# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/20/2024** | **Ali ARSLAN** |  |

## 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

[Insert your name here.]

**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?

Given that the company being analyzed is a financial institution and that the data entering and leaving the system pertains to important transactions and sensitive information, it is understood that secure communication is crucial for this organization. It is unknown whether Artemis Financial engages in international transactions, but it is known that investment firms typically have foreign investments or clients conducting transactions from abroad. Considering this information, it is very likely that international transactions occur.

The standards for secure communication are defined by ISO 27701 under ISO 27001. However, unlike many European countries, there is no requirement within the US for companies dealing with sensitive data to have this certification. Nevertheless, European clients and partners may request or prefer companies to have this certification. Additionally, implementing these standards can reduce or eliminate legal liability in the event of a data breach.

Currently and in the future, we can list external threats such as DOS, DDOS, MITM (Man in the Middle - where communication is intercepted and altered), APT (Advanced Persistent Threat), and Supply Chain Attack. Furthermore, some internal threats can also lead to external threats. These include mistakes made by service providers or personnel, either maliciously or accidentally, leaving the system vulnerable to external attacks.

System modernization is critically important for ensuring security. To elaborate, continuously updating open-source libraries to track and close security vulnerabilities or following best practices for the security issues arising from the use of web technologies like WASM (WebAssembly), PWA (Progressive Web Apps), and SPA (Single Page Applications), can minimize risk through regular updates and security checks.

**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.

For the Artemis application, the Level 1 assessment processes that need attention are as follows:

Input Validation: Unrestricted inputs on the software side can lead to numerous security vulnerabilities, from disabling the system to injection attacks (SQL, XSS) that expose sensitive data. Therefore, every piece of data coming from outside the system should be checked, limited, and access should be restricted.

APIs: During the development phase, we use many APIs to assist our work. However, correctly integrating and configuring these APIs, as well as continuously monitoring and updating them against potential security vulnerabilities, are among the best practices for keeping the system secure.

Code Errors: Mistakes in the code can expose sensitive information, allow attackers to gather information about the system, and lead to various security vulnerabilities, including DOS attacks that can disable the system. It is crucial to be cautious of these potential security issues.

Encapsulation: Restricting access to object properties and methods, and limiting access to only those who need it via getter and setter methods, helps to protect the system from external interference and prevents the disclosure of sensitive data. Therefore, always adhering to encapsulation principles is essential for system security.

**3. Manual Review**

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

**Input validations**

GreetingController.java Line 15 ,18

CRUDController.java Line 13,17

Define wrappers around native methods Guideline 5-3

GreetingController.java Line 15,18

CRUDController.java input validation wrapper Line 13,17

customer.java line 12

Check input is valid type or limit them

Validate inputs Guideline 5-1

**Encapsulation**

customer.java Line - 5 :

Limit the extensibility of classes and methods Guideline 4-5

**Client/Server**

DocData.java Line 22,32 : Database connection doesn't close

DOS-2: Release resources in all cases Guideline 1-2

**Code Error**

DocData.java Line 30 Sensitive data record in log files

Purge sensitive information from exceptions Guideline 2-1

**Code Quality**

DocData.java Line 27 hard codded credentials

**APIs**

Spring Framework

Apache Tomcat

Outdated should updated

**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

Upon examining the dependencies that triggered security warnings, it was found that they fall under two main packages:

**Spring Boot 2.2.4**

* Logback
* snakeyaml
* Hibernate
* Spring WebMVC
* spring-context
* spring-expression
* spring-beans
* spring-web

**Apache Tomcat 9.0.30**

* Bouncy Castle

It is believed that updating these two packages(Apache TomCat Spring Boot) will resolve the issue.

**Spring Boot 2.2.4**

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

**Code :** CVE-2022-27772

**Description :** 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

**Code :** CVE-2023-20873

**Description :** 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+.

**Code :** CVE-2023-20883

**Description :** 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.

**Bouncy Castle 1.46**

**bcprov-jdk15on-1.46.jar**

**Code :** 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.

**Code :**CVE-2015-6644

**Description :** Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows attackers to obtain sensitive information via a crafted application, aka internal bug 24106146.

**Code :** CVE-2015-7940

**Description :**The Bouncy Castle Java library before 1.51 does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack."

**Code :** 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.

**Code :** 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.

**Code :** 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.

**Code :** 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.

**Code :** 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.

**Code :**  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.

**Code :** 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.

**Code :** 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.

**Code :** 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.

**Code :** 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."

**Code :** 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.

**Code :** CVE-2020-0187

**Description :** In engineSetMode of BaseBlockCipher.java, there is a possible incorrect cryptographic algorithm chosen due to an incomplete comparison. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-10Android ID: A-148517383

**Code :** CVE-2020-26939

**Description :** In Legion of the Bouncy Castle BC before 1.61 and BC-FJA before 1.0.1.2, attackers can obtain sensitive information about a private exponent because of Observable Differences in Behavior to Error Inputs. This occurs in org.bouncycastle.crypto.encodings.OAEPEncoding. Sending invalid ciphertext that decrypts to a short payload in the OAEP Decoder could result in the throwing of an early exception, potentially leaking some information about the private exponent of the RSA private key performing the encryption.

**Code :** CVE-2023-33201

**Description :** Bouncy Castle For Java before 1.74 is affected by an LDAP injection vulnerability. The vulnerability only affects applications that use an LDAP CertStore from Bouncy Castle to validate X.509 certificates. During the certificate validation process, Bouncy Castle inserts the certificate's Subject Name into an LDAP search filter without any escaping, which leads to an LDAP injection vulnerability.

**Code :** CVE-2024-29857

**Description :** An issue was discovered in ECCurve.java and ECCurve.cs in Bouncy Castle Java (BC Java) before 1.78, BC Java LTS before 2.73.6, BC-FJA before 1.0.2.5, and BC C# .Net before 2.3.1. Importing an EC certificate with crafted F2m parameters can lead to excessive CPU consumption during the evaluation of the curve parameters.

**Code :** CVE-2024-30171

**Description :** An issue was discovered in Bouncy Castle Java TLS API and JSSE Provider before 1.78. Timing-based leakage may occur in RSA based handshakes because of exception processing.

**Code :** CVE-2024-34447

**Description :** bouncycastle - Improper Validation of Certificate with Host Mismatch. The software communicates with a host that provides a certificate, but the software does not properly ensure that the certificate is actually associated with that host.

**logback-core-1.2.3.jar**

**Code :** 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.

**Code :** CVE-2023-6378

Description : A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data.

**log4j-api-2.12.1.jar**

**Code :** 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

**Code :** 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.

**Code :** 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.

Code : 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.

**Code** : 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.

**snakeyaml-1.25.jar**

**Code** : CVE-2017-18640

**Description** : The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

**Code** : CVE-2021-4235

**Description** : Due to unbounded alias chasing, a maliciously crafted YAML file can cause the system to consume significant system resources. If parsing user input, this may be used as a denial of service vector.

**Code** : CVE-2022-1471

**Description** : SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond.

**Code** :CVE-2022-25857

**Description** : The package org.yaml:snakeyaml from 0 and before 1.31 are vulnerable to Denial of Service (DoS) due missing to nested depth limitation for collections.

**Code** : CVE-2022-3064

**Description** : Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory.

**Code** : CVE-2022-38749

**Description** : Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Code** : CVE-2022-38750

**Description** : Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Code** : CVE-2022-38751

**Description** : Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Code** : CVE-2022-38752

**Description** : Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack-overflow.

**Code** : CVE-2022-41854

**Description** :Those using Snakeyaml to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow. This effect may support a denial of service attack.

**jackson-databind-2.10.2.jar**

**Code** : 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.

**Code** : 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.

**Code** : CVE-2021-46877

**Description** : jackson-databind 2.10.x through 2.12.x before 2.12.6 and 2.13.x before 2.13.1 allows attackers to cause a denial of service (2 GB transient heap usage per read) in uncommon situations involving JsonNode JDK serialization.

**Code** : CVE-2022-42003

**Description** : In FasterXML jackson-databind before versions 2.13.4.1 and 2.12.17.1, resource exhaustion can occur because of a lack of a check in primitive value deserializers to avoid deep wrapper array nesting, when the UNWRAP\_SINGLE\_VALUE\_ARRAYS feature is enabled.

**Code** : CVE-2022-42004

**Description** : In FasterXML jackson-databind before 2.13.4, resource exhaustion can occur because of a lack of a check in BeanDeserializer.\_deserializeFromArray to prevent use of deeply nested arrays. An application is vulnerable only with certain customized choices for deserialization.

**Code** : CVE-2023-35116

**Description** : jackson-databind through 2.15.2 allows attackers to cause a denial of service or other unspecified impact via a crafted object that uses cyclic dependencies. NOTE: the vendor's perspective is that this is not a valid vulnerability report, because the steps of constructing a cyclic data structure and trying to serialize it cannot be achieved by an external attacker.

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

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : CVE-2022-42252

**Description** : If Apache Tomcat 8.5.0 to 8.5.82, 9.0.0-M1 to 9.0.67, 10.0.0-M1 to 10.0.26 or 10.1.0-M1 to 10.1.0 was configured to ignore invalid HTTP headers via setting rejectIllegalHeader to false (the default for 8.5.x only), Tomcat did not reject a request containing an invalid Content-Length header making a request smuggling attack possible if Tomcat was located behind a reverse proxy that also failed to reject the request with the invalid header.

**Code** : CVE-2023-44487

**Description** : 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.

**Code** : CVE-2023-46589

**Description** : 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.

**Code** : 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.

**Code** : 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

**Code** : CVE-2020-9484

**Description** : CVE-2020-9484 and the previously published mitigations for CVE-2020-9484 also apply to this issue.

**Code** : 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.

**Code** : CVE-2022-34305

**Description** : 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.

**Code** : CVE-2023-41080

**Description** : 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.

**Code** : 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.

**Code** : 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.

**Code** : CVE-2023-42795

**Description** : Incomplete Cleanup vulnerability in Apache Tomcat.When recycling various internal objects in Apache Tomcat from 11.0.0-M1 through 11.0.0-M11, from 10.1.0-M1 through 10.1.13, from 9.0.0-M1 through 9.0.80 and from 8.5.0 through 8.5.93, an error could

cause Tomcat to skip some parts of the recycling process leading to

information leaking from the current request/response to the next.

**Code** : CVE-2023-45648

**Description** : Improper Input Validation vulnerability in Apache Tomcat.Tomcat from 11.0.0-M1 through 11.0.0-M11, from 10.1.0-M1 through 10.1.13, from 9.0.0-M1 through 9.0.81 and from 8.5.0 through 8.5.93 did not correctly parse HTTP trailer headers. A specially

crafted, invalid trailer header could cause Tomcat to treat a single

request as multiple requests leading to the possibility of request

smuggling when behind a reverse proxy.

**Code** : CVE-2024-21733

**Description** : 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.

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** : CVE-2023-28708

**Description** : 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.

**Code** : CVE-2021-43980

**Description** : 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.

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

**Code** :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.

**spring-web-5.2.3.RELEASE.jar**

**Code** : 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.

**Code** : 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.

**Code** : 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.

**Code** :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. Sonatype's research suggests that this CVE's details differ from those defined at NVD. See https://ossindex.sonatype.org/vulnerability/CVE-2021-22118 for details

**Code** : CVE-2024-22243

**Description** :Applications that use UriComponentsBuilder to parse an externally provided URL (e.g. through a query parameter) AND perform validation checks on the host of the parsed URL may be vulnerable to a open redirect https://cwe.mitre.org/data/definitions/601.html attack or to a SSRF attack if the URL is used after passing validation checks.

**Code** : CVE-2024-22262

**Description** :Applications that use UriComponentsBuilder to parse an externally provided URL (e.g. through a query parameter) AND perform validation checks on the host of the parsed URL may be vulnerable to a open redirect https://cwe.mitre.org/data/definitions/601.html attack or to a SSRF attack if the URL is used after passing validation checks. This is the same as CVE-2024-22259 https://spring.io/security/cve-2024-22259 and CVE-2024-22243 https://spring.io/security/cve-2024-22243 , but with different input.

**spring-beans-5.2.3.RELEASE.jar**

**Code** : 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.

**spring-webmvc-5.2.3.RELEASE.jar**

**Code** :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.

**spring-context-5.2.3.RELEASE.jar**

**Code** : 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.

**spring-expression-5.2.3.RELEASE.jar**

**Code** : 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.

**Code** : CVE-2023-20861

**Description** : In Spring Framework versions 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.

**Code** : CVE-2023-20863

**Description** : 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.

**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.

Upon examining the Artemis Financial system, it was observed that the system is not suitable from a security standpoint and the following issues were identified:

* Outdated APIs
* Unencrypted data transfers
* Unfiltered inputs
* Code quality issues
* Encapsulation issues
* Client-server relation issues
* Code errors

The steps to resolve the issue are as follows:

* Upgrade to Spring Boot 3.4.0 and Apache Tomcat 10.1.26 versions.
* Configure HTTPS settings and certificates to ensure data exchanges are conducted over a secure protocol.
* Implement filtering and validation for all incoming data.
* Ensure that all resources opened in the code are released once the process is completed.
* Avoid logging sensitive data and sending it to the buffer as an error.
* Avoid hardcoding sensitive data within the code.

Implementing these steps will reduce the system's security risks.