Module 3 Project One

Anthony Lee

SNHU

CS-305-T3674 – Software Security

Professor Vivian Lyon

January 21, 2022

****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

Document Revision History 4

Client 4

Instructions 4

Developer 5

1. Interpreting Client Needs 5

2. Areas of Security 5

3. Manual Review 5

4. Static Testing 6

5. Mitigation Plan 13

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/21/2022 | Anthony Lee | Creation |

## Client

Artemis Financial

## 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

Anthony Lee

## 1. Interpreting Client Needs

The client, Artemis Financial, desires to modernize its operations. As a result, they want to implement and apply the most current and effective software security. The client already has a RESTful web API and is asking us, Global Rain, for our expertise in taking steps to protect the organizations from external threats.

The secure communication is important because Artemis Financial is a financial consulting company that develops individualized financial plans for their patrons. Since the plans are individualized, there is no international transactions being made. The client, being a financial consulting company, has many external threats to consider, such as phishing, injection attacks, and spoofing, to name a few.

Open-source libraries and evolving web application technologies are constantly being updated, which helps keep up with the modernization requirements the client wants. As such, keeping up-to-date with the latest version of the implemented libraries and applications is needed to keep up with security.

## 2. Areas of Security

APIs: RESTful API sections

Client/Server: This code is the server side of Client/Server interactions

Code Error: This code is reading files

Code Quality: Since the code implements prior architectures (APIs, Client/Server, Code Error), it is good practice to check for secure coding practices in the code base.

## 3. Manual Review

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

Graphical user interface, text, application, Word

Description automatically generated

See lines 21-31 of the DocData.java class. Because there is an outside file being read into the code, there is a vulnerability that could be a possible target for attack.

General areas to ensure good secure coding include dependency check for files (see attached picture of code); client/server for the dealing with clients’ personal information; code error and code quality because outside files being read into memory.

## 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

Graphical user interface, text, application, email

Description automatically generated

Vulnerability codes: (Note: all vulnerability codes in the report had their information taken from National Vulnerability Database (NVD))

**bcprov-jdk15on-1.46.jar:**

**CVE-2013-1624**

The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169. **No solution** found (openwall.com). **Published** 2/8/2013, **source** MITRE.

**CVE-2015-6644**

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. No solution given. Published 1/6/2016. Source Android.

**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." **No** **solution** given. **Published** 11/9/2015. **Source** MITRE

**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. **Soultion** apply update, back up existing installation (access.redhat.com). **Published** 6/1/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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. **Solution** apply update, back up existing installation(access.redhat.com). **Published** 6/4/2018. **Source** MITRE

**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." **Solution** disable TLS RSA, apply an update (kb.cert.org). **Published** 12/12/2017. **Source** CERT/CC.

**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. **Solution** apply Critical Patch Update (oracle.com). **Published** 01/14/2022. **Source** MITRE

**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**. No solution** **given**. **Published** 04/21/2021. **Source** CERT/CC.

**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. **Solution** do initial raw RSA decryption calculation simply to do a length check on the payload and then either decrypting a known payload and then returning failure where the original payload turns out to be too small, or passing the payload through for decryption where the payload is the correct size (github.com). **Published** 11/02/2020. **Source** MITRE

**logback-core-1.2.3.jar:**

**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. **Solution** is fixed and auto-remediated via an update to the fixed build. No further action required (security.netapp.com). **Published** 12/16/2021. **Source** Switzerland Government Common Vulnerability Program.

**log4j-api-2.12.1.jar:**

**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. **Solution** upgrade to 2.13.2 (apache.org). **Published** 04/27/2020. **Source** Apache Software Foundation

**CVE-2021-44228**

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects. **No further action needed.** **Published** 12/10/2021. **Source** Apache Software Foundation

**CVE-2021-44832**

Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2. **No further action needed**. **Published** 12/28/2021. **Source** Apache Software Foundation

**CVE-2021-45046**

It was found that the fix to address CVE-2021-44228 in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution in some environments and local code execution in all environments. Log4j 2.16.0 (Java 8) and 2.12.2 (Java 7) fix this issue by removing support for message lookup patterns and disabling JNDI functionality by default. **No further action required**. **Published** 12/14/2021 **Source** Apache Software Foundation

**CVE-2021-45105**

Apache Log4j2 versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3 and 2.3.1) did not protect from uncontrolled recursion from self-referential lookups. This allows an attacker with control over Thread Context Map data to cause a denial of service when a crafted string is interpreted. This issue was fixed in Log4j 2.17.0, 2.12.3, and 2.3.1. **No further action needed. Published** 12/18/2021 **Source** Apache Software Foundation

**snakeyaml-1.25.jar:**

**CVE-2017-18640**

The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564. **No solution given. Published** 12/11/2019. **Source** MITRE

**jackson-databind-2.10.2.jar:**

**CVE-2020-25649**

A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. **Solution** apply update. **Published** 12/03/2020 **Source** Red Hat, Inc.

t**omcat-embed-core-9.0.30.jar:**

**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. **No solution given**. **Published** 02/24/2020. **Source** Apache Software Foundation.

**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. **No solution given**. **Published** 06/26/2020. **Source** Apache Software Foundation

**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. **No solution given**. **Published** 07/14/2020. **Source** Apache Software Foundation.

**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. **No solution given**. **Published** 07/14/2020. **Source** Apache Software Foundation

**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. **No solution given**. **Published** 10/12/2020. **Source** Apache Software Foundation

**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. **No solution given**. **Published** 12/03/2020. **Source** Apache Software Foundation.

**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. **No solution given**. **Published** 02/24/2020. **Source** Apache Software Foundation.

**CVE-2020-1938**

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. **Solution** is to update to new version for security patches and disable connector. **Published** 02/24/2020. **Source** Apache Software Foundation

**CVE-2020-8022**

A Incorrect Default Permissions vulnerability in the packaging of tomcat on SUSE Enterprise Storage 5, OpenStack Cloud Crowbar 8 allows local attackers to escalate from group tomcat to root. **No solution given**, advised to change versions affected. **Published** 06/29/2020. **Source** SUSE

**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. **No solution given**. **Published** 05/20/2020. **Source** Apache Software Foundation.

**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. **No solutions given**. **Published** 01/14/2021. **Source** Apache Software Foundation.

**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. **Solution** is to apply Critical Patch Update. **Published** 03/01/2021 **Source** Apache Software Foundation.

**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. **No solution given.** **Published** 03/01/2021. **Source** Apache Software Foundation.

**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. **Solution** is to update to another version. **Published** 07/12/2021. **Source** Apache Software Foundation

**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. **Solution** is to change versions. **Published** 07/12/2021. **Source** Apache Source Foundation.

**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. **No solution given**. **Published** 09/16/2021. **Source** Apache Software Foundation.

**CVE-2021-42340**

The fix for bug 63362 present in Apache Tomcat 10.1.0-M1 to 10.1.0-M5, 10.0.0-M1 to 10.0.11, 9.0.40 to 9.0.53 and 8.5.60 to 8.5.71 introduced a memory leak. The object introduced to collect metrics for HTTP upgrade connections was not released for WebSocket connections once the connection was closed. This created a memory leak that, over time, could lead to a denial of service via an OutOfMemoryError. **No solution given.** **Published** 10/14/2021. **Source** Apache Software Foundation.

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

**CVE-2020-10693**

A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages**. No solution given. Published** 05/06/2020. **Source** Red Hat, Inc.

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

**CVE-2020-5421**

In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. **Solution** is to remain on actively-supported version and apply security patches without delay. **Published** 09/19/2020. **Source** Pivotal Software, Inc.

**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**. No solution given**. **Published** 01/10/2022. **Source** VMware.

**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**. No solutions given. Published** 10/28/2021. **Source** VMware.

**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. **No solution given. Published** 05/27/2021. **Source** VMware.

**spring-jcl-5.2.3.RELEASE.jar:**

**CVE-2020-5421**

In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. **No solution given, users advised to apply patch updates. Published** 09/19/2020. **Source** Pivotal Software, Inc.

## 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.

From the static testing, the solutions given have been to update/apply security patches. When there were no patches given, then the best course of action would be to switch versions from the affected builds.

From the manual review, the mitigation plan for each outlined areas of security:

* Dependency check for files: Use spring-data-rest-webmvc version 2.7.X or above; reference CVE and NVD databases.
* Client/Server: Denial of Service; Resopurce limit checks should not suffer form integer overflow
* Code Error: Error handling and logging; Ensure log entries that include un-trusted data will not execute as code in the intended log viewing interface or software.
* Code Quality: Memory Management; Double check that the buffer is large as specified and check buffer boundaries if calling the function in a loop. Make sure there is no danger of writing past the allocated space.

**References**

*Add support for specifying an SSL configuration for SmtpAppender*. [LOG4J2-2819] Add support for specifying an SSL configuration for SmtpAppender - ASF JIRA. (n.d.). Retrieved January 23, 2022, from https://issues.apache.org/jira/browse/LOG4J2-2819

Bcgit. (n.d.). *CVE 2020 26939 · bcgit/BC-java wiki*. GitHub. Retrieved January 23, 2022, from https://github.com/bcgit/bc-java/wiki/CVE-2020-26939

*CERT/CC Vulnerability note vu#144389*. VU#144389 - TLS implementations may disclose side channel information via discrepancies between valid and invalid PKCS#1 padding. (n.d.). Retrieved January 23, 2022, from https://www.kb.cert.org/vuls/id/144389

*CERT/CC Vulnerability Note vu#930724*. VU#930724 - Apache Log4j allows insecure JNDI lookups. (n.d.). Retrieved January 23, 2022, from https://www.kb.cert.org/vuls/id/930724

*CVE-2021-42550 Logback vulnerability in NetApp products*. CVE-2021-42550 Logback Vulnerability in NetApp Products | NetApp Product Security. (n.d.). Retrieved January 23, 2022, from https://security.netapp.com/advisory/ntap-20211229-0001/

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2013-1624

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2015-7940

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2015-6644

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000338

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000339

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000341

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000342

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000343

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000344

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000345

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000346

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2016-1000352

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2017-13098

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2018-1000613

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2018-5382

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-26939

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-42550

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-9488

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-44228

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-44832

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-45105

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-45046

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2017-18640

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-25649

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2019-17569

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-11996

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-13934

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-13935

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-13943

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-17527

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-1935

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-8022

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-9484

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-24122

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-25122

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-25329

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-30640

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-33037

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-41079

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-42340

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-10693

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-5421

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-22060

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-22096

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-22118

NVD. (n.d.). Retrieved January 23, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-5421

Oracle Critical Patch Update Advisory - April 2019. (n.d.). Retrieved January 23, 2022, from https://www.oracle.com/security-alerts/cpuapr2019.html

Oracle Critical Patch Update Advisory - January 2021. (n.d.). Retrieved January 23, 2022, from https://www.oracle.com/security-alerts/cpujan2021.html

*Oss-security - re: CVE request: TLS CBC padding timing flaw in various SSL / TLS implementations*. Openwall. (n.d.). Retrieved January 23, 2022, from https://www.openwall.com/lists/oss-security/2013/02/05/24

*RHSA-2018:2669 - Security Advisory*. Red Hat Customer Portal. (n.d.). Retrieved January 23, 2022, from https://access.redhat.com/errata/RHSA-2018:2669