# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/24/24** | **Justin Phillips** | **Initiated the document** |
|  |  |  |  |

## 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

Justin Phillips

**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 financial institution handling banking information for their webapp. They will be handling sensitive financial information and user account information. The company will need to consider international transactions, multifactor identification, identity verification, card controls, bank account verification, and an abundance of other security factors related to these functions. There are a lot of threats to defend from; Hackers, phishers in both the company and the customer’s end, company security, man in the middle attacks, and any packets received by the end user will need to be considered as liable threats. All libraries used must also be kept free of known vulnerabilities to the greatest extent possible; No areas of weakness can be afforded with such sensitive information. The system must also be available as much as possible, there is little room for downtime. Finally, government limits may exist based on where the server is housed and where sites are accessible, and the company must worry about the General Data Protection Regulation and Payment Card Industry Data Security Standard while operating internationally, an EU law which defend the privacy rights of the end users and a security standard for keeping online balance interactions secure respectively.

**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.

It is apparent then that there will need to be very strict input validation, scrutinizing APIs, top of the line cryptography for the entire communications pipeline, strict client/server communications privacy and security, safe error handling, code optimized for security, and strict instance and account encapsulation.

Input validation is important for all transactions of information and all out-of-system data must be assumed to be untrusted. Having definitions for specific acceptable headers is necessary to prevent spoofing or malicious header modifications. Filled at forms should be parsed through a filter to take out injections or other forms of potential attack. Failure to properly validate input may mean that an attacker can disguise themselves as another user, elevate their own permissions, or remotely execute malicious code.

APIs interfacing with our application will need to be monitored for their own vulnerabilities as well as their permission to interface with our application. We may work with popular APIs such as Plaid for interfacing with certain online ecosystems or allow some developers an API key for developing peripheral apps without directly interacting with data through requiring certain user credentials and by continuing to lock access to the full depth of the system from any external sources (limiting the API’s use to retrieving information from an already authenticated account and session, for instance). Failure to vet APIs interfacing with us could result in data leaks through someone else’s vulnerability. Failure to properly regulate our own API could result in data leaks through inappropriate access to information.

Encrypting communications between the client and server is critical to prevent man in the middle attacks, packet modifications, snooping or spying especially on an open network if the client is in a public place, and otherwise malicious activity. Failure to encrypt information could lead to leaking of user name and password sensitive information and a breach of security integrity for the client’s sensitive financial information.

Handling errors safely will prevent system downtime or denial of service attacks caused by things such as integer overflow, oversized headers on packets, inappropriate headers on packets, or other common exploits used to take systems down, drain their resources, or make them slow.

Building our application securely is also important. Everything from the design to the actual programming behind the project will need to be analyzed to ensure it is secure and free of bugs and exploits that could compromise security, such as a design flaw allowing for permissions elevation.

Encapsulation should occur per instance as well as per account. No account should be able to see any information related to any other account, and a user should only have one instance at a time possible per account. Failure to properly encapsulate could mean leaking of data or information between instances or accounts, or instance mixups where the information is sent to the wrong client.

**3. Manual Review**

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

CRUD and CRUDController.java are placeholder files. The first thing that should be implemented is using the RESTful API to implement security measures such as roles and user accounts to ensure that while our application is developed it cannot be utilized to alter our records without the correct permissions.

CRUDController.java uses business\_name as a request parameter which may be a privacy vulnerability.

In DocData.java, the DriverManager connection is hardcoded with the localhost connection port plainly written as well, allowing anyone with access to this file in a readable format to gain access to the DriverManager or to replace this port with a different location.

In customer.java, functions exist to showInfo() and deposit(int a), simple functions with no validation or verification of the customer’s identity, ID, session ID, or any other security measures. The deposit function has no check to ensure that the amount is valid; It just allows any arbitrary int. As well, this should be a double to allow for cents.

In GreetingController.java, the name parameter is unvalidated and gotten via a public request param. As an overview, it seems like all requests are made through the HTTP header, which is insecure as well as leaves vulnerabilities by exposing information in the URL header, allowing information to be retrieved by browser history as well as snooped fairly easily in public networks or by man in the middle attacks.

Observable in the controller files, raw user data is utilized and not parsed for security purposes in advance to being utilized.

In application.properties, the server port is 8081, TCP/UDP, which should be 443 or 8443 HTTPS.

In pom.xml, there is no observable dependency for a validator.

**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

This table has all of the information relevant to each dependency, courtesy of NVD.NIST.gov (*National Vulnerability Database,* 2024), with vulnerability descriptions borrowed from that site as well as research conducted by Justin Phillips on another related project utilizing many of the same dependencies (Phillips, 2024):

|  |  |
| --- | --- |
| Color | Purpose |
|  | Highest severity |
|  | High severity |
|  | Moderate risk |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dependency | Purpose | Vulnerabilities | Vulnerabilities description | Recommendation |
| Bcprov-jdk15on-1.46.jar | Bouncy Castle 1.46 is an encryption library for JDK 1.5. | CVE-2023-33202 | BC versions for Java prior to 1.73 have a vulnerability in the PEMParser file where ASN.1 data can be tailored to produce an OutOfMemory error. This creates potential for denial of service. (2018) (National Vulnerability Database, 2024) | Critical to update; Update to 1.77.  Specifically, older versions of Bouncy Castle have severe structural issues that impact the functionality of the dependency and don’t enable it to perform its essential functions. This must be updated. |
| CVE-2016-1000346 | The client (other party) DH public key was not fully validated until versions after 1.55, so invalid keys could be used. Newer versions check keys on handshake. (2018) (National Vulnerability Database, 2024) |
| CVE-2013-1624 | TLS before 1.48 did not properly time side-channel attacks if the MAC check operation was non-compliant due to malformed CBC padding. This allows attackers to conduct attacks such as plaintext recovery via analysis of timing data for crafted packets. (2013) (National Vulnerability Database, 2024) |
| CVE-2016-1000344 | DHIES allowed use of ECB mode; This is an unsafe mode. (2018) (National Vulnerability Database, 2024) |
| CVE-2016-1000343 | Default configuration DSA key pairs generated were weak, assuming 1024 bits of key size. This can be configured as a manual solution. (2018) (National Vulnerability Database, 2024) |
| CVE-2016-1000342 | 1.55 and older did not properly validate ASN.1 encoding of signature on verification; Extra elements can be injected into the sequence making up the signature while still passing validation, meaning that data can be injected and potentially remote code execution is possible. (2018) (National Vulnerability Database, 2024) |
| CVE-2016-1000339 | The primary engine class used for AES, AESFastEngine, allowed snooping of data on the CPU channel for the application, allowing the lookup tables to be monitored and allow information leaks on the key in use. This component was modified in versions 1.56 onwards to no longer leak. (2018) (National Vulnerability Database, 2024) |
| CVE-2017-13098 | RSA private keys can be recovered from vulnerable applications due to a weak Bleichenbacher oracle being provided when TLS cipher suites using RSA keys are negotiated. (2018) (National Vulnerability Database, 2024) |
| CVE-2018-5382 | The default keystore HMAC is only 16 bits long, allowing attackers to easily compromise it. 1.47 uses 160 bits instead, and applies this repair retroactively to any existing keystores. (2018) (National Vulnerability Database, 2024) |
| hibernate-validator-6.0.18.Final.jar | Hibernate validator 6.0.18 is a JavaBeans parser that verifies Beans meet particular criteria, such as ensuring a value exists and is not null or exists and is within a particular range, such as a version checker. | CVE-2020-10693 | In older versions of hibernate-validator, a bug in the message interpolation processor allowed invalid EL expressions to be evaluated as if they passed the validity requirements; This bypasses input sanitation and may allow injection of menacing data, though this only seems to influence user-controlled data in error messages specifically. (2020) (National Vulnerability Database, 2024) | This should be updated, though it is a moderate priority. The current version is 8.0.0, we are two major updates behind. There are performance improvements, new features, and bug fixes that are waiting to be taken advantage of. |
| jackson-databind-2.10.2.jar | Jackson Databind 2.10.2 is a general purpose data binder which allows processing and conversion between JSON and java files. | CVE-2023-35116 | Versions prior to and including 2.15.2 allow attackers to cause denial of service by crafting objects which use cyclic dependencies. (2023) (National Vulnerability Database, 2024) | Jackson databind is currently 2.18. Considering the quantity and severity of the vulnerabilities, including the data integrity risk, it must be updated to a newer version. |
| CVE-2021-46877 | Jackson Databind 2.10.X->2.12.x before 2.12.6 and 2.13.x before 2.13.1 have vulnerabilities allowing JsonNode JDK serialization flaws to be used to create a denial of service due to 2Gb per read heap usage bloat in uncommon situations, which is very severe. (2023) (National Vulnerability Database, 2024) |
| CVE-2022-42004 | In jackson-databind before 2.13.4, there is a lack of deserializer checks for deeply nested arrays, creating a vulnerability where deeply nested arrays can be used to exhaust resources. (2022) (National Vulnerability Database, 2024) |
| CVE-2022-42003 | A lack of a check in primitive value deserializers in versions before 2.13.4.1 and 2.12.17.1 can allow deep wrapper array nesting when a configuration variable UNWRAP\_SINGLE\_VALUE\_ARRAYS is true. (2022) (National Vulnerability Database, 2024) |
| CVE-2020-36518 | A large depth of nested objects can allow a Java StackOverflow exception and a denial of service prior to 2.13.0. (2022) (National Vulnerability Database, 2024) |
| CVE-2020-25649 | A vulnerability exists with entity expansion improper implementation. This flaw allows vulnerability through the attaching of external XML entities, which puts data integrity at risk. (2020) (National Vulnerability Database, 2024) |
| log4j-api-2.12.1.jar | Apache Log4J 2.12.1 is a Java logging framework. | CVE-2021-44832 | Versions 2.0-beta7 through 2.17.0 with exceptions for 2.3.2 and 2.12.4 are vulnerable to remote code execution attacks when an attacker’s configuration utilizes a Java Database Appender with a JNDI LDAP data source URI, occurring when the attacker has control of the target LDAP server. This issue can be countermeasured by limiting JNDI data sources to Java protocols in Log4J versions 2.17.1, 2.12.4, and 2.3.2, meaning this fix is not available on the version we currently utilize. (Phillips, 2024) | Log4J is now on release 2.23.1, and must be updated to take advantage of the latest security updates and improvements, as well as fixes for all of these issues. Alternatively, as Logback is intended as a successor for Log4J, it should be considered whether Log4J is necessary for our application; If not, it should be considered to be removed in favor of Logback.  Update or replace with modern Logback. |
| CVE-2021-45105 | 2.0-alpha1 to 2.16.0 with similar exclusions in 2.12.3 and 2.3.1 fail to protect from self-referential lookups, allowing attackers to cause denial of service. This issue is corrected in newer versions 2.17.0, 2.12.3, and 2.3.1. (Phillips, 2024) |
| CVE-2021-45046 | Certain non-default configurations allow allow vulnerabilities for attackers with control over the thread context map to input data into a JNDI lookup pattern, resulting in information leaks and remote code execution, as well as local code execution in all environments. This was fixed in newer versions 2.16.0 and 2.12.2 by removing support for message lookup patterns and by default disabling JNDI functionality. (Phillips, 2024) |
| CVE-2021-44228 | 2.2.0beta through 2.15.0, with exceptions for 2.12.2, 2.12.3, and 2.3.1, have vulnerabilities with JNDI features not protecting against attacker controlled JDNI endpoints; An attacker gaining control over log messages or their parameters may execute code loaded from LDAP servers when message lookup substitution is enabled. 2.15.0 disables this behavior, and newer versions 2.16.0, 2.12.2, 2.31, and later remove the functionality altogether. (Phillips, 2024) |
| CVE-2020-9488 | Improper validation of certificates is allowed with host mismatches in the SMTP appender, meaning an SMTPS connection can be intercepted by a man-in-the-middle attack, leaking logs sent through the appender, which is fixed in versions 2.12.3 and 2.13.1 and newer. (Phillips, 2024) |
| logback-core-1.2.3.jar | Logback Core 1.2.3 is a logging framework which may act as a successor to Log4j. | CVE-2023-6378 | A logback receiver component allows attackers to mount denial of service attacks by sending crafted data. (Phillips, 2024) | The current version is 1.5.6, and it should be updated. It is worth considering that Logback should come to replace Log4J. |
| CVE-2021-42550 | An attacker with certain privileges allowing them to edit configuration files could craft malicious configurations which allow code execution from LDAP servers. (Phillips, 2024) |
| snakeyaml-1.25.jar | SnakeYAML 1.25 is a Java library used for parsing the YAML language format for configuration files. | CVE-2022-1471 | Below version 2.0 where the Constructor() class does not restrict types able to be instantiated during deserialization, meaning that YAML content provided by an attacker can lead to remote code execution. There is a functional SafeConstructor method that restricts deserialization. (Phillips, 2024) | SnakeYAML is deprecated; SnakeYAML Engine offers more security and stability alongside new features such as YAML 1.2 support.   The transition should be made to SnakeYAML Engine 2.7. |
| CVE-2022-41854 and relative CVE-2022-25857 | User supplied input may allow an attacker to cause a stack overflow. Similarly, stack overflow can be caused between versions 0 and 1.31 due to a missing nested depth limitation for collections, which may cause a denial of service attack. (Phillips, 2024) |
| CVE-2022-25857 | The alias feature prior to 1.26 allows entity expansion during a load operation, which can cause resource drain. (Phillips, 2024) |
| spring-boot-2.2.4.RELEASE.jar and spring-boot-starter-web-2.2.4.RELEASE.jar | Spring Boot 2.2.4 is part of the Spring framework. It allows autoconfiguration of the application and creating a standalone application .jar file easily. Its Starter Web module is a starter that brings in all the dependencies needed to create a functional web-app, embedding tools like Tomcat for self-contained app servers, i.e. that Localhost port. | CVE-2023-20883 | “In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.” May 26, 2023; 1:15:14 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) | Spring boot 2.x is no longer supported. The current version is 3.2.5. We must update to resolve vulnerabilities. |
| CVE-2023-20873 | “In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+.” April 20, 2023; 5:15:08 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-27772 | “spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.  AbstractConfigurableWebServerFactory.  createTempDir method.” March 30, 2022; 2:15:08 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| spring-core-5.2.3.RELEASE.jar and spring-web-5.2.3.RELEASE.jar and spring-webmvc-5.2.3.RELEASE.jar | Spring Core 5.2.3 is part of the Spring framework. It is the core container for the Spring ecosystem. Spring web MVC is responsible for the handling of objects in the webapp related to the model, the view, and the object controllers, and is one of Spring Core’s complimentary components. | CVE-2023-20863 | “In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.” April 13, 2023; 4:15:07 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) | Spring 5.x is no longer supported. The current version is 6.0.11. We must update to resolve vulnerabilities. |
| CVE-2022-22971 | “In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, application with a STOMP over WebSocket endpoint is vulnerable to a denial of service attack by an authenticated user.” May 12, 2022; 4:15:15 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-22970 | “In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, applications that handle file uploads are vulnerable to DoS attack if they rely on data binding to set a MultipartFile or javax.servlet.Part to a field in a model object.” May 12, 2022; 4:15:15 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-22968 | “In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.” April 14, 2022; 5:15:08 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-22965 | “A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.” April 01, 2022; 7:15:13 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-22096 | “In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.” October 28, 2021; 12:15:07 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-22118 | “In Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7, a WebFlux application is vulnerable to a privilege escalation: by (re)creating the temporary storage directory, a locally authenticated malicious user can read or modify files that have been uploaded to the WebFlux application, or overwrite arbitrary files with multipart request data.” May 27, 2021; 11:15:07 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-5421 | “In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.” September 19, 2020; 12:15:11 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2016-1000027 | “Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.” January 02, 2020; 6:15:11 PM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| tomcat-embed-core-9.0.30.jar and tomcat-embed-websocket-9.0.30.jar | Tomcat 9.0.30 is a single Java web application and a server distribution compressed and packaged together in the form of a JAR, WAR, or ZIP file. It allows self-hosting servers in our app. | CVE-2024-21733 | “Generation of Error Message Containing Sensitive Information vulnerability in Apache Tomcat.This issue affects Apache Tomcat: from 8.5.7 through 8.5.63, from 9.0.0-M11 through 9.0.43. Users are recommended to upgrade to version 8.5.64 onwards or 9.0.44 onwards, which contain a fix for the issue.” January 19, 2024; 6:15:08 AM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) | Tomcat is severely outdated and has many significant flaws. It must be immediately updated to the current version 10.1.24, as it is a major version behind and the 9.x series of Tomcat has dropped support entirely, meaning new vulnerabilities will be discovered and will not be repaired. |
| CVE-2023-46589 | “Improper Input Validation vulnerability in Apache Tomcat.Tomcat from 11.0.0-M1 through 11.0.0-M10, from 10.1.0-M1 through 10.1.15, from 9.0.0-M1 through 9.0.82 and from 8.5.0 through 8.5.95 did not correctly parse HTTP trailer headers. A trailer header that exceeded the header size limit could cause Tomcat to treat a single request as multiple requests leading to the possibility of request smuggling when behind a reverse proxy. Users are recommended to upgrade to version 11.0.0-M11 onwards, 10.1.16 onwards, 9.0.83 onwards or 8.5.96 onwards, which fix the issue.” November 28, 2023; 11:15:06 AM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2023-44487 | “The HTTP/2 protocol allows a denial of service (server resource consumption) because request cancellation can reset many streams quickly, as exploited in the wild in August through October 2023.” NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2023-41080 | “URL Redirection to Untrusted Site ('Open Redirect') vulnerability in FORM authentication feature Apache Tomcat.This issue affects Apache Tomcat: from 11.0.0-M1 through 11.0.0-M10, from 10.1.0-M1 through 10.0.12, from 9.0.0-M1 through 9.0.79 and from 8.5.0 through 8.5.92. The vulnerability is limited to the ROOT (default) web application.” August 25, 2023; 5:15:09 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2023-2870 | “When using the RemoteIpFilter with requests received from a reverse proxy via HTTP that include the X-Forwarded-Proto header set to https, session cookies created by Apache Tomcat 11.0.0-M1 to 11.0.0.-M2, 10.1.0-M1 to 10.1.5, 9.0.0-M1 to 9.0.71 and 8.5.0 to 8.5.85 did not include the secure attribute. This could result in the user agent transmitting the session cookie over an insecure channel.” March 22, 2023; 7:15:10 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-43980 | “The simplified implementation of blocking reads and writes introduced in Tomcat 10 and back-ported to Tomcat 9.0.47 onwards exposed a long standing (but extremely hard to trigger) concurrency bug in Apache Tomcat 10.1.0 to 10.1.0-M12, 10.0.0-M1 to 10.0.18, 9.0.0-M1 to 9.0.60 and 8.5.0 to 8.5.77 that could cause client connections to share an Http11Processor instance resulting in responses, or part responses, to be received by the wrong client.” September 28, 2022; 10:15:09 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-34305 | “In Apache Tomcat 10.1.0-M1 to 10.1.0-M16, 10.0.0-M1 to 10.0.22, 9.0.30 to 9.0.64 and 8.5.50 to 8.5.81 the Form authentication example in the examples web application displayed user provided data without filtering, exposing a XSS vulnerability.” June 23, 2022; 7:15:07 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2022-29885 | “The documentation of Apache Tomcat 10.1.0-M1 to 10.1.0-M14, 10.0.0-M1 to 10.0.20, 9.0.13 to 9.0.62 and 8.5.38 to 8.5.78 for the EncryptInterceptor incorrectly stated it enabled Tomcat clustering to run over an untrusted network. This was not correct. While the EncryptInterceptor does provide confidentiality and integrity protection, it does not protect against all risks associated with running over any untrusted network, particularly DoS risks.” May 12, 2022; 4:15:07 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-41079 | “Apache Tomcat 8.5.0 to 8.5.63, 9.0.0-M1 to 9.0.43 and 10.0.0-M1 to 10.0.2 did not properly validate incoming TLS packets. When Tomcat was configured to use NIO+OpenSSL or NIO2+OpenSSL for TLS, a specially crafted packet could be used to trigger an infinite loop resulting in a denial of service.” September 16, 2021; 11:15:07 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-33037 | “Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honoured the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding.” July 12, 2021; 11:15:08 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-30640 | “A vulnerability in the JNDI Realm of Apache Tomcat allows an attacker to authenticate using variations of a valid user name and/or to bypass some of the protection provided by the LockOut Realm. This issue affects Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65.” July 12, 2021; 11:15:08 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-25122 | “When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 could duplicate request headers and a limited amount of request body from one request to another meaning user A and user B could both see the results of user A's request.” March 01, 2021; 7:15:13 AM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2021-24122 | “When serving resources from a network location using the NTFS file system, Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106 were susceptible to JSP source code disclosure in some configurations. The root cause was the unexpected behaviour of the JRE API File.getCanonicalPath() which in turn was caused by the inconsistent behaviour of the Windows API (FindFirstFileW) in some circumstances.” January 14, 2021; 10:15:13 AM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-17527 | “While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests.” December 03, 2020; 2:15:12 PM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-13943 | “If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers - including HTTP/2 pseudo headers - from a previous request rather than the intended headers. This could lead to users seeing responses for unexpected resources.” October 12, 2020; 10:15:12 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-13935 | “The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.” July 14, 2020; 11:15:11 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-8022 | “A Incorrect Default Permissions vulnerability in the packaging of tomcat on SUSE Enterprise Storage 5, SUSE Linux Enterprise Server 12-SP2-BCL, SUSE Linux Enterprise Server 12-SP2-LTSS, SUSE Linux Enterprise Server 12-SP3-BCL, SUSE Linux Enterprise Server 12-SP3-LTSS, SUSE Linux Enterprise Server 12-SP4, SUSE Linux Enterprise Server 12-SP5, SUSE Linux Enterprise Server 15-LTSS, SUSE Linux Enterprise Server for SAP 12-SP2, SUSE Linux Enterprise Server for SAP 12-SP3, SUSE Linux Enterprise Server for SAP 15, SUSE OpenStack Cloud 7, SUSE OpenStack Cloud 8, SUSE OpenStack Cloud Crowbar 8 allows local attackers to escalate from group tomcat to root.” June 29, 2020; 5:15:11 AM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-11996 | “A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.” June 26, 2020; 1:15:10 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-9484 | “    When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103 if a) an attacker is able to control the contents and name of a file on the server; and b) the server is configured to use the PersistenceManager with a FileStore; and c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter="null" (the default unless a SecurityManager is used) or a sufficiently lax filter to allow the attacker provided object to be deserialized; and d) the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over; then, using a specifically crafted request, the attacker will be able to trigger remote code execution via deserialization of the file under their control. Note that all of conditions a) to d) must be true for the attack to succeed.” May 20, 2020; 3:15:09 PM -0400 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-1938 | “When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified 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 (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.” February 24, 2020; 5:15:12 PM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |
| CVE-2020-1935 | “In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner.” February 24, 2020; 5:15:11 PM -0500 NVD NIST.GOV (National Vulnerability Database, 2024) |

**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.

All of the dependencies we rely upon should be made up-to-date. We must discuss at a meeting whether Log4J should be migrated to Logback entirely so we can drop that dependency. SnakeYAML should be dropped in favor of its successor, the SnakeYAML Engine, which offers support for the newer YAML 1.2 format. Our Spring dependencies must be updated. Tomcat must be updated. Bouncy Castle must be updated. Jackson Databind must be updated. Our codebase must be updated with new security feature implementations and the HTTP header technique for sending parameters must be ceased in favor of POST form requests that are encrypted with a modern SLA hash or similar effective encryption technology. As we update our dependencies we must investigate the dependencies for migration guides; We are significantly behind by many major versions on many of our requirements, and foundational frameworks like Spring are no longer maintained, leaving our project vulnerable by the nature of its architecture. Both Spring APIs we employ have namespace changes between our versions and their latest versions, so following the migration guides will help us navigate this transition. By implementing all these changes carefully, we can build a far more secure app for our clients to protect their information, as well as make ourselves robust.

REFERENCES:

*“National Vulnerability Database*.” National Institute of Standards and Technology, U.S. Department of Commerce, 2024

Phillips, Justin. *“CS 305 Module Two Coding Assignment Template”*. Southern New Hampshire University, CS 305. 5/20/24.