# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[11/17/2024]** | **[Noah Khomer]** | **Project 1 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

Noah Khomer

**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 is a consulting company which is focusing on individualized financial plans. The company is at a stage where they are trying to modernize their operations so they can meet the new challenges that come in evolving technological and security world. The key take away is that financial data has become one of the most sensitive assets in the industry and many hackers try to target this type of data for money purposes. At this point, software security has become critical so that the client information can be protected. It is important to remember that while client information is protected, another reason to follow these guidelines is because it helps a company keep up with industry regulatory practices as well as maintaining client trust. Think about it would you consider focusing on a financial company that had a reputation in security software challenges?

This vulnerability assessment is aimed to focus on the security aspect and posture of Artemis Financial web-based software application. By constantly focusing and addressing potential threats and vulnerabilities. This report will focus on providing a deep analysis of security areas which are critical for Artemis Financial to continue their operations while also focusing on actionable for the company to the point they can mitigate the proposed and identified risks. By the end of this document, Artemis Financial will be equipped better so that they can focus on modern operations while keeping their focus on secure financial services.

One of the most valuable things for Artemis Financial is secure communication, the reason for this is simple. Their business is sorted in a way that it involves a heavy moving of data which contains highly sensitive information about their customers such as data, savings, investments, their retirement plans, social security numbers, insurance policies and even more. Therefore, the company must place a strict emphasis on ethics such as confidentiality, integrity and client trust. The reason for this is they need to protect the client data to never have unauthorized access during any data transmission. Furthermore, they need to maintain integrity so that tampering of the financial records or information never happens. Lastly, secure communications provide a means of reassurance. Currently, reassurance is needed to run a company, it is even better if you have reassurance in the form of a real-world practice. For example, the company must focus on implementing HTTPS for every single client-server communication. This way the data is secure whether it is parsing through locations, logging credentials, printing financial details. In a manner, strict secure communication will make data encrypted and protect it from any hacker’s eyesight.

Based on all the information provided about Artemis Financial, I cannot tell whether the company handles international transactions or not. But what I can do is make logical inferences. It is reasonable for a financial company to be involved in international transactions and the reason for this is simple. Money is the language of the world, and it comes in many forms, currency and coins. It is reasonable to assume that Artemis Financial has international dealings at some point or maybe in likely foreseeable future. Even if Artemis Financial currently only operates domestically, it is important for the software application to be fully designed with international transactions in the mind. The software, therefore, must comply with data protection compliance requirements such as GDPR and CCPA. DGPR is the general data protection regulation center for clients who are based in Europe and its Union. Meanwhile California Consumer Privacy Act is based for all the United States based clients. Since it is apparent that just based on two of these data compliances Artemis Financial must be handling international transactions the next order of business is to implement something known as cross-border encryption. Each time data crosses borders there are a lot more liabilities and things in the way. Therefore, it is important to have encryption for this type of data to ensure confidentiality and avoid any type of interception from attackers or enemy nations. At the end of the day, all of this assumes that money indeed can be converted from one currency to another and can even have multilingual capabilities which causes Artemis Financial to work on international levels.

Yes, absolutely. Artemis Financial must comply with several governmental and industry specific restrictions for secure communication to protect any sensitive financial data so that they can maintain their compliance. I have already discussed how GDPR and CCPA are required for secure communications especially across borders. I will now talk in depth about the remaining regulatory compliances such as PCI DSS, EAR, SOX. One of the first governmental compliance for Artemis Financial would-be Payment card Industry Data Security Standard. This is one of the most standard govern which is held and overseen anytime there is handling of payment with card data. This government law requires that all the sensitive cardholder information must be encrypted when it is being sent over public networks. The second thing this law focuses on is to make sure there are access controls meaning only authorized individuals can access this type of payment system. Lastly it requires that all organizations must regularly assess the vulnerabilities of their system. The impact this law has on Artemis Financial is that they must focus on implementing a strong encryption standard such as TLS 1.3. Furthermore, the company must focus on regular vulnerability scans to understand the potential issues. The second regulation that can have a huge impact on Artemis Financial is known as Export Controls on Cryptography. While cryptography seems like a fancy topic there are restrictions on the export of cryptographic technologies. This government basically aims to prevent any misuse of encryption tool by any hostile nations and its partners. It can have a strong impact on Artemis Financial because they use strong encryption tools such as AES-256 and at the same time avoid using any of the banned algorithms for being weak in the world of cryptography such as MD5 or even SHA-1. The last law that can have a huge impact on the company Is known as Sarbanes Oxley Act. The reason for this is because this government focuses on having strict requirements for any financial recordkeeping and their data security. So, to align their standards SOX standards, the company should focus on implementing secure communication protocols such as the HTTPS so that they can protect their financial records.

Artemis Financial is a financial company so due to the nature of the business it is always going to have external threats that are present right now and even in the immediate future. Threats such as phishing attacks that are heavily focused on emails of the client and banking partners are always a current and future threat. More specific threats such as API exploitations and Ransomware are also a type of threat that can impact users and customers now and even in future because it can cause heavy downtime and loss of data. One of the most common threats such as SQL injection attacks are also highly relevant as seen in milestone 1 and 2 there can e input fields that do not properly sanitize any type of user input so hackers can easily put malicious commands to the database records.

There is a lot of potential and role to be considered when it comes to open-source libraries in the world of software security. Artemis Financial must fully understand the flaws and benefits of using open-source libraries. One of the first security risks is in the open-source components. There are so many libraries that are known to have vulnerabilities that can be exploited if they are not patched and updated to the newest version. For example, as discussed earlier outdated cryptographic libraries and web frameworks can allow hackers to bypass many types of authentications and lead to compromise of data. Therefore, one of the main things is to make use of tools such as the OWASP Dependency-Check which can help in regularly scan all the libraries for vulnerabilities. The benefit of this approach is that you can see the risk severity in builds. Usually, one of the common solutions is to upgrade to the latest stable versions which have addressed previous security flaws. Aside from the common solution, another solution is to rely on dependency management systems such as implementation of Maven or other tools which can help keep track of library and their versions. As for the evolving web application technologies right now, one of the most modern practices consists of RESTful API, Cloud integration and PWA. Restful API makes it possible to have Oauth 2.0 which secures all the API endpoints. At the same time cloud integration is a good solution for huge companies, especially in the world of banking. By storing everything on a cloud platform with huge margins for scalability and cost efficiencies that can help in putting more expense towards security challenges. Lastly PWA also known as progressive web applications are offering a modern user experience with features such as offline access to data, push notifications and react based fast loading times all that scale a business towards using their costs to invest in security aspects. In my opinion, one of the main things Artemis Financial should focus on while maintaining their modernization efforts is to comply with PCI DSS, ISO 27001 and lastly GDPR AND CCPA.

**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.

1. Input Validation

Input validation applies to Artemis Financials’ software application because it prevents any malicious data from being entered and compromising the application. The system is going to process a lot of input data which is user provided such as customer details, account numbers, their login credentials and if any of this data is unvalidated it can lead to injection attacks and even data manipulation. Therefore, implementing strict validation rules that can ensure that the data is acting as expected formation is a way of safeguarding the system from any types of exploits such as SQL injection and even XSS.

1. APIs

API also applies to Artemis Financials software design application because API enable the software to exchange data between different types of systems. Therefore, API can be a huge exploit if they are left unsecured and cause a lot of data to be compromised. There are many ways to secure API such as Oauth 2.0, rate limiting where the user cannot run scripts and have a time-based refresh period usually every 1.5 second etc. and even penetration testing all which is essential to prevent any unauthorized access by hackers. All the mentioned techniques employed can stop API abuse because the company handles sensitive financial data having API applies to the financial company.

1. Cryptography

Cryptography also applies to the company because as discussed earlier the financial data whether at rest or in transit must be encrypted per so many different governing bodies across the USA and Europe to keep up with the laws. Therefore, Artemis Financial must encrypt client information such as their account details, transaction records, pending transactions, future subscriptions bill etc. so that no one can have unauthorized access to this data or information. Practices such as using strong encryption standards like AES-256 and TLS 1.3 make the company follow regulatory requirements as the law requires that MD% and SHA-1 must be avoided.

1. Client/Server Security

Client server security also applies to the financial company because the communication between clients and servers needs to protect so that hackers or attackers are unable to intercept any of the sensitive data during the transmission as discussed in introduction answers. The Artemis Financial software must focus on enforcing HTTPs standards to secure all types of data from login credentials to transaction details. Furthermore, implementation of TLS 1.3 will focus on having the data encrypted whether it is at standpoint or moving from one place to another so that it is inaccessible by any hacker.

1. Code Error Handling

Code error handling applies to the financial company because it is necessary to prevent any type of exposure of sensitive system details that can leave some sort of trace behind and even give insights on how things work for the company. For example, sometimes when you run into an error you see debugging information which can give hints on how the program is working. This type of approach is unacceptable for any banking industry because the customer data is money and financial based. Therefore, Artemis Financial software must focus on implementing proper error logging and giving users facing error messages. What this means is displaying errors as 404 errors or even mentioning specific strings from the code such as “We are sorry, you have encountered an error, no further information disclosed about the error”. In this manner, the detailed logs are only accessible to the developers where the company can protect itself from any hackers looking to gain access to systems working.

1. Encapsulation

Encapsulation also applies to financial companies because it provides insights into the internal data structure. These structures can be anything such as account balances and even customer details reflecting their transactions and account money history etc. All of these can be protected using encapsulation because it utilizes proportions of controlled access methods, for example the usage of getters and setters. This way Artemis Financial can safeguard any of the critical financial data from any modification which is unintended. The approach makes sure that the system integrity is in place and reduces security risks.

1. Code Quality

I don’t think code quality applies directly even though it is very essential it is apparent that some apply and some don’t according the assignment instructions. For a banking financial industry to focus on code quality is important but even more than that is important to focus on relevant issues and issues with a risk severity of high or critical.

**3. Manual Review**

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

A screen shot of a computer code

Description automatically generated

One of the first manual findings in my review was that in the file of CrudController.java I noticed that the code for line 10-17 had issues with how the business \_name parameter is accepted without any validation. What this means is that it is leaving the endpoint more vulnerable to any injection attacks in case a hacker submits any harmful input.

A screen shot of a computer

Description automatically generated

In lines 26-28, for the file DocData.java is another issue where the hardcoding of the database credentials for the root, root is exposing sensitive information if the code ends up getting leaked. Therefore, the risk of having unauthorized database access later is inevitable.

A computer code with colorful text

Description automatically generated

Another issue I found was in the greetingcontroller.java file where for lines 14-20 there is an issue in how the name parameter is being utilized. For example, it is being directly used without any form of sanitization which makes it vulnerable to injection attacks from hackers. Furthermore, aside from injection attacks, hackers can also manipulate the values of the data because of the lack of sanitization.

A screen shot of a computer program

Description automatically generated

Another issue I found is that in application.properties file where HTTPS settings are missing. The main thing about this file is there is no mention of server.ssl.enable. So if the SSL is configurated to default behavior without that statement it means it is not set to true. Spring boot automatically chooses HTTP instead of the HTTPS. This means the server will only try to accept unencrypted HTTP connections which is a huge software security issue.

A computer screen with text and symbols

Description automatically generated

In lines 25-30 for the DocData.java, there is an issue where the printing of any stack traces is going to expose sensitive implementations details. These details can be anything such as the database structure or the connection values which can be bad thing for a hacker to gain access into.

A black background with white text

Description automatically generated

There is another issue in the customer.java file where the sensitive data is being stored insecurely. For example, in line 5-10 there is the word account balance being stored in simple plaintext which can make the situation very vulnerable if the database is compromised and taken over by hackers.

A black background with white text

Description automatically generated

There is another issue in the same statement for the same customer.java file where the account balance field is public which can lead to unauthorized modification. For the sake of encapsulation, it is very important to make sure that the private fields are used by using methods like getter and setter.

A screen shot of a computer code

Description automatically generated

Another issue can be seen in how the API authentication is not secured for the Crudcontroller.java file. The /read API endpoint has several issues where it lacks the usage of authentication. This way it allows any user to have access to sensitive information. Authorization check is required here to make sure role-based access is in place for example “a user that has role to access”.

**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

**Dependency name**: bcprov-jdk15on-1.46  
**Vulnerability ID**: cpe:2.3:a:bouncycastle:bouncy-castle  
**Issue mentioned** dependency report: The issue for this report is that there is insecure cryptography practices. The reason for this is because it is outdated so multiple known vulnerabilities exist.   
**Recommended fix** in the report is to update the bouncy castle cryptographic library to at least a version 1.70

**Dependency name:** hibernate-validator-6.0.18.Final.jar **Vulnerability ID:** cpe:2.3:a:redhat:hibernate\_validator:6.0 **Issue mentioned:** Input validation issue where the vulnerability can have potential unauthorized access. **Recommended fix:** Upgrading the hibernate validator library to 6.2.x

Dependency name: jackson-core-2.10.2.jar

Vulnerability ID: cpe:2.3:a:fasterxml:jackson-core

Issue mentioned: Remote code execution can happen because the vulnerability is letting deserialization for the untrusted data

Recommended fix: Upgrading the library of the Jackson core to version 2.13.x or more

Dependency name: jackson-databind-2.10.2.jar

Vulnerability ID: cpe:2.3:a:fasterxml:jackson-databind

Issue mentioned: Malicious payload injection can happen because of the deserialization that can cause a lot of security exploits.

Recommended fix: Upgrading the Jackson data bind library to 2.13.x or higher.

Dependency name: log4j-api-2.12.1.jar

Vulnerability ID: cpe:2.3:a:apache:log4j

Issue mentioned: Outdated version where a lot of DOS can happen (denial of service). One of the common issues with this type of problem is that it can cause a lot of potential information exposure.

Recommended fix: Upgrading the Apache Log4j library to a version of at least 2.17.1

Dependency name: logback-core-1.2.3.jar

Vulnerability ID: cpe:2.3:a:qos:logback

Issue mentioned: A huge vulnerability in the logging configuration that can also have DoS

Recommended fix: Upgrading the logback library to version 1.2.10 or newer

Dependency name: spring-boot-2.2.4.RELEASE.jar

Vulnerability ID: cpe:2.3:a:vmware:spring\_boot

Issue mentioned: There are multiple issues for this one where the framework is highly impacted because of potential privilege escalation.

Recommended fix: Upgrading Spring Boot framework to at least 2.6.15 or newer will resolve.

Dependency name: tomcat-embed-core-9.0.30.jar

Vulnerability ID: cpe:2.3:a:apache:tomcat

Issue mentioned: There is improper handling when it comes to the HTTP requests not being secure therefore information can be leaked or target for DOS attacks.

Recommended fix: Upgrading the tomcat embedded core library to 9.0.50 or newer is critical for these purposes.

**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 make sure the mitigation plan focuses on both solutions for the manual review and the static testing report. It is important to make the mitigation plan in a way that focuses on both steps for both parts.

1. Upgrading list for the vulnerable dependency.

The list for the vulnerable dependency and their fixes to exact build versions are listed

* Upgrade bcprov-jdk15on from version 1.46 to 1.70
* Update hibernate-validator from version 6.0.18 to 6.2.x
* Update jackson-core and jackson-databind from 2.10.2 to 2.13.x
* Upgrade log4j-api from 2.12.1 to 2.17.1
* Update logback-core from version 1.2.3 to 1.2.10
* Upgrade spring-boot from version 2.2.4 to 2.6.15
* Update tomcat-embed-core from version 9.0.30 to 9.0.50

1. Coding Review

After reviewing all the codes mentioned above in the manual review the solutions are proposed as such.

* For the problem in CrudController.Java with the business name being a parameter that is accepted without validation. It is important to mitigate the risks there by establishing what the input should match. For example, including a code like “input must match a-zA-Z-0-9” will validate and sanitize all user inputs in a way that only the safe characters are accepted
* The second issue found in the manual review was for the file DocData.java where database credentials were hardcoded. This is a very bad practice because it can show sensitive information if the code is leaked. Therefore, one of the obvious fixes is to include ENV files. The ENV files are like an environment variable file where sensitive credentials can be stored so that when the file handling for such a procedure happens it tries to get the credentials from this file. For example, “System.getenv(nameofenvfile” is a lot more secure practice than just listing credentials in an open codebase.
* Another issue we found was in GreetingController.java. For this section, the API security had unsecured endpoints. The reason for that was because for the name parameter there was no sanitation, and it was making it more vulnerable to any type of injection attacks and even data manipulation. Therefore, one of the common ways. To fix this, it's the same as problem one where we try to include what type of input should be coming in. For example, characters like A-Z, capital A-Z and 0 to 9.
* One of the biggest problems I found in my manual review was the application dot properties file where there was number HTTP enforcement. The whole thing was just missing which means that by default there will be going to a HTTP. Because of this reason not having a secure version of it. Allow sensitive data like login credential to go over insecure channels. Therefore, it is important to include and enable them true for the file in application properties, so default version is not used.
* As discussed in some of the earlier milestones, sometimes whenever there's an error, it's going to print stack traces which can show implementational details. These details can be anything like the database structure or even the connection strings, and hackers can use this. Type of information for their own advantages. Therefore, the print stack trace when logging in should print a statement saying that database connection just failed to go through.
* Another issue that I found was in the file called customer dot Java and this file integer for the account balance was being stored in just plain text and this is bad especially if the database is compromised because it's going to show exactly what everyone's account balance is. Therefore, one of the simpler ways to fix this is to encrypt any type of sensitive data before storing it, and then decrypt if only necessary.
* For the same file and the same integer variable, there is another problem where the whole field is just public. This can lead to any type of unauthorized modification. Therefore, one of the mitigation strategies for this is to make sure that the field is private. So, using validated sources like getter and setter methods are a lot better than leaving them public.
* The last problem I found in my manual review was for the file called CRUD controller dot Java and this file the API has an endpoint where there is no authentication being required, which is very bad because when someone wants to access. File there should be some sort of authorization check. Therefore, the mitigation strategy for this is to make sure that the code is somehow verifying that the user who is trying to access the file has already the role where it is needed to be given.