****

**CS-305 Project One**

# Artemis Financial Vulnerability Assessment Report

**Paige Bertram**

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** | **01/25/2023** | **Paige Bertram** |  |

## Client



## Developer

Paige Bertram

## Interpreting Client Needs

What is the value of secure communications to the company?

* Due to the nature of Artemis Financial’s work involving sensitive client information, implementing a web-based application with strong security while data is being moved is very important. This helps to ensure that while data is being moved over the network, it is protected from outside attacks and has encryption and multifactor authentication in place to add extra layers of security. Since we were not provided with the information that they make international transactions or have government restrictions, it is still important to implement security measures within the system that will also consider open-source libraries and evolving web application technologies.

## Areas of Security

* Input validation is an area of security since users are accessing a website that requires them to input personal information to access their personal data and information. Since Artemis Financial has a RESTful web application, having secure APIs in place helps ensure that the data is only being accessed by authorized users. Cryptography is relevant because it will ensure encryption is in place for sensitive data and protect it from third parties. Client / Server is also relevant because we are using APIs, and having a valid certificate will secure the web application and database. Code errors help to review code and find vulnerabilities. Code quality helps to follow coding best practices and will help to ensure there are no vulnerabilities in the code where possible attacks will be able to get into the system and access data. Encapsulation ensure data access protection and helps to reduce the risk of data leaks.

## Manual Review

* After reviewing the code, it was found that there is no authentication in place, as well as no input validation, and this will make the system vulnerable to outsider attacks or unauthorized access into the system. Cryptography is also missing, which means that there is no certificate in place to secure the application and the database through HTTPS. Business information is also available within the Java files, which leaves information open to third parties and possible data leaks.

## Static Testing

* bcprov-jdk15on-1.46.jar - Upgrade to a newer version
  + 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 CVE2013-0169.
  + 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 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.
  + 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 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.
* hibernate-validator-6.0.18.Final.jar - Upgrade to a newer version
  + 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 - Upgrade to a newer version
  + 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 - Upgrade to a newer version
  + 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-inthe-middle attack which could leak any log messages sent through that appender.

* snakeyaml-1.25.jar - Upgrade to a newer version
  + 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 - Upgrade to a newer version
  + 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.

* tomcat-embed-core-9.0.30.jar - Upgrade to a newer version
  + 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 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with

* + 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 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 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 web 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 accessible to untrusted users. Users wishing to take a

defence-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. Note that all of conditions a) to d) must be true for the

attack to succeed.

* + 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 behaviour of the JRE API

File.getCanonicalPath() which in turn was caused by the inconsistent behaviour 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 this 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 honoured the identify

encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the

final encoding.

* tomcat-embed-websocket-9.0.30.jar - Upgrade to a newer version
  + 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 7.0.104. 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 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 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 web 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 accessible to untrusted users. Users wishing

to take a defence-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-2020-8022: A Incorrect Default Permissions vulnerability in the packaging of tomcat

on SUSE Enterprise Storage 5, SUSE Linux Enterprise Server 12-SP2-BCL, SUSE Linux

Enterprise Server 12-SP2-LTSS, SUSE Linux Enterprise Server 12-SP3-BCL, SUSE Linux

Enterprise Server 12-SP3-LTSS, SUSE Linux Enterprise Server 12-SP4, SUSE Linux

Enterprise Server 12-SP5, SUSE Linux Enterprise Server 15-LTSS, SUSE Linux Enterprise

Server for SAP 12-SP2, SUSE Linux Enterprise Server for SAP 12-SP3, SUSE Linux

Enterprise Server for SAP 15, SUSE OpenStack Cloud 7, SUSE OpenStack Cloud 8, SUSE

OpenStack Cloud Crowbar 8 allows local attackers to escalate from group tomcat to

root. This issue affects: SUSE Enterprise Storage 5 tomcat versions prior to 8.0.53-

29.32.1. SUSE Linux Enterprise Server 12-SP2-BCL tomcat versions prior to 8.0.53-

29.32.1. SUSE Linux Enterprise Server 12-SP2-LTSS tomcat versions prior to 8.0.53-

29.32.1. SUSE Linux Enterprise Server 12-SP3-BCL tomcat versions prior to 8.0.53-

29.32.1. SUSE Linux Enterprise Server 12-SP3-LTSS tomcat versions prior to 8.0.53-

29.32.1. SUSE Linux Enterprise Server 12-SP4 tomcat versions prior to 9.0.35-3.39.1.

SUSE Linux Enterprise Server 12-SP5 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux

Enterprise Server 15-LTSS tomcat versions prior to 9.0.35-3.57.3. SUSE Linux Enterprise

Server for SAP 12-SP2 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise

Server for SAP 12-SP3 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise

Server for SAP 15 tomcat versions prior to 9.0.35-3.57.3. SUSE OpenStack Cloud 7

tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud 8 tomcat versions prior

to 8.0.53-29.32.1. SUSE OpenStack Cloud Crowbar 8 tomcat versions prior to 8.0.53-

29.32.1.

* + 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. Note that all of conditions a) to d) must be true for the

attack to succeed.

* + 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 behaviour of the JRE API

File.getCanonicalPath() which in turn was caused by the inconsistent behaviour 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 this 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 honoured the

identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding

was the final encoding.

## Mitigation Plan

* I would first implement input validation in GreetingController and multifactor authentication to ensure only authorized users are accessing the system and the data. I would also remove the business information from the Java files to reduce the risk of data leaks and request a certificate to secure communications between the web application and database. I would also recommend upgrading spring-core-5.2.3, log4j-api-2.12.1.jar, and tomcatembed-core-9.0.30.jar to the latest versions to resolve bug issues and have the most up-to-date patches and security updates to make the system more secure and functional.