# CS 305 Project One Template

**Document Revision History**

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | [1-21-26] | [Addison Janovich] | |

**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 following 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 sure 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**

[Addison Janovich]

**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?

[Secure communications to Artemis Financial are very valuable. First, it protects sensitive data. Artemis handles personally identifiable information and financial data. Secure communications help prevent data interception, tampering, and unauthorized disclosure. Also, clients expect confidentiality when sharing financial data; a breach would break that trust and possibly result in loss of clientele. Lastly, secure communication supports compliance with financial and data-protection regulations.

There is a possibility that Artemis conducts international transactions. A few examples include clients who don't live in the U.S. and investment portfolios that have foreign assets.

Yes, a few government restrictions on secure communications to consider are GLBA or Gramm-Leach-Bliley Act, which mandates safeguards for customer financial information. Also, some countries may actually regulate the use of strength or encryption rather than banks.

There are many current threats that need to be noted and learned how to prevent. Man-in-the-middle attacks are becoming more common when encryption is weak or misconfigured. Another example is SQL injection or input validation attacks. Automated bots attacking APIs might become a new threat once more development happens.

Some modernization requirements and considerations for including open-source libraries. Libraries must be updated regularly and monitored using tools such as dependency management software. Also, API design that might implement token-based authentication, implement rate limiting, and API gateways.

**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.

[Every area of the vulnerability assessment process flow diagram applies to the Artemis Financials software application in some way.

**Architecture reviews** help identify points of failure and insecure data flows between components, which is relevant because early architectural weaknesses can reveal other vulnerabilities.

**Input validation** is required because the REST API accepts user input, including financial data, credentials, and other sensitive information. Invalid or malicious input can lead to API injection attacks. Input validation is highly relevant given that APIs are increasingly under attack.

Again, **API**s are exposed externally and are prime targets for attack. APIs require authentication and authorization.

**Cryptography** applies because Artemis transmits and stores sensitive financial and personal data. Encryption is essential for all data, especially when financial data is moved between accounts. Strong encryption prevents attacks such as man-in-the-middle and data leakage.

**Client/server**: the application follows a client-server model, and communication occurs over networks that may not always be trusted. Client/server protects against replay attacks.

**Secure error handling** is very important because improper error messages can expose internal logic or database structure. This is very relevant because it is crucial that financial applications not leak any internal details.

**Code quality** is another important part of Artemis; high-quality code reduces vulnerabilities caused by insecure dependencies and logic flaws. High-quality code supports long-term maintainability.

**Encapsulation** applies because financial data must be strictly controlled and protected. This prevents unauthorized access.

Each **code review** is necessary because each layer may pose security risks; it is important to check each layer for vulnerabilities that could be exploited.

Lastly, a **mitigation plan** is required to document the risks and remediation steps, ensuring action is taken.   ]

**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. CRUDController class does not have any authentication or authorization at all.
2. Customer class- in the customer class the class name needs to be uppercased. Also a getter class, lastly a validation in deposit.
3. DocData- the use of “root”,”root” root has unrestricted database access so the enture database could be takenover if compromised.
4. GreetingsController- unauthenticated endpoint and input validation
5. myDateTime- no input validation

   ]

**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

[CVE-2016-1000338, DSA signature validation allows injection of extra elements, potentially introducing “invisible” data into signed structures., Upgrade to Bouncy Castle ≥ 1.56, Mailing lists, third-party advisories, MITRE: CVE
CVE-2020-15522, Timing issue in EC math library could expose private keys via multiple deterministic ECDSA signatures., Upgrade to Bouncy Castle ≥ 1.66, Vendor advisory: NetApp Advisory

CVE-2023-33202, PEMParser DoS via crafted ASN.1 data; may cause OutOfMemoryError.,Upgrade to Bouncy Castle ≥ 1.73, CVE-2015-7940, Invalid curve attack in ECDH key exchange; could leak private keys., Upgrade to Bouncy Castle ≥ 1.51,Red Hat, Debian, Oracle advisories, MITRE: CVE
CVE-2020-36518, Unsafe deserialization allows remote code execution., Upgrade to Jackson ≥ 2.15,MITRE: CVE
CVE-2020-9546, Deserialization vulnerability could execute arbitrary code., Upgrade to Jackson ≥ 2.15,MITRE: CVE
CVE-2019-16335, Deserialization leads to remote code execution., Upgrade to Jackson ≥ 2.12,MITRE: CVE
CVE-2020-5398, Remote code execution via Spring Expression Language injection., Upgrade Spring to ≥ 5.3.30 / Spring Boot ≥ 2.7,MITRE: CVE
CVE-2019-3799, RCE in Data Binding; attacker may exploit unsafe deserialization., Upgrade Spring to ≥ 5.2.13,MITRE: CVE
CVE-2018-1275, Unsafe deserialization leading to RCE., Upgrade Spring ≥ 5.2.3
CVE-2020-13935, Remote code execution via request parsing., Upgrade to Tomcat ≥ 9.0.54,MITRE: CVE
CVE-2020-1938, Ghostcat: AJP connector information disclosure.,Disable AJP or upgrade to Tomcat ≥ 9.0.31,MITRE: CVE
CVE-2020-2150, Unsafe YAML deserialization allows remote code execution., Upgrade SnakeYAML ≥ 2.x,MITRE: CVE
CVE-2021-27291, Deserialization of untrusted YAML may execute arbitrary code., Upgrade SnakeYAML ≥ 2.x,MITRE: CVE
CVE-2021-44228, Remote code execution via JNDI lookup (“Log4Shell”)., Upgrade Log4j ≥ 2.17.1,MITRE: CVE
CVE-2021-45046, Information leak / DoS via crafted log messages., Upgrade Log4j ≥ 2.17.1,MITRE: CVE

]

**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.

[After interpreting the results from the manual review and static testing report, I have identified steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application. What would solve all of the vulnerability issues is simply updating to the latest versions. Usually these versions contain security patches that address any vulnerability found and patch it up to keep everything protected.]