****

# 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** | **November 8th, 2022** | **Elizabeth Robles** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for 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 choose to include images or supporting materials. If you include them, make certain to insert them in all 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

Elizabeth Robles

## Interpreting Client Needs

Our client, Artemis Financial, is a company that develops customized financial plans for their customers, these plans include savings, retirement, investments, and insurance. This company values secure communications since having their customers financial information means that they have a great deal of valuable data that needs to be secured. It is not 100% clear if the company makes any international transactions, however, I will assume that they do since it does not state that the company only makes domestic transactions. There are also governmental restrictions about secure communications to consider as this company is in the financial sector, such as financial cybersecurity compliance, these regulations support data security. The external threats that may be present now and, in the future, include phishing, exploitation, DoS, insider threat, and ransomware attacks, just to name a few. The modernization requirements that we must consider include open-source libraries and web application technologies. Open-source libraries offer various benefits such as lower costs, the potential for scalability, support, and quality software. We will also need to consider building a web application, as they currently offer a desktop application, this would offer their customers more services they could use online.

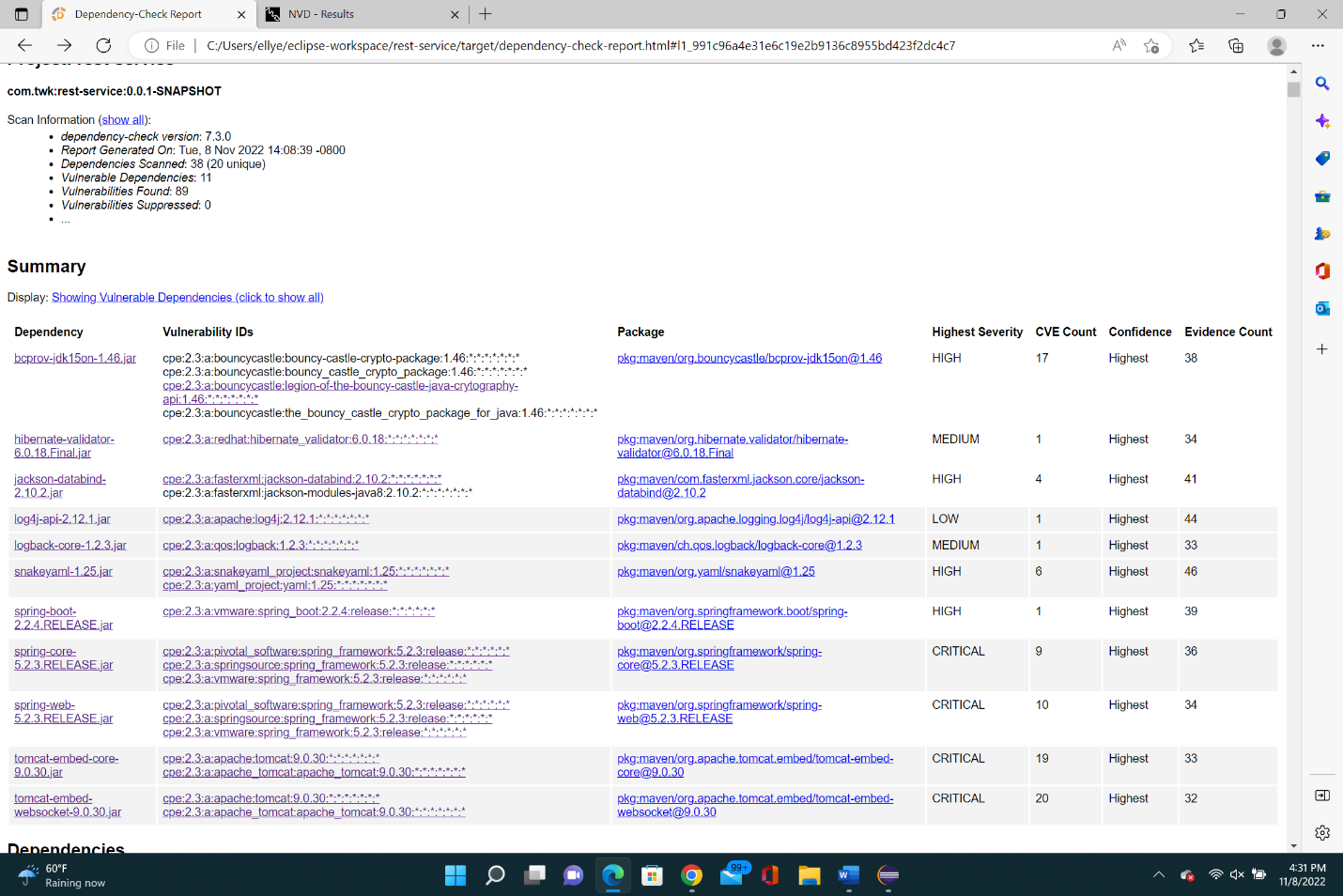
## Areas of Security

* Input Validation: the system will need to ensure that only correct data enters the workflow and prevents deformed data from entering. This can help reduce SQL injections as well as other attacks.
* APIs: Secure API interactions are important for Artemis Financial as they deal with financial data that may be shared amongst different services, banks, and other organizations.
* Cryptography: This will ensure that sensitive data such as client financial information is encrypted and unreadable by attackers, especially when dealing with clients that are in another country.
* Code Error: Simple code errors can become much larger problems for a system, such as revealing details to a hacker, so it is important for Global Rain to have a plan as to how errors will be handled and for all developers to understand the policy and make sure that the code the write follows that policy.
* Code Quality: Follow all secure coding practices/patterns such as data validation, authentication, authorization, storing sensitive information, database security, and following basic coding practices.
* Encapsulation: This will protect certain pieces of code and the data that it handles from the rest of the code, this can simply be done by making sure to declare variables in the class as private.

## Manual Review

After identifying out clients needs and deciphering the areas of security the Artemis Financial web application will need, I then looked over the project code base and determined that although the current code is not bad, it could be improved and there are some areas of security that need to be updated. Firstly, in the Customer.java file, it seems to be incomplete, with customer information that is not included, this could include other financial records, personal information, etc. In the GreetingController.java file I would also like to see some input validation such as authentication included in the code. CRUD is included which is an API that is helpful in database web applications.

## Static Testing



1. The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.
   1. [CVE-2016-1000352](https://nvd.nist.gov/vuln/detail/CVE-2016-1000352)**: earlier versions the ECIES implementation allows the use of ECB mode, it is unsafe and there is no support for it**
   2. [CVE-2016-1000346](https://nvd.nist.gov/vuln/detail/CVE-2016-1000346)**: in earlier versions DH public key is not completely validated; this can cause issues as these invalid keys may leak details to a 3rd party**
   3. **CVE-2016-1000345: in earlier versions the DHIES/ECIES CBC mode make it vulnerable to oracle attacks**
   4. **CVE-2016-1000344: in earlier versions the DHIES implements allows the use of ECB mode which makes it unsafe**
   5. **CVE-2016-1000343: in earlier versions the DSA key pair generator generates a weak private key; this can be dealt with by explicitly passing parameters to the key pair generator**
   6. **CVE-2016-1000342: earlier versions do not fully validate encoding of signature on verification, it may be possible to inject other elements in the sequence that makes up the signature and still have it validate**
   7. **CVE-2016-1000341: earlier versions and earlier DSA signature generation is vulnerable to timing attack.**
   8. **CVE-2016-1000339: in earlier versions the primary engine class used for AES was AESFastEngine. Due to the highly table-driven approach used in the algorithm, if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak info.**
   9. **CVE-2016-1000338: in earlier version the DSA does not completely validate ASN.1 encoding of signature on verification**
   10. **CVE-2018-5382: the default BKS keystore use an HMAC that is only 16 bits long, which allows an attacker to compromise the integrity of a BKS keystore.**
   11. **CVE-2017-13098: in earlier versions when configured to use the JCE for cryptographic functions, it provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated, an attacker can recover the private key from this vulnerable application**
   12. **CVE-2013-1624: in earlier versions it does not properly consider timing side-channel attacks on noncompliant MAC check operation during the processing of malformed CBC padding, this allows attackers to conduct distinguishing attacks and plaintext-recovery attacks**
2. Hibernate's Bean Validation (JSR-380) reference implementation.
   1. CVE-2020-10693: a bug was discovered in the message interpolation processor that enables invalid EL expressions to be evaluated as valid. Attackers can then bypass input sanitation controls
3. General data-binding functionality for Jackson: works on core streaming API
   1. CVE-2022-42004: in earlier versions resource exhaustion can occur because of a lack of BeanDeserializer.\_deserializeFromArray to prevent the use of deeply nested arrays
   2. CVE-2022-42003: in earlier version resource exhaustion can occur because of a lack of check in primitive value deserializers to avoid deep wrapper array nesting
   3. CVE-2020-36518: in earlier versions it allows a java StackOverflow exception and DoS via a large depth of nested objects
   4. CVE-2020-25649: an error was found where it did not have entity expansions secured properly, leaving it vulnerable to XXE attacks, which effects data integrity
4. The Apache Log4j API
   1. CVE-2021-44832: in earlier versions it is vulnerable to a remote code execution attack when a configurated uses JDBC Appender with a JNDI LDAP data source UIR when an attacker has control of the target LDAP server, issue can be fixed by limiting JNDI data source names
   2. CVE-2021-45105: in earlier versions it is not protected from uncontrolled recursion from self-referential lookups, this can allow an attacker to have control over Thread Context Map data
   3. CVE-2021-45046: a fix in ApAche Log4j 2.15.0 was not complete and could allow attackers with control over Thread Context Map input data when the logging configuration uses a non-default Patter Layout with either a Context Lookup or a Thread Context Map patter 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, newer versions fix this issue.
   4. CVE-2021-44228: in earlier versions JNDI features used in configurations, log messages, and parameters do not protect against attacker-controlled LDAP and other KNDI endpoints
   5. CVE-2020-9488: improper validation of certificate, this could allow an SMTPS connection to be intercepted by a man-in-the-middle attack
5. logback-core module
   1. CVE-2021-42559: in earlier versions an attacker with the right privileges to edit configurations files could create a malicious configuration
6. YAML 1.1 parser and emitter for Java
   1. CVE-2022-38752: using snakeYAML to parse untrusted YAML files can leave it vulnerable to DoS attacks
   2. CVE-2022-38751: using snakeYAML to parse untrusted YAML files can leave it vulnerable to DoS attacks
   3. CVE-2022-38750: using snakeYAML to parse untrusted YAML files can leave it vulnerable to DoS attacks
   4. CVE-2022-38749: using snakeYAML to parse untrusted YAML files can leave it vulnerable to DoS attacks
   5. CVE-2022-25857: earlier packages are vulnerable to DoS attacks due to missing to nested depth limitation for collections
   6. CVE-2017-18640: in earlier SnakeYAML, it allows for entity expansion during a load operation
7. Spring Boot
   1. CVE-2022-27772: in earlier spring-boot versions that are no longer supported by the maintainer, it was vulnerable to temporary directory hijacking
8. Spring Core
   1. CVE-2022-22971: in earlier version application with a STOMPY over WebSocket endpoint is vulnerable to a DoS attack
   2. CVE-2022-22970: in earlier version applications that handle file uploads are vulnerable to DoS attacks
   3. CVE-2022-22968: in earlier versions the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless itis listed with both upper and lower case for the first character of the filed including upper and lower case for the first character of all nested fields within the property path
   4. CVE-2022-22965: a Spring MVC or SpringWebFlux application running on JDK 9+ may be vulnerable to RCE via data binding
   5. CVE-2022-22950: in earlier and unsupported versions, its possible for a user to provide a specially crafted SpEL expression that may cause a DoS
   6. CVE-2021-22060: in earlier and unsupported versions its possible for a user to provide malicious input to cause the insertion of additional log entries
   7. CVE-2021-22096: in earlier and unsupported versions its possible for a user to provide malicious input to cause the insertion of additional log entries
   8. CVE-2021-22118: in earlier versions a WebFlux application is vulnerable to a privilege escalation, this can lead to a malicious user to read or modify files
   9. CVE-2020-5421: in earlier and unsupported versions the protections against RFD attacks may be bypassed depending on the browser used
   10. CVE-2016-1000027: Pivotal Spring Framework though 5.3.16 suffers from a potential RCE issue if used for Java deserialization of untrusted data, the issue may or may not occur depending on how the library is implemented within a product
9. Spring Web
   1. CVE-2022-22971: in earlier and unsupported versions, application with a SOMP over WebSocket endpoint us vulnerable to DoS attacks
   2. CVE-2022-22970: in earlier and unsupported versions, applications that handle file uploads are vulnerable to DoS attacks if they rely on data binding
   3. CVE-2022-22968: in earlier and unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive, so a field is not properly protected unless it is listed with both upper and lower case for the first character of the field
   4. CVE-2022-22965: may be vulnerable to RCE via data binding if a Spring MVC or Spring WebFlux app is running on JDK 9+
   5. CVE-2022-22950: in earlier and unsupported versions its possible for a user to provide a specially crafted SpEL expression that may cause a DoS
   6. CVE-2021-22060: in earlier and unsupported versions its possible for a user to provide malicious input to cause the insertion of additional log entries
   7. CVE-2021-22096: in earlier and unsupported versions it is possible for a user to provide malicious input to cause the insertion of additional log entries
   8. CVE-2021-22118: in certain versions a WebFlux app is vulnerable to a privilege escalation, a malicious user can read or modify files that have been uploaded to the WebFlux application
   9. CVE-2020-5421: in earlier and unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser using a jsessionid path parameter
   10. CVE-2016-1000027: in earlier versions, it suffers from a potential RCE issue if used for Java deserialization of untrusted data
10. Core Tomcat implementation
    1. CVE-2022-42252, CVE-2021-43980, CVE-2022-34305, CVE-2022-29885, CVE-2021-41079, CVE-2021-33037, CVE-2021-30640, CVE-2021-25329, CVE-2021-25122, CVE-2021-24122, CVE-2020-17527, CVE-2020-13943, CVE-2020-13935, CVE-2020-13934, CVE-2020-8022, CVE-2020-11996, CVE-2020-9484, CVE-2020-1938, CVE-2020-1935, CVE-2019-17569
       1. In certain versions of Apache Tomcat, there are many issues and attacks that may occur such as smuggling attacks, client connections to share anHTTp11Processor instance resulting in responses to be received by an unwanted client, exposing an XSS vulnerability, DoS risks, could trigger an infinite loop resulting in DoS, unresponsive servers, etc.

## Mitigation Plan

1. The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7
   1. The main issue with this vulnerability is the fact that newer versions of the Bouncy Castle Java are not being used, my suggestion to fix this is simply update to the newest version.
2. Hibernate's Bean Validation (JSR-380) reference implementation.
   1. Simply update to newest version of Hibernate Validator to avoid potential bypass of input sanitation controls.
3. General data-binding functionality for Jackson: works on core streaming API
   1. Update to newest version of FasterXML Jackson-databind to avoid DoS and XXE attacks.
4. The Apache Log4j API
   1. Update to newest version of Apache Log4j2 to avoid vulnerabilities such as leaks, DoS, and remote code execution attacks, etc.
5. logback-core module
   1. Simply update to newest version of logback version to avoid malicious configuration that allows the execution of arbitrary code.
6. YAML 1.1 parser and emitter for Java.
   1. Update to newest version of package org.yaml:snakeyaml and also do not allow the parser to run on user supplied input as it may cause various attacks.
7. Spring Boot
   1. Do not use an earlier or unsupported version of sbring-boot, update to the most recent version as issues only arise when versions are no longer supported by the maintainer.
8. Spring Core
   1. Do not use older version of spring framework as they are old and unsupported, please update to the most recent version.
9. Spring Web
   1. Update to the most recent version of Spring framework, using an unsupported version can lead to various attacks.
10. Core Tomcat implementation
    1. Update to most recent version of Apache Tomcat