# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **Mar 23 2025** | **Barbara Kendall** |  |

## 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

Barbara Kendall

**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?

Artemis greatly values secure communication as their customers require this as they are government agencies, entrepreneurs and businesses. Artemis. Financial handles their financials, as they are aptly named, and therefore there is a lot of sensitive data they deal with that hackers could greatly exploit. The financial transactions that Artemis conducts would contain sensitive data like account numbers and client information. Having this information hacked could lead to financial loss/ theft as well as the credibility and trust of Artemis as a business be at risk. Depending on what happened, Artemis could also be liable for government fines if they do not handle this sensitive information with the care expected.

* Are there any international transactions that the company produces?

Yes, Artemis conducts international transactions. They connect customers to accounts around the world as well as customer to customer. Scams are be prevalent here as well as country by country restrictions in sending money. Artemis must follow government guidelines here in order to avoid any issues there.

* Are there governmental restrictions on secure communications to consider?

There are European Union restrictions that come with the type of transactions that Artemis conducts. The General Data Protection Regulation regulates communications for example and requires data handlers/ service providers to make sure data is protected from potential threat. This means Artemis must take special care when handling customer data to minimize security threats.

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

The present external threats include authentication threats which are potential unauthorized access to the system via details on accounts in the system. Man-in-the-middle threats which is a compromise of the communication between client and application. As well as denial of service threats which is sending massive amounts of traffic to the application in order to use up memory and resources to slow down the system and crash it.

External threats in future would be API threats. API threats are important because one day there may be vulnerabilities with how the API methods and tokens are integrated into the system. This could mean compromised API calls.

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

Modernization requirements that must be considered include making small changes in the software to reduce the risk of crashing the system by making big changes on the crucial legacy and open source libraries. This way you can make sure the system is working well with each small change made before moving on to another one.

Modernization can also provide flexibility and accessibility with changes made. The idea is to make the application, technologies and source libraries more accessible which will allow future changes to the system be easier to make.

**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.

Artemis Financial has the following security concerns in its application that need to be addressed, including API security, secure coding practices, cryptography, and client-server communication. Since the application uses API calls, it is crucial to implement a secure configuration to reduce the risk of security breaches between the system and the API.

Cryptography is a key area to look at, as Artemis Financial transmits and accesses data over the internet. Without quality encryption, sensitive information like company or customer data could be compromised, which could affect company integrity and confidentiality. Ensuring proper certificate validation is very important.

Client-server security is also important, as it handles the communication between users and the system's backend. Since clients interact with the application through the front-end interface, securing this connection is very important to prevent unauthorized access or data breaches.

Secure coding is also necessary to maintain the integrity of the business. Proper code sanitization helps prevent vulnerabilities, especially with exception handling and error management, ensuring that the application will be secure.

**3. Manual Review**

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

**Direct Object Reference**

During the manual review, I was able to find a potential vulnerability in the CRUDController.java file where the application could potentially expose the internal objects. They are able to be passed and accessed through injections of code. When the value “business\_name” is passed through the CRUD method it may expose the DocData object database access vulnerability.

**Data Access**

I was also able to find a vulnerability when it comes to data access. In the DocData.java file, in the method to access data which involves the definition of the location of the database, the username and password have some potential vulnerabilities. The username and password are both root which is not recommended as the username and password combination could be easily guessed by anyone. This could allow unauthorized users to easily use a brute force attack to gain access to the system.

**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

1. Bouncy Castle Crypto Package (bcprov-jdk15on-1.46.jar)  
   This dependency has 15 known vulnerabilities, including CVE-2015-6644, which can lead to information disclosure. This would allow attackers to access private data stored in the database. To mitigate this risk, it is important to regularly update both the application and the underlying operating system to prevent security breaches. (Source, 2016).
2. Apache Log4j API (log4j-api-2.12.1.jar)  
   This dependency is affected by CVE-2020-9488, which results from improper certificate validation. This vulnerability allows an SMTPS connection to be intercepted by a man-in-the-middle attack, potentially exposing log messages if they are sent through the appender. We must upgrade to the latest version, which includes built-in verification to prevent this type of attack. (Apache, 2020).
3. YAML 1.1 Parser and Emitter for Java (snakeyaml-1.25.jar)  
   This dependency has a vulnerability, CVE-2017-18640, which allows entity expansion during operation. This can lead to excessive memory consumption, which could cause the system to crash. The recommended solution is to migrate to an updated version of SnakeYAML, which restricts the number of aliases for collections, preventing excessive memory use.
4. Spring Boot Starter for JSON Processing (spring-boot-starter-json-2.2.4.RELEASE.jar)  
   This dependency is affected by CVE-2020-7712, which allows attackers to inject commands via the parseLookup function. To fix this, we would restrict the supported syntax for bracketed parts of lookup strings, eliminating the need for eval and removing the possibility of command injection. This is a very easy fix and is important to maintaining application security.
5. Core Tomcat Implementation (tomcat-embed-core-9.0.30.jar)  
   This dependency has eight known vulnerabilities, many of which involve handling excessive requests, potentially leading to denial-of-service (DoS) attacks or causing the server to become unresponsive. Upgrading to a newer version of Tomcat can mitigate these risks. This is an essential fix, as high memory consumption could crash the system.  
   Another Tomcat vulnerability, CVE-2020-13935, affects WebSocket communication due to improper payload length validation. This can also lead to DoS attacks by triggering infinite loops. It is recommended to implement proper validation within the WebSocket framework to prevent such attacks.

**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.

Steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application would be to tackle the issue of data access username and password by creating a strong combination of alphanumeric characters for both the username and password. This would address the risk of a brute force attack on the system. I would also work on preventing the exploitation of vulnerability CVE-2020-9488 by updating the Apache Server to the newest version. Having the most updated version of any software may include some security updates that can be important in securing the system.

Then, I would go to the code review and modification and work on secure coding practices. With quality code, proper authentication and error catching will be easier to conduct. Doing so will help mitigate authentication vulnerabilities. Next, I would tackle the issues of certificate validation. The code should allow for proper validation and verification of all of the digital certificates for the app and server. Doing so will help prevent the exploitation of vulnerability CVE-2020-9488 and CVE-2020-13935.

Lastly, I would address the issue of TLS Certificate. Implementing a mutual check on both the client and server-side can be done through pinned certificates. This will help tackle the potential vulnerabilities found in the Bouncy Castle dependency and prevent the compromising of client API requests.

**6. Citations**

General Data Protection Regulation (GDPR). (n.d.). *GDPR Info*. Retrieved March 23, 2025, from [https://gdpr-info.eu](https://gdpr-info.eu/)

**Maven Repository.** (n.d.). *org.bouncycastle*. Retrieved March 25, 2025, from <https://mvnrepository.com/artifact/org.bouncycastle>