****

# 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** | **5/22/2022** | **Hillary Loyd** |  |

## 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

Hillary Loyd

## 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?
  + Artemis Financial flows their financial information from their servers to around the world locations, customers, and their vendors. Global Rains mission is “Security is everyone’s responsibility”. If data were to be infiltrated by an outside source, it could lead to any of the companies or patrons that Artemis Financial is working with to be stolen. Making our mission statement more valuable then ever. Keeping communications secure as well as the safety measures being taken this protects the information.
* Are there any international transactions that the company produces?
  + It is possible that Artemis Financial would work with patrons all over the world. However, with the idea that the company would stay in the US their patrons might work with others outside the country.
* Are there governmental restrictions about secure communications to consider?
  + There would be governmental restrictions when it comes to secure communications. There are many acts that are in place to help with this. Sarbanes-Oxley Act requires audits to be completed to ensure data is stored and managed in a secure manner. This is one of the many acts that are in place.
* What external threats might be present now and in the immediate future?
  + Some of the external threats that are or may become present are Malware, Phishing, Ransomware, DDos Attacks or other forms of hacking.
* What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies?
  + Modernization is very important for protecting data. Using modern techniques such as two-factor authentication can help protect private user data from threats. Newer versions of libraries may include some security patches that could also help.

[Include your findings here.]

## 2. Areas of Security

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

Input Validation – users will need to input login information or other potentially vital information. This information should be validated to prevent vulnerabilities.

APIs – Since this is a RESTful application, the REST API should be checked for potential vulnerabilities.

Cryptography – It is a requirement for financial data to be stored by financial institutions, so a review should be done on the methods for encryption.

Code Error – All code should be check for errors that would impact security

Code Quality – All code should be checked to see if secure practices and patterns are being utilized.

Encapsulation – The code contains data structures. These should be reviewed to ensure they are secured.

## 3. Manual Review

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

Data Access – In the DocData.java file the data access method is to access the data that involves the description of the location for the data in which is the username and password, this could be a potential for possible vulnerabilities. The root is used for the username and password, making the root password easy to figure out by simple deduction. This also makes it easier for unauthorized users to gain access.

Direct object reference – In the CRUDController.java file, there is a vulnerability where the application could be exposing the inside of objects and the retrieval and passing through a sequence of code interjection. Business\_name is passed through the CRUD method it is exposing the DocData object access vulnerability.

## 4. Static Testing

Run a dependency check on Artemis Financials’ 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

CVE-2018-1000613 - Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.60.

CVE-2015-6644 - An information disclosure vulnerability in Bouncy Castle could enable a local malicious application to gain access to user’s private information. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.49

CVE-2016-1000338 - 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 Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000342 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000343 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000344 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000352 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000341 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2016-1000345 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2017-13098 - 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." Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.60.

CVE-2020-15522 - Bouncy Castle BC Java before 1.66, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 have a timing issue within the EC math library that can expose information about the private key when an attacker is able to observe timing information for the generation of multiple deterministic ECDSA signatures. Recommended to upgrade to bcprov-jdk15on-1.46.jar to version 1.67.

CVE-2016-1000339 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

CVE-2020-26939 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.61.

CVE-2015-7940 - 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.". Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.51.

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. 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. Recommended to upgrade to bcprov-jdk15on-1.46.jar to version 1.47.

CVE-2013-1624 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.48.

CVE-2016-1000346 - 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. Recommended to upgrade bcprov-jdk15on-1.46.jar to version 1.56.

## 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 Financials’ software application.

1. The first step if the problem of data access username and password. This can be resolved by creating a strong combination of alphanumeric characters for both the username and the password.
2. The code review and modification by integrating secure coding practices. This should help the quality of code for having proper authentication and error catching.
3. . Upgrade bcprov-jdk15on-1.46.jar to version 1.67 (the highest required by the vulnerabilities discovered).