****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 5](#_Toc32574613)

[4. Static Testing 7](#_Toc32574614)

[5. Mitigation Plan 7](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/23/2024** | **Alex Frankel** | **Began work** |
| **1.337** | **1/27/24** | **Alex Frankel** | **Completed Vulnerability Assessment** |

## 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

Alex Frankel

## Interpreting Client Needs

For a financial consultant, securing their data means securing their client’s personal data as well as their livelihoods. It is best for this information, especially financial information, to be kept private.

Potential threats to this system are anyone who wishes to compromise its security. Hackers make no distinctions between potential targets- these threats will be around forever. They can be as small as an individual or a larger organization. It could even be a foreign government. This is done through exploiting a vulnerability in a system, which can be done technologically or personally. Any lapse of security, either online or in person, could result in a breach. The FTC enforces regulations on information security, and companies are potentially liable for data breaches! The FTC Safeguards rule “requires covered financial institutions to develop, implement, and maintain an information security program with administrative, technical, and physical safeguards designed to protect customer information,”(Ritchie, 2023). It is in our best interest to take information security seriously, rather than as an afterthought. Our credibility, and the well-being of both us and our clients, are at stake.

Something to keep in mind is that technology is ever-changing. Artemis Financial makes use of Spring, which is a contemporary API solution for Java programming that can be used for web applications. Web applications can make use of several API’s, or dependencies, and it is paramount to keep these up-to-date. It could mean the difference between patching the software and facing a life altering exploit.

## Areas of Security

Input Validation

Queries are going to be requested to and from this application, since it deals with financial databases. It is important that they cannot be tampered with through improper input and improper requests should not be accepted by this application.

API’s

The use of API’s is one of our most vulnerable areas of security since it involves an external party’s code. A codebase can be programmed securely, only to be vulnerable through the API’s they use. Static testing maps out a project’s dependencies as well as all documented vulnerabilities related to them AND known solutions/guidelines.

Cryptography

At some point, sensitive data is going to be communicated by this company. Proper encryption techniques (using HTTPS) will assure that even if this data is intercepted it cannot be used in a harmful manner. User authentication and access control strategies should be employed.

Code Error

Errors, caused by invalid requests, for example, should always be handled properly so unwanted data dumps or redirects are prevented.

Encapsulation

The idea is that only what is necessary at a given point in an application is made available. Only what is relevant to the trusted user is ever displayed or accessible. Every field should be private and used only through proper accessors and mutators.

## Manual Review

Use of @GetMapping- HTTP GET request.

A screenshot of a computer

Description automatically generated

Though it is not currently tied to sensitive data, the HTTP GET request (shown here as GetMapping) can sometimes include sensitive data in the URL.

No input validation

A screen shot of a computer

Description automatically generated

Along with the previous screenshot, the requests to the server are not being properly validated.

No error handling

A screenshot of a computer error page

Description automatically generated

The application has no error handling for invalid requests. This is poor practice

Access Control

A screen shot of a computer program

Description automatically generated

Connections to the SQL server are being made without any authorizations. Hardcoding the username and password in plaintext is not advised

A screen shot of a computer code

Description automatically generated

Customer’s information should not be stored in plaintext and processes involving account transactions should be encrypted and subject to authorization before being called.

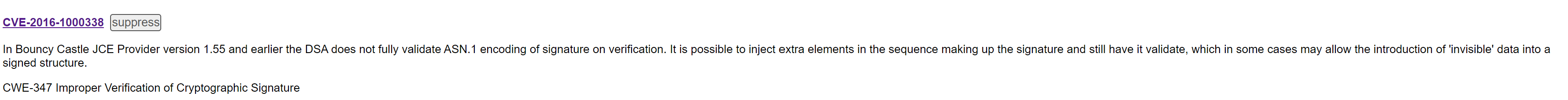
## Static Testing

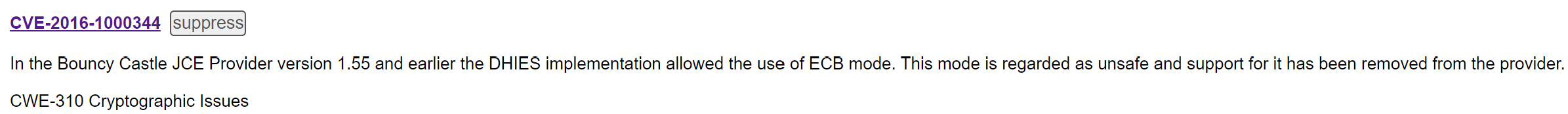
A screenshot of a computer

Description automatically generated

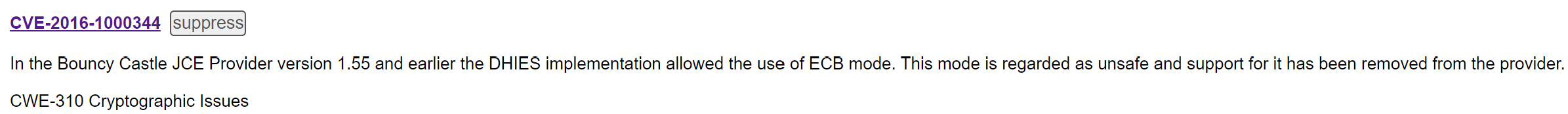
Above are all 13 dependencies this codebase uses with known vulnerabilities. I will list vulnerabilities for each of these dependencies below and link an example of the responses to them in section 5. Most severe vulnerabilities will be included.

Vulnerabilities related to Bouncy Castle Crypto

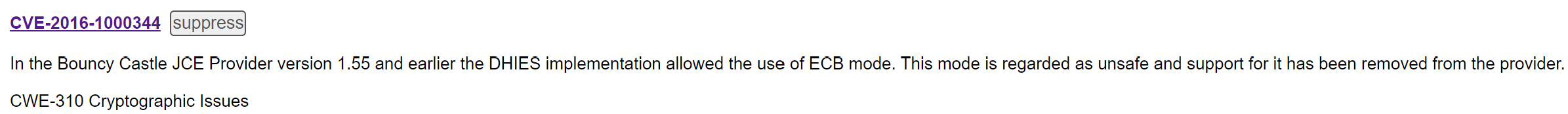




Vulnerabilities related to Hibernate’s Bean Validation



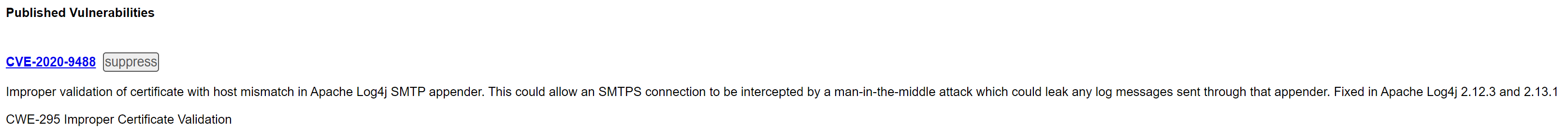
Vulnerabilities related to Jackson Databind



A close-up of a text

Description automatically generated

Vulnerabilities related to Apace Logj4 API

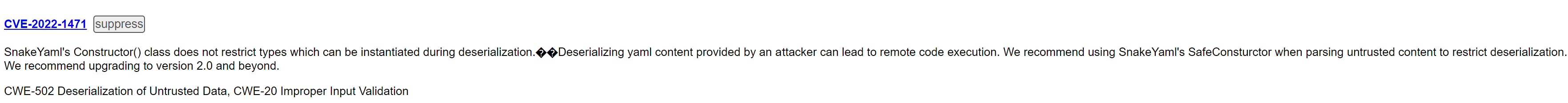


Vulnerabilities related to Logback Core Module

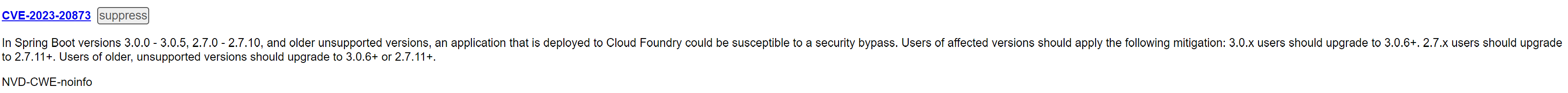
A close up of a text

Description automatically generated

Vulnerability related to YAML 1.1 parser and emitter for Java



Vulnerability related to SpringBoot

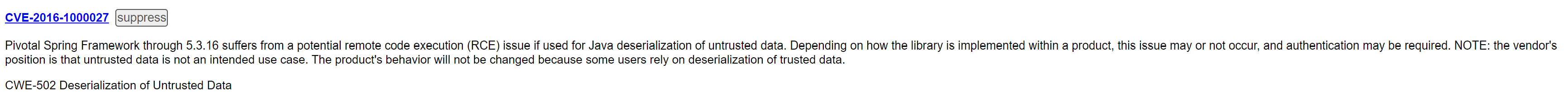


Vulnerability related to Spring Core and Spring Web, Spring Web MVC

A close-up of a white background

Description automatically generated

Vulnerability related to Spring Web



Vulnerability related to Tomcat Core

A close-up of a text

Description automatically generated

A close-up of a text

Description automatically generated

## Mitigation Plan

I will first address the results of my manual inspection and provide potential fixes.

**Input Validation and Secure Data**

GreetingController.java lines 15-18

CRUDController.java lines 12-16

HTTP GET requests can expose valuable information through the URL and it is stored in browser history. Replace this with a @RequestMapping or POST request that will not store data in this way.

Input validation should also be included within the program and NOT just plugged into the URL. Parameterizing input before sending requests to the server will be helpful to prevent DOS and Injection attacks.

**Error Handling**

As I mentioned above, the server has no endpoint to handle errors. Adding an @ExceptionHandler will properly redirect when a request is invalid.

**Access Control**

Secure data transfer can be accomplished with a proper HTTPS protocol. Additionally, users should not be able to access a server or a database without providing proper credentials. Access Control and Authorization ideally should be in place for all users wishing to connect to the server.

**API’s**

No matter how secure your codebase may be, if you rely on open source architectures for a project you are subjecting yourself to all of their flaws and oversights. Luckily, many open source technologies are continually updated by dedicated teams, which will save us from auditing potentially thousands of lines of code. Sometimes, the best thing we can do to prevent exploits is to keep our dependencies up to date. Below is an abbreviated list of proposed solutions to known vulnerabilities in the dependencies I listed above.

CVE-2016-1000338

A 3rd party advisory was released and a PATCH for the software was released that fixes the exploit

<https://github.com/bcgit/bc-java/commit/b0c3ce99d43d73a096268831d0d120ffc89eac7f#diff-3679f5a9d2b939d0d3ee1601a7774fb0>

CVE-2016-1000344

A 3rd party advisory was released and a PATH for the software was released that fixes the exploit

<https://github.com/bcgit/bc-java/commit/9385b0ebd277724b167fe1d1456e3c112112be1f>

CVE-2020-10963

A 3rd party advisory was released and a PATH for the software was released that fixes the exploit

<https://www.oracle.com/security-alerts/cpuapr2022.html>

CVE-2020-25649

A 3rd party advisory was released and a PATH for the software was released that fixes the exploit

<https://github.com/FasterXML/jackson-databind/issues/2589>

CVE-2020-36518

The issue is being actively tracked

<https://github.com/FasterXML/jackson-databind/issues/2816>

CVE-2020-9488

The issue has been patched

https://issues.apache.org/jira/browse/LOG4J2-2819

CVE-2020-6378

The issue has been patched

<https://logback.qos.ch/news.html#1.3.12>

CVE-2022-1471

The issue has been reported and potential solutions have been formulated

<https://github.com/google/security-research/security/advisories/GHSA-mjmj-j48q-9wg2>

CVE-2023-20873

Update version to patch exploit

<https://spring.io/blog/2023/05/18/spring-boot-2-5-15-and-2-6-15-available-now>

CVE-2023-22965

Issue has been patched

<https://www.oracle.com/security-alerts/cpujul2022.html>

CVE-2016-1000027

Issue is being tracked and advisories have been released

<https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2016-1000027>

CVE-2020-1938

Issue has been patched

<https://www.oracle.com/security-alerts/cpujan2021.html>

CVE-2023-4487

Dozens of advisories were released, patches as well

<https://cgit.freebsd.org/ports/commit/?id=c64c329c2c1752f46b73e3e6ce9f4329be6629f9>

WORKS CITED

Ritchie, J. N. & A., & Technology, T. O. of. (2023, October 6). FTC Safeguards Rule: What your business needs to know. Federal Trade Commission. https://www.ftc.gov/business-guidance/resources/ftc-safeguards-rule-what-your-business-needs-know