# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/15/2024** | **Daniel Gorelkin** |  |

## Client



## Developer

Daniel Gorelkin

1. **Interpreting Client Needs**

Artemis Financial is a financial institution that offers its service to multiple clients over the web, inside and outside the country's boundaries. Therefore, it must outstand the highest security standards to safeguard its reputation and the customer data it stores and manipulates. One of the internal considerations that we should overlook is client trust and reputation risks. A mishandled data leak or a breach of sensitive financial information can harm future business prospects and potentially lead to legal prosecution from clients, ultimately damaging the company's public image. Because Artemis Financial will provide global service and deal with cross-border data transfer, observing regulatory and international privacy laws should be the company's first priority. Manipulating customer data between countries can expose the company to violations of local and foreign laws such as GDPR, the General Data Protection Regulation in Europe (Wolford, 2024), and the Financial Industry Regulatory Authority FINRA (*About FINRA*, n.d.) within the states. Therefore, our software must comply with how personal data is collected, stored, and processed. In addition, providing services to government agencies worldwide could be a tempting target for attackers. Therefore, it must implement vital encryption standards such as PCI DSS for handling credit card transactions (PCI Security Standards Council, 2024) and support SSL/TLS protocols for all communications involving sensitive financial data; hence, a data leak could cause an international conflict.

Besides internal errors and software bugs, some external threats might be present. Despite Global Rain's ability to address most, if not all, of the modern threats such as phishing, SQL injections, permission escalation, authentication and authorization, downtime and service disruptions, input validation and sanitation, the use of modern and up-to-date frameworks to prevent future vulnerabilities from dependencies through its API, and the use of updated libraries is a long term concern. Therefore, we must regularly update libraries and security protocols to comply with the newest versions and standards and prevent possible future threats. However, open-source code libraries are public, making them prone to vulnerabilities and breaches. For example, the OWASP foundation provides records of common vulnerabilities. It suggests using Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) Tools to discover these open-source threats (*Free for Open Source Application Security Tools | OWASP Foundation*, n.d.) and even manual penetration testing to breach the software while it still in the safe testing phase. In contrast, Maven, an open-source software project management and comprehension tool, aims to gather current principles for best practices development and make the code secure and less vulnerable (*Maven – Introduction*, n.d.) while comparing the existing vulnerabilities against the National Vulnerability Database (NVD) and highlighting the threat’s severity and it’s history.

**2. Areas of Security**

Some security areas referred to the vulnerability assessment process flow diagram apply to Artemis Financials' software application. For example:

* The Artemis Financial app will read user input, forms, and potentially uploaded files. Improperly sanitized and fetched input could leave the door open for potential breaches and malicious usage. The input must be monitored to prevent malicious expressions leading to remote code execution. Untrusted data must be scanned or even declined before accessing the system.
* The app will be connected to a database by a JDBC connectivity API and various remote servers, potentially inside and outside the country's boundaries. This means potential access for malicious usage and SQL injection exists if quarries are not filtered and passed securely to and from the database.
* Because the system will serve thousands of users simultaneously, each user must be uniquely granted privileges and have a secure session token to prevent privilege escalation. This means cryptography and data encryption in the form of password hashing, prevention of Man-in-the-Middle (MITM) attacks, sensitive information obscuring, and prevention of access for unauthorized users via multi-factor authentication mechanisms must be present in the app.
* Client/server communication must be confidential through a secure HTTPS connection to prevent cross-site request forgery (CSRF) and/or resubmitting and redirecting a completed transaction or gaining the client’s personal session tokens.
* Code quality and the app's error-handling mechanism are not less important to look into to prevent unwanted software behavior. The app should expect a regular user (not an attacker) to be uncooperative by submitting empty forms, prohibited characters, or even providing the system with a numerical "zero" input where such is not permitted. This could lead to a software crash if no error handler exists.
* For later maintenance purposes, such as updating security protocols and vulnerability updates, the software must comply with best coding practices and provide human-understandable code with detailed documentation.

**3. Manual Review**

Manually reviewing the code shows the following issues:

1. In the file GreetingController.java -> greeting method, the system reads user input, formats it, and passes it to the Greeting class constructor without validating the input. A screenshot of a computer program

   Description automatically generated
2. The same happens in the CRUDController.java file using the CRUD Method. – No Input sanitization.A computer screen shot of a code

   Description automatically generated
3. The public class customer.java, the public method showInfo(), returns customer information to whom it may concern without any encapsulation or authorization needed. The customer’s information is exposed.
4. In the same class, the public void deposit(int a) method could lead to a buffer overflow or incorrect operation handling if adverse or unexpected input passes. No error-handling mechanism exists, nor is there a negative deposit value input check present.
5. The myDateTime class does not take into account the various possible user time zone entries and must validate passed user or system input.
6. The retrieveDateTime() method could lead to a buffer overflow by exceeding the int[] buffer size.

A screen shot of a computer program

Description automatically generated

1. In the DocData.java class, the read\_document() method does not validate the Strings key and value, and there are no signs of proper input sanitization or prepared statements, which could lead to a SQL Injection breach.
2. In the same class and method, database details such as: . "jdbc:mysql://localhost:3306/test","root","root"); should not be hard coded to the code but instead read from the system environment variables. A computer screen shot of a computer code

   Description automatically generated
3. In the same class and method, the user authentication and authorization mechanism is missing, which can allow attackers to access sensitive data.

**4. Static Testing**

Running a dependency check on Artemis Financial’s software application reveals several existing vulnerabilities in the project where nine of which are marked with a critical severity status:

A close-up of a number

Description automatically generated

The table below shows the names of the critical and high vulnerabilities, their descriptions and solutions, and their documentation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Solution** | **Documentation** |
| Apache Tomcat Improper Privilege Management Vulnerability  In: tomcat-embed-websocket-9.0.30.jar  And in: tomcat-embed-core-9.0.30.jar | A mechanism that allowed returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application then this, along with the ability to process a file as a JSP, made remote code execution possible. | Disable the AJP Connector if not required or  Upgrade To Version->9.0.31. | **CVE Dictionary Entry:** [CVE-2020-1938](https://cve.org/CVERecord?id=CVE-2020-1938) **NVD Published Date:** 02/24/2020 **NVD Last Modified:** 07/24/2024 **Source:** Apache Software Foundation. |
| Spring Framework JDK 9+ Remote Code Execution Vulnerability.  In: spring-webmvc-5.2.3.RELEASE.jar  In: spring-expression-5.2.3.RELEASE.jar  In: spring-core-5.2.3.RELEASE.jar | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. | Deploy as a Spring Boot executable jar or  Upgrade To Version->6.1.13. | **CVE Dictionary Entry:** [CVE-2022-22965](https://cve.org/CVERecord?id=CVE-2022-22965) **NVD Published Date:** 04/01/2022 **NVD Last Modified:** 02/08/2023 **Source:** VMware |
| Spring Framework JDK 9+ Remote Code Execution Vulnerability.  In: spring-web-5.2.3.RELEASE.jar | Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. | Implement authentication and validate untrusted data input.  Upgrade To Version->6.1.13. | **CVE Dictionary Entry:** [CVE-2016-1000027](https://cve.org/CVERecord?id=CVE-2016-1000027) **NVD Published Date:** 01/02/2020 **NVD Last Modified:** 04/20/2023 **Source:** MITRE |
| NVD-CWE-noinfo (Insufficient Information)  In: spring-boot-2.2.4.RELEASE.jar | An application that is deployed to Cloud Foundry could be susceptible to a security bypass. | Upgrade To Version->3.3.3 | **CVE Dictionary Entry:** [CVE-2023-20873](https://cve.org/CVERecord?id=CVE-2023-20873) **NVD Published Date:** 04/20/2023 **NVD Last Modified:** 08/28/2023 **Source:** VMware |
| Deserialization of Untrusted Data and Improper Input Validation. In: snakeyaml-1.25.jar | SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. | Use SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization.  Upgrade To Version->2.3 | **CVE Dictionary Entry:** [CVE-2022-1471](https://cve.org/CVERecord?id=CVE-2022-1471) **NVD Published Date:** 12/01/2022 **NVD Last Modified:** 06/21/2024 **Source:** Google Inc. |
| Bouncy Castle JCE - Improper Verification of Cryptographic Signature.  In: bcprov-jdk15on-1.46.jar | The DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure. | Upgrade To Bouncy Castle Cryptography APIs for JDK 1.8.  Upgrade To Version->1.78.1 | **CVE Dictionary Entry:** [CVE-2016-1000338](https://cve.org/CVERecord?id=CVE-2016-1000338) **NVD Published Date:** 06/01/2018 **NVD Last Modified:** 08/29/2024 **Source:** MITRE |
| logback-core module  In: logback-core-1.2.3.jar | A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data. | Upgrade To Version->1.5.8 | **CVE Dictionary Entry:** [CVE-2023-6378](https://cve.org/CVERecord?id=CVE-2023-6378) **NVD Published Date:** 11/29/2023 **NVD Last Modified:** 12/05/2023 **Source:** Switzerland Government Common Vulnerability Program |
| hibernate-validator.  In: hibernate-validator-6.0.18.Final.jar | A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages. | Upgrade To Version-> [8.0.1.Final](https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator/8.0.1.Final) | **CVE Dictionary Entry:** [CVE-2020-10693](https://cve.org/CVERecord?id=CVE-2020-10693) **NVD Published Date:** 05/06/2020 **NVD Last Modified:** 11/06/2023 **Source:** Red Hat, Inc. |

**5. Mitigation Plan – Recommended actions**

Manual review mitigation:

1. In the file GreetingController.java -> greeting() input validation mechanism should be implemented. For example, the [**Apache Commons Text**](https://mvnrepository.com/artifact/org.apache.commons/commons-text) plugin library could be imported, and a string sanitization method could be implemented to validate safe string input.

A screenshot of a computer screen

Description automatically generated

1. Implement input validation in the CRUDController.java file using the CRUD Method.
2. In the public class customer.java, the public showInfo() method must verify user authentication and authorization by a security mechanism to prevent the exposure of sensitive information to an illegal user. In addition, the account\_balance attribute should be modified to be private.
3. The customer.java, public void deposit(int a) method, must validate that an integer positive deposit amount was passed to prevent incorrect action and errors. In addition, an error handler should be added to verify that a valid int parameter was passed to the method.

A screenshot of a computer

Description automatically generated

1. The myDateTime class, setMyDateTime() method must use a standardized method like GMT to store correct user location time zone entries and must validate user or system input to be in the correct format. In addition, an error handler should be added to the method.
2. In the myDateTime class, the retrieveDateTime() method must verify that the return requested does not create a buffer overflow by verifying the size of the int[] array before accessing the stored data.
3. In the DocData.java class, read\_document() method implement input validation mechanism such as StringEscapeUtils.*escapeJava()* from the org.apache.commons.text library or add a local custom class and use existing Java string methods, such as isLetter/isDigit, etc., to filter and overwrite the input arguments. In addition, the PreparedStatment class methods from the java.sql library should be implemented to prevent SQL injection.

A screenshot of a computer program

Description automatically generated

1. In the DocData.java class, read\_document() method, hide the exposed database and the user’s name and password and load them from the system environment to prevent attackers from gaining access to the database or the source code.

A screen shot of a computer program

Description automatically generated

1. In the DocData.java class, the read\_document() method must Implement an authentication and authorization mechanism to deny unauthorized users access to the data.
2. Update Maven libraries to the most up-to-date ones:

* tomcat-embed-websocket-9.0.30.jar -> Upgrade To Version-> 9.0.31.
* tomcat-embed-core-9.0.30.jar -> Upgrade To Version-> 9.0.31.
* spring-webmvc-5.2.3.RELEASE.jar -> Upgrade To Version-> 6.1.13.
* spring-expression-5.2.3.RELEASE.jar -> Upgrade To Version-> 6.1.13.
* spring-core-5.2.3.RELEASE.jar -> Upgrade To Version-> 6.1.13.
* spring-web-5.2.3.RELEASE.jar -> Upgrade To Version-> 6.1.13.
* spring-boot-2.2.4.RELEASE.jar -> Upgrade To Version -> 3.3.3.
* snakeyaml-1.25.jar -> Upgrade To Version-> 2.3.
* bcprov-jdk15on-1.46.jar -> Upgrade To Version-> 1.78.1.
* logback-core-1.2.3.jar -> Upgrade To Version-> 1.58.
* hibernate-validator-6.0.18.Final.jar -> Upgrade To Version-> 8.0.1.Final.

**References:**

*About FINRA*. (n.d.). <https://www.finra.org/about>

*Free for Open Source Application Security Tools | OWASP Foundation*. (n.d.). [https://owasp.org/www-community/Free\_for\_Open\_Source\_Application\_Security\_Tools#](https://owasp.org/www-community/Free_for_Open_Source_Application_Security_Tools)

Manico, J., & Detlefsen, A. (n.d.). *Iron-Clad Java*. O’Reilly Online Learning. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch03.html#ch03lev2sec4>

*Maven – Introduction*. (n.d.). <https://maven.apache.org/what-is-maven.html>

PCI Security Standards Council. (2024, July 12). *PCI Security Standards Council – Protect Payment Data with Industry-driven Security Standards, Training, and Programs*. <https://www.pcisecuritystandards.org/standards/>

Wolford, B. (2024, August 29). *What is GDPR, the EU's new data protection law?* GDPR.eu. <https://gdpr.eu/what-is-gdpr/>