# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07-16-2025** | **Tanner Hunt** | **Wrote rationale for security needs, Identified pertinent areas of security, performed manual code review of existent code, performed static testing of dependencies, wrote mitigation plan** |

## 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

Tanner Hunt

**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?

Cybersecurity should be a high priority item for Artemis Financial. Complying with cybersecurity regulations increases the publics confidence in a product, protects the company from legal penalties, and protects the company from potential lawsuits involving lost wealth attributable to data breaches. This institution is subject to the Gramm-Leach-Bliley act, which governs financial institutions; failure to comply can lead to fines up to $100,000 per violation and criminal charges against security officers and directors (Koller, 2024). Management of online payments makes this institution subject to PCI-PTS standards. Additionally, international investing may make the company subject to GDPR laws, which are the EU’s data privacy laws; failure to comply could result in either a €20,000,000 fine or fine equal to 4% of the company’s global revenue – whichever one is higher (Whitford, n.d.).

Artemis Financial may be exposed to social engineering attacks, phishing, data breaches, privacy violations, insider threats, and man in the middle attacks. Protection against these evolving threats requires an equally sophisticated and evolving infrastructure. This includes mitigating risks through security officers, staff training, and identifying risks associated with supporting software libraries. While open-source libraries are cost-efficient in solving certain problems, their paid counterparts may offer an added layer of protection by obscuring their implementation.

**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.

The following areas of security are pertinent to Artemis Financial:

1. Input Validation – this service frequently interacts with untrusted data in the form of user inputs. Unvalidated user inputs can lead to unpredictable software behavior, like data breaches and destruction of data
2. API security – these services depend on outsourced software which may also be vulnerable to attackers
3. Cryptography – cryptography is a requirement in the Gramm-Leach-Bliley act, in the PCI-PTS standards, and in GDPR laws.
4. Client/ Server – Secure data is transmitted and stored between clients and the server. Keeping this data secure and authenticating user requests for this data is a requirement of the Gramm-Leach-Bliley act and GDPR laws.
5. Code Error – Improperly handled errors can lead to data breaches and loss of secure data
6. Code Quality – It becomes increasingly difficult to assess security vulnerabilities as the complexity of a project increases. Maintaining high coding standards throughout the development process makes code more resilient to security vulnerabilities and mitigates damages in the case of a security breach
7. Encapsulation – Sharing unencapsulated data to users may allow them to directly modify it.

**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. Classes that are not meant to be inherited from should be declared as final. For example:
   1. Public class myDateTime { … } should become public final class myDateTime { … }
2. Database searches should be parameterized, not built through string concatenation, such as in the GreetingController, where string.format() may allow injection of malicious data
3. Avoid using try-catch blocks to validate user info to avoid “Fail Open” attacks. For example, in DocData.java, an attacker can bypass the try-catch statement in read\_document by passing certain unexpected values, like “null” to the function because it does not properly trigger an exception, which lets the attacker enter the “//database name, root is username and password” block of code
4. Methods should implement authorization using the @Secured tag
   1. For example, in Crud.java getContent() could become @Secured({“ADMIN”, “OTHER\_ROLE”}) public getContent(){ … }
5. Mutable data should not be exposed through a “this” reference, rather it should be copied.
   1. For example, in Customer.java, showInfo() should return a final copy of the account\_number, not a reference to this.account number
6. Validate the size of inputs to prevent overflow errors, like in the customer.java classes deposit(int a) method
7. Enable ssl and HTTPS in the application.properties file
8. Verify untrusted data does not contain HTML or XML elements, like in the Greeting.java classes content member variable.

**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

|  |  |  |
| --- | --- | --- |
| **Code** | **Description** | **Attribution** |
| CVE-2024-34447 | Bouncycastle will verify a host provides a certificate, but does not verify that the certificate is associated with the host. | [[CVE-2024-34447] CWE-297: Improper Validation of Certificate with Host Mismatch](https://ossindex.sonatype.org/vulnerability/CVE-2024-34447?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2016-1000338 | Bouncycastle does not validate the format of certificates, which may allow attackers to inject hidden data into an application | [[CVE-2016-1000338] CWE-347: Improper Verification of Cryptographic Signature](https://ossindex.sonatype.org/vulnerability/CVE-2016-1000338?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2016-1000343 | Bouncy Castle, in versions before 1.55, generates weak DSA key pairs when using the default value | [[CVE-2016-1000343] CWE-310](https://ossindex.sonatype.org/vulnerability/CVE-2016-1000343?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2024-29857 | Bouncy Castle, in versions before 1.78, is susceptible to excessive CPU consumption when evaluating curve parameters | [[CVE-2024-29857] CWE-125: Out-of-bounds Read](https://ossindex.sonatype.org/vulnerability/CVE-2024-29857?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2016-1000341 | Bouncy Castle versions 1.55 and earlier are susceptible to timing attacks when generating signatures, which may leak private information | [[CVE-2016-1000341] CWE-361](https://ossindex.sonatype.org/vulnerability/CVE-2016-1000341?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2023-33202 | Bouncy Castle before version 1.73 is susceptible to DOS attacks when parsing OpenSSL PEM encoded streams containing X.509 certificates, PKCS8 encoded keys, and PKCS7 objects | [EXPLOIT,THIRD\_PARTY\_ADVISORY](https://github.com/bcgit/bc-java/wiki/CVE-2023-33202) |
| CVE-2025-35036 | Hibernate Validator, before version 7.0.0 may inject user-supplied input into violation messages, which makes the service susceptible to arbitrary code execution | [[CVE-2025-35036] CWE-94: Improper Control of Generation of Code ('Code Injection')](https://ossindex.sonatype.org/vulnerability/CVE-2025-35036?component-type=maven&component-name=org.hibernate.validator%2Fhibernate-validator&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2023-1932 | Hibernate-validators isValid() does not properly verify closing tags in HTML, making the service susceptible to XSS attacks | [ISSUE\_TRACKING,VENDOR\_ADVISORY](https://bugzilla.redhat.com/show_bug.cgi?id=1809444) |
| CVE-2025-52999 | Jackson-core is susceptible to stack overflow errors if it is given a deeply nested document in versions prior to 2.15 | [[CVE-2025-52999] CWE-121: Stack-based Buffer Overflow](https://ossindex.sonatype.org/vulnerability/CVE-2025-52999?component-type=maven&component-name=com.fasterxml.jackson.core%2Fjackson-core&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2020-9488 | Apache Log4j improperly validates certificates, which may lead to man-in-the-middle attacks in versions before 2.12.3 and 2.13.1 | [ISSUE\_TRACKING,MITIGATION,PATCH,VENDOR\_ADVISORY](https://issues.apache.org/jira/browse/LOG4J2-2819) |
| CVE-2023-6378 | Logback version 1.4.11 and earlier are vulnerable to DOS attacks when deserializing untrusted data | [[CVE-2023-6378] CWE-502: Deserialization of Untrusted Data](https://ossindex.sonatype.org/vulnerability/CVE-2023-6378?component-type=maven&component-name=ch.qos.logback%2Flogback-classic&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2021-42550 | Logback version 1.2.7 and earlier is susceptible to arbitrary code execution when attackers are able to edit configuration files | [[CVE-2021-42550] CWE-502: Deserialization of Untrusted Data](https://ossindex.sonatype.org/vulnerability/CVE-2021-42550?component-type=maven&component-name=ch.qos.logback%2Flogback-classic&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2022-1471 | SnakeYaml constructor() is susceptible to remote code execution; use SafeConstructor() instead | [[CVE-2022-1471] CWE-20: Improper Input Validation](https://ossindex.sonatype.org/vulnerability/CVE-2022-1471?component-type=maven&component-name=org.yaml%2Fsnakeyaml&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2022-25857 | Snakeyaml is susceptible to DoS attacks when parsing nested collections | [[CVE-2022-25857] CWE-776: Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')](https://ossindex.sonatype.org/vulnerability/CVE-2022-25857?component-type=maven&component-name=org.yaml%2Fsnakeyaml&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2022-27772 | Older versions of spring-boot is susceptible to directory hijacking. | [[CVE-2022-27772] CWE-668: Exposure of Resource to Wrong Sphere](https://ossindex.sonatype.org/vulnerability/CVE-2022-27772?component-type=maven&component-name=org.springframework.boot%2Fspring-boot&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2023-20883 | Spring boot is susceptible to DoS attacks when using Spring MVC with a reverse proxy cache | [VENDOR\_ADVISORY](https://spring.io/security/cve-2023-20883) |
| CVE-2021-22118 | Spring framework before versions 5.2 are susceptible to privilege escalation by recreating the temporary storage directory | [THIRD\_PARTY\_ADVISORY](https://security.netapp.com/advisory/ntap-20210713-0005/) |
| CVE-2020-5421 | Older versions of Spring Framework are susceptible to RFD attacks, where authentic looking prompts and downloads are injected into a users browsing session from incomplete server responses, such as missing a content-disposition header, lacking security headers, or lacking CSRF tokens (Twingate Team, 2024). | [VENDOR\_ADVISORY](https://tanzu.vmware.com/security/cve-2020-5421) |
| CVE-2016-1000027 | Pivotal Spring Framework is not meant to deserialize untrusted data and is susceptible to remote code execution | [[CVE-2016-1000027] CWE-502: Deserialization of Untrusted Data](https://ossindex.sonatype.org/vulnerability/CVE-2016-1000027?component-type=maven&component-name=org.springframework%2Fspring-web&utm_source=dependency-check&utm_medium=integration&utm_content=12.1.0) |
| CVE-2020-1938 | Apache JServ Protocol is susceptible to arbitrary code execution and attackers can gain access to any file on a web server on versions before Apache Tomcat 9.0.31 | [THIRD\_PARTY\_ADVISORY](http://lists.opensuse.org/opensuse-security-announce/2020-03/msg00025.html) |
| CVE-2023-44487 | Apache Tomcat is vulnerable to DoS attacks by request cancellation through the HTTP/2 protocol | [VENDOR\_ADVISORY](https://www.netlify.com/blog/netlify-successfully-mitigates-cve-2023-44487/) |

**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.

The following actions are recommended, based on the previous reviews. See “Manual Code Review” section for recommendations on code updates:

1. Legal Compliance
   1. Implement end to end encryption of payment data (PCI PTS section B, GDPR)
   2. Minimize data collected from users to the minimum necessary for business functions (GDPR)
   3. Delete data after it is no longer necessary (GDPR)
   4. Implement authorization features to view financial data (Gramm-Leach-Bliley)
   5. Enforce multi-factor authentication (GDPR, Gramm-Leach-Bliley)
   6. Implement logging for security exceptions and events (Gramm-Leach-Bliley)
   7. Implement staff training for security and social engineering attacks (Gram-Leach-Bliley)
   8. Regularly review the security of third-party applications and libraries through static analysis
2. Dependency Review
   1. Verify the size and depth of nested, untrusted data
   2. Update Apache Log4j to V2.25.0
   3. Update Logback to version 1.5.18
   4. Always use SafeConstructor() over Constructor() while using snakeYaml
   5. Update spring-boot to at least 3.4.x
   6. Update Spring Framework to at least 6.1.x
   7. Ensure all json responses contain a content-disposition header, have a CSRF token, have a security header (like X-Content-Type-Options and X-frame-Options), and json responses are sanitized for malicious data (Twingate team, 2024)
   8. Do not use the Spring Framework to deserialize untrusted data
   9. Update Apache Tomcat to 11.0.9 or later

**References**

Köller, J. (2024). GLBA Cybersecurity Requirements: What Gramm-Leach-Bliley Means for Your IT. <https://www.tenfold-security.com/en/glba-compliance/>

Modular Security Requirements. (2025). Payment Card Industry (PCI) PIN Transaction Security (PTS) Point of Interaction (POI). <https://docs-prv.pcisecuritystandards.org/PTS/Standard/PCI_PTS_POI_SRs_v7.0.pdf>

Twingate Team. (2024). What is a Reflected File Download? How it Works and Examples. <https://www.twingate.com/blog/glossary/reflected%20file%20download>

Whitford, B. (n.d.). What is GDPR, the EU’s New Data Protection Law?. GDPR.eu. https://gdpr.eu/what-is-gdpr/