****

# 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** | **March 18, 2023** | **Maurice Wesley** | **Dependency Check** |
| **1.1** | **March 19, 2023** | **Maurice Wesley** | **Interpreting Need** |
| **1.2** | **March 19, 2023** | **Maurice Wesley** | **Static Review** |
| **1.3** | **March 19, 2023** | **Maurice Wesley** | **Areas of Security** |
| **1.4** | **March19, 2023** | **Maurice Wesley** | **Mitigation Plan** |

## Client



## Developer

Maurice Wesley

## Interpreting Client Needs

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

At Global rain, we believe that security is for everyone. Artemis Financial develops individualized financial plans for their customers. These plans include savings, retirement, investments, and insurance. Artemis Financial budgets, allocates, and redistributes assets across multiple portfolios for their clients.

I looked at the SANS website that the professor provided for security research. I may have to create an account to access the content. Additionally, I have to agree to be contacted by a third party to access the content. Security is an intangible asset that is vital to the company’s success. Defined for the purposes of this prompt, secure communications mean “protection against attackers that are trying to either steal, tamper or block the information being sent” (Llanillo, 2018). The amount of security is positively correlated with the amount of risk and the severity of the risk is positively correlated the amount revenue lost, death, or injury. The loss of money and/or information can decrease the company’s revenue and decrease consumer confidence.

Conversely, security adds value to the company. If the consumer is confident that Artemis can conduct secure business transactions or that their financial investments are secure, then the consumer will invest more or recommend the company to a friend. Word of mouth advertising can generate repeat business, new customers, and new investments.

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

The scenario presented in Project One does not explicitly state whether the company conducts international transactions or where the customers are located. I will have to make an inference from the Global Rain company profile. The company provides software for entrepreneurs, businesses, and government agencies around the world. Artemis Financial is employing Global Rain’s services. That being said, I am inclined to believe Artemis Financial performs international transactions.

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

**International**

**Payment Card Industry Data Security Standard**: This data security standard is mandatory for organizations processing credit card data. Theses requirements cover the storing, processing, and transferring of payment card data. The application will have to implement these requirements if the customers will make investments or purchase insurance using a credit/debit card.

**Customer Security Programme**: A global security mandate specifying national and international cybersecurity requirements. Any financial institution that uses SWIFT services must complete an independent assessment from a third-party. Additionally, Artemis Financial must able to attest to their compliance against CSCF v2023.

It would be advantageous for Artemis Financial to leverage SWIFT’s infrastructure to implement instruction matching for treasury and forex transactions, processing payment instructions between banks, and the securities infrastructure for investments. In 2022, SWIFT’s market share was comprised of 11000 institutions from more than 200 countries. Additionally, SWIFT can impose sanctions that are in line with governmental requirements. Artemis Financial could enhance their global footprint with a proven leader in messaging (SWIFT, 2022).

**National**

**Gramm-Leach-Bliley Act**: An act passed by congress that requires companies, like Artemis Financial, that provides financial, investment, and insurance to explain the company’s information-sharing practices to their customers and to safeguard sensitive data (FTC, n.d.). Similarly, the Act establishes strict data access policies and regulates the way financial institutions handles customers’ private data.

**Sarbanes-Oxley Compliant**: Artemis Financial will be registered by the US Securities and Exchange Commission. It is mandatory for the company to follow the recommended practices to prevent fraudulent financial transactions. Concurrently, SOC has guidelines for the storage management, documentation, and protection of financial records.

**FINRA**: Artemis Financial engages in investment and insurance activities. These activities require brokers to facilitate these transactions. The FINRA is mandatory for all brokers in the United States. The FINRA identifies rules for detecting and mitigating cyber threats. Also, it directs companies to establish a written data protection policy.

**PSD 2**: Mandatory in the European Union, the PSD 2 regulates the way electronic payments are initiated and processed. Additionally, the mandate establishes strict rules for protecting customers’ private data.

**BSA**: Establishes the prevention and the notification by financial institutions for activities related to terrorist financing, money laundering, and tax evasion. Artemis Financial will need to develop an incident response plan that addresses cyber-related crimes.

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

In the first half of 2020, cyberattacks on financial institutions increase by 238% year over year with an average cost of $5.72 million per breach. Kaushik Sen covered the top 6 cyber threats to financial services in 2023.

**Phishing**: Stephen Moramarco wrote that an estimated 90% of all successful hacking scams are initiated with a phishing attack. Furthermore, “approximately 15 billion spam/phishing emails are sent per day with nearly half of which target or impersonate financial institutions” (Moramarco, 2019).

Attackers send seemingly official emails to employees/customers that contain malware in an attempt to obtain information and/or login credentials. The development team can implement secure coding practices but cannot prevent human error. Training, education, and control measures will need to be integrated into the development cycle and after deployment.

**Ransomware**: This cyber attack is an extension of phishing attacks. After an attacker gains access to an operating platform, system, or information, the user is either lock out of the system or threats are made unless a payment is made to the attacker.

**SQL Injections, XXS, Local File Inclusion**: These attacks are associated with common platform vulnerabilities and exposures. Generally, the attacks can be attributed to improper certificate validation, vulnerabilities in dependencies, and/or the algorithmic design of certain modes/modules. The preceding is not an exhaustive list. However, the weaknesses allow an attacker to create malicious files to perform remote code execution, privilege escalation, and/or obtain sensitive data.

**Denial of Service**: When certain dependency vulnerabilities are discovered, an attacker can cause denial of service. There are several forms of an attack that will cause DoS. When the parser is dependent on user input, data binding, or an application improperly handles file uploads, the attacker crafts malicious input that will cause a buffer overflow. The buffer overflow can cause the server to become slow or unresponsive. DoS was one of the two major threats to payment processes in 2020.

**Supply Chain Attacks**: An attack that occurs on system through a third-party vendor or source. The vendor generally does not share the same security concerns as the financial institution nor do not implement the same security checks. A single vendor can store or have access to sensitive information for all of its clients. I mentioned a Zero Trust architecture in a previous assignment. Irrespective of dependency vulnerabilities, Zero Trust puts every user in an untrusted state and only allows the user to perform certain activities. This will prevent privilege escalation and other attacks on financial institutions.

**Bank Drops**: A mandatory requirement for institutions in the US to monitor and notify any fraudulent activities. Attackers obscure their location and identity by obtaining another individual’s personal credentials. Afterwards, they store stolen funds in a fake bank account under the stolen identity. Kaushik mentions that the response to the threat is to implement controls that focus purely on credentials associated with opening new accounts.

**What are the modernization requirements that you must consider? For example:**

**The role of open-source libraries**: Developers are put under tremendous pressure to create complex applications in a short period of time. Based on this external pressure to perform, many developers rely on open-source libraries. OS drives modernization, innovation, and promote collaboration among developers. The libraries provide a framework for developers to quickly adapt to evolving web applications (see prompt below). The collaboration among developers leads to more productivity, flexibility, and lower costs.

Artemis Financial is currently using a legacy system. It would be cost effective for the development team to use an open-source library that works well with legacy digital transformations. Stated in an article by IBM, “an open-source toolbox is fundamental for programmers and IT managers and include a variety of utilities including technology refreshes, compression algorithms, cryptography libraries, development tools and languages” (Brooks, 2021). Additionally, open-source libraries aid in cybersecurity as collaboration helps catch and mitigate bugs after deployment.

**Evolving web application technologies**: Web applications have evolved from web pages to complex business applications for business to business and business to customer transactions/interactions. Consideration must be given to using an open-source framework to build an application that is supported in all web browsers. As of 2020, the trending frameworks for building modern web applications include: React, Angular, and jQuery. Also, the development team should consider the size of the application, memory requirements, screen size, and performance on different devices.

As of 2020, Python was the most admired open-source language to conduct back-end development. Artemis financial will need to track application use, create pipelines, and perform general updates and maintenance. The API’s library of known threats and vulnerabilities will need to be continuously monitored and tested (Nehra, 2022).

## Areas of Security

I was recently promoted to Global Rain’s new agile scrum team. I have been tasked with examining Artemis Financials’ web-based software application to identify any security vulnerabilities. As the agile security officer for the financial company, I need to ensure that the organization’s use of secure communications channels to domestic and international customers meets government export regulations.

**Cryptography**: According to the resource material in module one, I am inclined to investigate cryptography vulnerabilities because it is an export item. Additionally, my conclusion is based on the previous prompt when I had to consider governmental restrictions on secure communications.

## Manual Review

**Class CRUD Controller**

**SQL Injection:** @RequestMapping

The Spring model view controller automatically bind request parameters to beans declared as arguments of methods if they are annotated with @RequestMapping. Subsequently, it is possible to insert malicious input in the arguments of the @RequestMapping annotated methods. Conversely, @Entity or @Document objects are linked to the underlying database and updated automatically by the framework that Artemis Financials implements (Spring MongoDB).

The above conditions can lead to a malicious attack: if the persistent objects are used as an argument in a method annotated with @RequestMapping, it is possible to construct user input injections, to change the content in the database fields. “The use of @Entity or @Document objects as arguments in methods annotated with @RequestMapping should be avoided. In addition to @RequestMapping, this rule also considers the annotations introduced in Spring Framework 4.3: @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping” (Sonarsource, 2023).

To mitigate the @RequestMapping vulnerabilities, it is recommended declare a string variable and initialize it to parameterized request containing a request.getParameter() statement with “user” or “pass” passed as arguments. The string variables are passed into a query statement. The statement would not be vulnerable to SQL injection.

**Class Greeting Controller**

**Unrestricted View Name Manipulation**: @GetMapping

An article mentioned that there “may be a situation when a template name or a fragment are concatenated with untrusted data” (Stepankin, 2020). The public method in the Greeting Controller uses @RequestParam with a value, a default value, and an argument. The return statement in the method is susceptible to untrusted data and the template path is “tainted”. This leads to a possible path traversal vulnerability. If the method contains a parser expression, then the method would be susceptible to an expression language injection vulnerability (Stepankin, 2020).

**Doc Data Read Document method the JDBC is not secure**

The current code connects to a database, but it does not establish a secure connection. The financial institution needs a secure connection. The recommended method to set up a secure connection parameter that includes: the database name, integrated security, encryption, trust server certificate, trust store, trust store password, and host name in certificate.

**Database Name**: Specifies the name of the database to connect to. If it is not stated, then the connection is made to the default database.

**Integrated Security**: Will be set to true which instructs the JDBC driver to search the local computer credential cache for credentials that were provided when a user signed into the computer or network.

**Encrypt**: Will be set to true for the SQL server to use TLS encryption for all the data sent between the client and the server. The encrypt parameter works in conjunction with the trust server certificate parameter.

**Trust Server Certificate**: It is set to true when the server transport layer security is automatically trusted when the communication layer is encrypted using TLS. The financial institution will want the TSC set to false and have the driver validate the server TLS certificate.

**Trust Store**: Specifies the path to the certificate trust store file. It must be set with a trust store password.

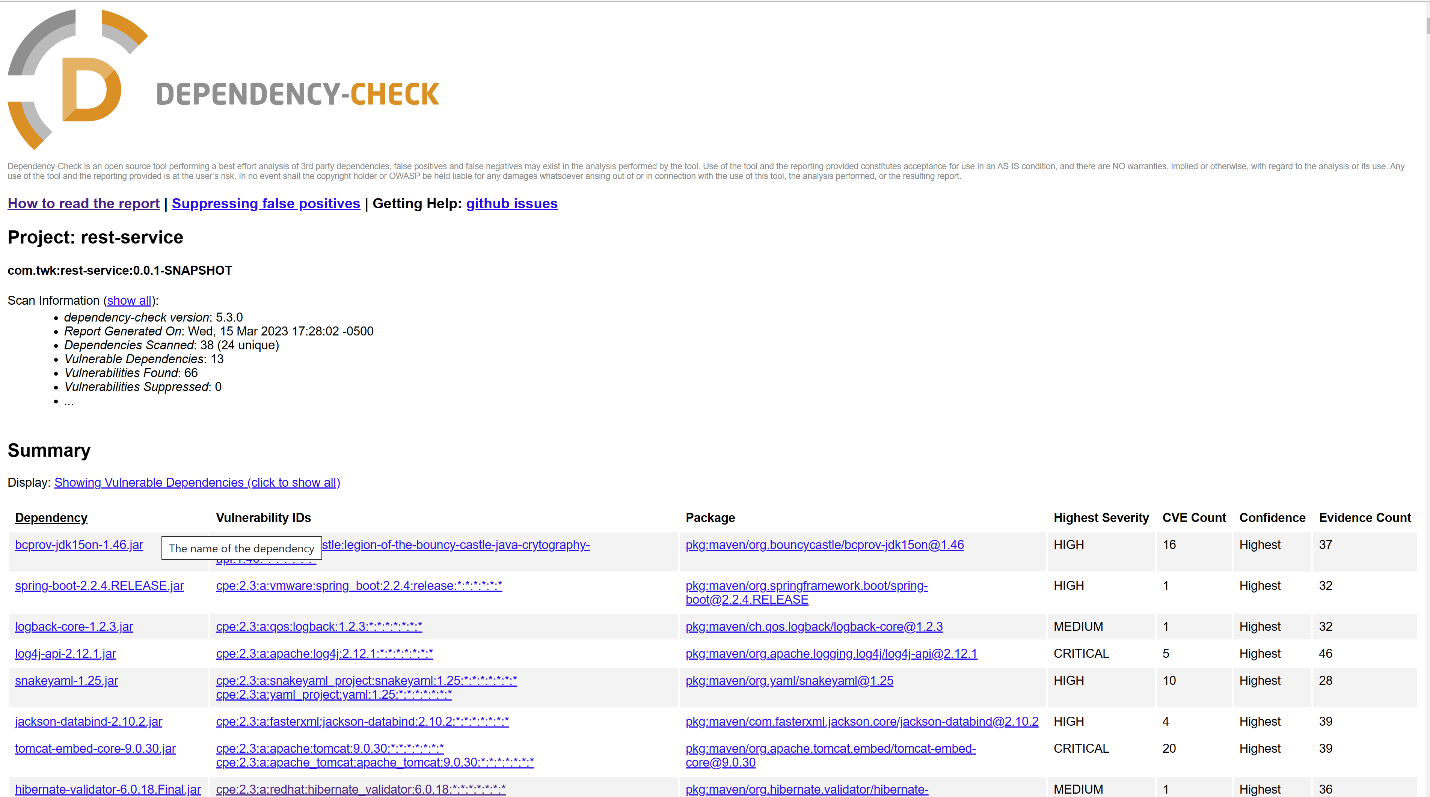
**Trust Store Password**: Specifies the password used to check the integrity of the trust store data.

**Host Name in Certificate**: Specifies the host name of the server.

**Customer Java File Customer Class**

The class member variable account balance is properly declared. The account number is listed private, but the account balance variable does not have an access modifier. For proper encapsulation, the account balance variable should be declared private to prevent unauthorized access.

## Static Testing

****

**Bcprov-jdk15on-1.46.jar**

Description**:** A java implementation package that provides a java cryptographic extension for JDK 1.5 – 1.7.

Severity: High

Vulnerabilities:

***CVE-2013-1624***: The transport layer security implementation in Bouncy java library before version 1.48 improperly conders timing side-channel attacks on noncompliant media access control check operation when processing malformed cypher block chaining. This vulnerability allows attackers to conduct distinguishing attacks using statistical analysis of timing data for crafted packets.

***CVE-2015-6644***: A crafted application allowed attackers to obtain sensitive information.

***CVE-2015-7940***: Bouncy Castle Java library before 1.51 does not validate a point that is withing the elliptic curve. This vulnerability allows attackers to obtain private keys using crafted elliptic curves.

***CVE-2016-1000338***: Does not fully validate the abstract syntax notation encoding of signatures. It allows attackers to inject extra elements in the sequence or invisible data into a signed signature.

***CVE-2016-1000339***: The table-driven algorithmic approach of the AESFastEngine combined with the data channel on the CPU being monitored, the lookup table accesses are sufficient to leak information on the AES key being used.

***CVE-2016-1000341***: Bouncy version 1.55 and earlier are vulnerable to timing attacks. Attackers may be able to obtain information about the signature’s k value and private value as well.

***CVE-2016-1000342***: Similar to stated above, the java cryptic extension does not fully validate the abstract syntax notation encoding of signature on verification.

***CVE-2016-1000343***: Bouncy key pair generator generates a weak private key if used with the default values. Solution would be to explicitly pass parameters to the key pair generator.

***CVE-2016-1000344***: Bouncy ECB mode is regarded as unsafe and support for it has been removed from the provider.

***CVE-2016-1000345***: The CBC mode of Bouncy is vulnerable to padding oracle attack. Bouncy version 1.55 and earlier environments timings can be observed enabling the attacker to identify when the decryption is failing due to padding.

***CVE-2016-1000346***: BC version 1.55 and earlier does not fully validate the DH public key. The attacker could use invalid keys to reveal details about the other party’s private key. This can be fixed with release 1.56.

***CVE-2016-1000352***: BC version 1.55 and earlier allowed the used of ECB mode which is unsafe and no longer supported by the provider

***CVE-2017-13098***: The java cryptographic extension in BC provides a weak Bleichenbacher oracle when any transport layer security cipher suite using RSA key exchange is negotiated.

***CVE-2018-5382***: The default BKS uses a HMAC that is only 16 bits long. This allows an attacker to compromise the integrity of the BKS keystone.

***CVE-2020-0187***: The base block cipher with an incomplete comparison may chose an incorrect cryptographic algorithm that could lead to local information disclosure with no additional execution privileges needed.

***CVE-2020-26939***: In BC before 1.61, the application could potentially leak some information about the private exponent of the RSA private key performing the encryption by sending invalid.

**Sprint-boot-2.2.4.RELEASE.jar**

Description**:** Spring Boot

Severity: High

Vulnerabilities:

***CVE-2022-27772***: Spring boot version prior to 2.2.11.RELEASE is vulnerable to temporary directory hijacking. Fortunately, this vulnerability can no longer be exploitable because this version of spring boot is no longer supported.

**Logback-core-1.2.3.jar**

Description**:** Log back core module

Severity: Medium

Vulnerabilities:

***CVE-2021-42550***: Version prior to and including 1.2.7 allowed an attacker with the required privileges to edit configuration files to craft malicious configurations allowing them to execute arbitrary code loaded from LDAP servers.

**Log4j-api-2.12.1.jar**

Description**:** The Apache Log4j API

Severity: Critical

Vulnerabilities:

***CVE-2020-9488*:** Apache Log4j SMTP appender does not properly validate certificate with host mismatch. An attack could intercept log messages sent through that appender.

***CVE-2021-44228***: Apache Log4j2 through 2.15.0 excluding some versions, an attacker who can control log messages or log message parameters can execute arbitrary code loaded form LDAP servers when message lookup substitution is enabled. The functionality has been completely removed in updated versions.

***CVE-2021-44832***: When a configuration uses a JDBC Appender with a JNDI LDAP data source URI, the application is vulnerable to remote code execution. The mitigation is to limit JNDI data source names to the java protocol.

***CVE-2021-45046***: A previous fix to CVE-2021-44228 was incomplete. The attackers were able to craft malicious input data to perform information leak, remote code execution, and local code execution.

***CVE-2021-45105***: Apache versions did not protect against uncontrolled recursion from self-referential lookups. An attacker could cause denial of service with maliciously crafted strings during interpretation.

**Snakeyaml-1.258.jar**

Description**:** The YAML 1.1 parser and emitter for Java

Severity: High

Vulnerabilities:

***CVE-2017-18640***: During load operation, it allows entity expansion reference

***CVE-2021-4235***: A maliciously crafted YAML file and parsing user input could cause denial of service vector.

***CVE-2022-1471***: Constructor does not restrict types during deserialization instantiation and can enable remote code execution.

***CVE-2022-25857***: Missing depth limitations for collections can cause denial of service

***CVE-2022-3064***: Parsing malicious YAML files can consume excessive amounts of CPU or memory and cause denial of service.

***CVE-2022-38749***: The attacker can supply content that causes stack overflow when the parser is dependent on user input.

***CVE-2022-38750***: Another instance of denial of service when the parser is dependent on user input.

***CVE-2022-38751***: Similar to the above, an attacker can cause denial of service when the parser is dependent on user input.

***CVE-2022-38752***: Another instance of denial of service when the parser is dependent on user input.

***CVE-2022-41854***: Allows stack overflow when the parser is running on user supplied input.

**Jackson-databind-2.10.2.jar**

Description**:** General Data Binding functionality for Jackson

Severity: High

Vulnerabilities:

***CVE-2020-25649***: The flaw enabled external entity attacks and threatened data integrity.

***CVE-2020-36518***: Allowed to many executions to be stored on the stack and cause denial of service.

***CVE-2022-42004***: Jackson-databind before 2.13.4, similar to above, does not prevent the use of deeply nested arrays. Some customized deserializations make the application vulnerable.

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

Description**:** Core tomcat implementation

Severity: Critical

Vulnerabilities:

***CVE-2019-17569***: When Tomcat is located behind a reverse proxy, certain versions introduced a regression enabling request smuggling.

***CVE-2020-11996***: A specialized sequence of HTTP requests could trigger high CPU usage over several seconds. With enough requests, this action could cause the server to become unresponsive.

***CVE-2020-13934***: Repeated HTTP requests could cause OutOfMemoryException leading to a DoS.

***CVE-2020-13935***: Invalid payload lengths could trigger an infinite loop. Thus, multiple requests could cause a DoS.

***CVE-2020-13943***: When a maximum number of concurrent streams for a connection was exceeded, a subsequent request could expose unexpected resources.

***CVE-2020-17527***: The re-use of an HTTP request header could cause information leak in between requests.

***CVE-2020-1935***: The end-of-line parsing allowed some invalid HTTP headers to be parsed as valid when Tomcat was located behind a reverse proxy.

***CVE-2020-1938***: An attacker was able to use the default AJP connector to upload files and then web application would be susceptible to remote code execution.

***CVE-2020-8022***: Tomcat had an incorrect default permissions vulnerability in a plethora of versions.

***CVE-2020-9484***: Enabled an attacker to implement remote code execution when certain conditions are met.

***CVE-2021-24122***: Tomcat disclosed the source code in some configuration.

***CVE-2021-25122***: Certain Apache Tomcat versions could duplicate request headers causing information leakage.

***CVE-2021-25329***: A previous fix for CVE-2020-9484 was incomplete

***CVE-2021-33037***: Tomcat did not properly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility of request smuggling.

***CVE-2021-30640***: Lock Out Realm allowed an attacker to authenticate using variations of a valid user name and could bypass some of the protections.

***CVE-2021-41079***: Apache Tomcat improperly validated incoming transport layer security packets. A malicious packet could cause an infinite loop causing DoS.

***CVE-2021-43980***: A concurrency bug in Apache Tomcat caused responses, or part responses, to be received by the wrong client.

***CVE-2022-29885***: The documentation incorrectly stated that the EncryptInterceptor enabled clustering over an untrusted network. It is susceptible to DoS when on an untrusted network.

***CVE-2022-34305***: The authentication example displayed user provided data without filtering which exposed a cross site scripting vulnerability.

***CVE-2022-42252***: Tomcat did not correctly reject an invalid Content-Length header. This made it possible for an attacker to smuggle an attack.

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

Description**:** Hibernate’s Bean Validation reference implementation

Severity: Medium

Vulnerabilities:

***CVE-2020-10693***:

**Spring-web-5.2.3.RELEASE.jar**

Description**:** Spring Web

Severity: High

Vulnerabilities:

***CVE-2016-1000027***: Depending on how the Pivotal Spring Framework is used, it enables the potential for remote code execution if used for deserialization of untrusted data.

***CVE-2020-5421***: Mentioned earlier, the security patch for CVE-2015-5211 can be bypassed depending on the browser used with jsessionid path parameter.

***CVE-2021-22096***: Enabled the insertion of additional log entries.

***CVE-2021-22118***: The Web Flux application is vulnerable to a privilege escalation.

**Spring-beans-5.2.3.RELEASE.jar**

Description**:** Spring Beans

Severity: High

Vulnerabilities:

***CVE-2022-22968***: The case sensitivity was not applied to the first character of the field in all nested fields within the property path.

**Spring-webmvc-5.2.3.RELEASE.jar**

Description**:** Spring Web MVC

Severity: Medium

Vulnerabilities:

***CVE-2021-22060***: Enable malicious input that caused the insertion of additional log entries.

**Spring-context-5.2.3.RELEASE.jar**

Description: Spring Context

Severity: Medium

Vulnerabilities:

***CVE-2022-22968***: The case sensitivity was not applied to the first character of the field of all nested fields within the property path.

**Spring-expression-5.2.3.RELEASE.jar**

Description**:** Spring Expression Language

Severity: Medium

Vulnerabilities:

***CVE-2022-22950***: Maliciously crafted SP expression language expressions may cause a denial of service.

## Mitigation Plan

**Log4j-api-2.12.1.jar**

***CVE-2021-45046(Critical)***: There was a previous fix to a critical vulnerability CVE-2021-44228. Previously, the JNDI features used in configuration, log messages, and parameters did not protect against attacker-controlled LDAP and other JNDI related endpoints. If the attacker had control of log messages and log message parameters, then the attacker was able to execute arbitrary code loaded from LDAP servers when message lookup substitution was enabled. The behavior was disabled and the functionality was removed.

However, it was incomplete in certain non-default configurations. Subsequently, the non-default pattern layout allowed attackers with control over Thread Context Map input data when combined with either Context Lookup or a Thread Context Map pattern. This enabled an attacker to craft malicious input data using a JNDI Lookup pattern which resulted in an information leak or remote code execution. Furthermore, this enabled local code execution in all environments. The requirements for the attack were:

* Logging configuration uses a non-default Payout
* Using either Context Lookup or Thread Context Map
* JNDI Lookup pattern

**Known Apache Log4j2 versions affected**:

* 2.0 – beta9 through 2.15.0
* along with (2.12, 2.12.2, 2.12.3, 2.3.1)

**Mitigation**:

I recommend upgrading to Log4j 2.16.0 in Java 8 or above. The issue was fixed by removing support for message lookup patterns and disabling JNDI functionality by default. Caution should be exercised as to not enable the JNDI functionality by mistake.

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

***CVE-2020-1938(Critical)***: Apache Tomcat treats AJP connections as a higher trust state than a similar hypertext transfer protocol connection. A known exploit was found on March 03. 2022 that enabled an attacker to access to the connection if the default Connector was not disabled. When curtain conditions are met, they allowed remote code execution. The requirements for a successful attack are:

* AJP Connector enabled
* Returning files from anywhere in the web application
* Application allows file upload and stores those files within the web application
* Processing the file in the web application as a java server page

**Known Apache Tomcat versions affected**:

* 7.0.0 – 7.0.99
* 8.5.0 - 8.5.50
* 9.0.0.M1 - 9.0.0.30

**Mitigation**:

The dependency check mentioned that it is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Mitigation of this vulnerability simply requires disabling the default connector. Alternative solutions include:

* Configure Tomcat to not trust the IP address provided.
* Configure Tomcat to authenticate the username provided.
* Find a way to guarantee only reverse proxies use the AJP port.
* If the client and Tomcat are on the same box: use lookback address when you trust all users
* If the client and Tomcat are on different boxes: setup a dedicated subnet for a client to Tomcat connection. Additionally, configure Tomcat to accept connections from specific hosts (Thomas, 2020).

Source

Llanillo, Nacho. (2018). *The value of secure communications*. [Blog Post]. [The value of secure communications | by Nacho Llanillo | Nacho’s Nook | Medium](https://medium.com/nachos-nook/the-value-of-secure-communications-432d4bcf1bf2)

SWIFT. (2023). *Customer Security Programme*. [Blog Post]. [Swift Customer Security Programme Information Sessions H1 | Swift](https://www.swift.com/news-events/webinars/swift-customer-security-programme-information-sessions-h1)

Shobhit, Seth. (2022). *How the Swift Banking System Works. Investopedia*. [Blog Post]. [How the SWIFT Banking System Works (investopedia.com)](https://www.investopedia.com/articles/personal-finance/050515/how-swift-system-works.asp)

Federal Trade Commission. (n.d.). *Gramm-Leach-Bliley Act.* [Blog Post]. [Gramm-Leach-Bliley Act | Federal Trade Commission (ftc.gov)](https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act)

Ekran. (2022). *7 Best Practices for Banking and Financial Cybersecurity Compliance*. [Blog Post]. [Banking & Financial Data Security Compliance: 12 Best Practices | Ekran System](https://www.ekransystem.com/en/blog/banking-and-financial-cyber-security-compliance)

Moramarco, Stephen. (2019). *Phishing attacks in the banking industry*. [Blog Post]. [Phishing attacks in the banking industry | Infosec Resources (infosecinstitute.com)](https://resources.infosecinstitute.com/topic/phishing-banking-industry/)

Kost, Edward. (2023). *The 6 Biggest Cyber Threats for Financial Services in 2023*. UpGuard. [Blog Post]. [The 6 Biggest Cyber Threats for Financial Services in 2023 | UpGuard](https://www.upguard.com/blog/biggest-cyber-threats-for-financial-services#:~:text=The%206%20Biggest%20Cyber%20Threats%20for%20Financial%20Services,Chain%20Attacks%20...%206%206.%20Bank%20Drops%20)

Brooks, Chuck. (2021). *Open source: A catalyst for modernization & innovation*. IBM. [Blog Post]. [Open source: A catalyst for modernization & innovation   - Servers & Storage (ibm.com)](https://www.ibm.com/blogs/systems/open-source-a-catalyst-for-modernization-innovation/)

Nehra, Mahipal. (2022). *The Evolution of Web Development & Its Modern Trends*. [Blog Post]. [The Evolution of Web Development & Its Modern Trends (decipherzone.com)](https://www.decipherzone.com/blog-detail/evolution-web-development)

Microsoft. (2023). Setting the connection properties. [White Papers]. [Setting the connection properties - JDBC Driver for SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/connect/jdbc/setting-the-connection-properties?view=sql-server-ver16)

Microsoft. (2023). Connection with encryption. [White Papers]. [Connecting with encryption - JDBC Driver for SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/connect/jdbc/connecting-with-ssl-encryption?view=sql-server-ver16)

SonarSource. (2023). Java static code analysis. Sonar Rules. [White Papers]. [Java static code analysis | spring: Persistent entities should not be used as arguments of "@RequestMapping" methods (sonarsource.com)](https://rules.sonarsource.com/java/tag/spring/RSPEC-4684)

Stepankin, Michael. (2020). Spring View Manipulation Vulnerability. [Blog Post]. [Spring View Manipulation Vulnerability | Veracode](https://www.veracode.com/blog/secure-development/spring-view-manipulation-vulnerability#:~:text=%40GetMapping%20%28%22%2Fmain%22%29%20public%20String%20fragment%20%28%29%20%7B%20return,or%20a%20fragment%20are%20concatenated%20with%20untrusted%20data.)

Thomas, Mark. (March 4, 2020). *Re: Fix for CVE-2020-1938*. [Thread]. [Re: Fix for CVE-2020-1938-Apache Mail Archives](https://lists.apache.org/thread/4gglghz0wofxthz7jkcp870wmn8hw1q7)