# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **Michael Abell** |  |

## 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

[Michael Abell]

**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?
  1. Artemis simply cannot offer a service to customers without security. Financial institutions are a major target for security breaches as they can lead the financial data. No matter how much money they may be able to help their clients earn, none of that matters if they cannot be safe from intrusion. We have been hired to ensure security and that is the number one priority.
  2. There are no specific mentions of international transactions, but they are a web based company. Considerations must be made for international threats and vulnerabilities.
  3. The Gramm-Leach-Bliley Act of 1999 that was updated in 2021 by the Federal Trade Commission stipulates that all communication must be secure. Guidelines to follow are on the Federal Trade Commission’s website.
  4. Every conceivable level of hacker, from individual to state sponsored can independently be a threat. Bots that scrape the internet looking for vulnerabilities such as outdated code can target our client if something is sniffed out.
  5. Immediately, all calls to libraries and API’s must be up to date. An audit of current tools utilized to compare other open-source resources that could be superior would be prudent as well.

**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.

1. **Input Validation**- Customers data must be secure. Not only do we need the basic username and password match for access, we need to be sure no malicious code can be injected into this input fields.
2. **APIs**- Relying on trusted and updated libraries and APIs is industry standard. That being the case we need to stay up to date and ensure we are using the most secure option.
3. **Cryptography**- Whether we use hashing or some other sort of encryption, it’s important that if there are data leaks, they are at least obfuscated trails of characters and not complete strings of actual data.
4. **Code Error**- Accounting for the unaccountable, we need all code to be able to handle unintended instructions. Locking down the code to expect certain data and ignoring all else is a start.
5. **Code Quality**- The formatting, the commenting and the structure of the code is to be predictable on the backend, with structured classes and syntax.

**3. Manual Review**

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

Input validation(1) in the customer.java class there is a simple formula to update the balance, but there are no exception handling methos for bad input.

Code Quality(4)- account balances should obviously be in Double instead of INT. Need to verify with the client that 100% of account numbers are only int also and no alpha characters. This was also found in the customer.java class. In DocData.java there seems to be a comment that says root is generically username and password? Very unsecure if true and should not be spelled out! In CRUD.java we have content being overloaded with two constructors. Need to make thos more unique

Encapsulation(1)- The crud.java class is totally public. Many of the values should be set to private to keep the code clean and secure.

APIs(1)- need to use the most current SpringBoot

**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

|  |  |
| --- | --- |
| [bcprov-jdk15on-1.46.jar](#RANGE!l2_991c96a4e31e6c19e2b9136c8955bd) | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms.- The fix is to “enable Java Security Manager to sandbox the evaluation of MVEL expressions” - https://access.redhat.com/errata/RHSA-2014:0372.html |
| [spring-boot-2.2.4.RELEASE.jar](#RANGE!l3_225a4fd31156c254e3bb92adb42ee8) | Vulnerability is the possibility of attacker creating temp directories and escalate their privileges. |
| [logback-core-1.2.3.jar](#RANGE!l4_864344400c3d4d92dfeb0a305dc87d) | The configuration file of logback can be modified or overwritten |
| [log4j-api-2.12.1.jar](#RANGE!l5_a55e6d987f50a515c9260b0451b4fa) | Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1 |
| [snakeyaml-1.25.jar](#RANGE!l8_8b6e01ef661d8378ae6dd7b511a7f2) | The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation |
|  |  |
| [jackson-databind-2.10.2.jar](#RANGE!l9_0528de95f198afafbcfb0c09d2e43b) | A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. |
| [tomcat-embed-core-9.0.30.jar](#RANGE!l13_ad32909314fe2ba02cec036434c0a) | Invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling |
| [hibernate-validator-6.0.18.Final.jar](#RANGE!l16_7fd00bcd87e14b6ba66279282ef15) | This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages |
| [spring-web-5.2.3.RELEASE.jar](#RANGE!l19_dd386a02e40b915ab400a3bf9f586) | A potential remote code execution (RCE) issue if used for Java deserialization of untrusted data |
| [spring-beans-5.2.3.RELEASE.jar](#RANGE!l20_0250c8c641433dc06b1b44e4563fa) | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding |
| [spring-webmvc-5.2.3.RELEASE.jar](#RANGE!l21_745a62502023d2496b565b7fe102b) | it is possible for a user to provide malicious input to cause the insertion of additional log entries |
| [spring-context-5.2.3.RELEASE.jar](#RANGE!l22_7750c95c96c7a1885c8b1b503ba91) | the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, |
| [spring-expression-5.2.3.RELEASE.jar](#RANGE!l23_d0c6bb10758805b2153c589686b80) | it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition |

**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.

[Include your findings here.]

As stated above, our immediate course of action is to update all libraries especially the Spring relates libraries. Their vulnerabilities have been address in the most current iteration of the software.

We will go to each dependency and update to the current version as well as cross reference any other libraries that may be superior.

The next action item is recode poor coding practices, such as doubles vs ints for storing financial information. There may be more than the identified issues to recode.