GLOBALRAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	2/4/24	Briana Long	

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
Briana Long

1. Interpreting Client Needs

Secure communications is crucial to the company Artemis Financial. The consulting company developes financial plans for their customers. This includes savings, retirement, investments and insurance to meet the needs of the individual customers. The company most likely does international transactions since the company uses an online application and includes a file the contains “myDateTime”.

There are no official government restrictions about secure communications. Financial companies need keep their customers information secure to prevent data leaks that can be damaging to their customers. This can include personal information such as social security numbers, names, addresses, personal accounts and account balances. Data leaks can harm the face value of the business and lead to less clients in the future.

The use of open-source libraries should be discouraged as this is an unsafe practice. These libraries need to be frequently updated and scanned to decrease vulnerabilities. Evolving web technologies includes microservices, dockers, orchestration tools and serverless computing. A microservice architecture would be beneficial as authentication and authorization can restrict the services that can be attacked. The security updates and patches can be updated frequently to combat vulnerabilities in the applications.

2. Areas of Security

The relevant areas of security are input validation, encapsulation, and code quality. Input validation is included to read the DocData, validates the customers account number and getting the time and date. The data in encapsulated in a separate data document that is read for the string key and values. The use of not directly including customers data is a good quality process that prevents data leaks.

3. Manual Review
customer.java

```java
package com.twk.restservice;

public class customer {
    private int account_number;
    int account_balance;

    public int showInfo() {
        //code to show customer information
        return this.account_number;
    }

    public void deposit(int a) {
            account_balance = account_balance + a;
    }
}
```

The customer.java class should not be public. It should be all be private as this is secure information. The information should be encapsulated to prevent easy access to the data. The deposit function also lacks a validation for the amount.

CRUD.java

```java
    public CRUD(String content) {
        this.content = content;
        this.content2 = content;
    }
```

In the CRUD.java class the second parameter should be named as content2 to prevent the recall of content1 with redundancy.

DocData.java

```java
public void read_document(String key, String value)
{
    /* implement read method */
    //Class.forName("com.mysql.jdbc.Driver");
    try {
        Connection con=DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/test","root","root");
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //here test is database name, root is username and password
```

The database connection is insecure and rooted into the code. This creates a security risk for the database.

4. Static Testing

bcprov-jdk15on-1.46.jar

CVE-2013-1624
The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets.

spring-boot-2.2.4.RELEASE.jar
cve-2022-27772
** UNSUPPORTED WHEN ASSIGNED ** spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

logback-core-1.2.3.jar
CVE-2021-42550
In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

log4j-api-2.12.1.jar
CVE-2020-9488
Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1

snakeyaml-1.25.jar
CVE-2017-18640
The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

jackson-databind-2.10.2.jar
CVE-2020-25649
A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.

tomcat-embed-core-9.0.30.jar
CVE-2020-25649
A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.

hibernate-validator-6.0.18.Final.jar
CVE-2020-10693
A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.

spring-web-5.2.3.RELEASE.jar
CVE-2016-1000027
Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

spring-beans-5.2.3.RELEASE.jar
CVE-2022-22965
A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

spring-webmvc-5.2.3.RELEASE.jar
CVE-2021-22060
In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.

spring-context-5.2.3.RELEASE.jar
CVE-2022-22968
In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.

spring-expression-5.2.3.RELEASE.jar
CVE-2022-22950
n Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition.

5. Mitigation Plan

The errors stated above in the manual review of the written code should be fixed. This will help secure the data more properly. Then all of the frameworks should be updated to the newest versions to combat all the vulnerabilities caused by unsupported and outdsated versions.