CS 305 Project One Template

Document Revision History

Version	Date	Author	Comments
1.0	05/19/2025	Eric Foster	

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
Eric Foster

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:

Overview

Artemis Financial is a company that creates personalized financial plans for individuals. These plans include savings, retirement, investments, and insurance. To better serve its customers and stay competitive, Artemis is updating its software systems and has developed a web-based application that allows for online communication and transactions. As part of this modernization, Artemis is focused on making sure the new system is safe and protected from online threats.

Artemis Financial has several important requirements when it comes to protecting its systems and customer data. First, because the company handles sensitive financial and personal information, it must ensure that all communication, such as sharing account or investment details, is kept private and secure. Using strong security measures to protect this information is essential for maintaining customer trust and complying with financial regulations.

In addition, Artemis may serve international clients, which means it must also follow global privacy laws like the General Data Protection Regulation (GDPR) in Europe and financial regulations such as the Gramm-Leach-Bliley Act in the United States (Federal Trade Commission, 2024; Intersoft Consulting, 2013). These laws require clear policies for handling and protecting customer data. As the company modernizes its systems using faster, more flexible tools, it must also be aware of growing cyber threats such as hackers, phishing scams, and

software vulnerabilities. To stay protected, Artemis must regularly test and update its systems and ensure that any new technologies meet strict security standards.

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.

It is recommended to focus on the following areas to cater to the needs above:

Area of Security	Description
Input Validation	This is very important in ensuring that all input data received is exactly what is expected to come through. This becomes important when receiving data such as account details, transaction searching, etc.
API Security	This ensures that each interaction with the online API is secure and does not leak any sensitive data, such as usernames, passwords, account information, etc.
Code Quality	Code Quality – This ensures that we are adhering to best security practices and patterns, which will subsequently result in better overall security for the updated system.
Cryptography	This ensures that all sensitive data has been encrypted and protects that data from man-in-the-middle attacks, which is described by IBM below titled "**NOTE.**"

By focusing on these areas of security, Artemis Financial will have a better understanding of how to better secure the new system to ensure it adheres to the goal of making sure the new system is safe and protected from online threats.

NOTE: A man-in-the-middle (MITM) attack is a cyberattack in which a hacker steals sensitive information by eavesdropping on communications between two online targets such as a user and a web application (Lindemulder & Kosinski, 2024).

3. Manual Review

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

- In file application.properties, on line 14, the server port is indicating an 80 port number, which translates to using HTTP protocol (Unsecured) instead of HTTPS protocol (Secured).
- For files customer.java and myDateTime.java, the naming conventions for these files and the subsequent class name (which is simply the file name) are bad practice forms of the naming conventions best practices. This can lead to difficulty reading and understanding what that class may be doing and ultimately can have a negative impact on code quality.
- In file myDateTime, the variables on line 5 – 7 and the methods on lines 9 and 14, do not explicitly state an access modifier. Because of this, java sets it to "default", which simply means that only classes within the same package can see, modify, and utilize that variable. While this may work for now, this can lead to possible unwanted behaviors and may cause unwanted changes to the variables due to access from other files in the codebase. This can also make it much more difficult to find bugs if a problem does arise.
- In file CRUDController.java, on line 13, this method does not include any sort of input validation for the parameter it is receiving. This can lead to unwanted behaviors and may also lead to allowing dangerous information through into the system.
- In file GreetingController.java, on line 16, this method does not include any form of input validation. This can lead to unwanted behaviors and opens doors to potentially dangerous information getting into the system.
- In file myDateTime.java, on line 14, this method does not include input validation. This can lead to unwanted behaviors such as incorrect time formatting and can also lead to potentially dangerous information getting into the system.

- In file DocData.java, on line 27, this line provides the information for connecting to the database link, the username, and the password. This line provides sensitive data publicly and in an unsecured manner. This information is used to access, at an administrative level, the database linked.
- In file customer.java, on line 7, this method provides sensitive data in an unsecured manner, giving access to sensitive data in a public manner.

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:

Dependency	Description	Highest Severity
spring-boot-2.2.4.RELEASE.jar	The foundation of the application. It simplifies setup and development of Java-based web services. The core framework has serious vulnerabilities that can expose the entire system to remote attackers or allow data leakage.	**CRITICAL**
log4j-api-2.12.1.jar	Helps record system activity and errors so developers can monitor and troubleshoot the application. This is part of a widely known security issue (Log4Shell). This could allow attackers to send specially crafted messages to take control of systems.	**CRITICAL**
snakeyaml-1.25.jar	Reads configuration settings written in a format called YAML, which is used to control how the application behaves. If exploited, it can let attackers run harmful commands from those files.	**CRITICAL**
tomcat-embed-core-9.0.30.jar	Powers the internal web server that runs the application so it can handle requests from users. Vulnerabilities here may expose private files or give attackers control over system behavior.	**CRITICAL**

Dependency	Description	Highest Severity
tomcat-embed-websocket-9.0.30.jar	Adds support for real-time communication features, such as live updates or chat functions. Vulnerable to attacks that can interfere with or monitor user data.	CRITICAL
spring-web-5.2.3.RELEASE.jar	Supports building web features and handles how the application connects and responds to users online. Weaknesses could allow attackers to bypass protections or expose sensitive info.	CRITICAL
spring-webmvc-5.2.3.RELEASE.jar	Manages how user requests are processed and how responses (like web pages or data) are returned. Flaws here may let attackers view or change private data.	CRITICAL
spring-context-5.2.3.RELEASE.jar	This helps organize and manage different parts of the application, so they work together smoothly. Security issues here could let attackers interfere with app behavior or access data.	CRITICAL
spring-expression-5.2.3.RELEASE.jar	Allows developers to write simple expressions to control logic or behavior within the application. If misused, it could let harmful commands be run inside the system.	CRITICAL
spring-core-5.2.3.RELEASE.jar	Contains the core tools that make the Spring framework function properly. Flaws here may threaten the app's stability and allow deep system attacks.	CRITICAL
bcprov-jdk15on-1.46.jar	Provides encryption tools to help protect sensitive data, such as passwords or financial details. Vulnerabilities could let attackers break encryption, weakening privacy protections.	HIGH
logback-classic-1.2.3.jar	Another tool for tracking what the application is doing, often used alongside log4j. If misconfigured or exploited, attackers might gain insight into the system or run unauthorized actions.	HIGH
logback-core-1.2.3.jar	A supporting component for logging activity within the system. Similar risks as above. This can be a backdoor if not securely configured.	HIGH

Dependency	Description	Highest Severity
jackson-databind-2.10.2.jar	Converts data between Java objects and formats like JSON, making it easier to send and receive information. It can be exploited to run unintended code if attackers send malicious data.	**HIGH**
jackson-core-2.10.2.jar	A helper library used by jackson-databind to process data formats. Less severe but may contribute to vulnerabilities in how data is handled.	**MEDIUM**
hibernate-validator-6.0.18.Final.jar	Ensures that user input (like form fields) meets certain rules, such as valid email addresses or required fields. If flawed, harmful or incorrect data may be accepted by the application.	**MEDIUM**

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.

To address the security risks identified in both the manual code review and automated dependency scan, Artemis Financial should take several important steps. First, the software should be updated to use secure communication methods by switching from HTTP to HTTPS, ensuring that customer data is encrypted while it's being transmitted. Input validation should be added to all parts of the application where users enter information. This helps prevent malicious or incorrect data from causing errors or vulnerabilities in the system. Additionally, database usernames and passwords should never be stored directly in the code. Instead, they should be moved to a secure configuration file or environment variables that are not publicly accessible.

Several outdated libraries used in the application contain serious known vulnerabilities that could allow attackers to take control of the system or gain access to private data. These include critical components such as the Spring Framework, Log4j, and Tomcat, which power the core functionality of the web application. Each of these should be updated to the latest secure versions as soon as possible. Other libraries like SnakeYAML, Jackson, and Bouncy Castle (used

for encryption) also need to be upgraded to ensure they do not introduce security flaws into the system. Keeping these tools up to date is one of the simplest and most effective ways to reduce risk (OWASP, 2021).

Finally, the overall quality and structure of the code should be improved by following consistent naming conventions and specifying clear access rules for variables and methods. These changes help make the code easier to read and maintain, reducing the chances of introducing new bugs in the future.

References

Federal Trade Commission. (2024). *Gramm-Leach-Bliley Act*. Federal Trade Commission. https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act

Intersoft Consulting. (2013). *General Data Protection Regulation (GDPR) – Final text neatly arranged*. General Data Protection Regulation (GDPR). https://gdpr-info.eu/art-3-gdpr/

Lindemulder, G., & Kosinski, M. (2024, June 11). *What Is a Man-in-the-Middle (MITM) Attack? | IBM*. IBM. https://www.ibm.com/think/topics/man-in-the-middle

OWASP. (2021). *A06 Vulnerable and Outdated Components - OWASP Top 10:2021*. Owasp.org. https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/