GLOBALRAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	1-17-2023	Vincent Snow	Vulnerability review & mitigation plan

Client

ARTEMIS FINANCIAL

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
Vincent Snow

1. Interpreting Client Needs

Artemis Financial is a company that uses sensitive information including clients' full names, contact information, financial account numbers and balances. Security is of utmost importance and carries with it high stakes for both the company and its customers. As a bank that may make international transactions transferring money, it must follow government regulations for its transactions that include encryption of user information and passwords. They face threats of attacks that attempt to prevent the service from functioning (DoS), to expose sensitive data, and to alter information and balances in their database (injections and privilege escalation Attacks). We must consider potential vulnerabilities in the libraries and dependencies used in the API, and upgrade or make changes where bugs have been found. Since technology is always evolving along with attackers finding new ways to exploit it, reliance on blacklisting known attack patterns will not suffice to protect the information.

2. Areas of Security

The areas of security we must review to ensure the application is secure are:

- Input Validations to prevent injections and be sure input does not interfere with the functioning of the system.
- Secure API Interactions because we are working with a RESTful API.
- Cryptography, because law requires a certain level of encryption for sensitive and financial data especially when sent internationally.
- Secure Client/Server interactions when sending data back and forth between the client (customer/user) and the bank's servers.
- Code Error handling to be sure that errors do not interfere with normal usage of the application or grant unauthorized access to alter information in the database
- Code Quality review to ensure secure and safe practices and coding patterns are being followed throughout the application that keep the data as safe as possible.

3. Manual Review

- The account balance is not a private variable in customer.java class.
- The method that updates an account balance does not have any validation or error handling for the input deposit amount.
- There is no communication with a database; it appears that text files may be used to store and retrieve information.
- There is no login interface or credentials to verify the users of the API.
- There is no safe error handling in the code; Improper input can crash the program or overload server.

4. Static Testing

CVE-2016-1000338 & CVE-2016-1000342 - Bouncy Castle does not fully validate encoding of signature; possible to inject extra elements and introduce ‘invisible’ data.

CVE-2016-1000343 - Bouncy Castle generates a weak private key if used with default values. Can be avoided by explicitly passing parameters to key pair generator.

CVE-2016-1000344 & CVE-2016-1000352 – Bouncy Castle 1.55 and earlier allow use of ECB mode; considered unsafe and removed from the provider.

CVE-2016-1000341 – Bouncy Castle is vulnerable to timing attacks that allows k value to be discovered.

CVE-2016-1000345 – Bouncy Castle 1.55 and earlier vulnerable to padding attack; can be watched and discovered when decryption is failing.

CVE-2017-13098 aka ‘ROBOT’ - Bouncy Castle 1.55 and earlier allows private key t be recovered from a vulnerable application.

CVE-2020-15522 - Timing information reveals information about the private key in Bouncy Castle.

CVE-2020-0187 (OSSINDEX) - incorrect algorithm may lcally disclose information with no privileges required.

CVE-2016-1000339 - Bouncy Castle 1.55 and earlier can leak information from tables in algorithms.

CVE-2020-26939 (OSSINDEX) - Bouncy Castle 1.61 and earlier allow attackers to extract information about the private key by throwing error exceptions.

CVE-2015-7940 - Bouncy Castle 1.51 and below allows attackers to obtain private keys via invalid curve attack

CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346 , CVE-2015-6644 (OSSINDEX) - Bouncy Castle 1.55 and earlier vulnerabilities that allow attackers to extract information about the private key or obtain sensitive information.

CVE-2020-10693 - Hibernate Validator version 6.1.2 does not properly validate input and allows bypassing it.

CVE-2020-25649 - Jackson Databind is vulnerable to XXE attacks, sacrificing data integrity

CVE-2020-36518 – Jackson databind before 2.13 allows Stack Overflow and DoS vis deep nested objects

CVE-2022-42003 & CVE-2022-42004 - Jackson databind before 2.14 resource exhaustion vulnerability.

CVE-2020-9488 - Apache Log4j vulnerability can leak log messages under certain circumstances. Fixed in new versions.

CVE-2021-42550 – In logback 1.2.7 and earlier, an attacker with high permissions could edit malicious configurations.

CVE-2022-1471 (OSSINDEX) - SnakeYaml's Constructor class is vulnerable to remote code execution & does not restrict types; Use SafeConstructor instead.

CVE-2017-18640 – SnakeYaml before 1.26 allows entity expansion during load operation

CVE-2022-25857 – SnakeYaml before 1.31 does not limit nesting depth; vulnerable to DoS

CVE-2022-3064, CVE-2022-38749, CVE-2022-38751 , CVE-2022-38752 , CVE-2022-41854,
CVE-2021-4235 , CVE-2022-38750 - Parsing large or malicious YAML docs can consume excessive memory; vulnerability to DoS

CVE-2022-27772 - Spring Boot prior to version 2.2.11 release vulnerable to directory hijacking.

CVE-2022-22965 - Spring MVC using JDK 9+ may be vulnerable to remote code execution through data binding. If executed as a JAR, it is not vulnerable to the exploit. Known exploit works only on Tomcat server.

CVE-2016-1000027 - Spring framework 5.3.16 vulnerable to potential remote code execution.

CVE-2021-22118 - Spring versions 5.2.x-5.2.15 and 5.3.x - 5.3.7 Webflux application vulnerability to privilege escalation.

CVE-2020-5421 - several Spring versions vulnerable to bypassing protections against RFD attacks.

CVE-2022-22950, CVE-2022-22971, CVE-2022-22970 – Spring versions prior to 5.3.2 vulnerable to DoS attacks.

CVE-2022-22968, CVE-2021-22060 , CVE-2021-22096 - Spring version 5.3.18 and below improperly validate input; vulnerable to malicious attacks.

CVE-2020-1938 - If AJP port is used in Apache Tomcat, potentially unauthorized access can be allowed to files.

CVE-2020-11996, CVE-2020-13934 - Some versions of Tomcat are vulnerable to DoS attack under certain HTTP/2 requests.

CVE-2020-13935, CVE-2021-41079 – Tomcat WebSocket vulnerability allows for DoS; payload lengths or specifically crafted TLS packets trigger infinite loop.

CVE-2020-17527 - Tomcat could re-use HTTP requests on HTTP/2 possibly leaking information.

CVE-2021-25122 – some Tomcat versions can duplicate HTTP requests and allow information to be leaked.

CVE-2022-29885 – Tomcat versions 0.1.0-M1 to 10.1.0-M14, 10.0.0-M1 to 10.0.20, 9.0.13 to 9.0.62 and 8.5.38 to 8.5.78: EncryptIntercept vulnerability to DoS

CVE-2022-42252 - Tomcat versions 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 allow HTTP request smuggling under certain circumstances.

CVE-2020-9484, CVE-2021-25329 - 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 allow remote code execution under certain set of known circumstances.

CVE-2021-30640 – improper validation issue in JNDI Realm of Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65.

CVE-2022-34305 - cross-site scripting vulnerability under certain circumstances in 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.

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 were susceptible to JSP source code disclosure in some configurations. JRE API File.getCanonicalPath() was the root cause.

CVE-2021-33037, CVE-2019-17569, CVE-2020-1935 - Apache Tomcat multiple versions have vulnerability to HTTP request smuggling.

CVE-2020-1394 – Tomcat servers vulnerability that may leak information in HTTP headers in HTTP/2 connections

CVE-2021-43980 - Tomcat vulnerability, client connections could share and HTTP11Processor instance and messages received by wrong client.

5. Mitigation Plan

- Implement an interface with secure validation for users to log in and verify their identity with passwords, allowing them a level of permission necessary to interact with their account.
- Implement a MySQL database to store customer information.
- Implement code to communicate with it to transfer the sensitive data using HTTPS.
- Implement error handling code in verifications and data transfers to prevent malfunctioning of the program or protect from vulnerabilities.
- Use encryption for sensitive data that adheres to legal requirements.
- Fix the security issues with the customer Java object that holds a copy of data: all fields should be private with getters and setters to change or access them.
- Upgrade the Bouncy Castle dependency past version 1.55, which has numerous critical vulnerabilities that allow attackers to obtain private encryption keys and personal data.
- Add additional validation to the code or upgrade Hibernate Validator over 6.1.2.
- Upgrade Jackson databind above version 2.14.
- Upgrade Apache Log4j dependency to 2.13.1
- Upgrade logback above 1.2.7
- Upgrade SnakeYaml above 1.31
- Use SafeConstructor instead of SnakeYaml's Constructor class
- Limit the size of YAML documents being read and parsed or use a different format and parser.
- Upgrade Spring Boot version in use to above 5.3.7 to avoid multiple vulnerabilities.
- Launch the application on an alternative to Tomcat server such as Jetty, or use JDK version less than 9. If a Tomcat server is used, use a version 10.1.5 or higher to avoid the potential vulnerabilities found in earlier versions.
- Upgrade Spring to a safe version according to CVE-2021-22118 if using WebFlux.