****

# 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** | **18 March 2022** | **Patrick Pattison** | **Project One Submission** |

## 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

Patrick Pattison

## 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?
  + Due to the company dealing with financial matters, secure communications are paramount.
* Are there any international transactions that the company produces?
  + Considering they want to develop an internet presence, I would assume that international transactions could be a distinct possibility.
* Are there governmental restrictions about secure communications to consider?
  + Absolutely, financial transactions are some of the most regulated there are.
* What external threats might be present now and in the immediate future?
  + Cyber criminals are the ever-presence threat. In addition, you have the threat of library introduced vulnerabilities.
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?
  + The modernization requirements were to add a web presence. These requirements will need the assistance of some open source libraries to handle the web communications. These libraries could be adding vulnerabilities into the system.

## 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.

The following areas of security need to be accounted for:

* Input Validation – Any time a program takes in input, especially over the internet, that input needs to be validated from a security stand point.
* APIs – APIs will be required to process the web communications and care will have to be exercised so as to not introduce more vulnerabilities.
* Cryptography – Not only will internet communications be used, but financial information will be processed. As such, cryptography will be required.
* Code Error – Securely handling errors is always a requirement for all software.
* Code Quality – Much like Code Error, Code Quality should always be a requirement for software.
* Encapsulation – Encapsulation is always a good idea when it makes sense. This allows any vulnerabilities that might be in the software to keep the damage localized and minimized.

## 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 Validation
  + In RestServiceApplication.java – Program args are passed onto Spring Framework with no validation and leaving all validation up to the API.
* Cryptography
  + Does not appear that cryptography is being utilized.
* Code Quality
  + In DocData.java – Database credentials are hard coded into the source code.

## 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

* Bcproxy-jdk15on-1.46.jar
  + CVE-2016-1000352 – Allows the use of ECB mode, which is regarded as unsafe and support for it has since been removed by the provider.
  + CVE-2016-1000346 – Other part Diffie-Hellman (DH) public is not fully validated.
  + CVE-2016-1000345 – 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.
  + CVE-2016-1000344 – The DHIES implementation allowed the use of ECB mode.
  + CVE-2016-1000343 – The DSA key pair generator generates a weak private key if used with default values.
  + CVE-2016-1000342 – ECDSA does not fully validate ASN.1 encoding of signature on verification.
  + CVE-2016-1000341 – 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.
  + CVE-2016-1000339 – 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.
  + CVE-2016-1000338 – The DSA does not fully validate ASN.1 encoding of signature on verification.
  + CVE-2018-5382 – 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.
  + CVE-2017-13098 – When configured to use the Java Cryptography Extension for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated.
  + CVE-2013-1624 – 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.
* Hibernate-validator-6.0.18.Final.jar
  + CVE-2020-10693 – A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid.
* Jackson-databind-2.10.2.jar
  + CVE-2020-36518 –
  + CVE-2020-25649 – This flaw allows vulnerablilites to XML external entity (XXE) attacks.
* Log4j-api-2.12.1.jar
  + CVE-2021-44832 – Vulnerability to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP source URI when an attacker has control of the target LDAP server.
  + CVE-2021-45105 – Vulnerability from uncontrolled recursion from self-referential lookups.
  + CVE-2021-45046 – The fix for CVE-2021-44228 was incomplete and in some non-default configurations an attacker with control over Thread Context Map input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup or a Thread Context Map pattern to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution.
  + CVE-2021-44228 – An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers
  + CVE-2020-9488 – Vulnerability 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
* Logback-core-1.2.3.jar
  + CVE-2021-422550 – An attacker with the required privileges to edit configuration files could craft malicious configuration allowing execution of arbitrary code loaded from LDAP servers.
* Snakeyaml-1.25.jar
  + CVE-2017-18640 – The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation.
* Spring-aop-5.2.3.RELEASE.jar and Spring-core-5.2.3.RELEASE.jar
  + CVE-2020-5421 – Protections against Reflected File Download attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.
  + CVE-2021-22060 – A follow-up to CVE-2021-22096 with additional types of inputs and in more places.
  + CVE-2021-22096 – It is possible for a user to provide malicious input to cause the insertion of additional log entries.
  + CVE-2021-22118 – A WebFlux application is vulnerable to a privilege escalation.
* Tomcat-embed-core-9.0.30.jar and Tomcat-embed-websocket-9.0.30.jar
  + CVE-2021-42340 – The fix for bug 63362 present in Apache Tomcat introduced a memory leak that could, over time, lead to a denial of service via an OutOfMemoryError.
  + CVE-2021-41079 – 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.
  + CVE-2021-33037 – Tomcat does 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.
  + CVE-2021-30640 – Vulnerability in the JNDI Realm of tomcat allows an attacker to authenticate using variations of a valid user name or bypass some of the protection provided by the LockOut Realm.
  + CVE-2021-25329 – The fix for CVE-2020-9484 was incomplete and a highly unlikely configuration edge case is still vulnerable.
  + CVE-2021-25122 – When responding to a new h2c connection request tomcat could duplicate request headers and a limited amount of request body from one request to another, resulting in two users seeing the results of one users request.
  + CVE-2021-24122 – When serving resources from a network location using the NTFS file system, tomcat is susceptible to JSP source code disclosure in some configurations.
  + CVE-2020-17527 – Tomcat could re-use an HTTP request header calue from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream.
  + CVE-2020-13943 – Exceeding the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol) is is possible that a subsequent request made on that connection could contain HTTP headers from a previous request rather than the intended headers.
  + CVE-2020-13935 – Invalid payload lengths of a WebSocket frame could trigger an infinite loop, leading to a denial of service.
  + CVE-2020-13934 – Tomcat did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of these requests were made, an OutOfMemoryException could occur leading to a denial of service.
  + CVE-2020-8022 – Incorrect default permissions in the packaging of tomcat on SUSE Linux.
  + CVE-2020-11996 – Specially crafted sequence of HTTP/2 requests could trigger high CPU usage for several seconds. A sufficient number of such requests could cause the server to become unresponsive.
  + CVE-2020-9484 – If four different conditions are all met an attacker can use a specifically crafted request and trigger a remote code execution via deserialization of the file that is under their control.
  + CVE-2020-1938 – Tomcat treats Apache JServ Protocol (AJP) connections as having higher trust than, for example, a similar HTTP connection.
  + CVE-2020-1935 – The HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid.
  + CVE-2019-17569 – 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.

## 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 experimentation, upgrading two main packages used will eliminate all of the vulnerabilities. Springframework.boot needs to be upgraded to version 2.6.4. Bouncycastle needs to be upgraded to version 1.70.