit. Versions of the library after 1.2.7 have resolved this issue. <https://github.com/cn-panda/logbackRceDemo> contains a demo of the vulnerability.

In snakeyaml-1.25.jar, CVE-2017-18640 has been identified. The bug has been fixed in versions 1.26 and newer.

Spring-boot-2.2.4 has vulnerability CVE-2022-27772. This bug can allow temporary directory hijacking. This version of spring-boot is no longer supported by the developers and should not be used. Updating to the latest version of spring-boot will alleviate this issue. This bug was discovered by @trugPa on GitHub and further information about its discover can be found here: <https://github.com/JLLeitschuh/security-research/security/advisories/GHSA-cm59-pr5q-cw85>.

Spring-core-5.2.3 has the following vulnerabilities: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, and CVE-2021-22096. Updating to spring-core 5.3.22 will resolve the vulnerabilities identified in the dependency vulnerability report.

Spring-web-5.2.3 has the following vulnerabilities: CVE-2016-1000027, CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, and CVE-2021-22096. Updating to the latest version of spring web will resolve these vulnerabilities. The latest version of spring-web is 5.3.22.

tomcat-embed-core-9.0.30 has the following vulnerabilities: CVE-2020-1938, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-17527, CVE-2021-25122, CVE-2021-41079, CVE-2022-29885, CVE-2020-9484, CVE-2021-25329, CVE-2021-30640, CVE-2022-34305, CVE-2021-24122, CVE-2021-33037, CVE-2019-17569, CVE-2020-1935, and CVE-2020-13943. Updating to version 10.1.0-M16 will resolve the identified vulnerabilities.

Tomcat-embed-websocket-9.0.30 has the following vulnerabilities: CVE-2020-1938, CVE-2020-8022, CVE-2020-11996, CVE-2020-13935, CVE-2020-13935, CVE-2020-17527, CVE-2021-25122, CVE-2021-41079, CVE-2022-29884, CVE-2022-9484, CVE-2021-25329, CVE-2021-30640, CVE-2022-34305, CVE-2021-24122, CVE-2021-33037, CVE-2019-17569, CVE-2020-1935, and CVE-2020-13943. Updating this dependency to version 10.1.0-M16 will resolve the vulnerabilities.

## 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.

Starting with the first problem identified in the manual code review, the development team needs to decide the best way to remove the database credentials from the code. I would advise prompting the user for the database credentials. Storing unencrypted credentials in source code is not safe and should be avoided at all costs.

To resolve the second problem, a potential integer overflow of the account balance, the team needs to implement bounds checking to ensure that the value of account\_balance will not exceed the maximum value of an integer prior to adding the deposit amount to the account\_balance variable. A simple if statement can check to see if adding the deposit amount will potentially overflow the account\_balance variable and refuse to add the deposit in this case.

The last issue I identified in the manual code review was a function that takes input from the user and concatenates it to a string without performing any input validation. This issue should be resolved by using regular expressions or some other method of input validation to ensure that the string does not contain any characters that could be interpreted as code. The development team should also check that the string is not null, empty, or contains an exceptionally large value as a part of input validation.

Finally, all the issues identified in the static testing tool which checked the dependencies of the project can be resolved by simply updating all the identified dependencies to their latest version. Most of the dependencies were pulled in from spring boot and will be updated by updating to the latest version of the spring libraries. For each dependency on the report, I have already included the latest version for the dependency in the section above titled “Static Testing”. Please ensure that the development team uses this information to update all the vulnerable libraries. I would highly suggest running the dependency checking report after updating the libraries to the latest version to ensure that there are no new vulnerabilities that have been identified in the newer versions of the libraries for added security.

References

CERN Computer Security. (n.d.). *How to keep secrets secret*. CERN computer security information. Retrieved July 16, 2022, from <https://security.web.cern.ch/recommendations/en/password_alternatives.shtml>

*Gramm-Leach-Bliley Act*. Federal Trade Commission. (2022, February 11). Retrieved July 14, 2022, from <https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act>

Manico, J., & Detlefsen, A. (2015). *Iron-clad java building secure web applications*. McGraw-Hill Education.

Nidecki, T. A. (2020, January 6). *What is integer overflow*. Acunetix. Retrieved July 16, 2022, from <https://www.acunetix.com/blog/web-security-zone/what-is-integer-overflow/>

University of Oregon. (n.d.). *Will your direct deposit leave the U.S.?* Business Affairs. Retrieved July 14, 2022, from <https://ba.uoregon.edu/content/will-your-direct-deposit-leave-us-international-ach>