# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09/19/2025** | **Michael Peck** | **Initial Assessment** |

## Client



## Developer

Michael Peck

**1. Interpreting Client Needs**

Because Artemis Financial focuses primarily on finance and insurance, security should be of the utmost importance across all other concerns. Customer information should be as safe and protected as if it were health information with HIPAA. When it comes to international transactions, while current documentation does not mention the possibility of international transactions, it should be assumed that they could eventually support it and thus are subject to international regulations. Financial institutions must follow strict governmental regulations, such as the Federal Reserve System (The Fed), the Security and Exchange Commission (SEC), the Consumer Financial Protection Bureau (CFPB), the Federal Deposit Insurance Corporation (FDIC), the Office of the Comptroller of the Currency (OCC), and the Financial Industry Regulatory Authority (FINRA) (StoneX, 2024). Financial institutions are one of the highest attacked industries out there and are susceptible to all sorts of external threats such as phishing and social engineering, insider attacks, data breaches, such as malware or ransomware, spoofing, distributed denial-of-service (DDoS) attacks or even possibly regulatory compliances such as the Sarbanes-Oxley Act, The Gramm-Leach-Bliley Act, or the PCI DDS (Camarda, 2025). Developers using open-source libraries should be very cautious about each library being used, as some open-source libraries are not as reliable as others and each come with their own vulnerabilities or even malware embedded in them. Depending on the size of the team and the application, updates may be required often, for bug fixes, security patches and any automation tools that may be required.

**2. Areas of Security**

Starting from the first point of the Vulnerability Assessment Process Flow diagram, input validation is a must, as most of the web applications will be reliant on input data, such as account information, monetary input values, and any non-valid data could impact the performance of the API. Secure API interactions is also a must, as security will be one of the driving factors given the regulations acting upon the application and the company. Cryptography follows suit with encryption algorithms to protect the customer’s investments and funds. The Client/Server model serves to only provide the client with appropriate information based on the client’s request; to ensure that only the information given is what is requested will help to protect the client’s information from being leaked or stolen. Secure Code Practices can help to prevent attacks by such actions as having the API avoid dynamic SQL for SQL injection attacks and using only pre-defined SQL sequences as needed (Oracle, 2019). Encapsulation is also going to be important to protect data as it’s being processed or accessed within the application.

**3. Manual Review**

Input Validation: In the CRUD class, the variables content and content2 (the variables should also be renamed more appropriately to self-identify their purpose) are not properly validated when they are assigned or even used. Properly validating them can prevent against SQL injections if used as they are, or they should be encrypted via HTTPS.

Secure Coding Practices: In the CRUDController, before the /read annotation, there is no authentication processed to prevent unauthorized users from reading any of the data that is processed.

Input Validation: In the DocData class, during the construction phase, the input given to the constructor method is not validated and checked beforehand and thus could create an issue where an attacker could execute a system command on the server (or directory access), including deleting the entire file system.

Encapsulation: In the customer class, the field account\_balance has a default visibility of public and not private, meaning that this field can be changed outside of the mutator method for changing this value from any class.

Input Validation: Also in the customer class, when accessing the deposit method, there is no validation that the input parameter for this function to ensure that the input is indeed an integer. Instead, it immediately adds the input parameter to the int value already established. Also, the account\_balance should be a double or float value, as integers would disallow cents to be displayed, only whole dollar amounts.

Encapsulation: In the myDateTime class, while there are no critical flaws in the system, poor encapsulation methods exist for the variables mySecond, myMinute, and myHour. All three variables should exist as private integers and should only be accessed by the accessor method. Leaving all three variables public by default, could allow attackers to change the date and time on the server, possibly to create a false timestamp of attacks on customers to create a false trail.

Input Validation: In the myDateTime class as well, there are no methods to ensure that the data being processed is data that is properly represented. Even if a user enters a number that is too long, it will still be misrepresented as values cannot exceed 59 for both seconds and minutes, and 23 for the hour (or 12 for 12-hour clocks). If a user entered a value higher than these, the application would display the new time, without validating these to be correct.

**4. Static Testing**

Running the dependency check report has reported 237 security vulnerabilities, but only some of the higher risk vulnerabilities will be documented and recommended solutions on how to patch them will be reported here.

Starting with alphabetical order, the package for Bouncy Castle, which is responsible for cryptographic algorithms, has 18 known CVE (common vulnerabilities and exposures) codes (CVE-2024-34447, CVE-2016-1000338, CVE-2016-1000342) that have to do with validating certain inputs. If inputs to these methods are incorrect, they can cause issues with certificates or signatures, where an attacker can provide extra data, creating “invisible” data. CVE-2016-1000343 reports that if default values are used when generating a DSA key pair, it creates a weak private key, which can be susceptible to attackers. CVE reports that explicitly passing parameters to the key generator can fix this issue. CVE-2020-15522 reports that there is a timing issue with the EC math library that can expose information about the private key to attackers and can observe timing information for the generation of deterministic ECDSA (elliptic curve digital signature algorithm) signatures. Performing your own blinding algorithms can help mitigate this risk. CVE-2023-33202 has a vulnerability that contains a possible Denial-of-Service (DoS) when parsing a file that can cause an OutOfMemoryError. To mitigate this, having limits on resources being distributed at any given time, or throttling, or caching result sets from database access could help to prevent these.

Moving on to Apache Commons BeanUtils, which is a wrapper around Java Beans, has a vulnerability (CVE-2025-48734) where an attacker can access the enum’s class loader property and execute arbitrary code. This can be mitigated simply by using version 1.9.2 or higher and disallowing the class level property access by default.

Skipping over medium or lower risk levels, H2, which is a database engine has a critical vulnerability (CVE-2021-42392), where during a connection to the database, an attacker can pass a JDNI (Java Naming and Directory Interface) driver name, causing remote code execution. Several suggestions report that deserializing could prevent the code from being carried over or overwritten. Another vulnerability (CVE-2022-23221) reports that the H2 Console can allow remote attackers to execute arbitrary code via a jdbc:2:mem URL substring. This can be mitigated by performing input validation on the console before execution to check for substrings that could be noted as attacking. CVE-2021-23463 is vulnerable to XML injection when it receives parsed string data, and if it executes the getSource() method while the parameter is DOMSource.class. To mitigate this, the XML parser or validator can be configured to disable external entity expansion. The vulnerability CVE-2022-45868 states the web-based admin console can be started via CLI and allows the user to specify the password in cleartext, but this can be solved by updating the package to 2.2.220 or later.

Jackson core is a streaming API and contains a high-level vulnerability (CVE-2025-52999) when it parses a user input file that may have deeply nested data. If the depth is large, a StackoverflowError could be thrown, or a StreamConstraintsException if a configured desired depth is reached before the end. This can be solved by avoiding the parsing of untrusted sources. Jackson databind works with the streaming API and contains a flaw (CVE-2020-25649) in the FasterXML databind where entity expansion is not secured properly and allows attacks from XML external entity and could impact data integrity. As with CVE-2021-23463, the XML parser and validator can be configured to disable external entity expansion. CVE-2020-36518 is associated with a Java Stackoverflow exception and DoS if there is a large depth of nested objects. This can be fixed by checking that the buffer is as large as needed before executing. CVE-2021-46877 has a DoS attack vulnerability that can be prevented by limiting or throttling all input to avoid any potential DoS.

Logback classic is a library that allows proper logging notes has vulnerabilities related to deserializing of untrusted data (CVE-2023-6378, CVE-2021-42550), where in one, poisoned data could be sent to mount a DoS style attack, and the other is when an attacker with required privileges could edit config files to allow arbitrary code to be loaded from LDAP servers. Both vulnerabilities can be mitigated (CWE-502) by preventing deserialization of objects.

Maven and Maven core has a vulnerability (CVE-2021-26291) that has a potential risk if a malicious actor takes over the repository listed in the POM or can insert themselves to pretend to be that repository. This is an architectural security fix (CWE-346).

Snakeyaml has several critical vulnerabilities, one of which (CVE-2022-1471) involves the standard constructor when deserialized could provide an attacker a way to remote code execution. Using the SafeConstructor could prevent deserialization of untrusted data or upgrading to version 2.0 or higher (CWE-502). Two vulnerabilities exist (CVE-2017-18640, CVE-2022-25857) are vulnerable to XML Entity Expansion) and can be stopped by scanning for recursive entity declarations and stop when parsing potentially insecure content (CWE-776). Vulnerabilities CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750 can all be found when YAML attempts to parse untrusted data and could lead to a DoS attack. Two solutions exists to prevent this; both in some way by checking the buffer size (CWE-787, CWE-121).

Spring AOP has several high and critical vulnerabilities. CVE-2022-22965 states that a spring mvc application can be vulnerable to remote code execution if run on Tomcat. Running and testing it in a sandbox could help to validate any input (CWE-94). CVE-2021-22118 is associated with a WebFlux application being vulnerable to a privilege escalation by creating a temporary storage directory. This can be mitigated by instituting the principle of least privilege when assigning access (CWE-269).

The Spring Boot and Spring Boot Starter Web packages have a combined 6 vulnerabilities. CVE-2023-20873 signifies that Spring Boot is susceptible to a security bypass if below a certain version (3.0.6+ or 2.7.11+). Upgrading the version can mitigate this. CVE-2022-27772 states that the packages are vulnerable to temporary directory hijacking but are only affected by versions no longer supported. Upgrading the packages to supported packages could mitigate these risks.

Spring Context has a listed 13 vulnerabilities but contain identical vulnerabilities from Spring Boot and Spring Boot Starter Web and won’t be listed. The vulnerability CVE-2020-5421 is associated with certain versions of Spring Framework, where protections against RFD attacks may be bypassed, depending on the browser through the use of a jsessionid path parameter. As of now, there are no known mitigations for this weakness, except using a non-listed version. In certain versions of Spring Framework, vulnerabilities CVE-2022-22950, CVE-2023-20861 and CVE-2023-20863 state that it is possible for a user to provide a specially crafted SpEL expression to start a DoS attack. Throttling or limiting resources could mitigate this (CWE-770).

Spring Core and Web also has similar vulnerabilities from other Spring framework packages listed above, so those will be ignored. Within the Spring framework, the annotation detection mechanism may not resolve or recognize annotations with vulnerability CVE-2025-41249, which can be a problem if authorization is required. Currently, this weakness is being analyzed, but the current workaround is to use the access control capabilities of your operating system to define your ACLs. Spring Web contains a vulnerability (CVE-2024-38809) with ETags request headers being vulnerable to DoS attacks, unless the application is using the corrected or fixed version (CWE-400). Similarly, the Spring MVC controller contains a method with the @RequestBody that is vulnerable to a DoS attack (CVE-2024-38828). Within the Spring Web framework, another vulnerability (CVE-2024-38816) states the WebMVC and WebFlux is vulnerable to path traversal attacks. An attacker pursuing this can craft malicious HTTP requests to obtain any file on the file system in which the Spring system is running on (CWE-22).

Finally, TomCat acts as a specialized web engine to run java-based applications, has a combined 91 across its packages, making it the most vulnerable package(s). CVE-2020-1938 reports that because TomCat trusts Apache JServ Protocol connections over HTTP, these AJP connections can be exploited due to this trust. CVE-2024-52316 states that if Tomcat is configured to a custom Jakarta authentication, ServerAuthContext may throw an exception during the process without setting an HTTP status, allowing attackers to bypass the authentication process. Input validation can correct this but is undergoing re-analysis (CWE-754). Tomcat contains a path equivalence vulnerability that allows an attacker to execute code with a partial PUT request (CVE-2025-24813). CVE-2020-13934 relates to an OutOfMemory exception could happen if a number of h2c direct connections to TomCat were to happen, causing a DoS (CWE-401). An infinite loop can occur with invalid payload lengths, leading to a DoS (CVE-2020-13935), or a specially crafted packet when TomCat is configured to use NIO+OpenSSL. TomCat also has a vulnerability with the HTTP/2 protocol allowing a DoS when request cancellation reset many streams quickly (CVE-2023-44487); throttling is the best solution here (CWE-400).

**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 Financials’ software application.

Starting with the input validations for the project code, all input should be checked for valid inputs and then processed appropriately. All int variables should be validated before being assigned to their assigned variables such as content and content2 inside the CRUD class, the deposit method inside the customer class should have its parameter checked before the parameter is passed through. For the DocData class constructor, before the constructor is called, the parameter passed through should be checked to ensure no command within the string passed could access the file system behind the API and validated to ensure the string contains only valid characters (alpha numerical, most likely, and no symbols).

Inside the myDateTime class, there exists two minor issues that could impact operations, and that is the input system for the variables designated for the time (seconds, minutes and hours). None of the three variables checks for proper integer values or even checks if they are within a valid range (0-12 for 12-hour clocks, or 0-23 for 24-hour clocks, and then 0-59 for seconds and minutes). Besides that, now having these variables not properly encapsulated means that an attacker could alter the time outside of any mutator methods and could disrupt the system by altering the time and issuing an attack in that fashion.

Considering all of the 237 detected vulnerabilities, the Common Weakness Enumeration does have a list of commonly known fixes for each of the known vulnerabilities. Below is a table of the more commonly known problems with the appropriate fixes. Not all will be listed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CVE Code | Package(s) | Brief Description | Severity | CWE Code |
| CVE-2024-34447 | bcprov-jdk15on-1.46.jar | Host provides a certificate, but the software cannot properly associate the two. | H | CWE-297  Ensure that all relevant properties of the certificate are fully validated before the certificate is pinned, including the hostname. |
| CVE-2016-1000338,  CVE-2016-1000342 | bcprov-jdk15on-1.46.jar | Unvalidated signature. Extra elements in sequence in signature creating a “false” positive signature | H | CWE-347  Architectural security tactic. |
| CVE-2021-42392 | h2-1.4.200.jar | An attacker may pass a JNDI driver name and a URL leading to a LDAP or RMI servers, causing remote code execution | C | CWE-502  Explicitly define a final object() to prevent deserialization. |
| CVE-2022-23221 | h2-1.4.200.jar | H2 Console before 2.1.210 allows remote attackers to execute arbitrary code | C | CWE-88  Convert to the expected data type, ensure that the input's values fall within the expected range of allowable values |
| CVE-2022-45868 | h2-1.4.200.jar | The web-based admin console in H2 Database Engine before 2.2.220 can be started via the CLI with the argument -webAdminPassword, which allows the user to specify the password in cleartext | H | CWE-312  Use the provider's (cloud provider) controls to encrypt the data at rest |
| CVE-2020-25649 | jackson-databind-2.10.2.jar | This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. | H | CWE-611  Many XML parsers and validators can be configured to disable external entity expansion. |
| CVE-2020-36518 | jackson-databind-2.10.2.jar | allows a Java StackOverflow exception and denial of service via a large depth of nested objects. | H | CWE-787  Double check that the buffer is as large as specified. |
| CVE-2021-46877 | jackson-databind-2.10.2.jar | allows attackers to cause a denial of service (2 GB transient heap usage per read) in uncommon situations involving JsonNode JDK serialization. | H | CWE-770  Limit the amount of resources that are accessible to unprivileged users. |
| CVE-2023-6378,  CVE-2023-6378 | logback-classic-1.2.3.jar | allows an attacker to mount a Denial-Of-Service  attack by sending poisoned data. | H | CWE-502  Explicitly define a final object() to prevent deserialization. |
| CVE-2021-26291 | maven-core-3.0.jar,  maven-settings-3.0.jar | A malicious actor takes over that repository or is able to insert themselves into a position to pretend to be that repository. | C | CWE-346  Architectural security tactic. |
| CVE-2022-1471 | snakeyaml-1.25.jar | 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. | C | CWE-502  Explicitly define a final object() to prevent deserialization. |
| CVE-2022-22965 | spring-aop-5.2.3.RELEASE.jar,  spring-context-5.2.3.RELEASE.jar,  spring-core-5.2.3.RELEASE.jar,  spring-expression-5.2.3.RELEASE.jar,  spring-web-5.2.3.RELEASE.jar,  spring-webmvc-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. | C | CWE-94  Use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications |
| CVE-2023-20873 | spring-boot-2.2.4.RELEASE.jar,  spring-boot-starter-web-2.2.4.RELEASE.jar, | an application that is deployed to Cloud Foundry could be susceptible to a security bypass. | C | NVD-CWE-noinfo |
| CVE-2022-27772 | spring-boot-2.2.4.RELEASE.jar,  spring-boot-starter-web-2.2.4.RELEASE.jar | vulnerable to temporary directory hijacking | H | CWE-668  Architectural security tactic. |
| CVE-2023-20883 | spring-boot-2.2.4.RELEASE.jar,  spring-boot-starter-web-2.2.4.RELEASE.jar | there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache. | H | CWE-400  Uniformly throttles all requests in order to make it more difficult to consume resources |
| CVE-2022-22950 | spring-aop-5.2.3.RELEASE.jar,  spring-context-5.2.3.RELEASE.jar,  spring-core-5.2.3.RELEASE.jar,  spring-expression-5.2.3.RELEASE.jar,  spring-web-5.2.3.RELEASE.jar,  spring-webmvc-5.2.3.RELEASE.jar | a user to provide a specially crafted SpEL expression that may cause a denial-of-service condition. | M | CWE-770  Use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications |
| CVE-2025-41249 | spring-core-5.2.3.RELEASE.jar,  spring-web-5.2.3.RELEASE.jar | annotation detection mechanism may not correctly resolve annotations on methods within type hierarchies | H | CWE-285  Ensure that you perform access control checks related to your business logic. |
| CVE-2016-1000027 | spring-web-5.2.3.RELEASE.jar | suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. | C | CWE-502  Explicitly define a final object() to prevent deserialization. |
| CVE-2024-38816 | spring-webmvc-5.2.3.RELEASE.jar | An attacker can craft malicious HTTP requests and obtain any file on the file system that is also accessible to the process in which the Spring application is running. | H | CWE-22  Use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications |
| CVE-2020-1938 | tomcat-embed-core-9.0.30.jar,  tomcat-embed-websocket-9.0.30.jar | Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. | C | NVD-CWE-Other |
| CVE-2024-52316 | tomcat-embed-core-9.0.30.jar,  tomcat-embed-websocket-9.0.30.jar | a custom Jakarta Authentication (formerly JASPIC) ServerAuthContext component which may throw an exception, allowing the user to bypass the authentication process. | C | CWE-754  Check the results of all functions that return a value and verify that the value is expected. |
| CVE-2025-24813 | tomcat-embed-core-9.0.30.jar,  tomcat-embed-websocket-9.0.30.jar | Path Equivalence: 'file.Name' (Internal Dot) leading to Remote Code Execution | C | CWE-44 |

References

StoneX. (2024). *Financial regulations*. @StoneX\_Official. <https://www.stonex.com/en/financial-glossary/financial-regulations/#:~:text=Banking%20Regulations:%20The%20Federal%20Reserve,and%20timely%20information%20to%20investors>.

‌ Camarda, V. (2025, April 30). *5 Identity Security Challenges in the Finance Industry | Identity Defined Security Alliance*. Identity Defined Security Alliance. <https://www.idsalliance.org/blog/5-identity-security-challenges-in-the-finance-industry/#:~:text=Cyberattacks%20are%20one%20of%20the,Estate%20documents>.

Oracle. (2019). *Secure Coding Guidelines for Java SE*. Oracle.com. <https://www.oracle.com/java/technologies/javase/seccodeguide.html>

NVD - Search and Statistics. (2025). Nist.gov. <https://nvd.nist.gov/vuln/search#/nvd/home?resultType=records>

Common Weakness Enumeration. (2025). CWE - Common Weakness Enumeration. Mitre.org. <https://cwe.mitre.org/index.html>

‌

‌