****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Artemis Financial Vulnerability Assessment Report 1](#_Toc145737065)

[Document Revision History 3](#_Toc145737066)

[Client 3](#_Toc145737067)

[Instructions 3](#_Toc145737068)

[Developer 4](#_Toc145737069)

[1. Interpreting Client Needs 4](#_Toc145737070)

[2. Areas Of Security 5](#_Toc145737071)

[3. Manual Review 5](#_Toc145737072)

[4. Static Testing 6](#_Toc145737073)

[5. Mitigation Plan 12](#_Toc145737074)

[6. Appendix 14](#_Toc145737075)

[7. References 22](#_Toc145737076)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9-17-2023** | **Danielle Eeg** | **First Draft** |

## 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

Danielle Eeg

## Interpreting Client Needs

At Artemis Financial, secure communications should be of the utmost importance for the safety of customer data. Finance data is inherently sensitive, and this company will interact with several types of finance data, related to savings, retirement, investment, and insurance plans. Why is data security so important? According to IBM’s Cost of a Data Breach Report 2023, the average cost of a data breach in 2023 is $4.45 million [1]. This is staggering motivation for prioritizing secure communications, but Artemis Financial should also be motivated by their reputation for growing and maintaining their customer base.

It is unclear whether Artemis Financial facilitates international transactions. If they do, security laws and regulations should be considered for all countries the organization services. Many countries have government regulations on cybersecurity strategies, such as the Sarbanes-Oxley act of 2002 (SOX) that regulates cybersecurity practices in public companies in the United States, and the Gramm-Leach-Bliley Act (GBLA), which regulates security controls for customer data at financial institutions in the United States [2]. There are far more than just these two acts that regulate data security in the United States, and even more beyond that passed in other countries. On March 1, 2023, the White House published the National Cybersecurity Strategy, which defines five pillars of cybersecurity [3]:

1. Defend critical infrastructure.
2. Disrupt and dismantle threat actors.
3. Shape market forces to drive security and resilience.
4. Invest in a resilient future.
5. Forge international partnerships to pursue shared goals.

Financial institutions are an obvious target for malicious attacks; they host financial information for their clientele and often require personal identification information such as social security numbers. Access to this type of data can yield a big payday for cyber criminals whether it’s through a ransomware payout or through other forms of financial fraud or identity theft. The types of threats an organization like Artemis Financial could face includes phishing attacks, social engineering attacks, ransomware attacks, SQL injection, brute force attacks, denial-of-service attacks, XSS attacks, as well as others.

As technology progresses, so will methods of cybercrimes. While the use of open-source libraries may be mandatory for the success of the application, as time goes on, these libraries must be evaluated for security updates. Applications cannot be left dormant once developed; leaving them static post-deployment will lead to security risks as library versions are updated to reflect the latest security technologies. The same thing goes for web application technologies. As technology develops, we applications will need to stay on top of the latest and greatest security techniques, tools, and frameworks to be able to stand against cyber-attacks. Furthermore, new government regulations will assuredly be put into place to regulate the use and communication of data. Companies such as Artemis Financial will need to ensure their security standards comply with legal requirements to avoid consequences such as fines or even jail time.

## Areas of Security

1. Input Validation

Input validation is important across most, if not all, programs that require input from users, but it is particularly critical for web applications such as this one where there is a high risk of things like SQL injection, privilege escalation, and other attacks. Input validation should have several lines of defense, such as white listing (for passing input matches patterns marked as “good”), input validation frameworks such as Apache Struts, limitations on input length (for preventing denial-of-service attacks), and regular expression validation for parameters of URLs [4]. This is not a comprehensive list; the methods of input validation vary based on application and data requirements.

Additionally, beyond the security standpoint, input validation can be helpful for an improved user experience. If a site is looking for certain input, it may not function the way the end user anticipates if wrong data is entered. Even if no damage is done from a security standpoint, this can be a frustrating experience for the user.

1. APIs

APIs are important to protect because insecure API connections can cause an attacker to have access to sensitive data, which can be costly and dangerous for the physical, financial, and emotional safety of users or companies. Attacks to an API can include everything from API injection, Denial of Service attacks, vulnerability attacks, cross site scripting attacks, as well as other ways [5]. In other words, there is no room for weakness when it comes to API security, particularly in the connect of Artemis Financial. Inherently, financial institutions handle sensitive data that needs to be protected for the integrity of the company and for the safety if its customers. Therefore, the API needs to have extremely vigilant security in place.

1. Cryptography

As much as user authentication, parameterization, input validation, and rock-solid API protocols can prevent sensitive data from getting into the hands of an attacker, cryptography is another piece of the puzzle that should act as a line of defense for Artemis Financial. Cryptography cannot be the only line of defense because it is possible for keys and ciphers to be leaked, but it is certainly an important line of defense to implement for this project.

1. Code Quality

Code quality is the fourth focus area of security in this case. If there are quality issues in code, particularly in the three previously mentioned areas, there is a window for an attacker to get through the program’s lines of defense. Every area of the program should meet a quality standard prior to deployment to ensure Artemis Financial is truly iron-clad.

## Manual Review

Within the Crud controller script request method, there is no validation of user input or authentication of user identity/permissions prior to giving read access to the document.

In the customer class definition, the showInfo() method that returns customer account number is a public method with apparently no user authentication. There should be validation of user identity so only the appropriate parties can view the account number (likely this would be financial institution administrators and the customer themselves). This same policy goes for the deposit method. However, in the case of the deposit, it may just be banking administration who gets to update the account balance. If a customer has permission to update the account balance, it is inevitable that some customer will add a balance to their account that does not truly belong to them, whether it is malicious or accidental.

For the DocData class read\_document method, where the user username and password is used to get read access to the document, some method of encryption and parameterization should be used to conceal the username and password in the request and prevent against SQL injection or other forms of input tampering. I am not certain how secure the document ID should be, but the getId() method should also be protected using user authentication and encryption if it is indeed sensitive or vulnerable to an injection attack.

In the Greeting class, there is no input validation for creating a new greeting object. This should be done to prevent a denial-of-service attack, or other forms of system penetration. For example, if there is a standard ID format, the anticipated type of input could be whitelisted while all others are rejected. Both the ID and the content should have their input length validated to prevent a denial-of-service attack.

In the GreetingController class, there should be validation to prevent unauthorized users from tampering with the get request. For example, there may be a way to inject code into the template to pull out more user data. For example, appending a request for the account SSN or other private data in addition to the name. Instead of using a string directly in the request, it could be pulled through an accessor, such as getGreeting. As another line of defense, as data is transferred from the API to the web application, there is risk of interception. There does not appear to be any decryption algorithm after the greeting is returned, so I assume the transferred data in not encrypted.

For the myDateTIme class, there is no input validation for the setMyDateTIme mutator method. There should be validation to prevent tampering that could lead to denial-of-service, or other date tampering. For example, changing the time in a banking app may cause certain time-sensitive functions to trigger prematurely (or not at all), payments to bounce, or other forms of error in reporting.

In the RestServiceApplication, there is no input validation for the args parameter. It is unclear what the program intends to use those args for, but based on their properties, a method of input validation should be chosen to protect the API from malicious requests or data.

## Static Testing

The following are all vulnerable dependencies, vulnerability code, description and mitigation plans, and identifiers for the static test of the project. To see a full list of vulnerabilities and their mitigation recommendations, see the Appendix (Page 14)

**Dependency 1:** bcprov-jdk15on-1.46.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\* |

**Description and Mitigations:**

This dependency is for the Bouncy Castle Cryptography package for cryptography algorithms in Java. To mitigate against this vulnerable dependency, use a version of Bouncy Castle that comes after 1.55 [6] [7] [8] [9], Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas Error messages should share minimal details to be helpful, but non-revealing of the methods used to call for the error. [10]

**Identifiers:**

1. bouncycastle/bcprov-jdk15on@1.46 (Confidence:High)
2. cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\* (Confidence:Low)
3. cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\* (Confidence:Low)
4. cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)
5. cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\* (Confidence:Low)

**Dependency 2:** hibernate-validator-6.0.18.Final.jar

**Vulnerability Codes:**

cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*

**Description and Mitigation**

This vulnerability was found in the Hibernate Validator that permits the bypass of input cleaning. Mitigation techniques can include having default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP [24]

**Identifiers:**

1. pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final (Confidence:High)
2. cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)

**Dependency 3:** jackson-databind-2.10.2.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This vulnerability is for the Jackson’s data-binding functions. Mitigation techniques may include: use of an XML parser and validation to prevent expansion issues [11] validation of input length, truncating if necessary, use of automatic buffer overflow detection for catching certain overflows. [12], signing and sealing to preserve serialized data, instead of deserializing, create a new object and validate all input into that object [13], and using Jackson-databind version after 2.15.2 [14]

**Identifiers:**

1. pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2 (Confidence:High)
2. cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)
3. cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* (Confidence:Low)

**Dependency 4:** log4j-api-2.12.1.jar

**Vulnerability Codes:**

cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

**Description and Mitigation:**

This vulnerability is for Apache Log4j API for logging implementations. To mitigate, use Apache Log4j 2.12.3 and 2.13.1.

**Identifiers:**

1. pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1 (Confidence:High)
2. cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)

**Dependency 5:** logback-core-1.2.3.jar

**Vulnerability Codes:**

cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

**Description and Mitigation:**

This is for the logback-core module, which is a logging framework. To prevent vulnerability, use versions after 1.2.7 and use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13]

**Identifiers:**

1. pkg:maven/ch.qos.logback/logback-core@1.2.3 (Confidence:High)
2. cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)

**Dependency 6:** snakeyaml-1.25.jar

**Vulnerability Codes:**

cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*

**Description and Mitigation:**

This vulnerability is for SnakeYAML, which is used to serialize Java objects. To prevent vulnerability, upgrade to SnakeYaml version 2.0 or beyond [15], use XML parser that restricts Document Type Definition entities when they expand recursively, and scan for recursive entity statements prior to parsing [16], and validate input length, truncating if necessary. Use automatic buffer overflow detection for catching certain overflows [12]

**Identifiers:**

1. pkg:maven/org.yaml/snakeyaml@1.25 (Confidence:High)
2. cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest)

**Dependency 7:** spring-boot-2.2.4.RELEASE.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is for Spring Boot, which is used for making Spring applications. To prevent this vulnerability, upgrade to versions 3.06+ or 2.7.11+ [17] and when resources have been overallocated, situate system in a safe state [18].

**Identifiers:**

1. pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE (Confidence:High)
2. cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\* (Confidence:Highest)
3. cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\* (Confidence:Highest)

**Dependency 8:** spring-boot-starter-web-2.2.4.RELEASE.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is for Spring Boots starter for Spring MVC web applications. To mitigate against vulnerabilities upgrade to versions 3.06+ or 2.7.11+ [40] and when resources have been overallocated, situate system in a safe state [43].

**Identifiers:**

1. pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE
2. cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\*
3. cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*
4. cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*

**Dependency 9:** spring-core-5.2.3.RELEASE.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency comes from spring core, the spring framework dependency injection and loC. To mitigate against it, default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP. Prevent command execution from using tented variables by removing the taint from non-dangerous inputs [19], Use a non-vulnerable version of Spring Framework: version 5.2.x after 5.2.14, or version 5.3.x 5.3.7 or later [20], and Ensure all programs are fail-closed, setting the system in a safe state [21]

**Identifiers:**

1. pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE
2. cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\*
3. cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*
4. cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*
5. cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

**Dependency 10:** spring-web-5.2.3.RELEASE.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is from the spring web services, to mitigate against this vulnerability, use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [30] and limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [21]

**Identifiers:**

1. springframework/spring-web@5.2.3.RELEASE
2. cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
3. cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
4. cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\*
5. cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
6. cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*

**Dependency 11:** spring-webmvc-5.2.3.RELEASE.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is from the Spring Web services MVC. To mitigate against it, limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [49]. And wherever possible, avoid basing logic on field names to avoid case sensitivity - triggered errors. [22] Additionally, when possible, clean user input to make uniform case (all upper or all upper)

**Identifiers:**

1. pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE
2. cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
3. cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
4. cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\*
5. cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
6. cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*

**Dependency 12:** tomcat-embed-core-9.0.30.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is for core Tomcat, which is used for HTTP Java web server environments. To mitigate against vulnerabilities, parameterization should be used to separate data from code and reduce the risk of SQL injection [69], use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13], use strict HTTP parsing framework and terminate client sessions after requests are complete [23], and use principle of least privilege across application. Use data access abstraction through Spring framework or other thread-safe features. [24]

**Identifiers:**

1. pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30 (Confidence:High)
2. cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* (Confidence:Highest) suppress
3. cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

**Dependency 13:** tomcat-embed-websocket-9.0.30.jar

**Vulnerability Codes:**

|  |
| --- |
| cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* |
| cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* |

**Description and Mitigation:**

This dependency is also for Tomcat core. Mitigation techniques should be the same as the prior dependency.

**Identifiers:**

1. pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30
2. cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*
3. cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

## Mitigation Plan

Address each of the risks described in the manual and static reviews. While this may seem like an extensive number of changes, it boils down to several key concepts, which align with the Areas of Security review. Across the board, most methods in the project need input validation. I recommend using Spring MCV for input validation (such as hibernator validator) to validate input prior to creating objects or making requests to the API. Additionally, parameterization and data encryption should be implemented in requests to the API to protect data transfer and reduce risk of injection.

The static test illuminated many areas for improvement, first and foremost, ensuring each dependency is up to date with the most recent security update. Most of the vulnerabilities (as you can see in the Appendix) are only pertinent to an older version of the dependency. For example. Updating Bouncy Castle (bcprov-jdk15on-1.46.jar) to a newer version than 1.55 eliminates over half its vulnerabilities (and updating it to versions after 1.74 removes even more). Beyond that, I recommend investigating and implementing most or all mitigative recommendations described in the previous section on the static test results for each vulnerable dependency.

Specific actions to take are as follows:

*From the manual review:*

* + 1. Add user authenication to CrudController prior to giving read access to user.
    2. Add user authentication to allow only the customer and administrators the ability to show account information in the customer class.
    3. The account number in the customer class should be encrypted prior to returning it with the accessor method.
    4. User authentication should be added to ensure only administrator privileges can update the account balance.
    5. Encryption and parameterization should be used to protect the username and password when requesting with the read\_document method in the DocData class.
    6. If document IDs are sensitive information, user authentication should be added prior to releasing the ID in the DocData method.
    7. Prior to creating a new greeting object, input validation should be used to ensure the ID argument entered is safe. This should be done using whitelisting, and input length validation.
    8. In the GreetingController class, input validation should be used to prevent tampering with the get request. Instead of issuing the request with a concatenated string, an accessor method should be used for parameters of the get request.
    9. In the myDateTime class, there should be input validation to ensure the date and time entered are physically possible (for example, seconds can integers 0-59, hours can be 0 – 23, etc.). This can be done with whitelisting or with validation framework.
    10. Prior to changing the time of the application in the myDateTime class, the user should be authenticated, and only administrators should be able to set the time.
    11. Input validation should be added to the RestServiceApplication to prevent malicious data from interacting with the API. It should certainly be checked for length, but likely other areas of validation too, depending on the expected value for args.

*From the static review:*

1. Use a version of Bouncy Castle that comes after 1.55 [6] [7] [8] [9]
2. Use separation of privilege and encapsulation to prevent attackers from accessing sensitive data within secure areas (addressed in manual review).
3. Ensure error messages share minimal details to be helpful, but non-revealing of the methods used to call for the error. [10]
4. Have default protocol be rejecting all input.
5. Use thorough validation to consider length of input, type of input, syntax, and conformity to expected input (addressed in manual review).
6. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP [24]
7. Use an XML parser and validation to prevent expansion issues [11]
8. Use automatic buffer overflow detection for catching certain overflows. [12]
9. Sign and seal to preserve serialized data, and instead of deserializing, create a new object and validate all input into that object [13]
10. Use Jackson-databind version after 2.15.2 [14]
11. Use Apache Log4j 2.12.3 and 2.13.1.
12. Use logback-core module versions after 1.2.
13. Upgrade to SnakeYaml version 2.0 or beyond [15]
14. Use XML parser that restricts Document Type Definition entities when they expand recursively.
15. Scan for recursive entity statements prior to parsing [16]
16. Upgrade Spring Boot to versions 3.06+ or 2.7.11+ [17]
17. When resources have been overallocated, situate system in a safe state [18].
18. Prevent command execution from using tented variables by removing the taint from non-dangerous inputs [19]
19. Use a non-vulnerable version of Spring Framework: version 5.2.x after 5.2.14, or version 5.3.x 5.3.7 or later [20]
20. Ensure all programs are fail-closed, setting the system in a safe state [21]
21. Wherever possible, avoid basing logic on field names to avoid case sensitivity - triggered errors. [22]
22. Parameterization should be used to separate data from code and reduce the risk of SQL injection [69
23. Use strict HTTP parsing framework and terminate client sessions after requests are complete [23]
24. Use data access abstraction through Spring framework or other thread-safe features. [24]

# Appendix

The following table includes all vulnerabilities found in the dependency report.

|  |  |  |
| --- | --- | --- |
| Vulnerability | Description | Solution |
| CVE-2016-1000338, CVE-2016-1000342 | In Bouncy Castle JCE Provider version 1.55 and earlier, signature injection can occur and still pass validation. | Use Static Application Security Testing (SAST) to identify sources and destinations of injection [25] |
| CVE-2016-1000343 | In Bouncy Castle JCE Provider version 1.55 and earlier, weak cryptographic private key generated when default values used | Ensure a version later than 1.55 of Bouncy Castle JCE is used |
| CVE-2016-1000344,  CVE-2016-1000352 | Malicious tampering of Electronic Code Book (cryptographic cypher) permitted with Bouncy Castle JCE Provider versions 1.55 and earlier [6]. | Ensure a version later than 1.55 of Bouncy Castle JCE is used |
| CVE-2016-1000341 | In Bouncy Castle JCE Provider version 1.55 and earlier, digital signature algorithm (DSA) is not time bound, allowing attacker to access unauthorized information. [7] | Ensure a version later than 1.55 of Bouncy Castle JCE is used |
| CVE-2016-1000345 | In Bouncy Castle JCE Provider version 1.55 and earlier, attackers are able to identify when decryption is failing (padding oracle attack) [8] | Ensure a version later than 1.55 of Bouncy Castle JCE is used |
| CVE-2017-13098 | BouncyCastle TLS prior to version 1.0.3 with JCE has a vulnerability that will allow an attacker to obtain the private key [26] | Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas. [10] |
| CVE-2020-15522 | Bouncy Castle BC Java before 1.66, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 may expose private key information due to EC math library timing vulnerability. [27] | Ensure all Bouncy Castle Javal, C#, FJA, and FNA area all kept up-to-date. Additionally, create independent accounts with minimal privileges for the purpose of singular tasks to reduce attacker access [28] |
| CVE-2020-0187 (OSSINDEX) | Possible incorrect cryptographic algorithm in engineSetMode of BaseBlockCipher.java [29] | I could not find any information on ways to resolve this issue, other than perhaps not using engineSetMode. This vulnerability does have a high confidentiality impact however, so future work should be done to address this vulnerability. [30] |
| CVE-2016-1000339 | In Bouncy Castle JCE Provider version 1.55 and earlier, the primary engine class used for AES was AESFastEngine, AES key information can be leaked. [9] | Ensure a version later than 1.55 of Bouncy Castle JCE is used |
| CVE-2020-26939 (OSSINDEX) | In Legion of the Bouncy Castle BC before 1.61 and BC-FJA before 1.0.1.2, RSA private key encryption exponent can be leaked. [31] | Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas. Error messages should share minimal details to be helpful, but non-revealing of the methods used to call for the error. [10] |
| CVE-2015-7940 | In Bouncy Castle Java library before 1.51, a lack of validation can allow attackers access to private keys [32] | Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas. [33] |
| 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” [34] | Check checksums prior to use to ensure they are aligned with the protocol [35]. |
| CVE-2013-1624 | Vulnerable to distinguishing attacks and plaintext-recovery attacks due to TLS in BC-Java before 1.48 and BC-C# before 1.8 [36] | Ensure newer verions of BC-Java and BC-C# are used then 1.48 and 1.8, respectively |
| CVE-2016-1000346 | In Bouncy Castle JCE Provider version 1.55 and earlier, other party DH public key validation error [37]. | Use Bouncy Castle JCE Provider version 1.56 or newer. |
| CVE-2015-6644 (OSSINDEX) | Sensitive information leak from Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 [38] | Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas. [33] |
| CVE-2020-10693 | Hibernate Validator 6.1.2 permits bypass of input cleaning [39] | Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP [40] |
| CVE-2020-25649 | XXE Attack vulnerability from entity expansion, from FasterXML Jackson Databind [41]. | Use an XML parser and validation to prevent expansion issues [11] |
| CVE-2020-36518 | DOS possible due to nesting for Jackson-databind prior to 2.13.0 [42] | Validate input length, truncating if necessary. Use automatic buffer overflow detection for catching certain overflows. Or, use a version of Jackson-databind after 2.13.0 [12] |
| CVE-2021-46877 | DOS risk for Jackson-databind 2.10 though 2.12.6 and 2.13.0 | Use a more recent version of Jackson Databind than 2.13.0 |
| CVE-2022-42003 | Resource exhaustion vulnerability when UNWRAP\_SINGLE\_  VALUE\_ARRAYS feature ennabled [43] | Use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13] |
| CVE-2022-42004 | Resource exhaustion vulnerability due to missing check for prevention of deep nested arrays [44] | Use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13] |
| CVE-2023-35116 | DOS from cyclic dependencies vulnerability in Jackson-databind versions through 2.15.2 [14] | Use Jackson-databind version after 2.15.2 |
| CVE-2020-9488 | SMTPS man-in-the-middle vulnerability due to poor certification validation in Apache Log4j SMTP appender [45] | Validate all pertinence certificate properties prior to pinning and manage certificates to ensure accurate encryption [46] |
| CVE-2021-42550 | Attackers can execute arbitrary code when they have required privileges, when logback bersion 1.2.7 and prior is used [47] | Use logback bersions newer then 1.2.7 and use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13] |
| CVE-2022-1471 | No type restriction on SnakeYaml constructor class, which creates a RCE vulnerability [15] | Upgrade to SnakeUaml version 2.0 or beyond [15] |
| CVE-2022-25857 | Missing nested depth limits create DoS vulnerability [48] | Use XML parser that restricts Document Type Definition entities when they expand recursively, and scan for recursive entity statements prior to parsing [16] |
| CVE-2022-38749, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750 | Attacker can cause stackoverflow and crash the parser when snakeYAML is used to parse untrusted YAML files [49] | Validate input length, truncating if necessary. Use automatic buffer overflow detection for catching certain overflows. [12] |
| CVE-2023-20873 | Security bypass vulnerability for Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older when an application is deployed to Cloud Foundry [17] | Upgrade to versions 3.06+ or 2.7.11+ [17] |
| CVE-2022-27772 | Directory hijack vulnerability for s pring-boot versions prior to version v2.2.11 [50]. | Upgrade to a supported version (versions after 2.2.11) |
| CVE-2023-20883 | DoS vulnerability if Spring MVC and revers proxy cache both used with Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 or earlier [51] | When resources have been overallocated, situate system in a safe state [18]. |
| CVE-2022-22965 | RCE vulnerability when application runs on Tomcat as WAR deployment. This applies to Spring MVC or Spring WebFlux applications running on JDK9+ [52] | Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP. Prevent command execution from using tented variables by removing the taint from non-dangerous inputs [19] |
| CVE-2021-22118 | Privilege escalation through duplicating a storage directory for tampering with files uploaded with WebFlux application. This vulnerability applies to Spring Framework versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7 [20] | Use a non-vulnerable version of Spring Framework: version 5.2.x after 5.2.14, or version 5.3.x 5.3.7 or later [20] |
| CVE-2020-5421 | Jsessionid path can be used to bypass CVE-2015-5211 attacks 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 or older versions [53] | Use an upgraded version of Spring Framework |
| CVE-2022-22950 | Vulnerability for input Spring expression language input o lead to DoS in Spring Framework 5.3.0-5.3.16 [54] | Limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [21] |
| CVE-2022-22971 | DOS vulnerability from app with STOMP over WebSocket endpoint [55] | Limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [21] |
| CVE-2023-20861 | Spring Framework prior to 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE has vulnerability for input Spring expression language input o lead to DoS in Spring Framework | When resources have been overallocated, situate system in a safe state [18]. Additionally, use newer versions of Spring that have this issue corrected. |
| CVE-2023-20863 | Spring Framework prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ has vulnerability for input Spring expression language input o lead to DoS in Spring Framework | When resources have been overallocated, situate system in a safe state [18]. Additionally, use newer versions of Spring that have this issue corrected. |
| CVE-2022-22968 | Spring Framework prior to versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20 have case sensitive fields for disallowedFields on DataBinder. This alters protection unless upper and lowercase variations are used for each field [56] | Wherever possible, avoid basing logic on field names to avoid case sensitivity - triggered errors. [22] Additionally, when possible, clean user input to make uniform case (all upper or all upper) |
| CVE-2022-22970 | File upload apps vulnerable to DOS in Spring Framework Versions prior to 5.3.20+ , 5.2.22+ | Limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [21] |
| CVE-2021-22060, CVE-2021-22096 | Log entry addition from malicious input allowable in Spring Framework versions 5.3.0 – 5.3.13 and 5.2.0 – 5.2.18 or older [57] [58] | Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP. Prevent command execution from using tented variables by removing the taint from non-dangerous inputs [19] |
| CVE-2016-1000027 | When deserializing malicious data in Jave, Spring Framework versions through 5.3.16 can have RCE [59] | Use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13] |
| CVE-2020-1938 | APJ connections have elevated trust when being used by Apache Tomcat. [60] | Apply updates to application based on vendor instructions [60] |
| CVE-2020-11996 | High CPU use age for Apache Tomcat requests may lead to unresponsiveness [61] | Limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [21] |
| CVE-2020-13934 | DoS risk due to OutOfMemory-Exception for non-release of HTTP/1.1 processor. This risk applies 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  [62] | Limit the number of resources allowed to each unauthorized user. Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Ensure all programs are fail-closed, setting the system in a safe state [49]. Additionally, use versions of Apache Tomcat that are not susceptible to this vulnerability. |
| CVE-2020-13935 | Incorrect payload validation for WebSocket Frame 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 may lead to DoS [63] | Use a version of Apache Tomcat that is not susceptible to this vulnerability |
| CVE-2020-17527 | Data leak risk between requests due to re-use of HTTP request headers from prior connection. This issue can occur in 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 [64] | Use encapsulation to create safe areas for data and to not allow its transmission outside of those areas. Use separation of privilege to only allow access to data by parties that truly need it. [33] |
| CVE-2021-25122 | Data sharing between users from duplicate request headers [65] | Use separation of privilege and encapsulation to prevent attackers to accessing sensitive data within secure areas. [10] |
| 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 improper validation increases infinite loop risk (DOS prone [66] | Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP [40] |
| CVE-2022-29885 | Apache Tomcat documentation error, stated it could run Tomcat clustering over untrusted network, when this could be quite risky [67] | When resources have been overallocated, situate system in a safe state [18]. |
| CVE-2022-42252 | If rejectlegalHeader setting was set to false in 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, vulnerable to request smuggling through header [68] | Use strict HTTP parsing framework and terminate client sessions after requests are complete [23] |
| CVE-2020-9484 | Attacker can trigger RCE via deserializations if a set of conditions are true for 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  [69] | Use signing and sealing to preserve serialized data. Instead of deserializing, create a new object and validate all input into that object [13] |
| CVE-2021-25329 | Fix incomplete for CVE-2020-9484 [70]. | See prior entry (CVE-2020-9484) |
| CVE-2021-30640 | JNDI Realm of Apache Tomcat permits unauthorized access to falsely authenticated credentials [71] | Parameterization should be used to separate data from code and reduce the risk of SQL injection [72] |
| CVE-2022-34305 | XSS Vulnerability due to unfiltered data 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 Form authentication [73] | Have default be rejecting all input; use thorough validation to consider length of input, type of input, syntax, and conformity to expected input. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP. Use a Firewall as a line of defense against attackers [74] |
| CVE-2023-41080 | Redirection to untrusted site in FORM authentication in Apache Tomcat: from 11.0.0-M1 through 11.0.0-M10, from 10.1.0-M1 through 10.0.12, from 9.0.0-M1 through 9.0.79 and from 8.5.0 through 8.5.92 [75] | Use an approved list of URLs available for redirections. Use server-side security checks instead of relying on client-side checks. Use input validation frameworks like Spring MVC or OWASP. Use a Firewall as a line of defense against attackers [76] |
| CVE-2021-24122 | 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 vulnerable to JSP source code disclosure [77] | Use a more recent version of Apache Tomcat than those vulnerable to code disclosure. Ensure permissions for viewing source code are as high as possible while still operating a functional site. |
| CVE-2021-33037 | Request smuggling vulnerability for 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 [78] | Use strict HTTP parsing framework and terminate client sessions after requests are complete [23] |
| CVE-2019-17569 | HTTP Request smuggling vulnerability when invalid transfer-encoding header mishandled [79] | Use strict HTTP parsing framework and terminate client sessions after requests are complete [23] |
| CVE-2020-1935 | Incorrect validation of invalid HTTP headers may result in HTTP request smuggling [80] | Use strict HTTP parsing framework and terminate client sessions after requests are complete [23] |
| CVE-2020-13943 | Excessive concurrent streams for a connection authorized when HTTP/2 client connects 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, resulting in unexpected outputs [81] | Use strict HTTP parsing framework and terminate client sessions after requests are complete [23] |
| CVE-2023-28708 | Insecure transmission of session cookies [82] | Use Secure Socket Layer for sign-in page and transferring any private data [83] |
| CVE-2021-43980 | Clients can receive wrong responses due to 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 [84] | Use principle of least privilege across application. Use data access abstraction through Spring framework or other thread-safe features. [24] |
| CVE-2020-8022 | Attackers have ability to escalate from tomcat all the way to the root [85] | Use principle of least privilege across application. Use data access abstraction through Spring framework or other thread-safe features. [86] |

# References

|  |  |
| --- | --- |
| [1] | IBM, "Cost of a Data Breach Report 2023," July 2023. [Online]. Available: https://www.ibm.com/reports/data-breach. [Accessed 10 09 2023]. |
| [2] | E. Kost, "Top 8 Cybersecurity Regulations for Financial Services," UpGuard, 27 June 2023. [Online]. Available: https://www.upguard.com/blog/cybersecurity-regulations-financial-industry. [Accessed 10 September 2023]. |
| [3] | T. W. House, "National Cybersecurity Strategy," 01 March 2023. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf. [Accessed 10 September 2023]. |
| [4] | J. Manico and A. Detlefsen, "Chapter 1: Web Application Security Basics," in *Iron-Clad Java*, McGraw-Hill Education, 2015. |
| [5] | A. Welekwe, "API Security Guide," Caomparitech, 18 11 2022. [Online]. Available: https://www.comparitech.com/net-admin/api-security/. [Accessed 15 09 2023]. |
| [6] | R. Awati, "Electronic Code Book," TechTarget, August 2021. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/Electronic-Code-Book. [Accessed 10 09 2023]. |
| [7] | "CVE-2016-1000341 Detail," NIST National Vulnerability Database, 20 10 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-1000341. [Accessed 10 09 2023]. |
| [8] | "CVE-2016-1000345 Detail," NIST National Vulnerability Database, 20 10 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-1000345. [Accessed 10 09 2023]. |
| [9] | "CVE-2016-1000339 Detail," NIST National Vulnerability Database, 20 10 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-1000339. [Accessed 10 09 2023]. |
| [10] | C. C. Team, "CWE-203: Observable Discrepancy," MITRE, 10 12 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/203.html. [Accessed 10 09 2023]. |
| [11] | "CWE611: Improper Restriction of XML External Entity Reference," MITRE, 21 02 2013. [Online]. Available: https://cwe.mitre.org/data/definitions/611.html. [Accessed 13 09 2023]. |
| [12] | "CWE-787: Out of Bound Write," MITRE, 27 04 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/787.html. [Accessed 13 09 2023]. |
| [13] | "CWE-502: Deserialization of Untrusted Data," MITRE, 25 06 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/502.html. [Accessed 12 09 2023]. |
| [14] | "CVE-2023-35116 Detail," NIST National Vulnerability Database, 26 07 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-35116. [Accessed 04 09 2023]. |
| [15] | "CVE-2022-1471 Detail," NIST National Vulnerability Database, 18 08 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-1471. [Accessed 04 09 2023]. |
| [16] | "CWE-776: IMproper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')," MITRE, 30 06 2009. [Online]. Available: https://cwe.mitre.org/data/definitions/776.html. [Accessed 13 09 2023]. |
| [17] | "CVE-2023-20873 Detail," NIST National Vulnerability Database, 28 08 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-20873. [Accessed 04 09 2023]. |
| [18] | "CWE-400: Uncontrolled Resource Consumption," MITRE, 08 11 2017. [Online]. Available: https://cwe.mitre.org/data/definitions/400.html. [Accessed 11 09 2023]. |
| [19] | "CWE-94: Improper Control of Generation Code ('Code Injection')," MITRE, 31 01 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/94.html. [Accessed 13 09 2023]. |
| [20] | "CVE-2021-22118 Detail," NIST National Vulnerability Database, 25 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-22118. [Accessed 04 09 2023]. |
| [21] | "CWE-770: Allocation of Resources Without Limits or Throttling," MITRE, 25 06 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/770.html. [Accessed 14 09 2023]. |
| [22] | "CWE-178: Improper Handling of Case Sensitivity," MITRE, 25 06 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/178.html. [Accessed 15 09 2023]. |
| [23] | "CWE-444 Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')," MITRE, 23 06 2014. [Online]. Available: https://cwe.mitre.org/data/definitions/444.html. [Accessed 12 09 2023]. |
| [24] | "CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')," MITRE, 11 05 2012. [Online]. Available: https://cwe.mitre.org/data/definitions/362.html. [Accessed 12 09 2023]. |
| [25] | "CWE-347: Improper Verification of Cryptographic Signature," MITRE, 25 04 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/347.html. [Accessed 10 09 2023]. |
| [26] | "CVE-2017-13098 Detail," NIST National Vulnerability Database, 20 10 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-13098. [Accessed 10 09 2023]. |
| [27] | "CVE-2020-15522 Detail," NIST National Vulnerability Database, 22 06 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-15522. [Accessed 10 09 2023]. |
| [28] | "CWE-362 Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')," MITRE, 13 12 2010. [Online]. Available: https://cwe.mitre.org/data/definitions/362.html. [Accessed 10 09 2023]. |
| [29] | "[CVE-2020-0187] CWE-310," Sonatype, [Online]. Available: https://ossindex.sonatype.org/vulnerability/CVE-2020-0187. [Accessed 10 09 2023]. |
| [30] | "Common Vulnerability Scoring System Calculator CVE-2020-0187," NIST National Vulnerability Database, [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2020-0187&vector=AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N&version=3.1&source=NIST. [Accessed 10 09 2023]. |
| [31] | "[CVE-2020-26939] CWE-203: Information Exposure Through Discrepancy," Sonatype, [Online]. Available: https://ossindex.sonatype.org/vulnerability/CVE-2020-26939. [Accessed 10 09 2023]. |
| [32] | "CVE-2015-7940 Detail," NIST National Vulnerability Database, 16 01 2019. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2015-7940. [Accessed 10 09 2023]. |
| [33] | "CWE-200 Exposure of Sensitive Information to an Unauthorized Actor," MITRE, 10 12 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/200.html. [Accessed 10 09 2023]. |
| [34] | "CVE-2018-5382 Detail," NIST National Vulnerability Database, 04 20 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-5382. [Accessed 10 09 2023]. |
| [35] | "CWE-354 Improper Validation of Integrity Check Value," MITRE, 19 07 2006. [Online]. Available: https://cwe.mitre.org/data/definitions/354.html. [Accessed 10 09 2023]. |
| [36] | "CVE-2013-1624 Detail," NIST National Vulnerability Database, 30 10 2018. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2013-1624. [Accessed 10 09 2023]. |
| [37] | "CVE-2016-1000346 Detail," NIST National Vulnerability Database, 20 10 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-1000346. [Accessed 10 09 2023]. |
| [38] | "[CVE-2015-6644] CWE-200: Information Exposure," Sonatype, 2018. [Online]. Available: https://ossindex.sonatype.org/vulnerability/CVE-2015-6644. [Accessed 10 09 2023]. |
| [39] | "CVE-2020-10693 Detail," NIST National Vulnerability Database, 10 05 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-10693. [Accessed 04 09 2023]. |
| [40] | C. C. Team, "CWE-20: Improper Input Validation," Common Weakness Enumeration, 29 06 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/20.html. [Accessed 04 09 2023]. |
| [41] | "CVE-2020-25649 Detail," NIST National Vulnerability Database, 02 02 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-25649. [Accessed 04 09 2023]. |
| [42] | "CVE-2020-36518 Detail," NIST National Vulnerability Database, 29 11 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-36518. [Accessed 04 09 2023]. |
| [43] | "CVE-2022-42003 Detail," NIST National Vulnerability Database, 02 12 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-42003. [Accessed 04 09 2023]. |
| [44] | "CVE-2022-42004 Detail," NIST National Vulnerability Database, 02 12 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-42004. [Accessed 13 09 2023]. |
| [45] | "CVE-2020-9488 Detail," NIST National Vulnerability Database, 12 05 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-9488. [Accessed 04 09 2023]. |
| [46] | "CWE-295: Improper Certificate Validation," MITRE, 27 03 2018. [Online]. Available: https://cwe.mitre.org/data/definitions/295.html. [Accessed 13 09 2023]. |
| [47] | "CVE-2021-42550 Detail," NIST National Vulnerability Database, 12 12 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-42550. [Accessed 04 09 2023]. |
| [48] | "CVE-2022-25857 Detail," NIST National Vulnerability Database, 08 08 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-25857. [Accessed 13 09 2023]. |
| [49] | "CVE-2022-38749 Detail," NIST National Vulnerability Database, 21 05 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-38749. [Accessed 04 09 2023]. |
| [50] | "CVE-2022-27772 Detail," NIST National Vulnerability Database, 07 04 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-27772. [Accessed 04 09 2023]. |
| [51] | "CVE-2023-20883 Detail," NIST National Vulnerability Database, 03 07 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-20883. [Accessed 04 09 2023]. |
| [52] | "CVE-2022-22965 Detail," NIST National Vulnerability Database, 08 02 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-22965. [Accessed 04 09 2023]. |
| [53] | "CVE-2020-5421 Detail," NIST National Vulnerability Database, 01 03 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-5421. [Accessed 04 09 2023]. |
| [54] | "CVE-2022-22950 Detail," NIST National Vulnerability Database, 22 06 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-22950. [Accessed 04 09 2023]. |
| [55] | "CVE-2022-22971 Detail," NIST National Vulnerability Database, 05 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-22971. [Accessed 04 09 2023]. |
| [56] | "CVE-2022-22968 Detail," NIST National Vulnerability Database, 19 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-22968. [Accessed 04 09 2023]. |
| [57] | "CVE-2021-22060 Detail," NIST National Vulnerability Database, 13 05 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-22060. [Accessed 04 09 2023]. |
| [58] | "CVE-2021-22096 Detail," NIST National Vulnerability Database, 28 04 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-22096. [Accessed 15 09 2023]. |
| [59] | "CVE-2016-1000027 Detail," NIST National Vulnerability Database, 20 04 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-1000027. [Accessed 04 09 2023]. |
| [60] | "CVE-2020-1938 Detail," NIST National Vulnerability Database, 12 07 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-1938. [Accessed 04 09 2023]. |
| [61] | "CVE-2020-11996 Detail," NIST National Vulnerability Database, 21 07 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-11996. [Accessed 04 09 2023]. |
| [62] | "CVE-2020-13934 Detail," NIST National Vulnerability Database, 01 03 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-13934. [Accessed 04 09 2023]. |
| [63] | "CVE-2020-13935 Detail," NIST National Vulnerability Database, 12 05 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-13935. [Accessed 04 09 2023]. |
| [64] | "CVE-2020-17527 Detail," NIST National Vulnerability Database, 12 05 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-17527. [Accessed 04 09 2023]. |
| [65] | "CVE-2021-25122 Detail," NIST National Vulnerability Database, 25 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-25122. [Accessed 04 09 2023]. |
| [66] | "CVE-2021-41079 Detail," NIST National Vulnerability Database, 25 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-41079. [Accessed 04 09 2023]. |
| [67] | "CVE-2022-29885 Detail," NIST National Vulnerability Database, 06 04 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-29885. [Accessed 04 09 2023]. |
| [68] | "CVE-2022-42252 Detail," NIST National Vulnerability Database, 30 05 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-42252. [Accessed 04 09 2029]. |
| [69] | "CVE-2020-9484 Detail," NIST National Vulnerability Database, 25 07 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-9484. [Accessed 04 09 2023]. |
| [70] | "CVE-2021-25329 Detail," NIST National Vulnerability Database, 26 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-25329. [Accessed 04 09 2023]. |
| [71] | "CVE-2021-30640 Detail," NIST National Vulnerability Database, 26 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-30640. [Accessed 04 09 2023]. |
| [72] | C. C. Team, "CWE-116 Improper Encoding or Escaping of Outpu," Common Weakness Enumeration, 29 06 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/116.html. [Accessed 04 09 2023]. |
| [73] | "CVE-2022-34305 Detail," NIST National Vulnerability Database, 26 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-34305. [Accessed 04 09 2023]. |
| [74] | "CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')," MITRE, 25 06 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/79.html. [Accessed 12 09 2023]. |
| [75] | "CVE-2023-41080 Detail," NIST National Vulnerability Database, 31 08 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-41080. [Accessed 04 09 2023]. |
| [76] | "CWE-601: URL Redirection to Untrusted Site ('Open Redirect')," MITRE, 24 02 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/601.html. [Accessed 12 09 2023]. |
| [77] | "CVE-2021-24122 Detail," NIST National Vulnerability Database, 24 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-24122. [Accessed 04 09 2023]. |
| [78] | "CVE-2021-33037 Detail," NIST National Vulnerability Database, 26 10 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-33037. [Accessed 04 09 2023]. |
| [79] | "CVE-2019-17569 Detail," NIST National Vulnerability Database, 02 09 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-17569. [Accessed 04 09 2029]. |
| [80] | "CVE-2020-1935 Detail," NIST National Vulnerability Database, 04 05 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-1935. [Accessed 04 09 2023]. |
| [81] | "CVE-2020-13943 Detail," NIST National Vulnerability Database, 21 01 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-13943. [Accessed 04 09 2023]. |
| [82] | "CVE-2023-28708 Detail," NIST National Vulnerability Database, 03 27 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-28708. [Accessed 04 09 2023]. |
| [83] | "CWE-523: Unprotected Transport of Credentials," MITRE, 30 10 2012. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-28708. [Accessed 12 09 2023]. |
| [84] | "CVE-2021-43980 Detail," NIST National Vulnerability Database, 09 11 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-43980. [Accessed 04 09 2023]. |
| [85] | "CVE-2020-8022 Detail," NIST National Vulnerability Database, 17 03 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-8022. [Accessed 04 09 2023]. |
| [86] | "CWE-276: Incorrect Defaul Permissions," MITRE, 20 08 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/276.html. [Accessed 15 09 2023]. |
| [87] | J. Nadeau, "Banking and Finance Data Breaches: Costs, Risks and More To Know," Security Intelligence, 6 October 2021. [Online]. Available: https://securityintelligence.com/articles/banking-finance-data-breach-costs-risks/. [Accessed 10 September 2023]. |