# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **Sept. 19, 2025** | **James Kofa** | **Initial Vulnerability** |

## Client



## Instructions

**Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend 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 include images or supporting materials. If you include them, make certain to insert them in 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

**James Kofa**

**1. Interpreting Client Needs**

**Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:**

* **What is the value of secure communications to the company?**
* **Are there any international transactions that the company produces?**
* **Are there governmental restrictions on secure communications to consider?**
* **What external threats might be present now and in the immediate future?**
* **What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?**

According to the project, Artemis Financial wants to modernize its operation to make sure that its asset is protected. Artemis Financial is an organization runs an internet-facing spring boot application that contain sensitive client’s data. The company do this so encrypted transport and Morden cryptograph are not an issue. The company even develops individualized financial plans to their clients. Some of those plans include, but not limited to savings and investments. The company does other things as well; they advised their clients on their personal information. Information like taxes, Pll, and more. It seems that this organization only operates in the United States, but the organization can be reach globally reachable. The organization I highly respectable, but a recent dependency information found some critical issue in its core libraries. Some of these libraries include Spring, Tomcat, Jackson, SnakeYAML, Logback, Bouncy Castle, and more. The organization need to priorities updating their security and software system periodically. With the organization current client base, technology progress, it is very critical that the organization discover any vulnerabilities quickly and repair it even faster. Within doing so, the clients should have already establish multi-factor authentication, and other web-based defensives.

**2. Areas of Security**

**Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application**.

 **Secure communications (TLS/HSTS):** Secure communication is one of the primary reasons for such application. The following app handles financial and Pll data. So, confidentiality, integrity, and server authenticity are mandatory. It is critical that the following is consider mTLS for service-to-service/admin paths.

 **Authentication & authorization (access control):** These functions build trust with the customer base. Admin and customers connections have to be enforced, least-privilege roles, and deny-by-default on URLs and method invocations. These action goes a long way in making sure that the clients’ information are safe and are being protected.

 **Input validation & output encoding:** to prevent injections like (SQL/SpEL), XSS, SSRF, path traversal, and template/deserialization abuse common in Java and other programming languages. Overall, this helps minimize any potential vulnerabilities. It also ensures API security is top priory.

 **Cryptography & sensitive-data protection:** This is essential especially when it comes to communication and storage. Being able to somehow ensure the organization integrity of their client’ data, that will bring a great trust between the organization and their clients.   
  
 **Dependency/supply-chain & build security:** This portion is not just about the clients, but about the entire lifecycle of that application. So, constant checking and updating, running SCA (Dependency-check), is great.

**3. Manual Review**

**Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.**

 **Hard-coded DB credentials & no TLS**  
**Location on Machine:** src/main/java/com/twk/restservice/DocData.java  
**Affect Area:** DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","root");  
**The Problem:** Secrets in code; connects without SSL; easy lateral movement if leaked.  
**How to solve the Problem:** Remove credentials from code

 **DB connection not closed (resource leak)**  
**Location on Machine:**  DocData.java  
**Affected Area:** Opens a JDBC connection but never closes it.  
**The Problem:** Exhausts connections; can cascade into DoS.  
**How to solve the Problem:** Use try-with-resources, maybe a datasource.

 **Stack trace printing**  
**Location on Machine:**  DocData.java   
**Affected Area/Problem:** Information disclosure in logs; may leak internals/paths.  
**How to solve the Problem:** Log sanitized message via a logger. Also, avoid printing raw stack traces in production.

 **Unauthenticated, wide-open endpoints**  
**Location on Machine:**  GreetingController.java   
**Affected Area/Problem:** Anyone can hit endpoints; foundation for enumeration/abuse.  
**How to solve the Problem:** Add Spring Security, require authn/authz (roles/claims), least privilege; restrict access where appropriate.

 **Overly broad HTTP method mapping**  
**Location on Machine:**  CRUDController.java   
**Affected Area/Problem:** Endpoint may accept POST/PUT/DELETE unintentionally (surprising surface).  
**How to solve the Problem:** Specify the method explicitly, e.g., @GetMapping("/read").

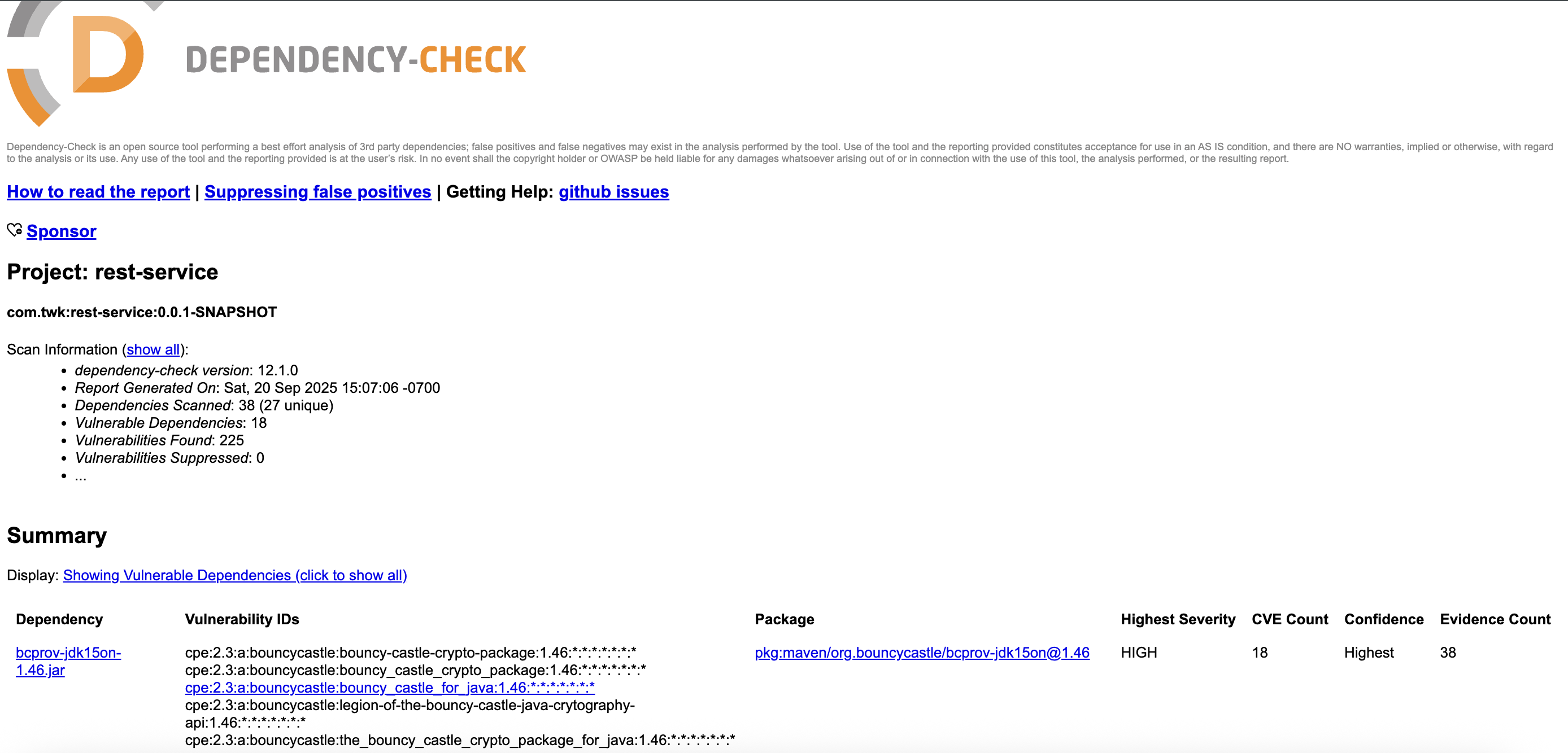
 **Unvalidated user input (potential injection/reflection)**  
**Location on Machine:**  GreetingController.greeting(@RequestParam String name).  
**Affected Area/Problem:** Input flows straight into responses/logic; if later rendered into HTML.   
**How to solve the Problem:** Validate/normalize inputs, or encode on output.

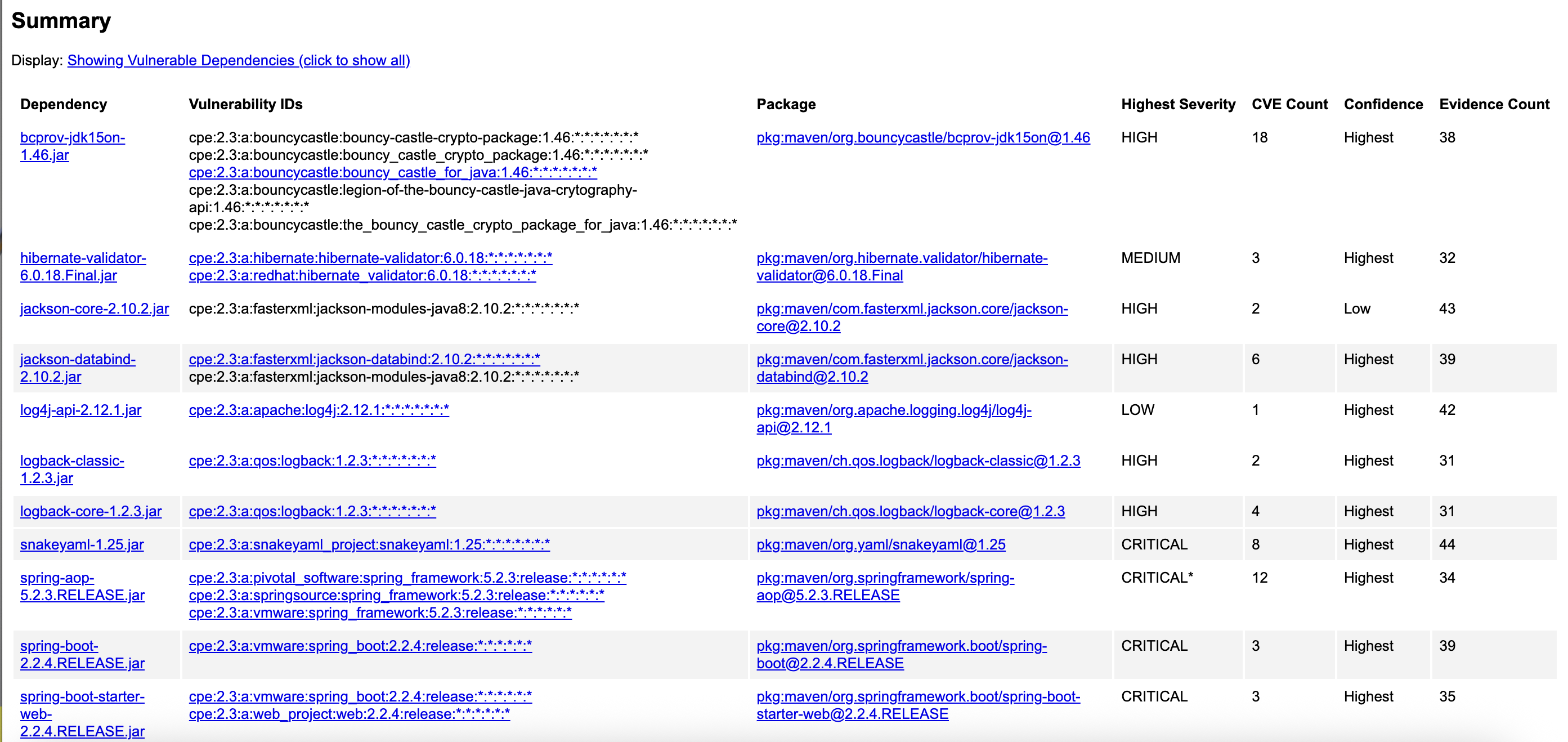
 **Multipart uploads enabled with very large limits**  
**Location on Machine:**  src/main/resources/application.properties  
**Affected Area/Problem:** max file **200MB**, request **215MB**. And storage abuse if upload endpoints are added.  
**How to solve the Problem:** Disable unless needed, or reduce limits.

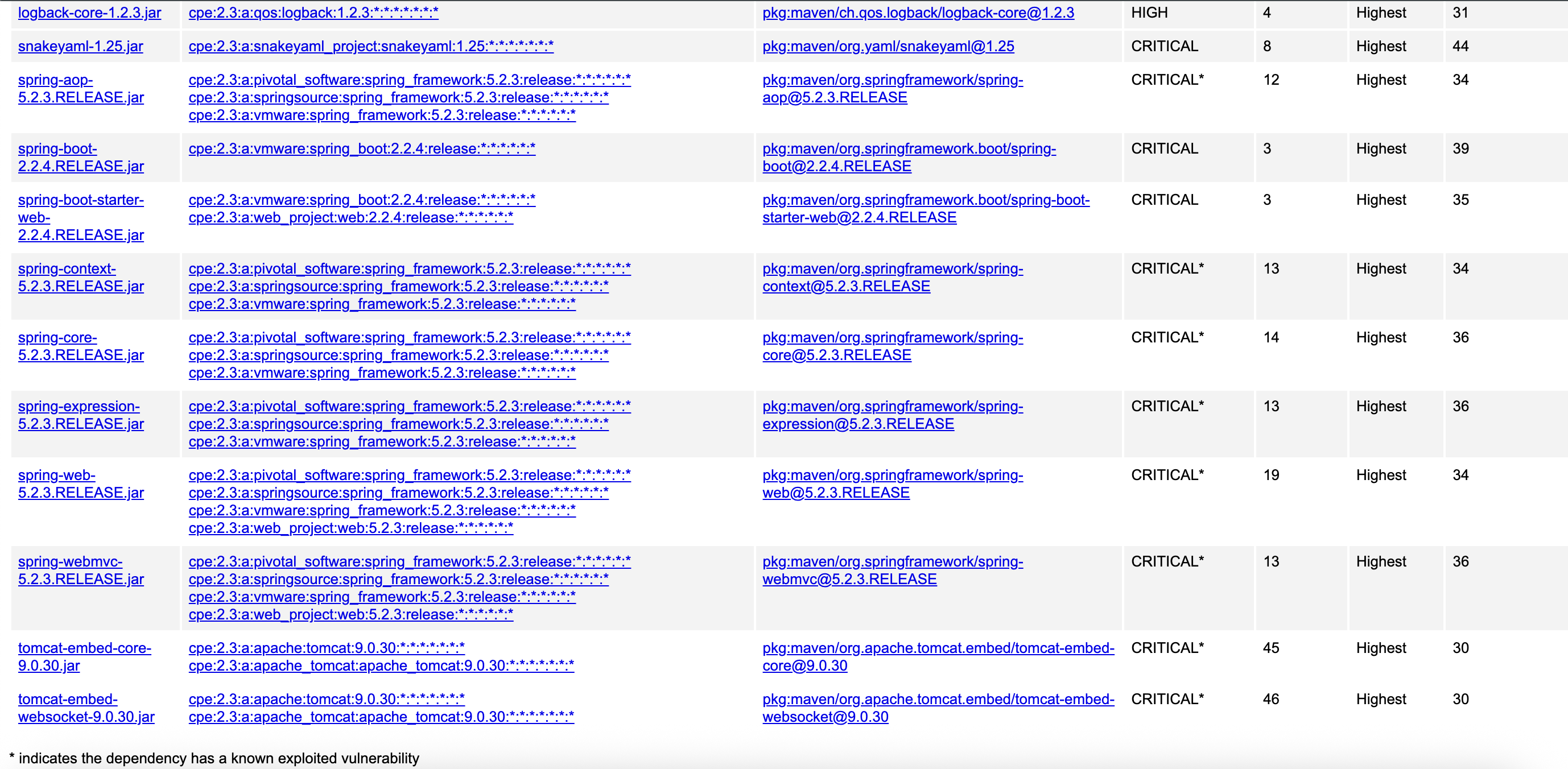
**4. Static Testing**

**Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:**

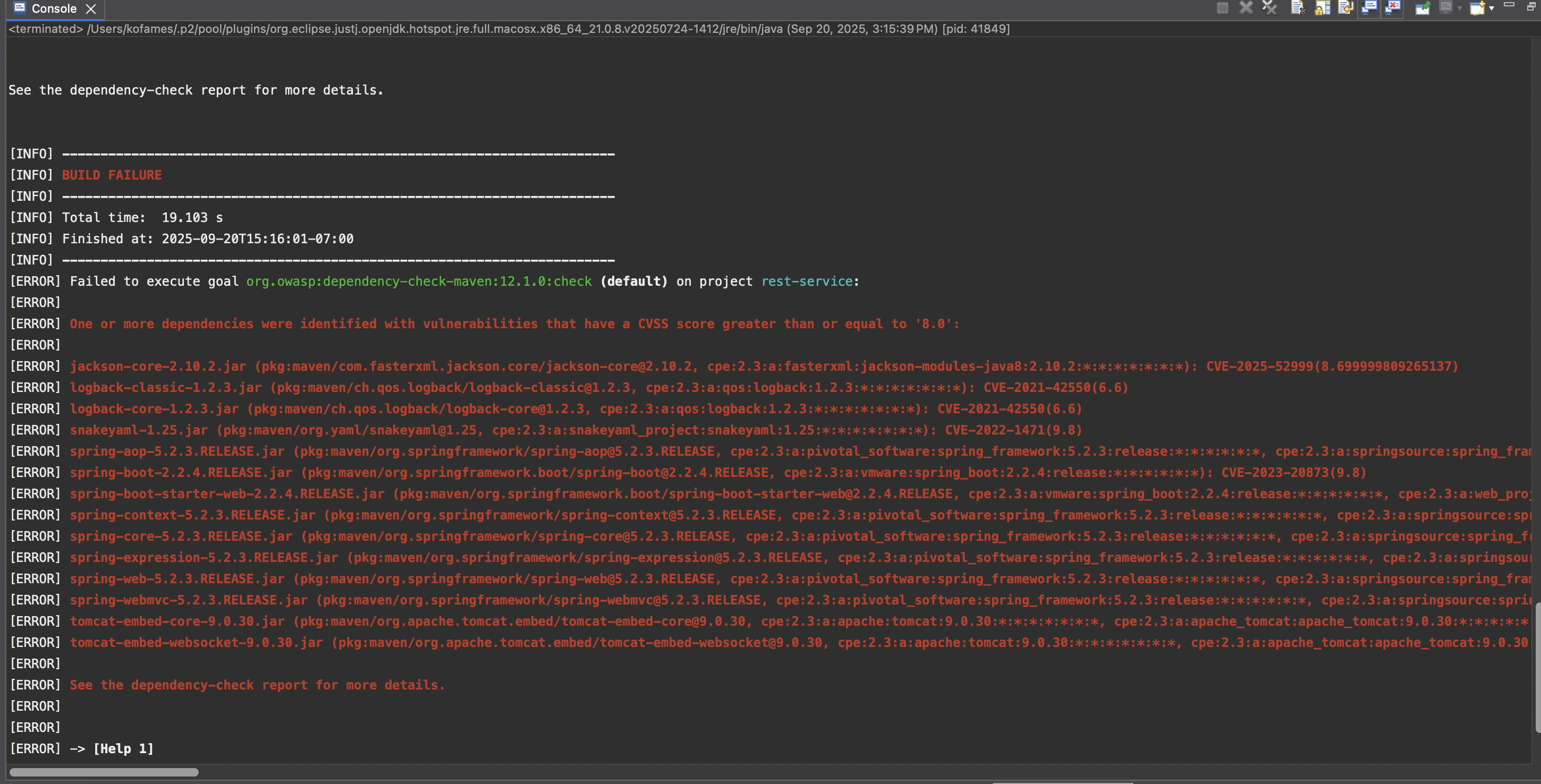
* **The names or vulnerability codes of the known vulnerabilities**
* **A brief description and recommended solutions provided by the dependency-check report**
* **Any attribution that documents how this vulnerability has been identified or documented previously**







**So, after running through the dependency check, I noticed 18 vulnerabilities that was flagged in the scan. I also notice that 13 vulnerabilities that contain at least one critical CVE. This along explain why the build fail. Please review the below pictures. I counted over 120 vulnerabilities in total.**



**5. Mitigation Plan**

**Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.**

From reviewing the overall report, I found issue such as missing authentication/authorization on endpoints, weak input validation, hard-coded credentials, and much more. After doing the static test, I confirmed some supplies chain risks to the system. Most of the issue the issue that was found, can easily be solved. Just by updating the system/software, can and will make a huge difference. New version, like Apache Tomcat, snakeyaml, and more. Periodically updating the software, changes things, and keep things up to date.