# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/2/2025** | **cody leveille** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Cody Leveille

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions 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 on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

Being a financial company it is critical that Artemis Financial has all of the necessary safety features to protect data, ranging from client personal information to the investment details of their clients. By making sure their data is protected and encrypted it will make sure their clients have confidence in them. Being a financial company it is possible that they may have international transactions. This being the case it means that their security features will have to be compliant with foreign policies otherwise they may not be allowed to do business within those countries. In the United States they will have to adhere to the Gramm Leach Bliley Act that will require them to use encrypted data for all financial transactions and make sure they have user authentication so if needed transactions can be tracked to an individual. When it comes to external threats there are people who will try attacking financial institutions with various attack styles. This can be using SQL injections, Man in the middle attacks and malware. With proper security procedures put in place a lot of these things can be prevented. When it comes to Open source libraries it is critical that you stay up to date with what is being used and there are no vulnerabilities on the current version that is being run. Being open source means everyone has access to these files and it can be easier for exploits to be found. For evolving web technologies using secure connections and an API authenticator are best to ensure data is protected.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

Architecture review- Being a financial institution it is critical that you identify potential weak points in the system. By making sure security is properly handled it will limit the amount of vulnerabilities within the system

Input validation- Being a financial institution it is important that data is secure and validated within the system, this will prevent data manipulation and fraud transactions from occurring from outside attacks.

APIs- Since RESTful APIs are the primary tool of Artemis Financial, making sure they are secure will prevent data breaches and ensure that data being exchanged between the server and client are not stolen or manipulated.

Cryptography- It is critical to protect sensitive data by using strong encryption it will protect this data from outside attacks. This data has federal requirements to protect so it is essential to ensure payment data and personal information stays secure.

Client/Server- By using a secure connection like HTTPS it will make sure data that is exchanged between the client and server are secure and are not vulnerable to man in the middle attacks.

Code Error - Make sure error handling is implemented into the code to prevent errors from being generated that might allow malicious people insight into the system and its inner workings. By implementing error handling we can ensure that information does not leak to those who are not supposed to see it.

Code Quality - Make sure industry standards are used for produced code, by using hard coded information or having flaws in the logic it might have vulnerabilities. By having good code it will prevent these and will allow us to implement good security measures.

Encapsulation - By using encapsulation it will make sure users do not have access to sensitive data and will not be able to manipulate that data within the system. By using proper encapsulation you will make sure data stays intact and reduce the risk of privilege escalation attacks.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

1. In the file DocData.java there is a vulnerability with credentials, currently it is hardcoded into the code to check for credentials. This could lead to unauthorized access to the database if it got leaked.
2. In DocData.Java the database is opened but never closed, this will drain resources within the system and may cause a memory leak.
3. in DocData.Java there is a vulnerability within the print stack trace, it can reveal stack traces to attackers.
4. In CRUDcontroller.java there is no input validation on line 13 this has a risk of being attacked through injections.
5. In CRUDcontroller on line 16 it uses an insecure use of toString and this may output a class name that can be used for potential attackers.
6. in Customer.java on line 5 the data there can be accessed because there is no encapsulation on it.
7. In customer.java on line 12 the deposit method allows the deposit of a negative value that can lead to fraudulent transactions that allow attackers to reduce values of accounts or withdraw money through the deposit method.
8. In the customer.java class there is currently no authorization to make sure that only authorized users can make changes on accounts.
9. In customer.java there is no encryption used, the data is stored in plain text so if an attacker were to gain access to this data everything would be available to them to see.
10. In RestServiceApplication,java there is no secure connection within the server, this will leave it vulnerable to man in the middle attacks.

**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 the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

After running the vulnerability dependency check there were 14 vulnerabilities overall detected.

1. cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\* - The version of bouncy castle is out of date and has cryptographic flaws that could lead to exposure and data breaches. it is recommended that it be updated to version 1.7 or higher because it fixes these known security issues. vulnerability id - CVE-2016-1000352, CVE-2016-1000343
2. cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\* - in this version of spring boot there are known vulnerabilities to denial of service attacks and has ways to bypass security features. It is recommended that spring boot be updated to version 3.0 or higher which will patch out these issues. vulnerability id - CVE-2023-20883
3. cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* - version 1.2.3 has a flaw that can allow the attacker to perform a DoS attack by sending poisoned data which will cause the application to crash. It is recommended to upgrade the logbook to a higher version such as 1.4.11 which removes this problem. Another issue with this version of the logbook allows an attacker with required privileges to edit configuration files which would allow them to execute loaded from LDAP servers. Vuln ID - CVE-2023-6378
4. cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* - The current version 2.12.1 has vulnerabilities that could allow an attacker to execute remote code, DoS attacks and view sensitive information through exploits. These exploits can allow an attacker to expose sensitive data, disrupt system operations and execute code on the system. it is recommended that it be updated to version 2.17.1 if on Java 8+ or update to version 2.12.4 if on Java 7. vuln ID - CVE-2021-44832
5. cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* and cpe:2.3:a:yaml\_project:yaml:1.25:\*:\*:\*:\*:\*:\*:\* The constructor class in Snake YAML version 1.25 does not restrict the types that can be instantiated during deserialization which can provide an attacker the ability to do remote code execution. recommendations of parsing untrusted content through snakeYaml safe constructor and updating to version 2.0 to remove this vulnerability. Vuln ID - CVE-2022-1471
6. cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* - The vulnerabilities listed for this can allow different versions of DoS attacks, resource exhaustion and security breaches. These issues stem from improper handling of deeply nested objects and lack of validation. It is recommended that the database be updated to version 2.15.2 to avoid this issue. vuln ID CVE-2021-46877
7. cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* and cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* - version 9,0.30 has multiple vulnerabilities that can lead to remote code execution and DoS attacks and session hijacking. These vulnerabilities happen from improper validation and cleanup processes. and insecure HTTP handling, this can lead to compromises within the system. The recommendation is to update to Apache Tomcat version 9.0,83 where these vulnerabilities have been patched. Vuln IDs - CVE-2023-46589, CVE-2023-45648, CVE-2023-44487, CVE-2021-41079, CVE-2020-13935
8. cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* - A flaw in the hibernate validator allows attackers to bypass input sanitation and may expose data in error messages it is recommended that it be updated to 6.1.5.Final to avoid this problem Vuln ID - CVE-2020-10693

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

From the manual review of code and static testing report there are a few critical vulnerabilities that need to be fixed within the system. With the manual code review in DocData.java the credentials should not be hardcoded into the program. This can lead to data breaches in the future, it is best to use environment variables to store sensitive data. Database connections should be properly closed once they are done being used, this will prevent resource leaks and memory exhaustion. With the Printstacktrace method this can reveal sensitive information about the system and would be best fixed with an error generated if it occurs. instead of revealing the data to potential attackers. In the CRUDcontroller.java there is currently no input validation for the system, this can leave it vulnerable to attacks through injections. To fix this issue simply add input validation to prevent this. The use of toString on line 16 can expose sensitive information about the system. It is best to sanitize this output to prevent this leak. In customer.java on line 5 there is no encapsulation on account balance, you want to use a private variable to prevent this data from being manipulated outside its intended uses. On line 12 of this same class there is no validation for fraudulent transactions. Currently as it is an attacker can deposit a negative amount allowing accounts to be drained of their funds. To fix this simply add a check for negative values in the amount the user wishes to deposit. In this same class there is currently no user authentication check either, this allows users to manipulate data on others accounts. The best course of action for this is adding role based security access to make sure users can not access sensitive data. Currently user data is stored as plain text, this makes it very easy for attackers to steal this data and use it to their advantage. It is recommended that some sort of encryption is used for this data to be compliant with federal regulations and laws. In RestServiceApplication.java there is no secure connection between user and server connection. This leaves it vulnerable to man in the middle attacks and is fixed by using HTTPS connections.

For the static testing report there were several vulnerabilities within the packages used in the system, these can be fixed by updating the out of date packages. The first thing identified within the report is a critical issue with bouncy castle version 1.46 and it is recommended that it be updated to version 1.7 to fix the known issues with the current version. Spring Boot version 2.2.4 needs to be updated to version 3.0 or higher to avoid DoS attacks and bypass techniques. Logback version 1.2.3 has an exploit that allows DoS attacks and allows for remote code execution to fix this problem it is recommended that it be updated to version 1.4.11. Log4j version 2.12.1 has vulnerabilities that allow attackers to do DoS attacks, remote code execution and information leaking to fix this issue it is recommended that Java8+ update to version 2.17.1 and Java 7 be updated to version 2.12.4 to patch out these issues. The current version of SnakeYML version 1.25 allows unrestricted deserialization which allows remote code execution; it is recommended to use SafeConstructor when parsing untrusted data and upgrade to version 2.0. Jackson -databind version version 2.10.2 has vulnerabilities that can lead to DoS attacks and resource exhaustion along with security breaches to fix this problem it is recommended that it be updated to version 2.15.2. Apache Tomcat version 9.0.30 has issues with remote code execution DoS attacks and session hijacking and insecure HTTP handling this can lead to a complete system compromise and to fix this problem it is critical to update to version 9.0.83. Hibernate validator version 6.0.18 allows attackers to bypass input sanitation and expose data with error messages, to fix this it is recommended that it be updated to version 6.1.5.Final.

To fix the vulnerabilities in the Artemis Financial application it is recommended that secure coding practices be implemented and the issues mentioned in the manual coding review be added as soon as possible. As well as all of the databases that were highlighted in the maven report be updated to prevent attacks on the system and to help prevent data being manipulated or leaked.