GLOBALRAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History 3
Client 3
Instructions 3
Developer 4
1. Interpreting Client Needs 4
2. Areas of Security 4
3. Manual Review 5
4. Static Testing 5
5. Mitigation Plan 6

Document Revision History

Version	Date	Author	Comments
1.0	**September 17, 2023**	**Tonessa V. Chatten**	

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
Tonessa V. Chatten

1. Interpreting Client Needs

Secure Communication
Secure communications are essential for Artemis Financial, a financial institution that holds sensitive customer information and conducts financial transactions. A breach in their communications could lead to severe economic losses and damage their reputation and credibility. Artemis Financial could also be subject to fines from government bodies, depending on the specifics of the breach.

International Transactions
Artemis Financial does process international transactions. International transactions can be complex, and Artemis Financial takes steps to protect its customers from security risks and scams, which are more common in international transactions.

Government Restrictions
Companies must comply with the General Data Protection Regulation (GDPR) when processing the personal data of individuals in the European Union (EU). The GDPR regulates communications and requires data handlers and service providers to protect customer data from potential threats. This places a high standard of scrutiny on companies, which must ensure that they move data in a way that minimizes security threats.

External Threats
With external threats to the application, a few come to mind: Authentication threats, Man-in-the-middle attacks, API threats, and DOS (Denial-of-service) attacks. Authentication threats allow unauthorized users to access accounts in the system. Man-in-the-middle attacks occur when a malicious person intercepts communication between a client and the application. DOS attacks involve sending massive amounts of traffic to the application to cause it to become overwhelmed and crash. With API threats, a malicious person can exploit it to compromise the application or even access sensitive data.

Modernization Requirements
There should be two key topics regarding modernization: incremental modernization and flexibility and accessibility. Incremental modernization involves making small, manageable changes to legacy systems to reduce risk and improve accessibility. This allows us to test and validate changes before moving on and helps minimize disruption. With flexibility and accessibility, the applications and technologies are more accessible in terms of reducing costs and improving agility. This allows us to respond more quickly to market and customer needs changes.

2. Areas of Security

In reviewing the Vulnerability Assessment Process Flow Diagram, the areas of security applicable to Artemis Financial's software application is: Input Validation, API, Cryptography, Client/Server, and Code Quality. I will explain in bullet points why Artemis Financial should focus on those five.

- Input Validation – Since this is financial software, the right people must be accessing the correct accounts.

- API – Artemis Financial's web application uses API calls, so it is important to secure the APIs to prevent security breaches between the system and the API connection.
- Cryptography – Artemis Financial accesses and transmits data over the internet, so it is important to encrypt the data to protect its integrity and confidentiality. Proper certificate validation should also be implemented.
- Client/Server – The web application communicates between clients and servers. It is important to secure this communication, as clients communicate with the system's backend through the front-end UI.
- Code Quality – Regarding code quality, I'll be specifying more on Secure Coding. There are many situations where code sanitization is required to maintain consistency in the application's business logic. For example, exception and error handling and checks directly apply to the application's code.

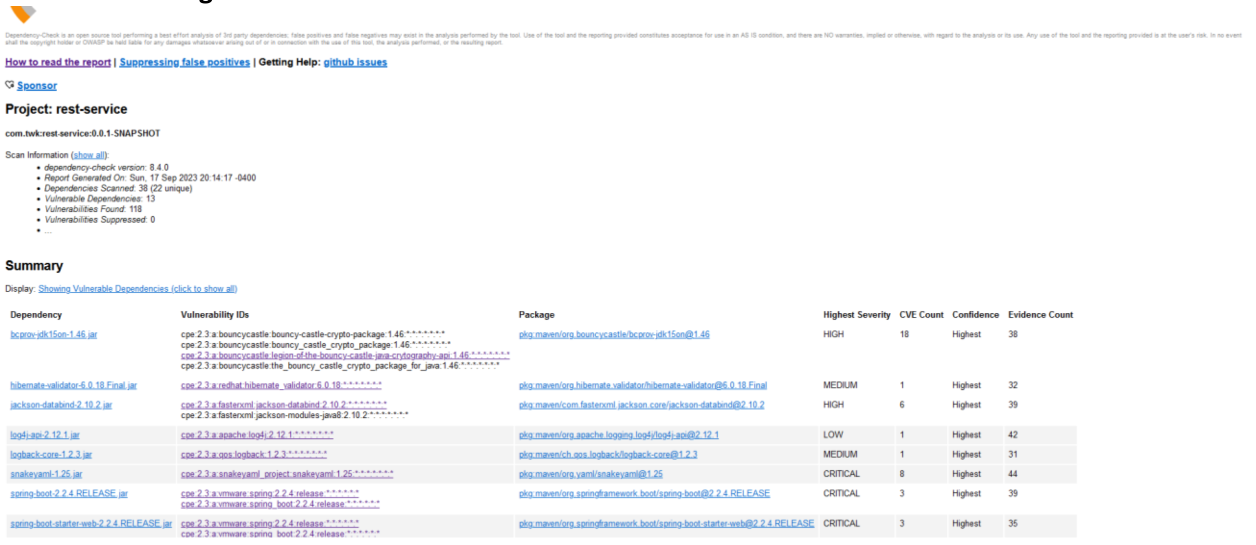
3. Manual Review

CRUDController.java:
This file shows that "business_name" is passed through the CRUD method and may expose the DocData object.

DocData.java:
This is a data access vulnerability where the username and password are both root – which isn't recommended as anyone could guess this. Even hackers can easily access this type of information via a brute force attack and paste it as a Pastebin text file.

Dependency Checker Upgrade:
I noticed that I needed to upgrade the dependency checker to my latest one. I also noticed that some other frameworks needed to be updated such as Spring.

4. Static Testing



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: rest-service

com.twk:rest-service:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 8.4.0
- *Report Generated On*: Sun, 17 Sep 2023 20:14:17 -0400
- *Dependencies Scanned*: 38 (22 unique)
- *Vulnerable Dependencies*: 13
- *Vulnerabilities Found*: 118
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bcprov-jdk15on-1.46.jar	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:*:*:*:*:*:*:*	pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46	HIGH	18	Highest	38
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	1	Highest	32
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:* cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	6	Highest	39
log4j-api-2.12.1.jar	cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:*	pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1	LOW	1	Highest	42
logback-core-1.2.3.jar	cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.3	MEDIUM	1	Highest	31
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	CRITICAL	8	Highest	44
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	CRITICAL	3	Highest	39
spring-boot-starter-web-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE	CRITICAL	3	Highest	35

I had 13 vulnerabilities come up. Most of these came up from a previous project. With those the mitigation plan was to just update them to the current versions (such as Log4J, Logback, Spring Framework, TomCat, etc.). In this project, I'll discuss the one difference that came to be –bcprovjdk15on-1.46.jar – The Bouncy Castle Crypto Package. Based on the dependency report, The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This vulnerability can result in the disclosure of information – which can allow a malicious application to access private data stored within the database. Because this affects packages 1.5 to 1.7 in the jar file, I recommend updating this to a current version.

5. Mitigation Plan

I would first address the username/password root issue when accessing the Artemis Financial software application. I would create a strong password with a minimum of 15 characters and alphanumerical. I would also even suggest two-factor authentication to guarantee that whoever is logging in, is more than likely the person. I would also upgrade the Log4J and Logback Apache applications. This should take care of the Client/Server issue. I would also upgrade the Spring frameworks around the same time as upgrading the Apache servers. Finally, I would upgrade the Core Tomcat to eliminate the change of a DOS attack vulnerability.