# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/19/25** | **Justin Perez** | **Added Information** |
|  |  |  |  |

## 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

Justin Perez

**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?***
  1. Secure communications are extremely important to this company due to their handling of sensitive data. Artemis financial handles a lot of material such as savings, retirement, investments, and insurance data so having encrypted communication is the most important. A secure communication network will ensure no one unauthorized has access to client’s data as well as preventing data breaches. I recommend SHA to encrypt data due to how hard it is to crack as of currently.
* ***Are there any international transactions that the company produces?***
  1. Artemis Financial has clients globally so it handles international transactions. This means we must encrypt data according to different countries’ regulations. The best approach to trying to ensure your software is up to standard in every country’s regulation is to investigate GDPR in Europe. It has some of the tightest regulations around data protection and control so if your program passes the GDPR then it will most likely pass almost all other regulations. Whatever it doesn’t you will have to simply add a bit to the software.
* ***Are there governmental restrictions on secure communications to consider?***
  1. As spoken about in the last point, regulations are all over the world. When developing the software ensuring that it meets proper regulations it’s easier to see the most regulated popular country than going down the list such as GDPR. Other countries like the USA have smaller restrictions like COPPA which protects children from having their data collected.
* ***What external threats might be present now and in the immediate future?***
  1. Because they are a financial company there are quite a few threats that could happen to them. They also handle sensitive data which places them even more in danger. A couple of these are:
     1. (MitM)-Man in the middle attacks that target APIS
     2. Phishing – An attack targeting customers and employees to get access
     3. Ransomware – An attack that targets the company holding their data hostage for money
     4. (DDoS) – An attack that aims to take down the website or to ruin the servers
     5. Malware – An attack that targets Customers and employees to watch their computer for multiple reasons
     6. (SQL Injection) An attack that executes commands in unexpected ways.
* ***What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?***
  1. The use of open-source libraries will heavily speed up development, however, will need to be watched. The reason open source is open source because anyone can look at the code and look at what is inside of it. While most use it for learning or creation some use it to find exploits to break into some programs which is why you will need to watch to ensure no vulnerabilities are found in the library you are using. You will also need to keep the software up to date to ensure there isn’t a vulnerability that shows up in the future.
  2. Artemis Financial should use newer web application technologies to stay secure. The RESTful API should use standards like OAuth 2.0 and JWT to control access. All data must go through HTTPS with TLS 1.3 to stay safe. Using tools that scan code and check for issues during development can help find problems early. Updating frameworks and using secure cloud tools will also make the system safer and faster.

**2. Areas of Security**

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

* ***Input validation***
  + The application will need to accept a lot of input from the user such as usernames, names, billing, birthdays, passwords, and income. So, input validation makes it to the data is safely collected and stores so there is no possibility of attacks that could happen like SQL injections where people will attempt to create code inside of the form.
* ***APIs***
  + The program will use APIs to manage their data. This means API security will be a top priority. An API will protect your data if you protect it.
* ***Cryptography***
  + Cryptography handles the system financial data and encrypts it for storage and transmitting it. This prevents MitM attacks.
* ***Client/server***
  + Client and server systems use both client and server components to run the app. This could be done as an example important client information is held in the server as you wouldn’t want a client to be able to easily view that information and tamper with it. A client could hold information such as simple Java code telling the app instructions of how to look/run.
* ***Code Errors***
  + Error handling helps prevent the application from not only crashing but revealing the wrong information. To illustrate this when you can’t connect to a website it will say “error can’t connect to website name” instead of “error can’t connect to admin/users/10202892892/data/credit card”. This is why handling code errors is important.
* ***Code Quality***
  + Standard coding practice reduces the chance of bugs and vulnerabilities. It will also speed up development heavily as well as make potential third-party clients be able to work with the code easily. It also supports input validation.
* ***Encapsulation***
  + Encapsulation means to protect internal data and the logic of an object from the outside in case an unauthorized user gets in. This will help prevent data leaks and keep apps secure.

**3. Manual Review**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability | Description | Fixes | Snip |
| Old Versions | Old plugins being used which could have vulnerabilities | * Update plugins if you can make it up to date * If you can’t update software figure out what vulnerabilities are known and plan accordingly to prevent them. |  |
| Bad Practice | “SQL Exception e” Is consider bad practice to be your only error message. This is due to it being extremely imprecise and hard to shoot. | * Add more catches that go to specific errors instead of a generalized error |  |
| No input Validation | There is no input validation when requesting “name” which could allow the user to put malicious code | * Create standard Input validation protocols. * A name shouldn’t have any characters or numbers inside of it. * A name shouldn’t be empty or over 50 characters long. |  |
| Naming Conventions | Proper naming conventions are not being followed which can cause development to slow down. | * Replace “myDateTime” to “MyDateTime” for consistency |  |
| Encapsulation | A account balance should be private as that’s information that users would not like to share. | * Rename line to “Private int account\_balance;” |  |

**4. Static Testing**

Run a dependency check on Artemis Financials’ software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| vulnerability | Case Score | Date | Description | Recommendation | Fixes |
| [CVE-2022-1471](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471) | CRITICAL (9.8) | N/A | YAML 1.1 parser and emitter for Java | SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond. | * Upgrade to 2.0 and Beyond * Use Yaml’s safe constructor when parsing untrusted content to restrict deserialization |
| [CVE-2023-20873](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20873) | CRITICAL (9.8) | N/A | Spring Boot | In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+. | * 3.0.x Users should upgrade to 3.0.6+ * 2.7.x Users should upgrade to 2.7.11+ * Users of older unsupported version should upgrade to 3.0.6+ or 2.7.11+. |
| [CVE-2022-27772](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772) | HIGH (7.8) | N/A | Spring Boot | spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org. springframework .boot .web .server .Abstract Configurable Web Server Factory .createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer | * Upgrade to version 2.2.11 release or later * Set up a temp directory to mitigate the less of a chance to guess the directory name |
| [CVE-2022-22965](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) | CRITICAL (9.8) | 2022/  04 / 04 | Spring Core | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it. | * Ensure that application isn’t deployed on spring boot Executable jar. If it’s don’t worry about the exploit * Upgrade spring boot too 2.5x oor later |
| [CVE-2020-5421](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421) | MEDIUM (6.5) | 2022/  04 / 04 | Spring Expression Language (SpEL) | In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 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 through the use of a jsessionid path parameter. | * Upgrade spring boot too 2.5x or later * Implement additional security measures like: Input validation, Content-disposition, and Monitor |
| [CVE-2022-22971](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971) | MEDIUM (6.5) | N/A | Spring Web | In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, application with a STOMP over WebSocket endpoint is vulnerable to a denial of service attack by an authenticated user. | * Upgrade spring boot too 2.5x or later * Implement additional security measures like: Input validation, Rate Limiter, and Monitor |
| [CVE-2021-22096](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | MEDIUM (4.3) | 2022/  04 / 04 | Spring Web MVC | In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. | * Upgrade spring boot to 5.3.x or later * Input Validation |

**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 Financials’ software application.

To immediately solve MOST of the vulnerabilities that are present a version change of all programs will be required to ensure we are up to date and safe. If unable to do a version change then we will need to do a lot of work ensuring input is validated, Standard security Monitoring, A temp directory will need to be made, and a rate limiter.  
  
The only other notable vulnerability solved with a version change easily is snakeYaml where we must restrict the use of it heavily or use a Safe Constructor which might be hard to implement