****

# Artemis Financial Vulnerability Assessment Report

**Table of Contents**

[Document Revision History 3](#_gjdgxs)

[Client 3](#_30j0zll)

[Instructions 3](#_1fob9te)

[Developer 4](#_3znysh7)

[1. Interpreting Client Needs 4](#_2et92p0)

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

[3. Manual Review 4](#_3dy6vkm)

[4. Static Testing 4](#_1t3h5sf)

[5. Mitigation Plan 4](#_4d34og8)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **Nov 10th 2023** | **Nate Dukes** | **Initial Revision** |
| **1.1** | **Nov 12th** | **Nate Dukes** | **“Investigated the static report”** |

## 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

Nate Dukes

## Interpreting Client Needs

Artemis financial is a consulting company which helps clients to better plan out their finances, including savings, retirement, investments, and insurance planning. Their goal is to update their software so as to be as secure as possible, as they deal with the financial stability and goals of their clients, potentially life changing money. The company will have to deal with international transactions, which could complicate the design of the software- there are many considerations to be made and compromises will inevitably need to fall into place where the law is not so clear. Any exploitation of the user’s data could be a catastrophic failure, where in there could be data breaches, identities stolen, or potentially even worst. With the power of modern computers, and clever software engineers, there is an ongoing battle between hackers and software security developers with strengthening encryption. This will be the focal point of the modernization of the software- protecting the important information of the clients.

## Areas of Security

1. Input validation: Perhaps the most obvious thing to ensure is secure is the input validation. The whole point of this update is to make sure that exactly this happens- effective and secure validation of input, so that nobody unwanted can make their way in...

2. Encryption/Cryptography: Second only to input validation in importance here, transferring and using the data that the users maintain is the next place that things can go wrong- even before the data is transferred. Making sure that the data is not insecure through obfuscation is another essential part of securing the user’s data.

3. APIs: Pretty much everything to do with finances is going to involve an API, this software is no different. Having a solid API to transfer and transform the user’s data is essential.

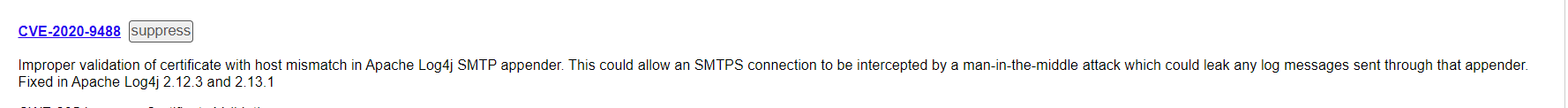
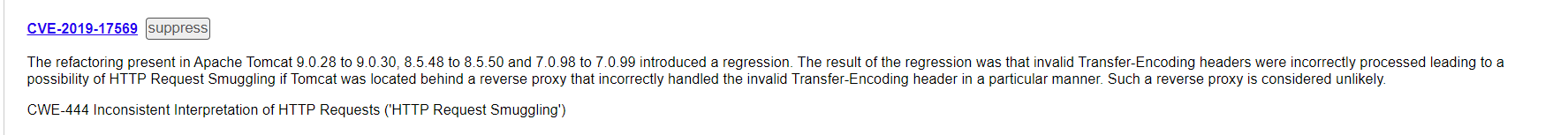
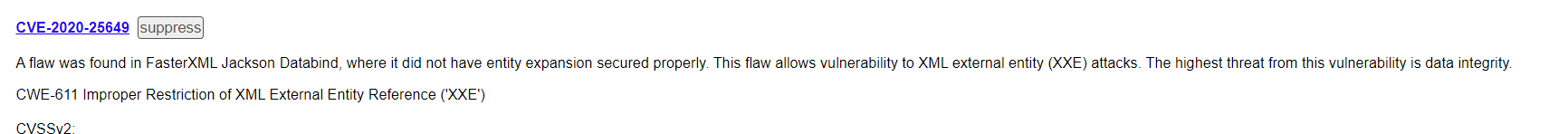
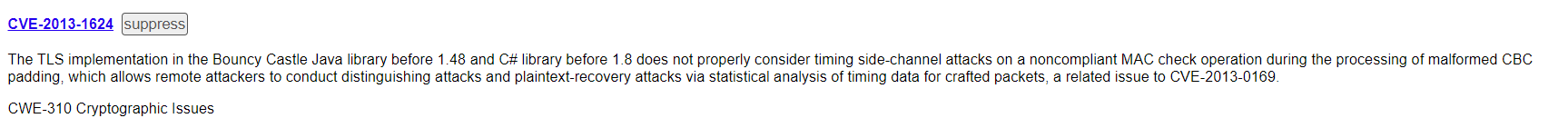
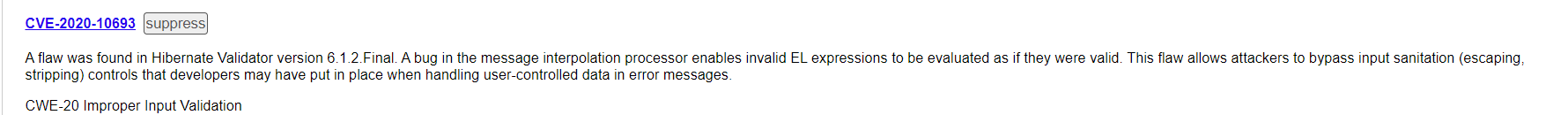
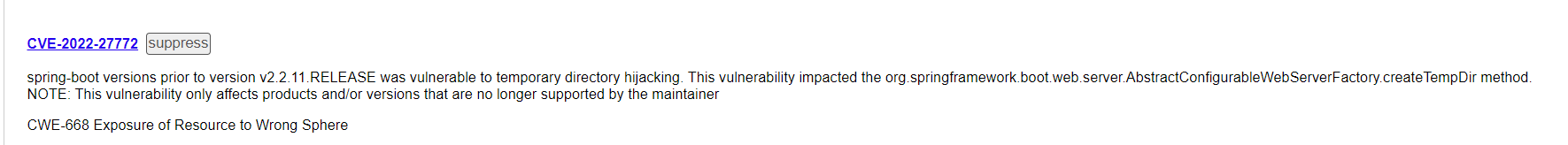
4/5. Code quality and error management. If at any point these two things are not taken into consideration, the failure is likely to be catastrophic, and cause major problems for both the user, and the company that was assigned to protect and advise them on their finances.

## 

## Manual Review

It is apparent from a brief overview of the code that there are numerous outdated sections. Chief among them is that lots of things are outdated. The spring framework is vulnerable to numerous attacks, possibly redundant to even mention as these will all show up in the static testing section. The code itself seems okay, not spectacular, a little bit messy with some redundancy for sure, but they have made so much of the objects public and their methods. I didn’t look in depth into all of them because it was so clear that these methods needed to be hidden behind a layer or two of abstraction. I didn’t see any input validation methods in the code, eg when depositing balance. Upscaling this software for international, modern, wide-scale use will take a lot of work.

## 4. Static Testing



As expected, the most evident CVEs are those that deal with input validation, cryptography, and data leaks. For being a company focused on updating security, it almost seems like it was intentionally left insecure and open to attackers. Who would have guessed? The solution for all of these is to update the software. Jackson, Oracle, bouncy castle, and especially spring framework have released patches, and the rest are solvable by making proper input validation according to the sources listed below the cve ids in the given report. This advice is applicable to every single CVE on this vulnerabilities list. Update and maintain, and develop proper input validation.

## 5. Mitigation Plan

As of now, I think it is in the company’s best interest to update all their software, hardware, and any plugins, apis, or frameworks that they have been using. Many of them are out of date. Next, it will be vital to take a good look over the code to see where permissions are, whom they are given to, and how to deny these permissions from outside sources. There are too many vulnerabilities as a result of the current layout, and updating it to validate input and encrypt the data more effectively will be essential going forwards.