# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **03/20/2025** | **Jian Wang** |  |

## 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

Jian Wang

**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?

Because this is a financial company, cybersecurity is everything to the company. All data and money transactions depend on cybersecurity.

In the company's description it is stated that “The software is for entrepreneurs, businesses, and government agencies around the world.”, so the company's business involves international transactions.

There are no restrictions from the government yet.

External threats have always existed, mainly from hacker organizations and individuals, and their actions are aimed at stealing company data. Such as DDoS attacks, stealing data through unencrypted HTTP connections, exploiting known vulnerabilities in open-source libraries, and attacking through insecure third-party software and remote control of insecure hardware.

First, authentication and authorization are required to prevent malicious access. At the same time, a web application firewall is needed for real-time protection. Mandatory use of HTTPS for communication encryption. Although open-source libraries are important for software development, vulnerabilities in open-source libraries can also be easily exploited. Therefore, known vulnerabilities in open-source libraries should be quickly fixed and the latest patches should be kept up to date.

**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.

**Input Validation** Input validation is the first line of defense to protect customer data, ensuring that unauthorized users cannot access it. At the same time, it can also prevent system paralysis caused by many automatically logging in.

**APIs** If the data transmitted through the API interface is sent directly to the backend without being encrypted, attackers may intercept the data, resulting in data leakage and security risks.

**Cryptography** The latest encryption technology can implement input verification and API encryption to prevent data from being cracked.

**Code Quality** High-quality code reduces the introduction of vulnerabilities. High-quality code uses the principle of least privilege to limit the scope of risks.

**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 DocData.java, “root” and “root” may be the login account and password and should not be displayed directly. Instead, they should be assigned with parameters.

In CRUD.java, “content2” is not used and is redundant code and should be deleted.

In CRUDcontroller.java, "business\_name" is not verified, which may lead to invalid login.

In customer.java, the “c” in “customer” should be uppercase, in line with industry standards.

The number of Deposit should not be “a”, but should be “amount” or something like that. And the amount should be limited, amount>0, otherwise an error will occur.

myDataTime.java should use MyDataTime. Return new int should not contain 3, but should be mySecond myMinute myHour.

When I tried to run the application, I found that the program could not run. The system prompted “java.lang.NoClassDefFoundError: junit/framework/TestCase”, and the corresponding plug-in needed to be loaded in the pom.xml file.

**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](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) [**CVE-2023-33202**](https://nvd.nist.gov/vuln/detail/CVE-2023-33202)Bouncy Castle for Java before 1.73 contains a potential Denial of Service (DoS) issue within the Bouncy Castle org.bouncycastle.openssl.PEMParser class. This class parses OpenSSL PEM encoded streams containing X.509 certificates, PKCS8 encoded keys, and PKCS7 objects. Parsing a file that has crafted ASN.1 data through the PEMParser causes an OutOfMemoryError, which can enable a denial of service attack. (For users of the FIPS Java API: BC-FJA 1.0.2.3 and earlier are affected; BC-FJA 1.0.2.4 is fixed.)

[hibernate-validator-6.0.18.Final.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492) [**CVE-2020-10693**](https://nvd.nist.gov/vuln/detail/CVE-2020-10693)A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. 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.

[jackson-databind-2.10.2.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) [**CVE-2023-35116**](https://nvd.nist.gov/vuln/detail/CVE-2023-35116)jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. NOTE: the vendor's perspective is that this is not a valid vulnerability report, because the steps of constructing a cyclic data structure and trying to serialize it cannot be achieved by an external attacker.

[log4j-api-2.12.1.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb) [**CVE-2021-44832**](https://nvd.nist.gov/vuln/detail/CVE-2021-44832)Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.

[logback-classic-1.2.3.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l12_7c4f3c474fb2c041d8028740440937705ebb473a) [**CVE-2023-6378**](https://nvd.nist.gov/vuln/detail/CVE-2023-6378)A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data.

[snakeyaml-1.25.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l15_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) [**CVE-2022-1471**](https://nvd.nist.gov/vuln/detail/CVE-2022-1471)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.

[spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l16_225a4fd31156c254e3bb92adb42ee8c6de812714) [**CVE-2023-20883**](https://nvd.nist.gov/vuln/detail/CVE-2023-20883)In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.

[spring-core-5.2.3.RELEASE.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l18_3734223040040e8c3fecd5faa3ae8a1ed6da146b) [**CVE-2023-20863**](https://nvd.nist.gov/vuln/detail/CVE-2023-20863)In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.

[tomcat-embed-core-9.0.30.jar](file:///C:\Users\wjss\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_ad32909314fe2ba02cec036434c0addd19bcc580) [**CVE-2025-24813**](https://nvd.nist.gov/vuln/detail/CVE-2025-24813)Path Equivalence: 'file.Name' (Internal Dot) leading to Remote Code Execution and/or Information disclosure and/or malicious content added to uploaded files via write enabled Default Servlet in Apache Tomcat. This issue affects Apache Tomcat: from 11.0.0-M1 through 11.0.2, from 10.1.0-M1 through 10.1.34, from 9.0.0.M1 through 9.0.98. If all of the following were true, a malicious user was able to view security sensitive files and/or inject content into those files: - writes enabled for the default servlet (disabled by default) - support for partial PUT (enabled by default) - a target URL for security sensitive uploads that was a sub-directory of a target URL for public uploads - attacker knowledge of the names of security sensitive files being uploaded - the security sensitive files also being uploaded via partial PUT If all of the following were true, a malicious user was able to perform remote code execution: - writes enabled for the default servlet (disabled by default) - support for partial PUT (enabled by default) - application was using Tomcat's file based session persistence with the default storage location - application included a library that may be leveraged in a deserialization attack Users are recommended to upgrade to version 11.0.3, 10.1.35 or 9.0.99, which fixes the issue.

**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.

Most vulnerabilities can be solved by upgrading through Maven to the latest version.

bcprov-jdk15on-1.46.jar, the latest version given by the official website is 1.70,

hibernate-validator-6.0.18.Final.jar has been officially upgraded to 9.0.0,

jackson-databind-2.10.2.jar can be upgraded to 2.18.3

og4j-api-2.12.1.jar can be upgraded to 2.20.0

logback-classic-1.2.3.jar can be upgraded to 1.5.18

snakeyaml-1.25.jar can be upgraded to 2.4

spring-boot-2.2.4.RELEASE.jar can be upgraded to 3.4.4

spring-core-5.2.3.RELEASE.jar can be upgraded to 7.0.0

tomcat-embed-core-9.0.30.jar can be upgraded to 11.0.5

After the upgrade is completed, keep the version up to date and test it to ensure the security of the function.

Regarding the code errors, the error codes need to be modified according to industry rules.