****

# 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** | **11/14/2022** | **Dylan Coulter** |  |

## 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

Dylan Coulter

## Interpreting Client Needs

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

The importance of secure communications is high for the company Artemis Financial due to the fact they are a financial company and the information and data they hold is sensitive and needs to be protected as such. A data breach could mean massive losses for both the customer and company. It could also lead to customers deciding not to do business with the company as they may view it as a risk to their privacy.

**Does the company make any international transactions?**

Artemis Financial is a web-based company which means the company can connect with customers from all around the world. International transitions are possible if someone accesses the data internationally. A customer using their account while in another country will produce international transactions.

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

Yes, seeing as Artemis Financial makes international transactions they would be subject, not only to the Unites States governments regulations, but also to the regulations of the countries they are completing transactions in. In November 2021, the FDIC and OCC past a rule that banking organizations would need to notify the primary federal regulator within 36 hours of a security incident (Magrath, 2022). In August 2021 the Federal Financial Institutions Examination Council (FFIEC) updated its Authentication and Access to Financial Institution Services and Systems Guidance. This is the first time in a decade this has been done. This guidance encourages FIs to identify their users and customers. The users and customers to be identified are the ones who warrant authentication and access management controls. This is in addition to users and customers who may warrant more enhanced authentication controls (Magrath, 2022).

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

There are multiple threats that are present now and will be in the immediate future. These include authentication threats. This can be in the form of brute force attacks, dictionary attacks, password spraying attacks, and Credential Stuffing Attacks. There is the possibility of man-in-the-middle attacks these attacks can happen from issues such as a compromised WIFI network. Then there are DDOS attacks where the system is overloaded to the point it crashes.

**What are the modernization requirements that you must consider?**

As the company is dealing with sensitive customer information, the first modernization requirement would be to utilize HTTPS as it is more secure than HTTP. Another requirement would be to use multi-authentication to ensure the users information is secure also this will help protect against brute force attacks. Also, the company can become fully web-based and this can benefit the company as it will have an open-source library to add future layers of security and functionality.

## Areas of Security

**Input Validation-** RESTful API can take user inputs and this means the user input must be accepted and validated in a secure way. Without this the system would be vulnerable.

**API-** This is one of the most integral areas of security as web applications make use of API calls. A secure configuration is important so all get requests between the user and the system as well other information will be secure.API also includes open authentication (OAuth) which is where unrelated servers and services will safely grant authenticated access to their assets without a single logon credential.

**Cryptography-** Is used so that information can be passed over the internet. Without encryption the data would be compromised. Utilizing this will help to better protect both Artemis Financial and their customers data.

**Client/Server-** This is the communication between the server and the client which when considering Artemis Financial will be happening a lot due to the fact the company is using a web-based platform. Ensuring the connection between the backend and the front-end UI is secure will be imperative.

**Secure Coding-** Developing structured secure code is important to the overall security of Artemis Financials company. Without secure code it is easier to perform attacks on the system.

## Manual Review

During the manual inspection of the code here are the issues that we found:

* HTTPS is not being utilized which is recommended when the system is working with sensitive information.
* The system has no authentication in place to be use for verification usages.
* Any request is not validated which can open the system up for an attack from outsiders.
* In the CRUD controller class business names are sent as request parameters. This can cause information leaks to outsiders.

## Static Testing

**bcprov-jdk15on-1.46.jar**

**Highest Severity- High**

**Evidence Count- 38**

[**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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."

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522)

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

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**CVE-2020-0187** (OSSINDEX)

In engineSetMode of BaseBlockCipher.java, there is a possible incorrect cryptographic algorithm chosen due to an incomplete comparison. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-10Android ID: A-148517383

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**CVE-2020-26939** (OSSINDEX)

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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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."

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**CVE-2015-6644** (OSSINDEX)

Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows attackers to obtain sensitive information via a crafted application, aka internal bug 24106146.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**hibernate-validator-6.0.18.Final.jar**

**Highest Severity- Medium**

**Evidence Count- 34**

[**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**jackson-databind-2.10.2.jar**

**Highest Severity- Low**

**Evidence Count- 41**

[**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-36518**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518)

jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-42003**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42003)

In FasterXML jackson-databind before 2.14.0-rc1, resource exhaustion can occur because of a lack of a check in primitive value deserializers to avoid deep wrapper array nesting, when the UNWRAP\_SINGLE\_VALUE\_ARRAYS feature is enabled. Additional fix version in 2.13.4.1 and 2.12.17.1

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-42004**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42004)

In FasterXML jackson-databind before 2.13.4, resource exhaustion can occur because of a lack of a check in BeanDeserializer.\_deserializeFromArray to prevent use of deeply nested arrays. An application is vulnerable only with certain customized choices for deserialization.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**log4j-api-2.12.1.jar**

**Highest Severity- Medium**

**Evidence Count- 44**

[**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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. Fixed in Apache Log4j 2.12.3 and 2.13.1

logback-core-1.2.3.jar

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**logback-core-1.2.3.jar**

**Highest Severity- Medium**

**Evidence Count- 33**

[**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550)

In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**snakeyaml-1.25.jar**

**Highest Severity- High**

**Evidence Count- 46**

[**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640)

The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-25857**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-25857)

The package org.yaml:snakeyaml from 0 and before 1.31 are vulnerable to Denial of Service (DoS) due missing to nested depth limitation for collections.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-38749**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38749)

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-38751**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38751)

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-38752**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38752)

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack-overflow.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-41854**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-41854)

Those using Snakeyaml to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow. This effect may support a denial of service attack.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-38750**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38750)

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**spring-boot-2.2.4.RELEASE.jar**

**Highest Severity- High**

**Evidence Count- 39**

[**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)

\*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

spring-core-5.2.3.RELEASE.jar

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**spring-core-5.2.3.RELEASE.jar**

**Highest Severity- Critical**

**Evidence Count- 36**

[**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965)

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22971**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971)

In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, application with a STOMP over WebSocket endpoint is vulnerable to a denial of service attack by an authenticated user.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968)

In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22970**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970)

In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, applications that handle file uploads are vulnerable to DoS attack if they rely on data binding to set a MultipartFile or javax.servlet.Part to a field in a model object.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)

In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)

In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**spring-web-5.2.3.RELEASE.jar**

**Highest Severity- Critical**

**Evidence Count- 34**

[**CVE-2016-1000027**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027)

Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965)

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950)

In Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22971**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971)

In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, application with a STOMP over WebSocket endpoint is vulnerable to a denial of service attack by an authenticated user.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968)

In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-22970**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970)

In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, applications that handle file uploads are vulnerable to DoS attack if they rely on data binding to set a MultipartFile or javax.servlet.Part to a field in a model object.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)

In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)

In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**tomcat-embed-core-9.0.30.jar**

**Highest Severity- Critical**

**Evidence Count- 33**

[**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)

Apache Tomcat 8.5.0 to 8.5.63, 9.0.0-M1 to 9.0.43 and 10.0.0-M1 to 10.0.2 did not properly validate incoming TLS packets. When Tomcat was configured to use NIO+OpenSSL or NIO2+OpenSSL for TLS, a specially crafted packet could be used to trigger an infinite loop resulting in a denial of service.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885)

The documentation of Apache Tomcat 10.1.0-M1 to 10.1.0-M14, 10.0.0-M1 to 10.0.20, 9.0.13 to 9.0.62 and 8.5.38 to 8.5.78 for the EncryptInterceptor incorrectly stated it enabled Tomcat clustering to run over an untrusted network. This was not correct. While the EncryptInterceptor does provide confidentiality and integrity protection, it does not protect against all risks associated with running over any untrusted network, particularly DoS risks.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-42252**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252)

If Apache Tomcat 8.5.0 to 8.5.82, 9.0.0-M1 to 9.0.67, 10.0.0-M1 to 10.0.26 or 10.1.0-M1 to 10.1.0 was configured to ignore invalid HTTP headers via setting rejectIllegalHeader to false (the default for 8.5.x only), Tomcat did not reject a request containing an invalid Content-Length header making a request smuggling attack possible if Tomcat was located behind a reverse proxy that also failed to reject the request with the invalid header.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)

A vulnerability in the JNDI Realm of Apache Tomcat allows an attacker to authenticate using variations of a valid user name and/or to bypass some of the protection provided by the LockOut Realm. This issue affects Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305)

In Apache Tomcat 10.1.0-M1 to 10.1.0-M16, 10.0.0-M1 to 10.0.22, 9.0.30 to 9.0.64 and 8.5.50 to 8.5.81 the Form authentication example in the examples web application displayed user provided data without filtering, exposing a XSS vulnerability.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-43980**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980)

The simplified implementation of blocking reads and writes introduced in Tomcat 10 and back-ported to Tomcat 9.0.47 onwards exposed a long standing (but extremely hard to trigger) concurrency bug in Apache Tomcat 10.1.0 to 10.1.0-M12, 10.0.0-M1 to 10.0.18, 9.0.0-M1 to 9.0.60 and 8.5.0 to 8.5.77 that could cause client connections to share an Http11Processor instance resulting in responses, or part responses, to be received by the wrong client.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

**tomcat-embed-websocket-9.0.30.jar**

**Highest Severity- Critical**

**Evidence Count- 32**

[**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)

Apache Tomcat 8.5.0 to 8.5.63, 9.0.0-M1 to 9.0.43 and 10.0.0-M1 to 10.0.2 did not properly validate incoming TLS packets. When Tomcat was configured to use NIO+OpenSSL or NIO2+OpenSSL for TLS, a specially crafted packet could be used to trigger an infinite loop resulting in a denial of service.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885)

The documentation of Apache Tomcat 10.1.0-M1 to 10.1.0-M14, 10.0.0-M1 to 10.0.20, 9.0.13 to 9.0.62 and 8.5.38 to 8.5.78 for the EncryptInterceptor incorrectly stated it enabled Tomcat clustering to run over an untrusted network. This was not correct. While the EncryptInterceptor does provide confidentiality and integrity protection, it does not protect against all risks associated with running over any untrusted network, particularly DoS risks.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-42252**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252)

If Apache Tomcat 8.5.0 to 8.5.82, 9.0.0-M1 to 9.0.67, 10.0.0-M1 to 10.0.26 or 10.1.0-M1 to 10.1.0 was configured to ignore invalid HTTP headers via setting rejectIllegalHeader to false (the default for 8.5.x only), Tomcat did not reject a request containing an invalid Content-Length header making a request smuggling attack possible if Tomcat was located behind a reverse proxy that also failed to reject the request with the invalid header.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)

A vulnerability in the JNDI Realm of Apache Tomcat allows an attacker to authenticate using variations of a valid user name and/or to bypass some of the protection provided by the LockOut Realm. This issue affects Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305)

In Apache Tomcat 10.1.0-M1 to 10.1.0-M16, 10.0.0-M1 to 10.0.22, 9.0.30 to 9.0.64 and 8.5.50 to 8.5.81 the Form authentication example in the examples web application displayed user provided data without filtering, exposing a XSS vulnerability.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

[**CVE-2021-43980**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980)

The simplified implementation of blocking reads and writes introduced in Tomcat 10 and back-ported to Tomcat 9.0.47 onwards exposed a long standing (but extremely hard to trigger) concurrency bug in Apache Tomcat 10.1.0 to 10.1.0-M12, 10.0.0-M1 to 10.0.18, 9.0.0-M1 to 9.0.60 and 8.5.0 to 8.5.77 that could cause client connections to share an Http11Processor instance resulting in responses, or part responses, to be received by the wrong client.

**Resource**- <https://cve.mitre.org/cve/search_cve_list.html>

## Mitigation Plan

**Perform Regular Updates/Maintenance**

In order for Artemis Financials software to be secure updates need to be performed. We can see this as the recommended fixes all revolve around performing updates. This shows us how fast technology and programs grow and change and a company’s security can be compromised if certain actions are not taken. One way for Artemis Financial to ensure they limit their vulnerabilities is to make sure the program that Global Rain develops for them is up to date to limit vulnerabilities. This is not just a one-time thing, continuous system updates are integral to ensuring security is at its peak.

**Change to HTTPS**

The system should be switch over to HTTPS for better security and to prevent and snooping by unwanted sources.

**Implement Two Factor Authentication**

To better protect sensitive data and user accounts the implementation of two factor Authentication should be considered.

**Business Names**

Business names should not be within hard coded data base credentials these should be removed

**References**

# Magrath, M. (2022, January 24).  *Top Banking Regulations & Security Compliance Requirements [2022]* https://www.onespan.com/blog/top-banking-regulations-security-compliance-requirements