****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **05/19/2022** | **Melanie Probst** |  |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Melanie Probst

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following 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 about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

With the modernization of companies in every field, strong cybersecurity is more important than ever, especially with those who have or would like to implement cloud services like Artemis Financial. With the inclusion of a RESTful web API, there will be another front to protect from potential attackers. Since Artemis Financial is a consulting company, it is imperative that their data and their client’s data are protected from those who wish to do them harm. There may be situations where clients of Artemis Financial would be abroad and in need of assistance. In such a situation, the company must be equipped to handle those requests. Aside from that, there does not appear to be any need for the company to facilitate international transactions themselves. There are both federal and state laws regarding cybersecurity that must be complied to by private companies. Each state has their own different laws and regulations, so if Artemis Financial wishes to operate in those states, they must abide by them. On the federal level, there are laws such as the SEC Regulation S-P under 17 CFR Part 248, Subpart A, which is an information security regulation that applies to all US and foreign investment advisers (among others) registered with the SEC (IT Governance, n.d.). Since Artemis Financial handles investments, this law will need to be looked into and compliance must be assured. There is also another regulation which needs to be followed, known as the Federal Trade Commission Act §5, which has established defined rules for companies within its jurisdiction to follow (IT Governance, n.d.). The external threat present now and in the future are the same: attackers who wish to steal confidential client and company data. As a result, the libraries and resources used in the construction of the Artemis Financial project must be secure and reliable. If any vulnerabilities or openings for attackers exist, they must be patched and sealed in order to protect our client from potential threats. As it stands now, Artemis Financial wishes to modernize their operations, particularly by securing the RESTful web API they have implemented. As stated previously, the API offers more methods of attack for those who wish to do harm. This means that there will be additional program requirements to ensure that the API is secure. A few examples of additional security requirements include ensuring a secure connection between the client and the server, and that all input is properly validated. Open source code is always a great resource for innovation and improvements. However, open source code can be dangerous since anyone can look at it and find a vulnerability. Special care should be taken to secure any loopholes in the code we use.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

When consulting the Vulnerability Assessment Process Flow Diagram, there are a few areas of security that need to be addressed for this project. One area is Input Validation, which dictates that any and all input and representations should be validated (Marquez, 2022). This is important to the application because any input that isn’t validated is at risk for injection from an attacker. Similarly, if representations are not validated, an attacker or user could gain access to information or abilities that they aren’t authorized to have. Encryption is also a concern for this application, since there is a structured database connected to the application (Marquez, 2022). Access to this database must be secure in order to protect the information contained within it. This means that the username and password to this database must not be directly available within the application to avoid any attackers gaining access to it. Due to the use of a REST API, client server connection security and code error handling must be addressed (Marquez, 2022). Any possibility for an attacker to intercept the data being transmitted must be eliminated. Similarly, the server must be protected from a DDOS attack, which includes error handling. If the errors produced in such an attack are correctly implemented, then the severity of the attack will be mitigated. Naturally, the REST API must also be properly secured, as defined by the API area of security (Marquez, 2022).

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

In the code itself, I see that there is a direct connection to the SQL database within the DocData class, and that the username and password for the database has been hard coded into the program. This is dangerous because if an attacker found an injection loophole or a way into the source code, then they would also have access to all the customer data found in the database. I also noticed that a template is used in the GreetingController class. This leaves the program vulnerable to injection, which is especially risky due to the username and password being hard coded. Additionally, I noticed that while the customer’s account number is a private variable, there is a way to see it through the showInfo() method in the Customer class. While this is not usually a problem, since customers should be able to see their account number, without the proper input validation and the present risk of injection throughout the program, this information should be better secured.

## 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 dependency check report. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

**Dependency:** bcprov-jdk15on-1.46.jar

**Vulnerability ID:** CVE-2016-1000352

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000346

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the other party DH public key is not fully validated. This can cause issues as invalid keys can be used to reveal details about the other party's private key where static Diffie-Hellman is in use. As of release 1.56 the key parameters are checked on agreement calculation.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000345

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES CBC mode vulnerable to padding oracle attack. For BC 1.55 and older, in an environment where timings can be easily observed, it is possible with enough observations to identify when the decryption is failing due to padding.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000344

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000343

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the DSA key pair generator generates a weak private key if used with default values. If the JCA key pair generator is not explicitly initialised with DSA parameters, 1.55 and earlier generates a private value assuming a 1024 bit key size. In earlier releases this can be dealt with by explicitly passing parameters to the key pair generator.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000342

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier ECDSA 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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000341

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier DSA signature generation is vulnerable to timing attack. Where timings can be closely observed for the generation of signatures, the lack of blinding in 1.55, or earlier, may allow an attacker to gain information about the signature's k value and ultimately the private value as well.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000339

**Description:** “In the Bouncy Castle JCE Provider version 1.55 and earlier the primary engine class used for AES was AESFastEngine. Due to the highly table driven approach used in the algorithm it turns out that if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak information on the AES key being used. There was also a leak in AESEngine although it was substantially less. AESEngine has been modified to remove any signs of leakage (testing carried out on Intel X86-64) and is now the primary AES class for the BC JCE provider from 1.56. Use of AESFastEngine is now only recommended where otherwise deemed appropriate.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000338

**Description:** “In Bouncy Castle JCE Provider version 1.55 and earlier 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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2018-5382

**Description:** “The default BKS keystore use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore. Bouncy Castle release 1.47 changes the BKS format to a format which uses a 160 bit HMAC instead. This applies to any BKS keystore generated prior to BC 1.47. For situations where people need to create the files for legacy reasons a specific keystore type "BKS-V1" was introduced in 1.49. It should be noted that the use of "BKS-V1" is discouraged by the library authors and should only be used where it is otherwise safe to do so, as in where the use of a 16 bit checksum for the file integrity check is not going to cause a security issue in itself.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2017-13098

**Description:** “BouncyCastle TLS prior to version 1.0.3, when configured to use the JCE (Java Cryptography Extension) for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. This vulnerability is referred to as "ROBOT."” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2013-1624

**Description:** “The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.” (National Institute of Standards and Technology, n.d.)

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

**Vulnerability ID:** CVE-2022-27772

**Description:** “\*\* UNSUPPORTED WHEN ASSIGNED \*\* 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. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2022-22968

Description: “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2022-22965

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2016-1000027  
**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2014-0054  
**Description:** “The Jaxb2RootElementHttpMessageConverter in Spring MVC in Spring Framework before 3.2.8 and 4.0.0 before 4.0.2 does not disable external entity resolution, which allows remote attackers to read arbitrary files, cause a denial of service, and conduct CSRF attacks via crafted XML, aka an XML External Entity (XXE) issue. NOTE: this vulnerability exists because of an incomplete fix for CVE-2013-4152, CVE-2013-7315, and CVE-2013-6429.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2013-7315  
**Description:** “The Spring MVC in Spring Framework before 3.2.4 and 4.0.0.M1 through 4.0.0.M2 does not disable external entity resolution for the StAX XMLInputFactory, which allows context-dependent attackers to read arbitrary files, cause a denial of service, and conduct CSRF attacks via crafted XML with JAXB, aka an XML External Entity (XXE) issue, and a different vulnerability than CVE-2013-4152. NOTE: this issue was SPLIT from CVE-2013-4152 due to different affected versions.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2013-4152  
**Description:** “The Spring OXM wrapper in Spring Framework before 3.2.4 and 4.0.0.M1, when using the JAXB marshaller, does not disable entity resolution, which allows context-dependent attackers to read arbitrary files, cause a denial of service, and conduct CSRF attacks via an XML external entity declaration in conjunction with an entity reference in a (1) DOMSource, (2) StAXSource, (3) SAXSource, or (4) StreamSource, aka an XML External Entity (XXE) issue.” (National Institute of Standards and Technology, n.d.)

**Dependency:** spring-core-5.2.3.RELEASE.jar

**Vulnerability ID:** CVE-2022-22950

Description: “n Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-22060

**Description:** “In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-22096

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-22118

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-5421

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**\*\*This dependency also contains the vulnerability IDs CVE-2022-22968, CVE-2022-22965, and CVE-2016-1000027 (listed above in the spring-boot-2.2.4.RELEASE.jar section)** (National Institute of Standards and Technology, n.d.)

**Dependency:** spring-jcl-5.2.3.RELEASE.jar

**\*\*This dependency contains the vulnerability ID CVE-2022-22950 (listed above in the spring-core-5.2.3.RELEASE.jar section)** (National Institute of Standards and Technology, n.d.)

**Dependency:** logback-core-1.2.3.jar

**Vulnerability ID:** CVE-2021-42550  
**Description:** "In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers." (National Institute of Standards and Technology, n.d.)

**Dependency:** log4j-api-2.12.1.jar

**Vulnerability ID:** CVE-2021-44832

**Description:** “Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-45105

**Description:** “Apache Log4j2 versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3 and 2.3.1) did not protect from uncontrolled recursion from self-referential lookups. This allows an attacker with control over Thread Context Map data to cause a denial of service when a crafted string is interpreted. This issue was fixed in Log4j 2.17.0, 2.12.3, and 2.3.1.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-45046

**Description:** “It was found that the fix to address CVE-2021-44228 in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution in some environments and local code execution in all environments. Log4j 2.16.0 (Java 8) and 2.12.2 (Java 7) fix this issue by removing support for message lookup patterns and disabling JNDI functionality by default.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-44228  
**Description:**  “Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-9488

**Description:** “Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1” (National Institute of Standards and Technology, n.d.)

**Dependency:** snakeyaml-1.25.jar

**Vulnerability ID:** CVE-2017-18640

**Description:** “The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.” (National Institute of Standards and Technology, n.d.)

**Dependency:** jackson-databind-2.10.2.jar

**Vulnerability ID:** CVE-2020-36518

**Description:** “jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-25649

**Description:** "A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.” (National Institute of Standards and Technology, n.d.)

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

**Vulnerability ID:** CVE-2022-29885

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-41079

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-33037

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-30640

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-25329

**Description:** “The fix for CVE-2020-9484 was incomplete. When using Apache Tomcat 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. to 7.0.107 with a configuration edge case that was highly unlikely to be used, the Tomcat instance was still vulnerable to CVE-2020-9494. Note that both the previously published prerequisites for CVE-2020-9484 and the previously published mitigations for CVE-2020-9484 also apply to this issue.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-25122

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2021-24122

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-17527

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-13943

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-13935

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-13934

**Description:** “An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-8022

**Description:** “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. This issue affects: SUSE Enterprise Storage 5 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP4 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 12-SP5 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 15-LTSS tomcat versions prior to 9.0.35-3.57.3. SUSE Linux Enterprise Server for SAP 12-SP2 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 12-SP3 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 15 tomcat versions prior to 9.0.35-3.57.3. SUSE OpenStack Cloud 7 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud 8 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud Crowbar 8 tomcat versions prior to 8.0.53-29.32.1.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-11996

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-9484

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-1938

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2020-1935

**Description:** “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. Such a reverse proxy is considered unlikely.” (National Institute of Standards and Technology, n.d.)

**Vulnerability ID:** CVE-2019-17569

**Description:** “The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading 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. Such a reverse proxy is considered unlikely.” (National Institute of Standards and Technology, n.d.)

**Dependency:** tomcat-embed-websocket-9.0.30.jar

**Vulnerability ID:** CVE-2022-29885

**Description:** “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.” (National Institute of Standards and Technology, n.d.)

**\*\*This dependency also shares all the same vulnerabilities as the dependency known as tomcat-embed-core-9.0.30.jar**

**Dependency:** hibernate-validator-6.0.18.Final.jar

**Vulnerability ID:** CVE-2020-10693

**Description:** “A flaw was found in Hibernate Validator version 6.1.2.Final. 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.” (National Institute of Standards and Technology, n.d.)

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

After review, the security of this program can be ensured with a few changes in the tools used and the code written. Any tools that have unfixed bugs or flaws that allow attackers to breach our security should be removed and replaced with another tool that accomplishes the task. If the flaw or bug can be fixed by using a later, updated version, then that version should be used instead of the one we currently have. For example, the Bouncy Castle JCE Provider currently being used is out of date, and all the issues present are due to that fact. Additionally, the Spring framework is out of date, and should be updated to avoid any flaws from the previous versions. There are some exploits within these tools that are due to the way the program is structured. Tomcat is one of those tools, as there is a flaw which allows attackers to smuggle requests if Tomcat was located behind a reverse proxy. If we restructure the code so Tomcat is not behind a reverse proxy, then this will not be an issue. Also, there are many exploits that exist due to incorrect error handling issues, the lack of proper input validation, and injection risks within the code. If these can be mended, the program will be safe for use by the client and their customers.

**Sources**

IT Governance. (n.d.). *Federal Cybersecurity and Data Privacy Laws Directory*. IT Governance. Retrieved from itgovernanceusa.com/federal-cybersecurity-and-privacy-laws

Marquez, J. (2022). *Vulnerability Assessment Process Flow Diagram* [PDF]. Southern New Hampshire University. Retrieved from <https://learn.snhu.edu/content/enforced/1069785-CS-305-T5607-OL-TRAD-UG.22EW5/course_documents/CS%20305%20Vulnerability%20Assessment%20Process%20Flow%20Diagram.pdf?_&d2lSessionVal=DbWZpf6sxO8xzooN9PCsRPy42&ou=1069785>.

National Institute of Standards and Technology. (n.d.). National Vulnerability Database. National Institute of Standards and Technology. Retrieved from https://nvd.nist.gov/vuln