****

# 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 5](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan 16](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/13** | **Raphael Coloma** | **Initial assessment** |

## 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

Raphael Coloma

## Interpreting Client Needs

Secure communication between the client and Artemis Financial is paramount. Since Artemis Financial will be dealing with sensitive data, it is crucial that the client’s information remains confidential. Data breaches and information leaks will damage the integrity and reputation of Artemis Financial, affecting current and future customers. Reliability and trust are key components to maintaining and growing clientele.

Though it is not explicitly stated, it is highly possible that Artemis Financial will have foreign clients and deal with international transactions. It is important that the application adheres to international regulations and privacy laws when dealing with global clients. Another restriction to consider is the Graham-Leach-Bliley Act. Under the Graham-Leach-Bliley Act, the information-sharing practices of Artemis Financial must be publicly disclosed and consumer data must be kept secure and protected.

Artemis Financial currently has a RESTful web application programing interface (API). A common external threat to RESTful API security is injection attacks, most notably SQL injections. Untrusted data is “injected” into the API as part of a query or command and is implemented by the interpreter allowing the attacker unauthorized access to the application or database. Another common threat comes with the authentication process and access controls for customers and employees. Missing or inadequate authentication and access controls can allow attackers to access/manipulate user accounts and data.

To modernize their operations, Artemis Financial will need to ensure the all open-source libraries and frameworks for its web application are up to date and periodically maintained. In addition, should Artemis Financial choose to move to cloud computing/databasing, the APIs used are properly coded/implemented to ensure secure and intentional requests for cloud systems.

## Areas of Security

**Input Validation**

User input to the web application should be validated. Input validation limits what can be accepted by the web application. It ensures that only appropriate data is passed and prevents exploitation using input fields. Since user input is expected, the web application should validate all data entered to protect against possible exploits

**Secure API**

APIs allow third-party outsiders to access data through an endpoint. Secure APIs authenticate user requests and respond with applicable access controls for the user. The web applications should authenticate users accessing the application and assign the appropriate authority level for access.

**Cyptopgraphy**

Sensitive data requires encryption at rest or in transit. Encrypting sensitive data ensures that data that is stolen or intercepted cannot is still protected and cannot be read. Since sensitive data will be transmitted from the client to the server and vice versa, steps should be taken to ensure that that information is encrypted.

**Code Quality**

As time goes on, developers have learned from previous attacks how to recognize weak coding practices and patterns. Ensuring code quality by implementing proper coding practices and patterns helps to bypass known exploits and vulnerabilities.

## Manual Review

I started in the pom.xml file to check the framework and dependencies as well as their versions. The Maven Dependency-Check Plugin will evaluate these for potential vulnerabilities. I was not able to determine any input validation while reviewing the code. User input is not limited and not specified to data types which can lead to injection vulnerabilities. There does not appear to be any form of authentication for users accessing the application. The customer class initializes account\_number and account\_balance variables, but there are no requests for usernames or passwords. In that same vein, there are no access controls implemented. Accessing the applications grants users unrestricted access to the application and its data. Running the applications, were no error handling codes, so crashes were met with a generic 404 page with no available messages. The application does not appear to be using a Secure Sockets Layer (SSL) and Transport Layer Security (TLS) to ensure information is encrypted during requests and responses. The DocData class does not parameterize its queries. Raw user input is used in the “mysql” query allowing injection vulnerabilities.

## Static Testing

**bcprov-jdk15on-1.46.jar**

**Vuln ID**: CVE-2013-1624

The TLS implementation in the Bouncy Castle Java library 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.

**Recommended Solution**: Update to current version

**References:** MISC - <http://www.isg.rhul.ac.uk/tls/TLStiming.pdf>

**Vuln ID**: 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.

**Recommended Solution**: Require users to update to the latest version of Android

**References:** OSSINDEX - [[CVE-2015-6644] CWE-200: Information Exposure](https://ossindex.sonatype.org/vulnerability/CVE-2015-6644?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: CVE-2015-7940 (OSSINDEX)

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."

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2015-7940] CWE-200: Information Exposure](https://ossindex.sonatype.org/vulnerability/CVE-2015-7940?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/b0c3ce99d43d73a096268831d0d120ffc89eac7f#diff-3679f5a9d2b939d0d3ee1601a7774fb0>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/413b42f4d770456508585c830cfcde95f9b0e93b#diff-54656f860db94b867ba7542430cd2ef0>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/acaac81f96fec91ab45bd0412beaf9c3acd8defa#diff-e75226a9ca49217a7276b29242ec59ce>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/843c2e60f67d71faf81d236f448ebbe56c62c647#diff-25c3c78db788365f36839b3f2d3016b9>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/50a53068c094d6cff37659da33c9b4505becd389#diff-5578e61500abb2b87b300d3114bdfd7d>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/9385b0ebd277724b167fe1d1456e3c112112be1f>

**Vuln ID**: CVE-2016-1000345

In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES CBC mode vulnerable to padding oracle attack. 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/21dcb3d9744c83dcf2ff8fcee06dbca7bfa4ef35#diff-4439ce586bf9a13bfec05c0d113b8098>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/1127131c89021612c6eefa26dbe5714c194e7495#diff-d525a20b8acaed791ae2f0f770eb5937>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://github.com/bcgit/bc-java/commit/9385b0ebd277724b167fe1d1456e3c112112be1f>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** BID - [103453](http://www.securityfocus.com/bid/103453)

**Vuln ID**: 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

**Recommended Solution**: Require users to update to the latest version of Android

**References:** OSSINDEX - [[CVE-2020-0187] CWE-310](https://ossindex.sonatype.org/vulnerability/CVE-2020-0187?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2020-26939] CWE-203: Information Exposure Through Discrepancy](https://ossindex.sonatype.org/vulnerability/CVE-2020-26939?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**logback-core-1.2.3.jar**

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <http://logback.qos.ch/news.html>

**log4j-api-2.12.1.jar**

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://issues.apache.org/jira/browse/LOG4J2-2819>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CERT-VN - [VU#930724](https://www.kb.cert.org/vuls/id/930724)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CISCO - [20211210 Vulnerabilities in Apache Log4j Library Affecting Cisco Products: December 2021](https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-apache-log4j-qRuKNEbd)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CERT-VN - [VU#930724](https://www.kb.cert.org/vuls/id/930724)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CERT-VN - [VU#930724](https://www.kb.cert.org/vuls/id/930724)

**snakeyaml-1.25.jar**

**Vuln ID**: CVE-2017-18640

The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation.

**Recommended Solution**: Update to current version

**References:** FEDORA - [FEDORA-2020-23012fafbc](https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/PTVJC54XGX26UJVVYCXZ7D25X3R5T2G6/)

**Vuln ID**: CVE-2021-4235

Due to unbounded alias chasing, a maliciously crafted YAML file can cause the system to consume significant system resources. If parsing user input, this may be used as a denial of service vector.

**Recommended Solution**: Bound alias chasing

**References:** MISC - <https://github.com/go-yaml/yaml/commit/bb4e33bf68bf89cad44d386192cbed201f35b241>

**Vuln ID**: CVE-2022-1471 (OSSINDEX)

SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution.

**Recommended Solution**: Only use SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization.

**References:** OSSINDEX - [[CVE-2022-1471] CWE-502: Deserialization of Untrusted Data](https://ossindex.sonatype.org/vulnerability/CVE-2022-1471?component-type=maven&component-name=org.yaml%2Fsnakeyaml&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - [bitbucket.org](https://bitbucket.org/snakeyaml/snakeyaml/commits/fc300780da21f4bb92c148bc90257201220cf174)

**Vuln ID**: CVE-2022-3064

Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory.

**Recommended Solution**: Limit YAML document size when parsing

**References:** MISC - <https://github.com/go-yaml/yaml/commit/f221b8435cfb71e54062f6c6e99e9ade30b124d5>

**Vuln ID**: CVE-2022-38749, CVE-2022-38750, CVE-2022-38752, CVE-2022-41854 & 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.

**Recommended Solution**: Validate YAML files before parsing

**References:**

MISC - <https://bitbucket.org/snakeyaml/snakeyaml/issues/525/got-stackoverflowerror-for-many-open>

MISC - <https://bitbucket.org/snakeyaml/snakeyaml/issues/526/stackoverflow-oss-fuzz-47027>

MISC - <https://bitbucket.org/snakeyaml/snakeyaml/issues/530/stackoverflow-oss-fuzz-47039>

MISC - <https://bitbucket.org/snakeyaml/snakeyaml/issues/531/stackoverflow-oss-fuzz-47081>

CONFIRM - <https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=50355>

**jackson-databind-2.10.2.jar**

**Vuln ID**: 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.

**Recommended Solution**: Secure entity expansion

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20210108-0007/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20220506-0004/>

**Vuln ID**: 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

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20221124-0004/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20221118-0008/>

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

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20200327-0005/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://lists.apache.org/thread.html/r5541ef6b6b68b49f76fc4c45695940116da2bcbe0312ef204a00a2e0%40%3Cannounce.tomcat.apache.org%3E>

**Vuln ID**: CVE-2020-11996

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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20200724-0003/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://kc.mcafee.com/corporate/index?page=content&id=SB10332>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20201016-0007/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20201210-0003/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20200327-0005/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <http://support.blackberry.com/kb/articleDetail?articleNumber=000062739>

**Vuln ID**: CVE-2020-8022

An 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://bugzilla.suse.com/show_bug.cgi?id=1172405>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://kc.mcafee.com/corporate/index?page=content&id=SB10332>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20210212-0008/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - [https://lists.apache.org/thread/7vykb4mjgvj7lvzng7n9b8z40b768c29](https://lists.apache.org/thread.html/r7b95bc248603360501f18c8eb03bb6001ec0ee3296205b34b07105b7%40%3Cannounce.tomcat.apache.org%3E)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - [https://lists.apache.org/thread/hd79q54gvvrc6p7yvh8bofkygkvlfzy3](https://lists.apache.org/thread.html/rfe62fbf9d4c314f166fe8c668e50e5d9dd882a99447f26f0367474bf%40%3Cannounce.tomcat.apache.org%3E)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20210827-0007/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://kc.mcafee.com/corporate/index?page=content&id=SB10366>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20211008-0005/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** DEBIAN - [DSA-5265](https://www.debian.org/security/2022/dsa-5265)

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://security.netapp.com/advisory/ntap-20220629-0002/>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** CONFIRM - <https://lists.apache.org/thread/k04zk0nq6w57m72w5gb0r6z9ryhmvr4k>

**Vuln ID**: 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.

**Recommended Solution**: Update to current version

**References:** MISC - <https://lists.apache.org/thread/zzcxzvqfdqn515zfs3dxb7n8gty589sq>

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

**Vuln ID**: CVE-2016-1000027 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2016-1000027] CWE-502: Deserialization of Untrusted Data](https://ossindex.sonatype.org/vulnerability/CVE-2016-1000027?component-type=maven&component-name=org.springframework%2Fspring-web&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: CVE-2020-5421 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2020-5421] CWE-20: Improper Input Validation](https://ossindex.sonatype.org/vulnerability/CVE-2020-5421?component-type=maven&component-name=org.springframework%2Fspring-web&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: CVE-2021-22096 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2021-22096] CWE-117: Improper Output Neutralization for Logs](https://ossindex.sonatype.org/vulnerability/CVE-2021-22096?component-type=maven&component-name=org.springframework%2Fspring-web&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**Vuln ID**: CVE-2021-22118 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2021-22118] CWE-668: Exposure of Resource to Wrong Sphere](https://ossindex.sonatype.org/vulnerability/CVE-2021-22118?component-type=maven&component-name=org.springframework%2Fspring-web&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**spring-beans-5.2.3.RELEASE.jar**

**Vuln ID**: CVE-2022-22965 (OSSINDEX)

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.

**Recommended Solution**: Ensure Spring Boot is run only as an executable jar

**References:** OSSINDEX - [[CVE-2022-22965] CWE-94: Improper Control of Generation of Code ('Code Injection')](https://ossindex.sonatype.org/vulnerability/CVE-2022-22965?component-type=maven&component-name=org.springframework%2Fspring-beans&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**spring-webmvc-5.2.3.RELEASE.jar**

**Vuln ID**: CVE-2021-22060 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2021-22060] CWE-117: Improper Output Neutralization for Logs](https://ossindex.sonatype.org/vulnerability/CVE-2021-22060?component-type=maven&component-name=org.springframework%2Fspring-webmvc&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**spring-context-5.2.3.RELEASE.jar**

**Vuln ID**: CVE-2022-22968 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2022-22968] CWE-178: Improper Handling of Case Sensitivity](https://ossindex.sonatype.org/vulnerability/CVE-2022-22968?component-type=maven&component-name=org.springframework%2Fspring-context&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

**spring-expression-5.2.3.RELEASE.jar**

**Vuln ID**: CVE-2022-22950 (OSSINDEX)

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.

**Recommended Solution**: Update to current version

**References:** OSSINDEX - [[CVE-2022-22950] CWE-770: Allocation of Resources Without Limits or Throttling](https://ossindex.sonatype.org/vulnerability/CVE-2022-22950?component-type=maven&component-name=org.springframework%2Fspring-expression&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0)

## Mitigation Plan

* The first step to ensuring the web application is secure and the data stored is protected is to implement user authentication. Users should have password protect accounts. This safeguards that user data by validating the user’s identity before granting access to their sensitive data.
* Once user authentication is created, access controls should be implemented. Access controls guarantee that users are allowed access to functions and data appropriate for what is needed following the principle of least privilege. Access controls create restrictions to protect the users’ data and the application.
* Input validation would be the next step to protecting the application and sensitive data. Input fields requiring untrusted data should be limited to protect from possible exploits. Data input should conform to patterns anticipated by the developer to defend from injection attacks.
* Error handling code should be implemented to deal with potential threats. Points for potential errors in the code should be contained to provide controllable outcomes and insight into the error.
* Since the web application will be transmitting important information, sensitive data should be encrypted using SSL and TLS to ensure privacy and protection.
* Implement parameterized queries into SQL with prepared statements. By binding untrusted data into placeholders within a query, the web application will be protected from SQL injection.
* Many of the dependence vulnerabilities stem from previous library versions. Please see the recommended solution for each vulnerability ID. We need to ensure that the framework and libraries being used by the application are up to date and maintained. This will protect the application from possible exploits. There still exist a few dependence vulnerabilities that require configuration changes or users to use the update their version of Android.