****

# 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** | **07/14/2021** | **Abby Ramatowski** |  |

## 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

Abby Ramatowski

## 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?
* Are there any international transactions that the company produces?
* Are there governmental restrictions about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

**What is the value of secure communications to the company?**

Artemis Financial is a financial consulting company and develops individualized financial plans for savings, retirement, investments, and insurance. Because Artemis Financial has requested that any security vulnerabilities in the current software is identified and because valuable personal information and financial information is at risk, I have concluded that security for Artemis’ Financial is a high priority.

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

Based on the given scenario, we are unaware if Artemis’ Financial produces any international transactions, but Global Rain does provide custom software design and development throughout the world.

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

Financial institutions, like Artemis Financial, must have insurance and there are government restrictions about secure communication applied to the company’s that insure them. For example, Federal Deposit Insurance Company (FDIC) has the following restrictions that apply to cybersecurity and secure communication:

* “Appendix A to Part 364 – Interagency Guidelines Establishing Standards for Safety and Soundness provide operational and managerial standards that address internal controls and information systems
* Appendix B to Part 364 – Interagency Guidelines Establishing Information Security Standards address administrative, technical, and physical safeguards to protect the security, confidentiality, and integrity of customer information
* Section 304.3(d) – Reports addresses requirements for regulatory notification of certain service provider relationships
* The Bank Service Company Act establishes FDIC regulation and examination authority over certain service providers”, (*Information Technology and Cybersecurity,* n.d.).

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

There are a number of threats that are common to financial companies. These include phishing, where the financial company is deceived by the attacker into believing they are trustworthy. The attacker does this through electronic communication and then will prompt the company to install malicious software onto the corporate device. This can result in monetary loss for the company or its customers, as well as identity theft. Another threat is reverse social engineering. This is when the attacker portrays themselves to be trustworthy and as someone who can help with IT related issues on the company’s website. A user will then communicate with the attacker for help and then the attacker forces the user to comply with their requirements. The results of this kind of attack could be a data breach of valuable company and customer information. Waterholing is another example of an external threat. This is when attackers infiltrate the website with malware. This can result in a more difficult to detect attack with loss of valuable personal information. Another threat is false websites. This is when attackers create a false website that is similar to the company’s website. They trick users to use the false website and can result in data breach of valuable information. (Cyriac N.T, 2019).

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

Ensuring that the RESTful API is secure must be considered. A RESTful API is an API (application programming interface) that follows a set of rules defined by the REST architectural structure to allow access and manipulation to a website. It is important to follow the best practices for securing RESTful API. These include keeping it simple – the more complex it is made to be, the more likely you will leave areas unsecure. Using HTTPS are also essential. HTTPS can protect the website from phishing, and other data breaches. Hashing algorithms can be used to protect passwords, and ensuring to never expose usernames, passwords, or other valuable information on URLs. All of these should be considered to improve the security of the website. (Fielding, R.T., 2020).

## 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.

**Input Validation**

Input validation is relevant and should be considered in Artemis Financial’s software application. Input validation is a process of testing and validating input received by the application. It is relevant because it can help to prevent data that has been incorrectly formed from entering the application. This information could possibly be malicious and, therefore, input validation will improve the security of the software supplication.

**APIs**

APIs provide a way for two applications to communicate. There are many ways to ensure that APIs are secure. APIs must be tracked and managed regularly, encrypting APIs, and using strong authorization and authentication techniques are some of the ways to ensure API security. Using secure APIs are relevant to Artemis Financial’s software application because there must be a way to securely communicate with the application to avoid leaking personal and valuable information to attackers.

**Cryptography**

Cryptography is relevant and should be considered for Artemis Financial’s software application. Cryptography is the process of encrypting data and allows sensitive and personal information to be stored and protected. This is very important for protecting Artemis Financial’s and their customers personal information.

**Client/Server**

Using a client server network will improve security and should be considered for Artemis Financial’s software application. A client server network involves a central server with which provides service to many clients on remote processors. This is relevant to the website because there will be many clients on remote processors making requests which should be received by a central server.

**Code Error**

Code error should be used for Artemis Financial’s software application rather than descriptive error messages. This is similar to the benefits of encrypting personal information. Using code error will make it more difficult for attackers to view data and information within the application.

**Code Quality**

Code quality is relevant because higher quality code provides improved security.

**Encapsulation**

Encapsulation is relevant to the security of Artemis Financial’s software application because it can protect it from unwanted access.

## 3. Manual Review

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

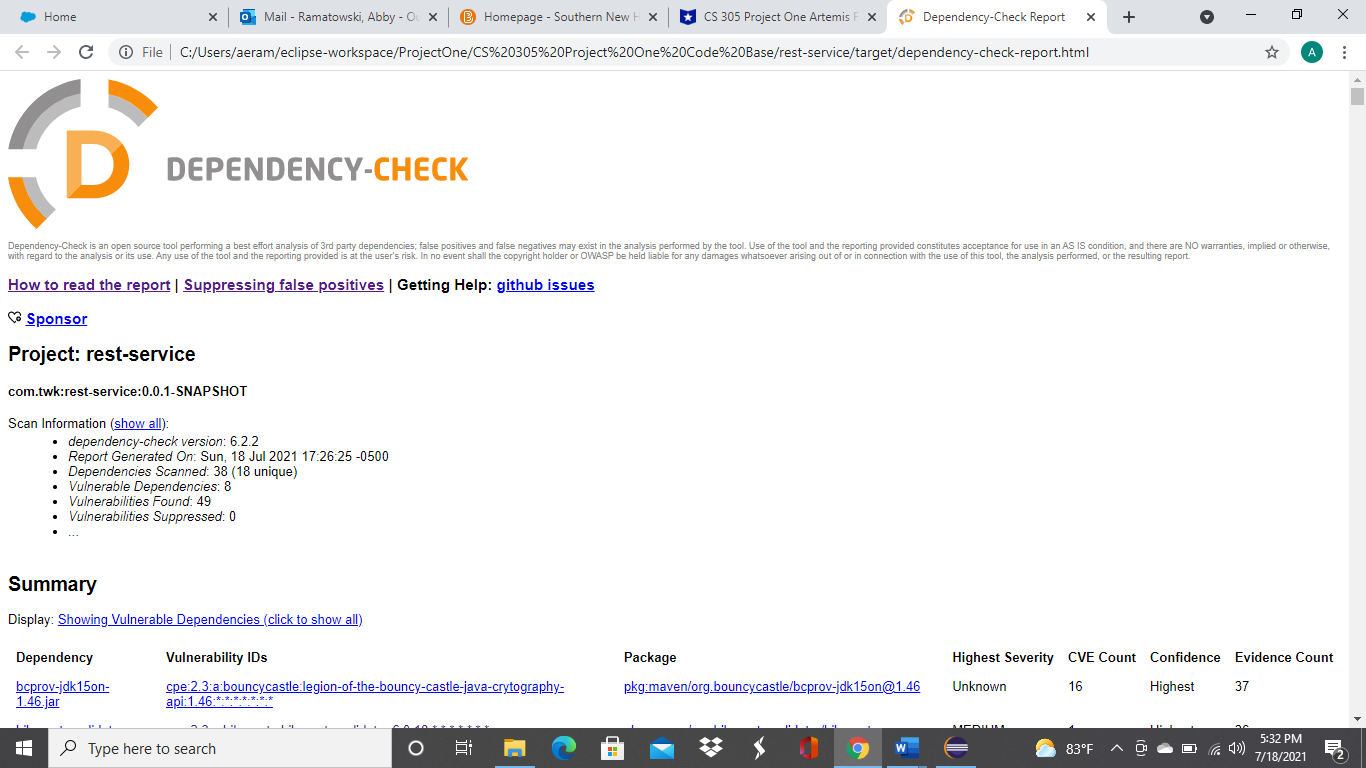
The following vulnerabilities were found when inspecting the code:

* I did not find the use of HTTPS in any area of the provided code. Use of HTTPS is important to ensure protection when sharing valuable information.
* In CRUDController.java, the business name is used as a parameter. This is sensitive information that could be leaked to an attacker when used as a parameter.
* I did not find an authentication system.
* There also is not an input validation system in place for requests in the system.

## 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



**Dependencies:**

**Bcprov-jdk15on-1.46.jar**

Description: 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.

Published Vulnerabilities:

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 values and ultimately the private value as well.

CVE-2016-10000342

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 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 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 execution 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.

**Hibernate-validator-6.0.18.Final.jar**

Description: Hibernate’s Bean Validation (JSR-380) reference implementation.

Published Vulnerabilities:

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.

**Jackson-databind-2.10.2.jar**

Description: General data-binding functionality for Jackson: works on core streaming API

Published Vulnerabilities:

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.

**Log4j-api-2.12.1.jar**

Description: The Apache Log4j API

Published Vulnerabilities:

CVE-2020-9488

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.

**Snakeyaml-1.25.jar**

Description: YAML 1.1 parser and emitter for Java

Published Vulnerabilities:

CVE-2017-18640

The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

**Spring-core-5.2.3.RELEASE.jar**

Description: Spring Core

Published Vulnerabilities:

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 excalation: 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.

**Tomcat-embed-core-9.0.30.jar**

Description: Core Tomcat implementation

Published Vulnerabilities:

CVE-2019-17569

The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that 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. Such a reverse proxy is considered unlikely.

CVE-2020-11996

A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.

CVE-2020-13934

An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.

CVE-2020-13935

The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.1042. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.

CVE-2020-13943

If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeding the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers – including HTTP/2 pseudo headers – from a previous request rather than the intended headers. This could lead to users seeing responses for unexpected resources.

CVE-2020-17527

While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests.

CVE-2020-1935

In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led 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. Such a reverse proxy is considered unlikely.

CVE-2020-1938

When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application – processing any file in the web application as a JSP Further, if the we application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessibility to untrusted users. Users wishing to take a defense-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

CVE-2020-9484

When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103 if a) an attacker is able to control the contents and name of a file on the server; and b) the server is configured to use the PersistenceManager with a FileStore; and c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter=”null” (the default unless a SecurityManager is used) or a sufficiently lax filter to allow the attacker provided object to be deserialized; and d) the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over; then, using a specifically crafted request, the attacker will be able to trigger remote code execution via deserialization of the file under their control.

CVE-2021-24122

When serving resources from a network location using the NTFS file system, Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106 were susceptible to JSP source code disclosure in some configurations. The root cause was the unexpected behavior of the JRE API File.getCanonicalPath() which in turn was caused by the inconsistent behavior of the Windows API (FindFirstFileW) in some circumstances.

CVE-2021-25122

When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 could duplicate request headers and a limited amount of request body from one request to another meaning user A and user B could both see the results of user A’s request.

CVE-2021-25329

The fix for CVE-2020-9484 was incomplete. When using Apache Tomcat 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. to 7.0.107 with a configuration edge case that was highly unlikely to be used, the Tomcat instance was still vulnerable to CVE-2020-9494. Note that both the previously published prerequisites for CVE-2020-9484 and the previously published mitigations for CVE-2020-9484 also apply to the issue.

CVE-2021-33037

Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did 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. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honored the identity encoding; and – Tomcat did not ensure that , if present, the chunked encoding was the final encoding.

**Tomcat-embed-websocket-9.0.30.jar**

Description: Core Tomcat implementation

Published Vulnerabilities:

CVE-2019-17569

The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling of Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

CVE-2020-11996

A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35, and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.

CVE-2020-13934

An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.

CVE-2020-13935

The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.

CVE-2020-13943

If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connections (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers – including HTTP/2 pseudo headers – from a previous request rather than the intended headers. This could lead to users seeing responses for unexpected resources.

CVE-2020-17527

While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests.

CVE-2020-1935

In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led 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. Such a reverse proxy is considered unlikely.

CVE-2020-1938

When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connections. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses.

## 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.

The first step is to switch to HTTPS for all communication to keep the company and their customer’s information secure. Next, I will remove business names from the parameters in CRUDController.java. I will move any request parameters to headers. Updating all of the above listed dependencies from the dependency check will also need to be completed. Finally, implementing two-factor authentication to protect the company and it’s customer’s information.

References

Cyriac, N.T., & Sadath, L. (2019). *Is Cyber Security Enough – A study on Big Data Security Breaches in Financial Institutions.* 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Information Systems and Computer Networks (ISCON), 2019 4th International Conference On. Retrieved from <https://eds-b-ebscohost-com.ezproxy.snhu.edu/eds/detail/detail?vid=3&sid=51ed0991-04b1-4730-a5e4-d98e6a2b9089%40sessionmgr102&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=edseee.9036294&db=edseee>

Fielding, R.T. (2020). *REST API Security Essentials.* RESTful API. Retrieved from <https://restfulapi.net/security-essentials/>

*Information Technology (IT) and Cybersecurity.* (n.d.) Federal Deposit Insurance Company. Retrieved from <https://www.fdic.gov/resources/bankers/information-technology/>