# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/21/2024** | **Mark Turner** |  |

## 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

Mark Turner

**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?

As an aggregator of personally identifiable financial information (PIFI), Artemis Financial must place extremely high value on securing communications. The company may facilitate international transactions to diversify client portfolios, expand its global presence, and assist clients seeking to retire abroad.

When communicating internationally, Artemis must comply with laws on cross-border information sharing, including EU regulations on personal data transfer. California and EU countries have specific laws governing how corporations handle personal information. The Data Privacy Act and Gramm-Leach-Bliley Act also require institutions to safeguard sensitive data and inform consumers about their privacy rights. Artemis should consider the many governmental restrictions regarding the transfer and storage of personal data.

Artemis should take steps to protect clients and employees from phishing, credential theft, and social engineering attacks. Implementing multi-factor authentication can help mitigate unauthorized access to sensitive information. Outdated third-party packages and internal technologies also pose significant risks; modernizing these systems with up-to-date web technologies can enhance data security. While open-source libraries can accelerate development, they must be properly managed to avoid vulnerabilities. Adopting cloud technologies and a microservices architecture can further reduce the risk of single points of failure, enhancing overall system integrity.

**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.

**Secure API Interactions**

Due to the nature of the data Artemis Financial collects and exposes, it’s necessary to ensure that information is encrypted both in transit and at rest. Using secure protocols like TLS or HTTPS mitigates the risk of an unauthorized user gaining access to sensitive information.

**Input Validation**

It’s important that Artemis Financial sanitizes all input going through their internal and external systems. This reduces the risk of injection attacks.

**Code Error Handling**

As a REST API, the application should return appropriate HTTP error codes that coincide with the REST API standards. This can be done with proper error handling.

**Secure Distributed Composing**

Artemis need to ensure data integrity and confidentiality across decentralized systems, protecting sensitive financial information from potential breaches. Secure Distributed composing allows for the integration of various services and applications while maintaining robust security protocols.

**Code Quality**

Taking advantage of secure coding practices and common development patterns can increase the maintainability of Artemis’ systems. When using open-source software, it’s also vital to ensure that the developers of said software utilize these practices to mitigate the vulnerability of the system.

**Encryption**

Protecting client financial data at rest and in transit through strong encryption protocols is vital for remaining in line with data protection regulations and preventing the exposure of sensitive information.

**3. Manual Review**

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

* Ln#9 in the CRUD object sets the `content` parameter passed into the constructor to both the private `content` and `content2` variables
* Ln#13 in the CRUDController takes in a string request parameter which is not used.
* Ln#16 in the CRUDController returns a `toString()` version of the entire DocData object
* Ln#16 in the GreetingController takes in and uses a request parameter without performing validation to determine if it’s a valid name.
* The Greeting method in the GreetingController does not properly handle the scenario where the consumer of the endpoint passes in an invalid name.
* Ln#13 in the Customer object simply adds whatever integer value is passed into the function’s ‘a’ parameter without checking to see if it’s a valid amount.
* Ln#12 in the customer object takes in an integer type which can be positive or negative and only a whole number.
* The `read\_document` function in the DocData object creates a connection to the database without properly closing the connection after use. Additionally, it hardcodes the password and database URL instead of storing and retrieving that information from a secrets 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 (HIGH) – Update to latest compatible package version

## CVE-2016-1000352: In Bouncy Castle JCE Provider v1.55 and earlier, ECIES allowed unsafe ECB mode; support has been removed.

## CVE-2016-1000346: In v1.55 and earlier, DH public key validation was incomplete, potentially exposing private key details.

## CVE-2016-1000345: DHIES/ECIES in v1.55 and earlier was vulnerable to padding oracle attacks, allowing decryption failure observations.

## CVE-2016-1000344: DHIES implementation allowed unsafe ECB mode in v1.55 and earlier; support has been removed.

## CVE-2016-1000343: DSA key generator in v1.55 and earlier could produce weak keys if not explicitly initialized with parameters.

## CVE-2016-1000342: ECDSA in v1.55 and earlier did not fully validate ASN.1 encoding, enabling signature injection attacks.

## CVE-2016-1000341: DSA signature generation in v1.55 and earlier was vulnerable to timing attacks, potentially revealing private values.

## CVE-2016-1000339: AESFastEngine in v1.55 and earlier leaked AES key information via table-driven access patterns.

## CVE-2016-1000338: DSA did not fully validate ASN.1 encoding in v1.55 and earlier, allowing signature injection attacks.

## CVE-2018-5382: Default BKS keystore used a weak 16-bit HMAC, compromising integrity; stronger HMAC introduced in Bouncy Castle v1.47.

## CVE-2017-13098: BouncyCastle TLS <v1.0.3 had a weak Bleichenbacher oracle for RSA key exchanges, enabling private key recovery.

## CVE-2013-1624: TLS implementation <v1.48 did not mitigate timing side-channel attacks during CBC padding, allowing plaintext recovery.

## log4j-api-2.12.1.jar (CRITICAL) – Update to latest compatible package version

## CVE-2021-44832: Versions 2.0-beta7 through 2.17.0 (excluding 2.3.2, 2.12.4) are vulnerable to remote code execution (RCE) via JDBC Appender with JNDI LDAP when an attacker controls the LDAP server. Fixed in versions 2.17.1, 2.12.4, and 2.3.2.

## CVE-2021-45105: Versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3, 2.3.1) were susceptible to uncontrolled recursion from self-referential lookups, leading to denial of service. Fixed in versions 2.17.0, 2.12.3, and 2.3.1.

## CVE-2021-45046: The fix for CVE-2021-44228 in version 2.15.0 was incomplete, allowing information leaks and RCE through crafted inputs in non-default configurations. Fixed in versions 2.16.0 (Java 8) and 2.12.2 (Java 7).

## CVE-2021-44228: Versions 2.0-beta9 through 2.15.0 (excluding 2.12.2, 2.12.3, 2.3.1) did not adequately protect against controlled JNDI endpoints, allowing arbitrary code execution from LDAP servers. Behavior disabled by default in version 2.15.0 and removed in 2.16.0.

## CVE-2020-9488: Improper certificate validation in the SMTP appender could allow man-in-the-middle attacks, leaking log messages. Fixed in versions 2.12.3 and 2.13.1.

## spring-beans-5.2.3.RELEASE.jar (HIGH) – Update to latest compatible package version

## CVE-2022-22965: Spring MVC or Spring WebFlux applications on JDK 9+ deployed as WAR files on Tomcat may be vulnerable to remote code execution (RCE) via data binding. Applications packaged as Spring Boot executable JARs are not affected, but other potential exploit vectors may exist.

## spring-webmvc-5.2.3.RELEASE.jar (HIGH) – Update to latest compatible package version

## CVE-2021-22060: In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.

## tomcat-embed-core-9.0.30.jar (Critical) – Update to latest compatible package version

* CVE-2024-21733: Sensitive information exposed in error messages; upgrade recommended.
* CVE-2023-46589: Improper input validation leads to request smuggling; upgrade advised.
* CVE-2023-45648: Invalid trailer headers cause request smuggling; users should upgrade.
* CVE-2023-42795: Incomplete cleanup may leak information between requests; upgrade needed.
* CVE-2023-44487: HTTP/2 request cancellation allows denial of service; patch recommended.
* CVE-2023-41080: Open redirect vulnerability in FORM authentication; upgrade Tomcat versions.
* CVE-2023-28708: Session cookies missing secure attribute; update Tomcat to secure.
* CVE-2022-42252: Invalid Content-Length header permits request smuggling; upgrade required.
* CVE-2021-43980: Improper TLS packet validation causes denial of service; update.
* CVE-2021-33037: Incorrect HTTP header parsing risks request smuggling; upgrade needed.
* CVE-2021-30640: JNDI Realm vulnerability allows username authentication bypass; update advised.
* CVE-2021-25329: Header duplication exposes responses across connections; patch recommended.
* CVE-2021-24122: JSP source code disclosure possible via NTFS configurations; upgrade.
* CVE-2020-17527: Reused HTTP headers could leak information between requests; patch necessary.
* CVE-2020-13943: Exceeded streams may leak previous request headers; upgrade needed.
* CVE-2020-13935: Invalid WebSocket frame payloads can trigger denial of service; upgrade.

## snakeyaml-1.25.jar (Critical) – Update to latest compatible package version

* CVE-2022-1471: SnakeYaml's Constructor() allows remote code execution; use SafeConstructor.
* CVE-2022-41854: Untrusted YAML parsing can lead to DoS attacks; upgrade recommended.
* CVE-2022-38752: Vulnerable to DoS via stack overflow when parsing untrusted input.
* CVE-2022-38751: Denial of Service risk from stack overflow in untrusted YAML parsing.
* CVE-2022-38750: Stack overflow during untrusted YAML parsing can lead to DoS.
* CVE-2022-38749: Potential for DoS through stack overflow with untrusted YAML content.
* CVE-2022-25857: Missing nested depth limit in SnakeYaml enables DoS attacks.
* CVE-2017-18640: Entity expansion issue in SnakeYAML prior to 1.26; related to CVE-2003-1564.

## spring-boot-2.2.4.RELEASE.jar (Critical) – Update to latest compatible package version

* CVE-2023-20883: Spring Boot may allow DoS attacks with reverse proxy caches.
* CVE-2023-20873: Cloud Foundry deployments in Spring Boot can lead to security bypass.
* CVE-2022-27772: Temporary directory hijacking vulnerability in unsupported Spring Boot versions.

## jackson-databind-2.10.2.jar (HIGH) – Update to latest compatible package version

* CVE-2023-35116: Jackson-databind allows DoS via crafted cyclic dependencies.
* CVE-2021-46877: Jackson-databind can cause DoS with excessive heap usage.
* CVE-2022-42004: Resource exhaustion from deeply nested arrays in Jackson-databind.
* CVE-2022-42003: Deep wrapper array nesting causes resource exhaustion in Jackson-databind.
* CVE-2020-36518: Nested objects can trigger StackOverflow, leading to DoS in Jackson-databind.
* CVE-2020-25649: Jackson-databind vulnerable to XML external entity (XXE) attacks affecting data integrity.

## logback-core-1.2.3.jar (HIGH) – Update to latest compatible package version

* CVE-2023-6378: Logback's serialization vulnerability allows DoS via poisoned data.
* CVE-2021-42550: Malicious config in Logback can execute arbitrary code from LDAP.

## hibernate-validator-6.0.18.Final.jar (MEDIUM) – Update to latest compatible package version

* **CVE-2020-10693**: Flaw in Hibernate Validator 6.1.2.Final allows invalid EL expressions to bypass input sanitation controls, risking user data.

## spring-context-5.2.3.RELEASE.jar (MEDIUM) – Update to latest compatible package version

* **CVE-2022-22968**: In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, 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, including upper and lower case for the first character of all nested fields within the property path.

## spring-expression-5.2.3.RELEASE.jar (MEDIUM) – Update to latest compatible package version

* **CVE-2022-22950** in Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, 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.

Manual review and static testing of Artemis’ software identified multiple security vulnerabilities related to improper input validation, serialization flaws, and outdated libraries. Several bits of code in controllers and data objects are vulnerable due to unvalidated input and poor handling of sensitive data. Static analysis revealed libraries with known vulnerabilities, such as Log4j, Jackson, and Spring Framework, requiring immediate updates.

**Code Issue Resolutions**

* Fix the first CRUD Constructor to only assign content to the ‘content’ private member.
* Eliminate the unused string request parameter in CRUDController.
* Add a toString() method override to DocData to avoid returning sensitive information.
* Implement validation checks for request parameters in GreetingController to ensure names are valid.
* Add error handling in the GreetingController to manage invalid name scenarios gracefully.
* In the Customer object, check the validity of the integer parameter before processing it.
* Modify the Deposit function in the Customer Object to only allow for Unsigned integers
* Add a `withdrawal` function to handle the deduction of funds from the customer’s account
* Ensure the read\_document function properly closes the database connection and avoid hardcoding sensitive data.