****

# 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** | **12 Jul 2021** | **J.F.Kiddy** | **Initial document** |

## 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

J.F. Kiddy

## 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?**

Securing all transactions and communications are of paramount importance to any business, but especially for a financial institution. We have been assigned the task of examining the vulnerabilities of Artemis Financials' web-based software application. So, during this assessment, we need to ensure the protection of the customer/client data and Atriums Financials’ information as well as their intellectual property. Also, since the Artemis Financials’ web-based software is a financial consulting company that develops individualized financial plans, it is important to secure every single communication that involves customer's account. (Username, password, bank account numbers, and other PPI) Are there any international transactions that the company produces?

* **Are there any international transactions that the company produces?**

Artemis Financials’ does have international transactions due to it being a web-based company. These transactions are consisted of the buying and selling of stock to make money for client’s savings, retirement, and investments. There are the customers personal transactions to consider as well, as customers could travel to various countries and would need to access their accounts overseas. However, she should work with a legal team to ensure that there are no loopholes which can compromise Artemis Financials' ability to conduct international transactions via the internet.

* **Are there governmental restrictions about secure communications to consider?**

There are numerous laws and regulations that cover Internet, data security, and privacy in the United States that should be considered. One is the Privacy Act of 1974, arguably this being the foundation for it all security requirements now. The Privacy Act of 1974 was established to ensure control over the collection, maintenance, use, and dissemination of personal information by agencies in the U.S. government, however it has been adapted and incorporated into many businesses. We may need to consider abiding by such as the Fair Credit Reporting Act, Fair Debt Collection Practices Act, Electronic Communications Privacy Act, Cyber Intelligence Sharing and Protection Act and should any transactions extend into the European Union, there is the General Data Protection Regulation. However, she should work with a legal team to ensure that there are no security loopholes which can compromise the confidentiality of the sensitive customer’s information.

* **What external threats might be present now and in the immediate future?**

Threats now and in the future can, include rival companies, foreign governments, foreign government funded hacker groups and anonymous hacker groups. With this information, security is extremely important for the company and the protection of its users. The current threats we are facing now can include Authentication threats: The access of the details to the various accounts in the system through unauthorized access. Man-in-the-middle threats: The compromise of the communication between the application and the client. Denial of service threats: The probing of the application with numerous traffic that can clog and make the application to crash. The threats in the immediate future include some that we are starting to see already. Ransomware: The unauthorized use of information on a secure system which is then used for blackmail. Blockchain Hacking and Cryptojacking: The unauthorized use of a computer to mine cryptocurrency via in inserted cryptomining code. AI-Based Hacking: Using corrupted A.I that can think in various ways to by pass security features.

* **What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies?**

Adding extra security will always be a constant battle as new threats emerge. Upgrading both code and system infrastructure to combat new and evolving threats, this alone will always require modernization. To this end, it could include advancements in making a web application that works in a sand-boxed mode, a testing environment that isolates untested code changes and outright experimentation from the production environment to ensure greater security. We could also take advantage of the open-source library to adding further layers of protection and functionality to Artemis Financials’ application. We also need to ensure monoliths are not created, doing due diligence on the framework and the tools and ensuring that we abstract the dependencies. Lastly, one future modernization requirement for this system could include making the company completely web based.

## 2. Areas of Security

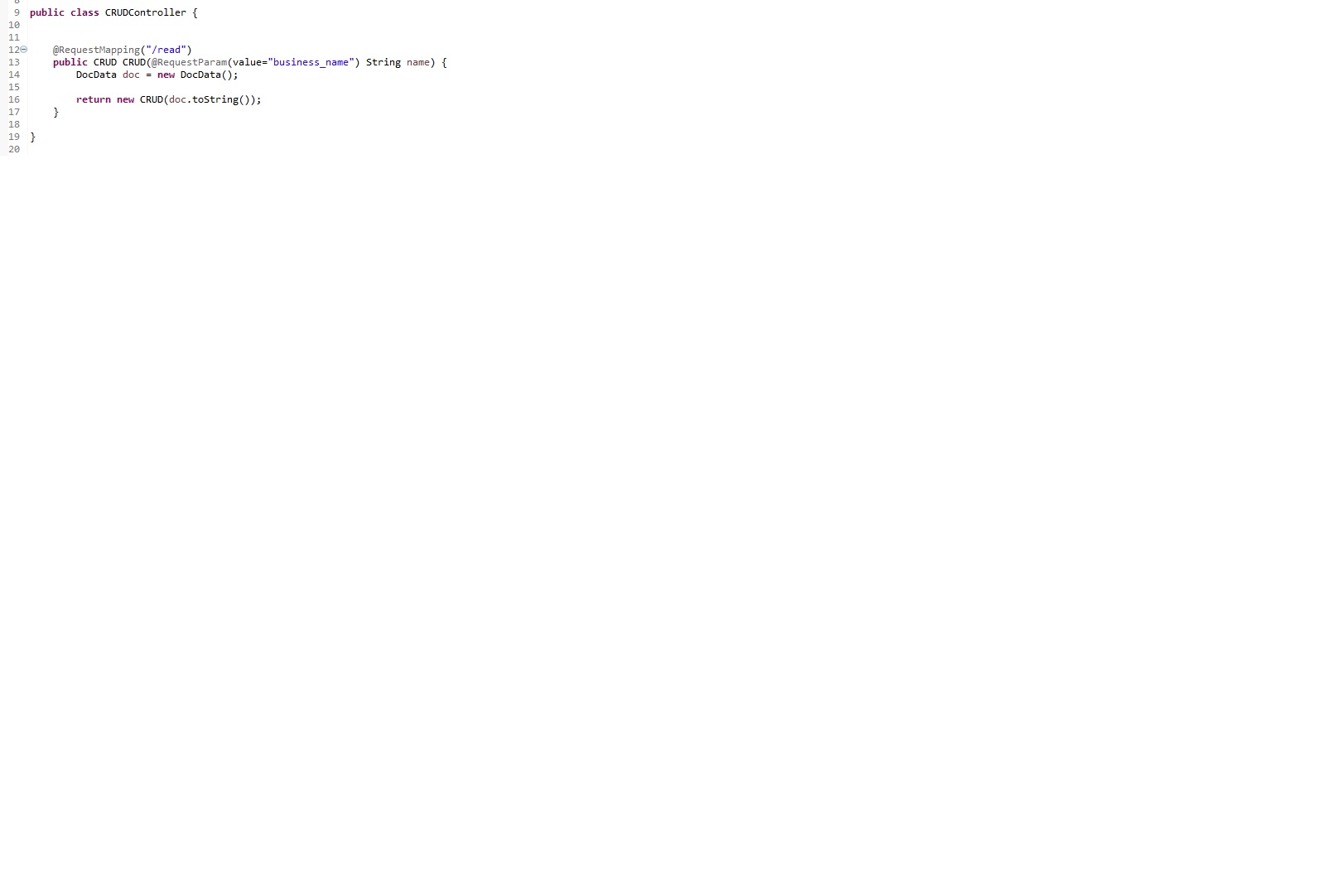
**Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.**

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. API: This needs to be a secured configuration to ensure that there is no compromise of the security between the system and the API connection. Cryptography: Lack of proper encryption can result to a compromise of the integrity and confidentiality of the information when it is transmitted over the internet. Client/server: Secure communication between the customer (client) and the backend of the web application (server) for financial transactions. Secure coding: Good code sanitization, such as exception handling and error checks, maintain the consistency in the business logic of the application.

## 3. Manual Review

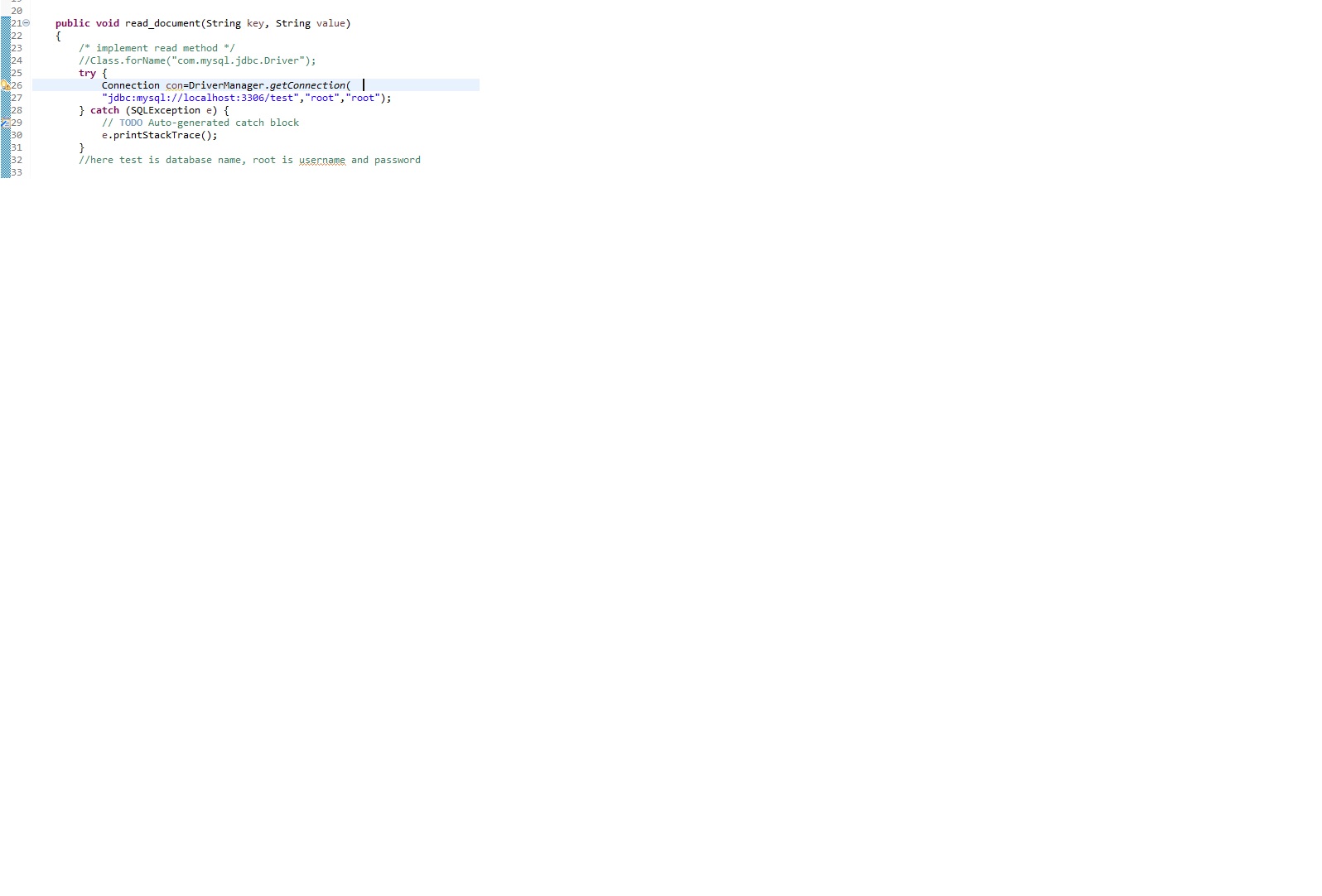
**Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.**

After performing a manual review of the code two vulnerabilities stand out immediately. First is an Encapsulation/controller issue, in the CRUDController.java has a Direct Object Reference that can possibly result in the application exposing internal objects. Theoretically they could be passed and accessed through injections of code, the Value=”business\_name” (See below) passed in the CRUD method.



## 

## Second is a data access vulnerability. In the DocData.java code the data access method that involves the definition of the location of the database. The root password could be easily guessed or compromised through Brute-force attacks however, the password and username are given, given plain English. The Value=” business\_name” is passed in the CRUD method which may expose the DocData object database access vulnerability. (See below)



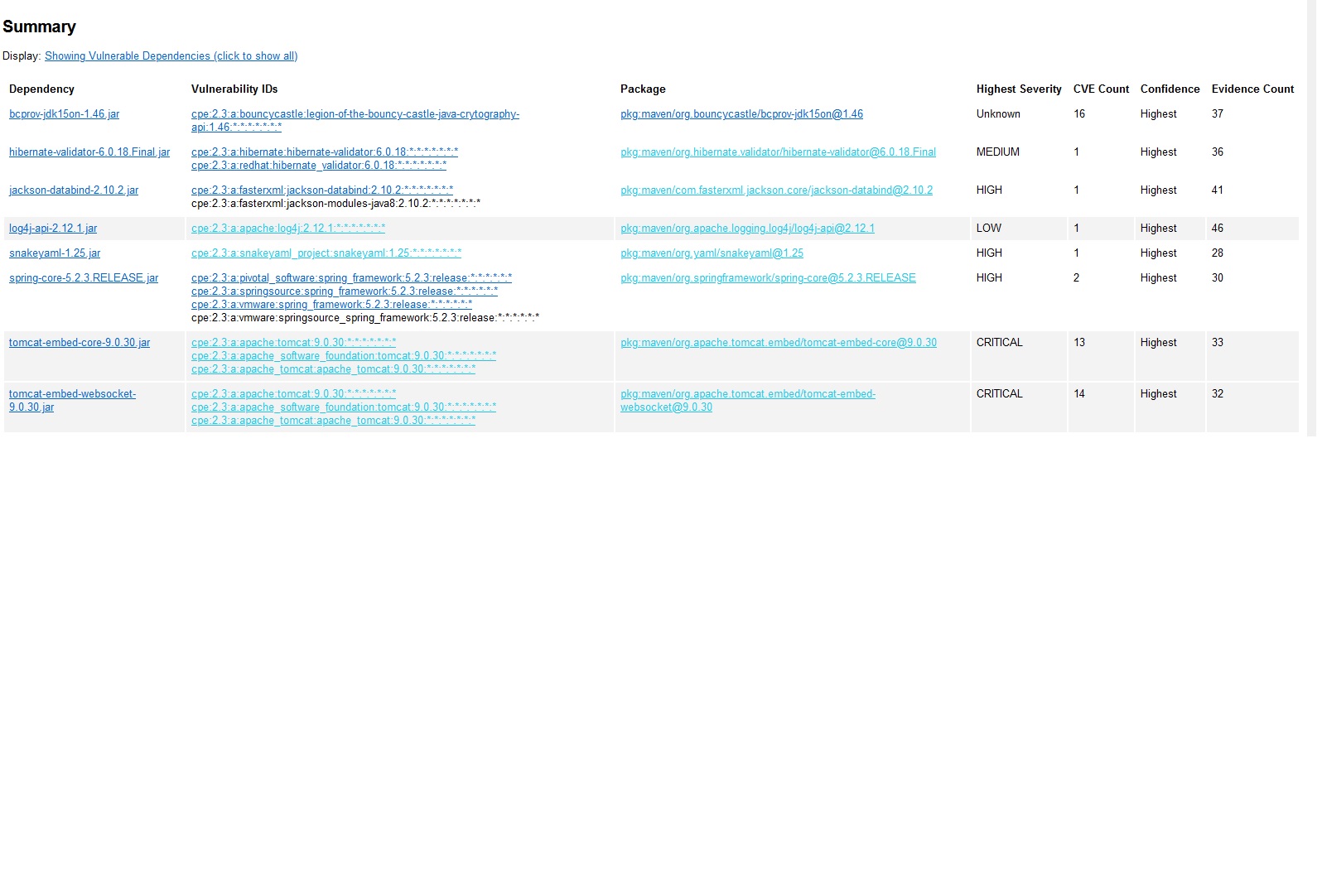
## 4. Static Testing

**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:**

One of the ways of ensuring secure and effective dependencies is to incorporate a dependency-check plugin scanner into our code. The Maven application is a non-intrusive test which would is a great method to ensure code security and all dependencies are current. Of course, there is the chance for false positives, when there is a supposed vulnerability when in reality there is no vulnerability. And on the other side of that, false negatives will arise when, as an example, not all dependencies have been tested or a name has been changed. Using the CPE dictionary search and to verify that there is a CPE to match is a good verification that a false negative has been found. So, it is rather easy to verify your code Based and the false positives can be eliminated from display by toggling the suppress icons on your report.

As stated previously, from a previous check, one of the reasons for our security vulnerabilities is the use of older versions of software in use. Firstly, and critically, the description of the components in the log4j-api-2.12.1.jar show that the most current version is 2.14.1. Based on the understanding of the causes of the security vulnerability, using Apache log4j API Dependencies has the potential for “*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*.” And while this threat is considered relatively low this is a prime example of keeping all of our software up to date. The next critical point is described as tomcat-embed-core-9.0.30.jar which is an open-source implementation for the Jakarta EE platform. This dependency has a very high influence in our code’s security. As such, it labeled as a critical risk. The CVE count shows us that the dependency was revealed twelve times and has nine critical and two severe vulnerabilities. And they mostly stem from the cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* which houses several high and critical vulnerabilities according to the NVD listing. And again, another example of software needing an update to 10.1.0-M2. And lastly the description snakeyaml-1.25.jar which is a parser and emitter for Jaza Dependencies for Core Tomcat implementation. The description focuses on the error which apparently expands on a vulnerability in a previous version as described by CVE-2003-1564. We also see that with a quick check a new version 1.29 of the software is available. And yet again we see how having the newer version can have a critical impact on our coding, and as such the issues of out-of-date software should be addressed immediately. NVD count shows us as single vulnerability which is a high risk and CVE shows us the same vulnerability as a critical issue. Here is a table of other vulnerabilities that have not previously been discussed.

|  |  |  |
| --- | --- | --- |
| bcprov-jdk15on-1.46.jar | **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.  **CVE-2015-6644**: An information disclosure vulnerability in Bouncy Castle could enable a local malicious application to gain access to user’s private information  **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."  **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.  **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.  **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.  **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.  **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 initialized 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.  **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.  **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.  **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.  **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.  **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."  **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 appears 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.  **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.  **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. | Updating to version 1.69 |
| hibernate-validator-6.0.18.Final.jar | **CVE-2020-10693**: 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. | Update to version 7.0.1 Final |
| jackson-databind-2.10.2.jar | **CVE-2020-25649**: 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. | Updating, to version is 2.13.0-rc1 |
| spring-core-5.2.3.RELEASE.jar | **CVE-2020-5421**: 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.  **CVE-2021-22118**: 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. | Update to version 5.3.9 |



## 

## 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.**

After reviewing Artemis Financials’ web-based software application for a vulnerability assessment, we have found several issues that need immediate addressing. During the manual review, we identified two vulnerabilities in the code base. In the DocData.java code displays the user’s name and password, and its simplicity makes it easy to crack by brute force attacks. Second is a vulnerability in the CRUDController.java where it is a possible to explore the application through The Value=”business\_name” and expose the DocData object database access vulnerability. During the static tests we found that there are several dependencies that require updates to make the app more secure, database access username and password, certificate Validation, code review and modification and TLS Certificate mutual checking. We also need to ensure that the website does not violate any international laws and security regulations, by setting up a conference with legal professionals. Lastly, we need to insure we are setting up Artemis Financials’ with a application that will last and be as future proof as can be made.