# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[07/19/2025]** | **[Derek Castro]** | **Week 3** |

## 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

[Derek Castro]

**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?

Artemis Financial handles customers’ personal financial plans including saving, retirement, investments, and insurance. It is crucial that communications remain secure due to the handling of this important information which may include other sensitive information such as date of birth, social security numbers, addresses, etc. Although it wasn’t stated in the scenario whether Artemis Financial handles international business, this may be implied since Global Rain specializes in international affairs. Most government agencies require security clearance which should be considered when handling sensitive data involving government operations. This may require some form of encryption language to be utilized to minimize any risk of exposure. Most modern businesses utilize cloud services for improved scalability and cost-effectiveness, therefore it is adamant that Artemis Financial maintains up-to-date software maintenance to reduce the risk of potential bugs which can result in potential vulnerabilities towards their web-based securities.

**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 – Important area to act as the first line of defense when handling the aforementioned sensitive information. Strict validation helps minimize malicious threats.
* Code Quality – The quality of the code reflects the overall security of the application. This includes up-to-date dependencies and logic errors which can lead to potential vulnerabilities.
* APIs – Monitoring API behaviors help reduce the risk of potential leaks since each request may contain sensitive information.
* Code Error – Error handling is one of the most critical areas of security hackers abuse to obtain sensitive data, or potential attacks. Artemis Financial should maintain proper error handling to ensure there is no risk for potential abuse.
* Cryptography – Encryption should be a requirement for this application due to the sensitive information that will continuously be handled.

**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
  + CRUDController.java is missing input validation for “business\_name”. This can lead to vulnerabilities like crashes through excessively long user input.
  + GreetingController.java is missing input validation for “name”. This can lead to vulnerabilities like crashes through excessively long user input.
* Code Quality
  + Customer.java is missing a modifier which limits how it can be handled. This is potentially leaving it vulnerable for misuse.
* APIs
  + It seems like anyone can request an API since there is no sign of a login mechanic.
  + Endpoints /read and /greeting both do not require any form of access control.
* Code Error
  + DocData.java handles exceptions which are printed through the console. If logs become visible, then sensitive information like class names can be exposed through the console.
* Cryptography
  + There are no encryption codes included in the above files.
  + After manually reviewing the dependencies through the pom.xml file, “bouncycastle” is included but shows an outdated version “1.46” compared to its latest version “1.81”. If ever utilized in the application, this can lead to many bugs, and potential vulnerabilities related to earlier releases.

**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

|  |  |  |
| --- | --- | --- |
| Name | CVE Code | Description |
| Bouncy Castle bcprov-jdk15on-1.46.jar | CVE-2013-1624, CVE-2015-6644, CVE-2015-7940, CVE-2016-1000338, CVE-2016-1000341, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000345, CVE-2016-1000346, CVE-2018-5382, CVE-2020-0187, CVE-2020-26939, CVE-2023-33201, CVE-2023-33202, CVE-2024-29857, CVE-2024-30171, CVE-2024-34447 | Contains multiple vulnerabilities and deserialization issues that can lead to RCE or Dos.   Solution includes update to later version 1.8+. |
| Spring Boot and Framework spring-boot-2.2.4.RELEASE.jar spring-boot@2.2.4.RELEASE | CVE-2016-1000027, CVE-2018-11039, CVE-2018-11040, CVE-2018-1257, CVE-2020-5421, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-27772, CVE-2023-20861, CVE-2023-20873, CVE-2023-20883, CVE-2024-22259 | Vulnerable to RCE, injection, and path traversal attacks.  Solution includes updating to later Spring Boot (3.5.3) and Spring Framework (6.2.8) versions. |
| logback-classic-1.2.3.jar | CVE-2021-42550, CVE-2023-6378 | Vulnerabilities towards malicious file configuration to execute arbitrary code.  Solution includes updating to later versions that are later than at least 1.2.7 |
| logback-core-1.2.3.jar | CVE-2021-42550, CVE-2023-6378, CVE-2024-12798, CVE-2024-12801 | Vulnerabilities towards malicious file configuration to execute arbitrary code.  Solution includes updating to later versions that are later than at least 1.2.7 |
| log4j-api-2.12.1.jar | CVE-2020-9488, CVE-2021-44228, CVE-2021-44832, CVE-2021-45046, CVE-2021-45105 | Vulnerable to logs being leaked, Dos attacks, and RCE.   Solution is included in Apache Log4j 2.12.3 and 2.13.1. Recommended to update to versions 2.16.0 or later. |
| snakeyaml-1.25.jar | CVE-2017-18640, CVE-2021-4235, CVE-2022-1471, CVE-2022-25857, CVE-2022-3064, CVE-2022-38749, CVE-2022-38750, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854 | Vulnerable to deserialization, and DoS attacks  Recommended to update SnakeYaml later than 1.26. |
| jackson-databind-2.10.2.jar jackson-core-2.10.2.jar | CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116 CVE-2025-49128 | Vulnerable to DoS, and RCE such as external entity attacks (XXE) which threatens data integrity.   Solution is to update version later than 2.15.2 |
| tomcat-embed-core-9.0.30.jar | CVE-2019-17569, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-13943, CVE-2020-17527, CVE-2020-1935, CVE-2020-1938, CVE-2020-8022, CVE-2020-9484, CVE-2021-24122, CVE-2021-25122, CVE-2021-25329, CVE-2021-30640, CVE-2021-33037, CVE-2021-41079, CVE-2021-43980, CVE-2022-29885, CVE-2022-34305, CVE-2022-42252, CVE-2023-28708, CVE-2023-41080, CVE-2023-42795, CVE-2023-44487, CVE-2023-45648, CVE-2023-46589, CVE-2024-21733, CVE-2024-23672, CVE-2024-24549, CVE-2024-34750, CVE-2024-38286, CVE-2024-50379, CVE-2024-52316, CVE-2024-54677, CVE-2024-56337, CVE-2025-24813, CVE-2025-31651, CVE-2025-46701, CVE-2025-48988, CVE-2025-49124, CVE-2025-49125 | Vulnerable to DoS attacks through HTTP Request Smuggling, RCE attacks through internal Dot, and/or malicious content that can be added through uploaded files via Apache Tomcat Servlet.   Recommended Solution: Update to later versions including 11.0.8, 10.1.42, or 9.0.106 |
| hibernate-validator-6.0.18.Final.jar | CVE-2020-10693, CVE-2023-1932, CVE-2025-35036 | Bypass vulnerabilities were possible via bug through the message interpolation processor, data leak, HTML injection and/or XSS attack  Recommended to update to version later than 7.0.0. |
| spring-web-5.2.3.RELEASE.jar | CVE-2016-1000027, CVE-2020-5421, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-22243, CVE-2024-22259, CVE-2024-22262, CVE-2024-38809, CVE-2024-38828, CVE-2025-41234 | Vulnerable to Dos Attack, malicious inputs, HTML injections, and/or SSRF attack. |
| spring-webmvc-5.2.3.RELEASE.jar | CVE-2016-1000027, CVE-2020-5421, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-22259, CVE-2024-38816 | Solution includes updating to later Spring Boot (3.5.3) and Spring Framework (6.2.8) versions. |
| spring-context-5.2.3.RELEASE.jar | CVE-2016-1000027, CVE-2020-5421, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-22259, CVE-2025-22233 | Vulnerable to Dos Attack, malicious inputs, HTML injections, and/or SSRF attack.  Solution includes updating to later Spring Boot (3.5.3) and Spring Framework (6.2.8) versions. |
| spring-expression-5.2.3.RELEASE.jar | CVE-2016-1000027, CVE-2020-5421, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-22259, CVE-2024-38808 | Vulnerable to Dos Attack, malicious inputs, HTML injections, and/or SSRF attack.  Solution includes updating to later Spring Boot (3.5.3) and Spring Framework (6.2.8) versions. |
| spring-core-5.2.3.RELEASE.jar | CVE-2016-1000027, CVE-2020-5421, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2022-22950, CVE-2022-22965, CVE-2022-22968, CVE-2022-22970, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-22259 | Vulnerable to Dos Attack, malicious inputs, HTML injections, and/or SSRF attack.  Solution includes updating to later Spring Boot (3.5.3) and Spring Framework (6.2.8) versions. |

All findings were done through the Maven Dependency Check through Eclipse-Workspace. All codes were analyzed through the CVE.org record database.

**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.

* Based on the excessive vulnerabilities found through the dependency check, of the most important steps is to upgrade all software versions to up-to-date releases that fix all the previously stated vulnerabilities. This software includes Spring Framework, Spring Boot, Jackson, Bouncy Castle, Hibernate Validator, and Tomcat.
* Input validation limitations need to be included in the controller fields such as “business\_name”, and “name” to avoid malicious input.
* There also needs to be access control that requires authentication/authorization checks on exposed endpoints such as /read and /greeting.
* Logs need to either be private or handled to not print sensitive information to the console when handling exceptions/errors.
* Add modifiers to customer.java classes to reduce unintended access.

These steps should help minimize the risk of malicious abuse and improve the overall security of the application.