****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[CS 305 Project One 1](#_Toc98515532)

[Document Revision History 3](#_Toc98515533)

[Client 3](#_Toc98515534)

[Developer 4](#_Toc98515535)

[1. Interpreting Client Needs 4](#_Toc98515536)

[2. Areas of Security 4](#_Toc98515537)

[3. Manual Review 5](#_Toc98515538)

[4. Static Testing 6](#_Toc98515539)

[5. Mitigation Plan 12](#_Toc98515540)

[6. References 13](#_Toc98515541)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/15/2022** | **Paul Kenaga** | **Initial draft** |

## Client



## Developer

Paul Kenaga

## Interpreting Client Needs

The value of secure communications to Artemis Financial is very significant because of the highly sensitive data that it is responsible for. Security is a crucial element to the success of its custom software. They seek to implement the most current and effective software security to their web-based software application to protect their organization from external threats and modernize its operations.

As a financial consulting company that develops individualized financial plans for savings, retirement, investments, and insurance for their patrons, the production of international transactions is an aspect of the client needs that must be considered. Patrons may live outside of the country and international investment opportunities can provide portfolio growth and diversification.

The sensitive data that Artemis Financial is responsible for requires their company to comply with multiple governmental restrictions to ensure secure communication. These include the Sarbanes-Oxley Act (SOX), Gramm-Leach-Bliley Act (GLBA), General Data Protection Regulation (GDPR), and California Consumer Privacy Act. The Sarbanes-Oxley Act applies great pressure on the company to provide accurate financial reports and to verify the accuracy within their reporting. The Gramm-Leach-Bliley Act, General Data Protection Regulation, and California Consumer Privacy Act require the company to secure the collection, storage, and use of its patrons’ data, in addition to communication protocols (Mehta, 2021).

The external threats that might be present now and in the immediate future for the client’s web-based software application are injection attacks, man-in-the-middle attacks, cross-site request forgery, clickjacking, and cross-site scripting. These attacks are common and should be planned for.

The modernization requirements that must be considered are relative to updating Artemis Financials’ RESTful web application programming interface (API), including libraries and dependencies, to the most recent and supported version. By modernizing the internal structure of the application, the company can mitigate current and future software security concerns while improving efficiency (Takyar, n.d.). It is critical for the success of this project to implement and apply the most current and effective software security as part of the modernization requirements.

## 2. Areas of Security

The areas of security that are applicable to Artemis Financials’ software application are input validation, cryptography, client / server, and code quality. Ensuring secure input and representations through input validation is relevant because patrons of the company exchange data with the software application. Input validation helps secure the application’s functionality for malfunctions and from attacks like cross-site scripting and Structured Query Language (SQL) injection (Input Validation Cheat Sheet, n.d.).

Encryption use and vulnerabilities from cryptography is an applicable area of security because of the sensitive data that the company handles and the regulations and laws that must be followed. Encryption can be added at the application and database levels to improve security protection from external threats. Sensitive data should be encrypted in transit and at rest. It is also important that the proper cryptographic algorithm is used (Cryptographic Storage Cheat Sheet, n.d.).

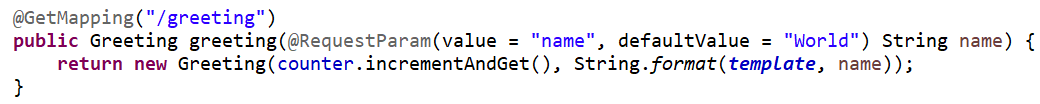
Client-server is applicable to the software application because of its RESTful web API. It is important to establish secure connections within the distributed system because both the client and server sides are vulnerable to external threats in this model (Distributed computing for efficient digital infrastructures, 2013). This places an emphasis on the importance of authentication because a compromised client can be detrimental.

Code quality through secure coding practices and patterns is an important area of security for Artemis Financials’ software application because the goal of this project is to identify any security vulnerabilities in their current software. High code quality is essential to support the defense against attackers that seek to exploit flaws that may exist within the application’s code.

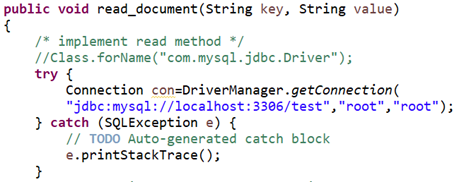
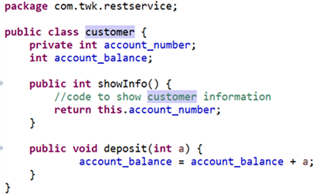
## 3. Manual Review

Data access and APIs are major vulnerabilities that exist within the code base of Artemis Financials’ software application because of an absence of input validation, authentication, and authorization . Within the CRUDController and GreetingController classes, input requested from the client is not validated before it is sent to the server nor is a user validated.

Text

Description automatically generated**CrudController.Java****GreetingController.Java**

An attacker can abuse these flaws by entering malicious code that can compromise sensitive data, lead to injection attacks, and harm the software application’s functionality. Within the DocData class, improper exception handling by using an exception stack trace leads to the exposure of sensitive data within a user’s browser. The server error message can reveal too much information when the SQLException is caught.

**DocData.Java** ** customer.Java**

The CRUD, customer, Greeting, and myDateTime classes are vulnerable to outside threats because there is no defense stopping attackers from accessing their variables and methods. The biggest threat is within the customer class where information regarding patrons’ account information can be disclosed or manipulated. The software application is not configured properly for authentication and authorization. The dependencies within the pom.xml file do not include spring security which offers basic role-based access control capabilities. Additionally, there are incomplete functions contained in the myDateTime class that should be completed to improve code quality.

**A screenshot of a computer

Description automatically generated with medium confidence pom.xml**  **Text

Description automatically generated with medium confidence myDateTime.Java**

## 4. Static Testing

Graphical user interface, text, application

Description automatically generated

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

**Published Vulnerabilities**:

* CVE-2013-1624 – Versions of Bouncy Castle Java library before 1.48 are vulnerable to cryptographic issues from remote attackers because the trusted layer security (TLS) implementation does not always consider timing side-channel attacks. The solution to this vulnerability is to update the Bouncy Castle library to the newest version, 1.70.
  + This vulnerability was identified on a noncompliant MAC check operation during the processing of malformed Cipher-Block Chaining (CBC) padding.
  + Attribution: <https://nvd.nist.gov/vuln/detail/CVE-2013-1624>
* CVE-2015-7940 – Versions of Bouncy Castle Java library before 1.51 are vulnerable to invalid curve attacks that threaten private keys used in elliptic curve cryptography. The solution to this vulnerability is to upgrade to bouncycastle 1.51 or later.
  + This vulnerability was identified through improper validation within an elliptic curve while exchanging keys.
  + Attribution: https://bugzilla.redhat.com/show\_bug.cgi?id=1276272
* CVE-2016-1000338 –The Digital Signature Algorithm (DSA) in versions of Bouncy Castle Java Cryptography Extension (JCE) Provider before and including 1.55 does not completely validate ASN.1 encoding of signature which could cause invisible data to be introduced. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified by a flaw within the DSA.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000339 – In versions of Bouncy Castle JCE Provider before and including 1.55 are vulnerable to exposure and modification of sensitive data because of leaks within the AESEngine and the algorithm used. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified while monitoring a CPU data channel.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000341 – The DSA signature generation in versions of Bouncy Castle JCE Provider before and including 1.55 are vulnerable to timing attacks because of a lack of binding. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified because of the lack of binding during DSA signature generation.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000342 – The Elliptic Curve Digital Signature Algorithm (ECDSA) causes versions of Bouncy Castle JCE Provider before and including 1.55 to be vulnerable to the introduction of invisible data because it does not fully validate ASN.1 encoding of signature on verification. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified by the possibility of injection of extra elements within a validated signature.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000343 – The DSA key pair generator in versions of Bouncy Castle JCE Provider before and including 1.55 does not generate a strong private key if default values are used which creates a vulnerability of degradation to the quality of data. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified through the initialization of the Java Cryptography Architecture key pair generator.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000344 – The Diffie-Hellman Integrated Encryption Scheme (DHIES) implementation causes Bouncy Castle JCE Provider before and including 1.55 to be vulnerable to data leaks because of the ability to use electronic codebook (ECB) mode. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified through the use of ECB mode which has been removed from the provider.
  + Attribution: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-1000344
* CVE-2016-1000345 – In versions of Bouncy Castle 1.55 and older, the cipher block chaining mode created a vulnerability to padding oracle attacks. The solution to this vulnerability is to update the Bouncy Castle to a later version.
  + This vulnerability was identified because of failures in decryption due to padding.
  + Attributes: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000346 – Public Diffie-Hellman (DH) keys in versions of Bouncy Castle JCE Provider before and including 1.55 are not fully validated which creates vulnerabilities with data security. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified by issues caused where static Diffie-Hellman is in use.
  + Attribution: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html
* CVE-2016-1000352 – The Elliptic Curve Integrated Encryption Scheme implementation in versions of Bouncy Castle JCE Provider before and including 1.55 allow the use of ECB mode which is unsafe due to cryptographic issues and vulnerabilities with the disclosure of sensitive information. The solution to this vulnerability is to update the Bouncy Castle JCE Provider to a later version.
  + This vulnerability was identified by the use of use of ECB mode which is know to be unsafe.
  + Attribution: https://nvd.nist.gov/vuln/detail/CVE-2016-1000352
* CVE-2017-13098 – The TLS implementation in Bouncy Castle versions before 1.0.3 are vulnerable to ROBOT attacks because of weak Bleichenbacher oracle that could allow the decryption of TLS traffic. The solution to this vulnerability is to update Bouncy Castle to a later version.
  + This vulnerability was identified when TLS was configured to use the JCE for cryptographic functions.
  + Attribution: https://nvd.nist.gov/vuln/detail/CVE-2017-13098
* CVE-2018-1000613 – Versions of bouncy Castle before 1.60 use externally-controlled input to select classes (CWE-470) which causes a vulnerability which leads to the deserialization of private keys. The solution to this vulnerability is to update Bouncy Castle to 1.60 or later.
  + This vulnerability was identified through CWE-470.
  + Attribution: https://nvd.nist.gov/vuln/detail/CVE-2018-1000613
* CVE-2018-5382 – Versions of Bouncy Castle 1.46 and earlier are vulnerable to brute force cracking because of insufficient protection that allows attackers to bypass BKS integrity checking. The solution to this vulnerability is to update the Bouncy Castle to a later version.
  + This vulnerability was identified within a BKS file that was created as the "BKS-V1" format.
  + Attribution: https://www.kb.cert.org/vuls/id/306792
* CVE-2020-15522 – Versions of Bouncy Castle BC Java before 1.66 have a timing issue that can expose private key information. The solution to this vulnerability is to update Bouncy Castle to a later version.
  + This vulnerability was identified within the EC math library.
  + Attribution: https://nvd.nist.gov/vuln/detail/CVE-2020-15522
* CVE-2020-26939 – Versions of Legion of the Bouncy Castle BC before 1.61 are vulnerable to sensitive data exposure because of flaws within error inputs. The solution to this vulnerability is to update Bouncy Castle to a later version.
  + This vulnerability was identified in org.bouncycastle.crypto.encodings.OAEPEncoding when a short payload is decrypted by invalid ciphertext.
  + Attribution: https://nvd.nist.gov/vuln/detail/CVE-2020-26939

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

**Published Vulnerability**:

* CVE-2020-10693 **–** Input sanitation can be bypassed because of a bug in the message interpolation processor that allowed invalid expression language (EL) expressions to be evaluated as valid. The solution to this vulnerability is to update the hibernate-validator to hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, or hibernate-validator 6.0.20.Final.
  + This vulnerability was identified by a bug in the message interpolation processor.
  + Attribution: https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693

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

**Published Vulnerability**:

* CVE-2020-25649 **–** A bug in FasterXML Jackson Databind impacted security of entity expansion that make data integrity vulnerable to Extensible Markup Language (XML) external entity (XXE) attacks. The recommended solution is to update the FasterXML Jackson Databind to jackson-databind-2.11.0.
  + This vulnerability was identified through setExpandEntityReferences(false) within the DOMDeserializer class.
  + Attribution: <https://github.com/FasterXML/jackson-databind/blob/master/src/main/java/com/fasterxml/jackson/databind/ext/DOMDeserializer.java#L30>

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

**Published Vulnerability**:

* CVE-2020-9488 **–** Incorrect validation within Apache Log4j SMTP appender could allow man-in-the-middle attacks through the interception of a Simple Mail Transfer Protocol Secure (SMTPS) connection causing log message leaks. The solution to this vulnerability is to upgrade the SMTP appender to 2.13.2.
  + This vulnerability was identified when Apache Log4j SMTP appender did not confirm that the SMTPS connection’s SSL/TLS certificate matched with the host name.
  + Attribution: https://issues.apache.org/jira/browse/LOG4J2-2819

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

**Published Vulnerability**:

* CVE-2021-42550 **–** Logback versions before and including 1.2.7 are vulnerable to attacks that could execute arbitrary code loaded from lightweight directory access protocol (LDAP) servers by manipulating configuration files. The solution to this vulnerability is to upgrade to logback version 1.2.9 and to set configuration files as read-only.
  + This vulnerability was identified when it was discovered that attackers with write access could manipulate configuration files.
  + Attribution: <https://logback.qos.ch/news.html>

**Dependency**: snakeyaml-1.25.jar

**Published Vulnerability**:

* CVE-2017-18640 **–** Some versions of SnakeYAML allow for entity expansion through one of the features used during a load operation. The solution to this vulnerability is to update to version 1.26.
  + This vulnerability was identified during a load operation when the Alias feature allowed entity expansion. This can lead to denial-of-service (DOS).
  + Attribution: https://bugzilla.redhat.com/show\_bug.cgi?id=1785377

**Dependencies:** spring-aop-5.2.3.RELEASE.jar, spring-core-5.2.3.RELEASE.jar

**Published Vulnerability**:

* CVE-2020-5421 **–** Certain Spring Framework versions lack proper protection from Reflected File Download (RFD) attacks through the use of a jsessionid path parameter. The solution to this vulnerability is to update the Spring Framework to version greater than 5.2.9.
  + This vulnerability was identified through the lack of protection from RFD attacks that came from another CVE, CVE-2015-5211.
  + Attribution: <https://tanzu.vmware.com/security/cve-2020-5421>

**Dependencies**: tomcat-embed-core-9.0.30.jar, tomcat-embed-websocket-9.0.30.jar

**Published Vulnerabilities**:

* CVE-2019-17569– Some versions of Apache Tomcat introduced a regression that created the possibility of Hypertext Transfer Protocol (HTTP) Request Smuggling. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.31 or later.
  + This vulnerability was identified as a result of the regression and processing of invalid Transfer-Encoding headers.
  + Attribution: https://lists.apache.org/thread/shxw4wz09kkq3tnbowxqo445gjhy9o4s
* CVE-2020-11996 – Some versions of Apache Tomcat could cause servers to become unresponsive if high CPU usage is triggered by HTTP/2 requests made on concurrent HTTP/2 connections. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.36 or later.
  + This vulnerability was identified through HTTP/2 requests sent to Apache Tomcat.
  + Attribution: https://lists.apache.org/thread/sqyqm0nn7g26bbsd2rm0g4sc3woo32mc
* CVE-2020-13934 **–** Some versions of Apache Tomcat are vulnerable to a denial of service due to an OutOfMemoryException from requests for h2c, HTTP/2 but without TLS, direct connection. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.37 or later.
  + This vulnerability was identified when the HTTP/1.1 processor was not released by an h2c direct connection after the upgrade to HTTP/2.
  + Attribution: https://lists.apache.org/thread/0n4tnnrc1wf9l9c3k834tzl76kpjdc8m
* CVE-2020-13935 **–** Some versions of Apache Tomcat are vulnerable to a denial of service caused by infinite loops from invalid payload lengths. The payload length is not being properly validated. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.37 or later.
  + This vulnerability was identified through the improper validation of the payload length in a WebSocket frame.
  + Attribution: https://lists.apache.org/thread/r7m5zthg1k9grytzqz0cwlnfb7wjfonz
* CVE-2020-13943 **–** Some versions of Apache Tomcat are vulnerable to having users see responses for unexpected resources if an HTTP/2 client exceeds the number of concurrent streams for a connection. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.38 or later.
  + This vulnerability was identified as a result of HTTP/2 protocol violation.
  + Attribution: https://lists.apache.org/thread/dkhymygyk9nn6lwpt52fq6pqn60p2d3m
* CVE-2020-17527 –Some versions of Apache Tomcat are vulnerable to data leaks if an HTTP request header value from a previous stream is reused for the request associated with the next stream. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.40 or later.
  + This vulnerability was identified by the Apache Tomcat Security Team while investigating a bug.
  + Attribution: https://lists.apache.org/thread/0nl7j4lr5j6jvwh45d2lhtjysqvnwnqx
* CVE-2020-1935 –Some versions of Apache Tomcat are vulnerable to HTTP Request Smuggling because of a faulty HTTP header parsing code approach that allowed considered some invalid HTTP headers to be parsed as valid. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.31 or later.
  + This vulnerability was identified by invalid HTTP headers being parsed as valid by HTTP header parsing code.
  + Attribution: https://lists.apache.org/thread/rjh93xslbk6q32kxlrq8loj0ggzkjzjb
* CVE-2020-1938 – Some versions of Apache Tomcat treat Apache JServ Protocol (AJP) connections with higher trust than HTTP connections that could lead to remote code execution attacks if an AJP port is accessible to untrusted users. The solutions this vulnerability is to upgrade to Apache Tomcat 9.0.31 or later, configure network firewall rules, configure a shared secret for the AJP connection, and configure an explicit address attribute to the connector.
  + This vulnerability was identified through incoming connections to Apache Tomcat.
  + Attribution: https://lists.apache.org/thread/bnys5lvg1875dsslkkx2vmwxv833l35x
* CVE-2020-9484 – Under certain conditions, some versions of Apache Tomcat are vulnerable to remote code execution by deserialization of a file under the control of an attacker. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.43 or later.
  + This vulnerability was identified through configuration issues with the PersistenceManager and the possibility of unauthorized file control.
  + Attribution: https://lists.apache.org/thread/jgqg9ftb9zwpd0kt7mxb5sj0hfnd4xym
* CVE-2021-24122 – Some versions of Apache Tomcat are vulnerable to the disclosure of Java Server Pages (JSP) source code caused by inconsistent behavior of the Windows API. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.40 or later.
  + This vulnerability was identified while using the New Technology File System (NTFS) file system.
  + Attribution: https://lists.apache.org/thread/1ydr25ylpw9rq4v29l4jtvkk1pb9y9n1
* CVE-2021-25122 – Some versions of Apache Tomcat are vulnerable to data leaks caused by duplication from h2c connection requests. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.43 or later.
  + This vulnerability was identified when responding to new h2c connection requests.
  + Attribution: https://lists.apache.org/thread/7vykb4mjgvj7lvzng7n9b8z40b768c29
* CVE-2021-25329 – Some versions of Apache Tomcat are vulnerable because the fix for CVE-2020-9484 was incomplete. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.43 or later.
  + This vulnerability was identified when using a highly unlikely configuration edge case.
  + Attribution: https://lists.apache.org/thread/hd79q54gvvrc6p7yvh8bofkygkvlfzy3
* CVE-2021-30640 – Some versions of Apache Tomcat allow attackers to authenticate using variations of a valid username and/or bypass protection from LockOut Realm. A solution to this vulnerability is to upgrade to Apache Tomcat 9.0.46 or later.
  + This vulnerability was identified by escape issues with parameters from queries made by the Java Naming and Directory Interface (JNDI) Realm.
  + Attribution: https://lists.apache.org/thread/3qcmnprbtft3phn3vyk06hpycm7p4sgx
* CVE-2021-33037 – Some versions of Apache Tomcat are vulnerable to request smuggling because, under some circumstances, they do not correctly parse the HTTP transfer-encoding request header. The solution to this vulnerability is to Upgrade to Apache Tomcat 9.0.48 or later.
  + This vulnerability was identified when the transfer-encoding header was wrongly ignored.
  + Attribution: https://lists.apache.org/thread/kovg1bft77xo34ksrcskh5nl50p69962
* CVE-2021-41079 – Some versions of Apache Tomcat are vulnerable to denial of service because they do not properly validate incoming TLS packets that could trigger an infinite loop. The solution to this vulnerability is to upgrade to Apache Tomcat 9.0.44 or later.
  + This vulnerability was identified when Tomcat was configured with certain non-blocking I/O (NIO) +OpenSSL or NIO2+OpenSSL for TLS.
  + Attribution: https://lists.apache.org/thread/wkvrs06sw7ct014npfpcrvjjpf6r1n6b

## 5. Mitigation Plan

The steps to remedy the identified security vulnerabilities can be accomplished by completing the following action list:

* To restrict access to sensitive information in Artemis Financials’ software application, “spring-boot-starter-security” should be added to the dependencies within the pom.xml file. The Spring Boot Security starter will provide the application with security features like authentication, authorization, and cryptography. Authentication can be implemented to identify and track patrons as they access the platform. Authorization will prevent patrons and attackers from accessing private information through role-based access control (RBAC) where the principle of least privilege is enforced. The cryptography module can be used to encrypt passwords and sensitive data in transit. Additionally, Spring Boot Security offers protection against CSRF attacks by using the Synchronizer Token Pattern.
  + Attribution: https://docs.spring.io/spring-security/reference/
* Input validation can be strengthened through blacklist and whitelist validation, length requirements, and query parameterization. These techniques should be used in conjunction with one with another. The blacklist will address known threats while the whitelist and length requirements will define acceptable input. It is also highly recommended to mitigate SQL injection through the use of query parameterization for all queries and all variables, especially within the CRUDController and GreetingController classes.
* The Bouncy Castle dependency is a provider for the Java Cryptography Extension. It is important that this dependency I upgraded to version 1.70 to resolve the many common vulnerabilities and exposures (CVEs) associated with the outdated version being currently used by Artemis Financial.
* The Hibernate Validator dependency should be upgraded to version hibernate-validator 6.2.2.Final. This change will secure input sanitation controls.
* The Jackson – Data Bind dependency is used for the conversion of JavaScript Object Notation (JSON) to Plain Old Java Object (POJO). This dependency needs to be upgraded to the most recent version, 2.13.2.
* The Apache Log4j API dependency should be upgraded to version 2.12.4 to resolve the associated CVE.
* The dependency for the logback-core module should be upgraded to 1.2.11 and logback configuration files should be set to read-only if possible.
* SnakeYAML is a YAML parser and emitter. The dependency used in this software application should upgrade to version 1.30 to mitigate the associated CVE.
* The CVE related to the “spring-aop-5.2.3.RELEASE.jar” and “spring-core-5.2.3.RELEASE.jar” dependencies can be remedied by upgrading the Spring Framework to version 5.3.17. Some false positives from the dependency check were excluded from this report to prevent redundant work.
* To address the CVEs related to the Core Tomcat implementation, the dependencies should be upgraded to one of the more recent versions like 10.0.18.

## 6. References

*Cryptographic Storage Cheat Sheet*. (n.d.). OWASP Cheat Sheet Series.

<https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html>

*Distributed computing for efficient digital infrastructures*. (2013, October 20). Digital Guide IONOS.

<https://www.ionos.com/digitalguide/server/know-how/what-is-distributed-computing/>

*Input Validation Cheat Sheet*. (n.d.). OWASP Cheat Sheet Series.

<https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html>

Mehta, M. (2021, October 22). *5 Important Data Privacy Laws: CCPA, HIPAA, GDPR, GLBA, and LGPD*.

Infosec Insights. <https://sectigostore.com/blog/data-privacy-laws-ccpa-hipaa-gdpr-glba-lgpd/>

Takyar, A. (n.d.). *What is application modernization?* LeewayHertz.

<https://www.leewayhertz.com/application-modernization/>