# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/24/2025** | **Karol Guerra** |  |

## 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

Karol Guerra

**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?

The company values secure communications to ensure that the movement and keeping of data complies with the CIA triad (confidentiality, integrity and authenticity). Having secure communications protects against attacks or unauthorized access to data.

* Are there any international transactions that the company produces?

When a company does business internationally, it must adhere to a multitude of global data protection regulations. This includes significant laws such as the General Data Protection Regulation (GDPR) to protect data. Compliance with these laws is for a company as it ensures the secure storage and processing of data which can prevent legal consequences and repercussions.

* Are there governmental restrictions on secure communications to consider?

Adhering to government restrictions, regulations, and procedures crucial for offering any application in a foreign nation. Some common government laws that can help businesses conduct appropriately would be FTCA, or GLBA.

* What external threats might be present now and in the immediate future?

Threats can exist in many different areas of an organization. When discussing external threats it is important to be able to recognize threats, breaches, exploits, CVE’s etc. in a timely manner to minimize the impact that an external threat may create. These threats can include malicious actors, DOS attacks, social engineering, phishing, etc.

* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

When conducting an assessment of the security, it is vital to consider that all systems and security measures are kept up to date and regularly maintained. This includes the integration of things such as regular vulnerability scans, SOAR, open source libraries, microservices, etc, while prioritizing the secure implementation of these elements.

**2. Areas of Security**

After conducting a thorough dependency check on Artemis Financials software application and evaluating the process flow diagram, several critical security concerns have come to light.

One of the primary vulnerabilities identified is related to input validation. Specifically, one of the dependencies, Jackson-Databind, poses a significant risk of deserialization attacks. To mitigate this threat, it is essential to implement robust input validation to ensure that only expected and safe data is processed.

One area observed in the flowchart is the use of secure APIs for enhancing overall application security. The Apache Log4j API has a few vulnerabilities that can allow attackers to control log messages or execute arbitrary code from LDAP servers. By leveraging well-designed APIs that adhere to security best practices, Artemis Financial can minimize risks related to unauthorized access or data breaches. Employing API security measures, such as authentication and proper access controls, will further protect sensitive operations and data exchanges between components.

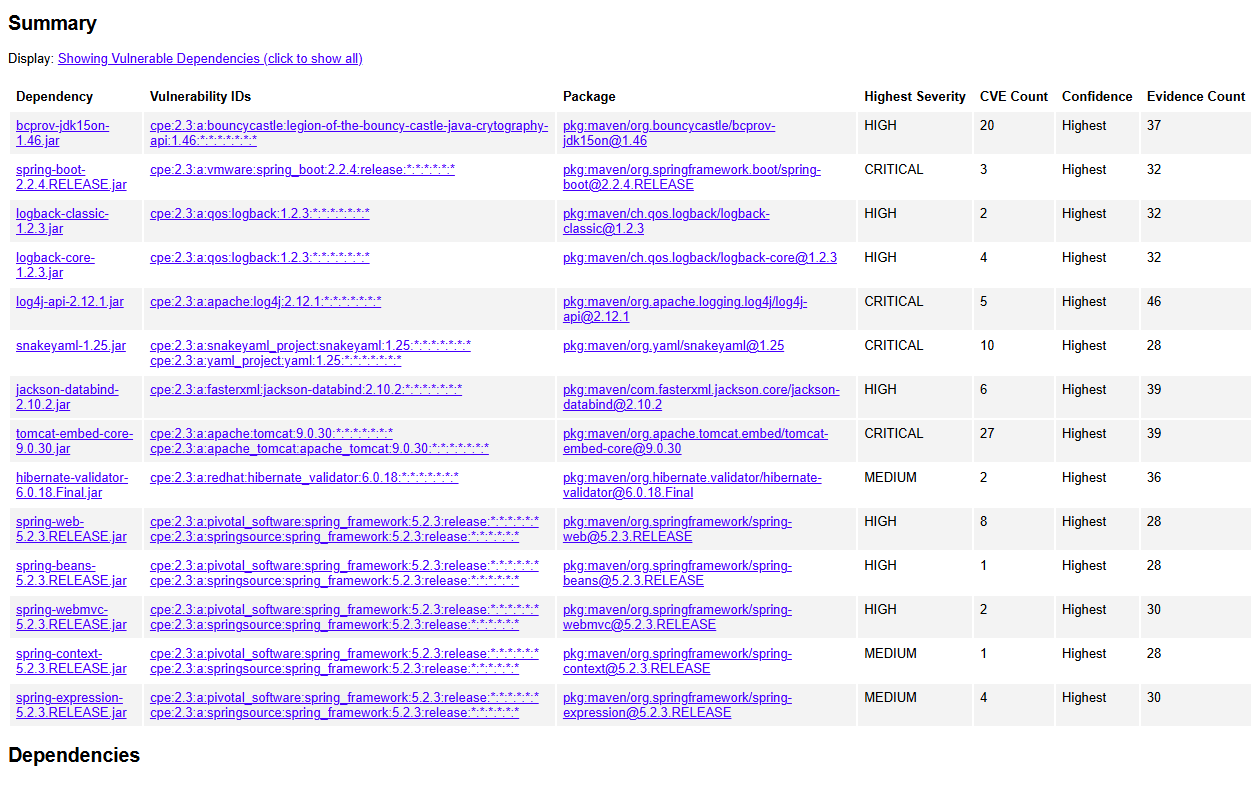
Another crucial aspect of security is encryption, which plays a vital role in safeguarding financial data both at rest and in transit. The dependency check report has flagged vulnerabilities in cryptographic libraries, including Bouncy Castle, which may lead to weak or improperly configured encryption mechanisms. Establishing secure encryption protocols will help maintain the integrity and confidentiality of sensitive data within the application.

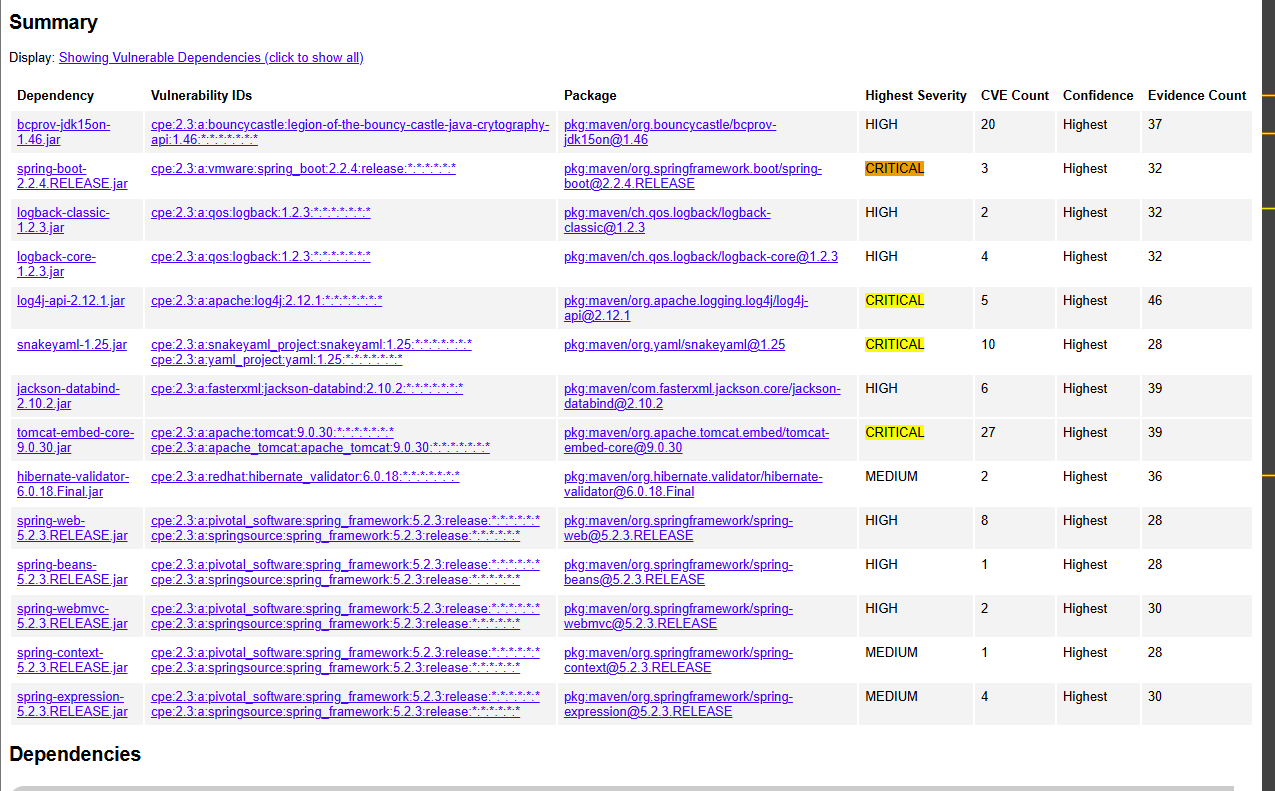
Error handling and logging are equally important areas of focus. Vulnerabilities found in components such as Tomcat Embed Core could potentially result in unintentional information disclosure. By adopting secure logging practices and crafting user-friendly error messages, the risk of exposing confidential information can be significantly reduced.

By addressing these security concerns—input validation, encryption, and error handling—Artemis Financial can strengthen its software application, ensuring it complies with the stringent security standards required in the financial industry.

**3. Manual Review**

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



****

**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

Some of the vulnerability codes of known vulnerabilities are Bouncy Castle with as CVE count of 20, Tomcat embedded Core with a CVE count of 27, Apache Log4j-api with a CVE count of 5, and project snakeyaml with a CVE count of 10. These are some of the critical vulnerabilities that were found in the report and the other less criticals are posted above in the screen shot with the CVE report.

* A brief description and recommended solutions provided by the dependency-check report

To address the issues found with the vulnerabilities it is important to read through the dependency checker and navigate the information provided about the vulnerabilities.

To provide resolutions on the issues associated with Bouncy Castle, it is recommended to upgrade to the latest version that resolves known vulnerabilities. Additionally, validating all cryptographic operations is critical as well and ensuring that certificates are properly validated. Updating is important as stated by the dependency check Versions 1.55 and earlier are more vulnerable for attacks.

Tomcat vulnerabilities include directory traversal attacks, file exposure issues, and DoS risks due to resource exhaustion. Updating to a patched version of Tomcat greater than 9.0.45 or 10.0.5 or later and applying secure configurations will help reduce vulnerabilities and reduce avenues for attacks.

Vulnerabilities in Log4j-api, regarding improper handling of user input during logging, can allow attackers to exploit logging mechanisms for code injection or information leakage. 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.

Vulnerabilities in SnakeYAML involve insecure processing of untrusted input. This could lead to deserialization attacks or denial of service. Upgrading to a secure version of SnakeYAML at least 2.0 or higher as stated in the dependency checker

* Any attribution that documents how this vulnerability has been identified or documented previously

[Include your findings here.]

CVE-2021-44832

<https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-apache-log4j-qRuKNEbd>

CVE-2016-1000341

<https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html>

CVE-2021-25329  
 <https://lists.debian.org/debian-security-announce/2021/msg00072.html>

CVE-2022-1471

* - <https://github.com/go-yaml/yaml/pull/375>
* - <https://lists.debian.org/debian-lts-announce/2023/07/msg00001.html>

**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 manual review and static testing results reveal critical vulnerabilities in Artemis Financial’s software application that appear to stem from outdated and insecure dependencies. Some key findings include vulnerabilities in Bouncy Castle, Tomcat Embedded Core, Apache Log4j-api, and SnakeYAML dependencies within the program. These pose risks such as weak encryption, directory traversal attacks, code injection, deserialization attacks, and denial of service attacks. Examples would be Tomcats vulnerabilities which can allow unauthorized access to sensitive files, while vulnerabilities in SnakeYAML expose the application to deserialization risks that could compromise its stability and security. Additionally, the presence of known critical vulnerabilities in Log4j highlights the risk of exploitation via improperly logged user input, potentially leading to unauthorized code execution.

The findings produced by the dependency checker address the importance of the security vulnerabilities within third-party libraries that are used in the Artemis Financial’s software, which can be detrimental for financial applications handling sensitive user data. The risks identified include data breaches, unauthorized access, and operational disruptions, all of which could undermine user trust and regulatory compliance. As a result, the application requires immediate attention to its critical vulnerabilities and others to secure its core components and mitigate these critical vulnerabilities.

After running the vulnerability scan there were a few results that populated some potential CVE’s with critical ratings ranging from medium risk up to high and critical risk. When vulnerabilities are present this means that threats either