# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09.18.2024** | **Remy Welham** |  |

## 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

Remy Welham

**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?

The client Artemis Financial is looking to create a web-based software application that can provide and develop individualized financial plans for its customers. This would include things such as savings, retirement, investments, and insurance. Any and all finances are extremely important to keep secure and safe so things such as bank information and account information doesn’t become accessible to those who would do harm with it. A company that works with these finances also needs to be and remain trustworthy or they will lose customers and the ability to function as a business.

The company doesn’t specify any international transactions or governmental restrictions on communications. However, this doesn’t mean there can’t be future security to consider for these categories. An area to keep in mind for international transactions would be if they get customers internationally or if someone who is a customer needs to access their information and accounts from abroad. These transactions need to be secure. For the governmental restrictions, there is any financial planning that goes into social security or medicare for insurance. Both of these would need to be protected to the governmental standard.

There are a few external threats that could be present now and in the immediate future. The biggest one would be a hacker getting access to any database that holds a client’s banking, insurance, retirement, and investment information. A leak of this information would be very bad. Another threat would be phishing emails or fake emails meant to trick people into giving up login information for the application. There’s also the chance of logins and certain pages being improperly created that would create vulnerabilities. For modernization requirements, source libraries are constantly being updated and the standards for different programming languages can also change. There must be updates that happen regularly so long as the application is in use. New threats and attacks are documented frequently. There’s also the act of being careful that these updates don’t break anything vital to the application.

**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.

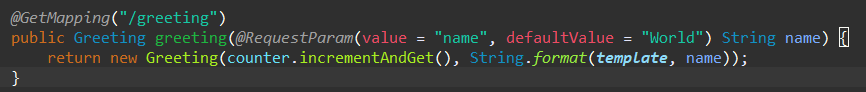
* **Input Validation:** For a software application that needs financial information to keep secure and have access to, there are going to be accounts and input coming through the application frequently. Without satisfactory input validation, the application is vulnerable to injections and service denial attacks.
* **APIs:** When you’re working with finances online, a lot of information and manipulation of the information will be going through different APIs. If insurance is being monitored and changed, there’s the API the Artemis Financial uses for their application and then there’s the API the insurance uses. Making sure these interact securely is vital to keeping data safe.
* **Client/Server:** When you have an application that has clients who need to be able to access and review things, in this case finances and financial advice, these interactions need to be secure. It also needs to be securely distributed across the application to keep any vulnerabilities safe. The application is only as secure as its least secure point.
* **Code Error:** Any application that has human interaction and untrusted data coming in needs to be ready to handle Errors that occur. This can come in the form of trying to convert a String of text to and Integer and making sure if it’s a String that cannot be converted, the Error is handled properly. If it’s not, this can cause many problems.
* **Encapsulation:** With any application that is handling multiple sources of data that need to be accessed by different people with different levels of access, the data needs to be structured in a secure way to ensure that those who don’t need to manipulate or touch pieces of a program literally cannot access them.

**3. Manual Review**

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

* One vulnerability is that the program doesn’t use the most up to date version of the Spring Framework in the pom.xml file as shown below.   
  A screen shot of a computer screen

  Description automatically generated
* The program also doesn’t use the most up to date version of Java which can introduce vulnerabilities.  
   A computer code with white text

  Description automatically generated
* In GreetingController, there’s no check for making sure that input is valid when given for the name given.   
  
* In the pom.xml file there is no Apache validator.
* In customer.java there is a method for showing a customer’s information that isn’t securely protected.   
  A screenshot of a computer program

  Description automatically generated
* In the DocData.java file there is a Connection con that isn’t used within the method.   
  A screen shot of a computer screen

  Description automatically generated
* In the CRUDController there is a call for doc.toString() without any error catching.   
  A computer code with colorful text

  Description automatically generated

**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 | Vulnerability IDs | Description | Solution |
| bcprov-jdk15on-1.46.jar | cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:bouncy\_castle\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*: | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7. | Update to most recent version. |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* | Hibernate's Bean Validation (JSR-380) reference implementation. | Update to most recent version. |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* | General data-binding functionality for Jackson: works on core streaming API | Update to most recent version. |
| log4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* | The Apache Log4j API | Update to most recent version. |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* | logback-core module | Update to most recent version. |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* | YAML 1.1 parser and emitter for Java | Treat untrusted data files with caution and do not use the parser on user supplied input. |
| spring-boot-starter-web-2.2.4.RELEASE.jar spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\* | Starter for building web, including RESTful, applications using Spring  MVC. Uses Tomcat as the default embedded container | Update to most recent version. |
| spring-webmvc-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\* | Spring Web MVC | Update to most recent version. |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | Core Tomcat implementation | Update to most recent version and do proper input validation. |

**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.

Most of the vulnerabilities can be fixed by either updating to the most recent version of the various dependencies and systems used within the program and by making sure there is proper error handling and input validation. Being able to keep things up to date helps keep known vulnerabilities from being exploited. Any data coming into the program from an outside source should be treated as untrusted data that need validation. Finally, anything that can cause an error to pop needs to be handled properly or it can create a number of issues.