GLOBALRAIN

# Practices for Secure Software Report

## Table of Contents

**Document Revision History**

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | Aug. 6, 2023 | Alexander Ahmann | |

**Client**

ARTEMIS
FINANCIAL

**Developer**
Alexander Ahmann

## 1. Algorithm Cipher

*Artemis Financial* is, as its name implies, a financial firm, further implying that the confidentiality and integrity[1] of the data are of top priority. Accessibility is important as well, but information relating to the client and its accuracy is more important, so it is of secondary importance in this author's opinion.[2] For this project, I will use the *Advanced Encryption Standard*[3] (specifically, AES-256) and the *Secure Hash Algorithm*[4] (specifically, SHA-512/256).

The Advanced Encryption Standard is an encryption procedure[5] that is used to transform plaintext into ciphertext, with AES-256 being the most secure implementation to my knowledge. The *Secure Hashing Algorithm* is a hashing procedure that takes a data structure of finite size and transforms it into a fixed string that has a significant change to its output if a bit in the data structure is changed. SHA-512/256 is the most secure variant to my knowledge.

The level of security that is provided by both an encryption and hashing procedure is in no small part determined by its key length as measured in bits.[6] The key size of a particular encryption or hashing procedure is typically denoted by a number on the right-hand side of the abbreviation of its name (s.a. AES-256 or SHA-512/256). A larger key size implies a more secure implementation or variant of the procedure in question.

---

[1] The terms *confidentiality*, *integrity* and *accessibility* are derived from the *CIA triad* used to conceptialise what kind of security engineering problems are to be solved; see Fortinet (n.d.). *What is the CIA triad and Why is it Important*? Retrieved on Aug. 14, 2023 from: https://www.fortinet.com/resources/cyberglossary/cia-triad

[2] This project's technical writeup is biased in favour of maintaining confidentiality and integrity. Others may disagree and believe that there should be equal balance between the three components of the CIA triad; or even a bias in favour of accessibility and convenience at the cost of maintaining confidentiality and integrity.

[3] For a detailed description, see: Selent, D. (2015). Advanced Encryption Standard. In Rivier Academic Journal (Volume 6, No. 2). Retrieved on Aug. 14, 2023 from: https://www2.rivier.edu/journal/roaj-fall-2010/j455-selent-aes.pdf

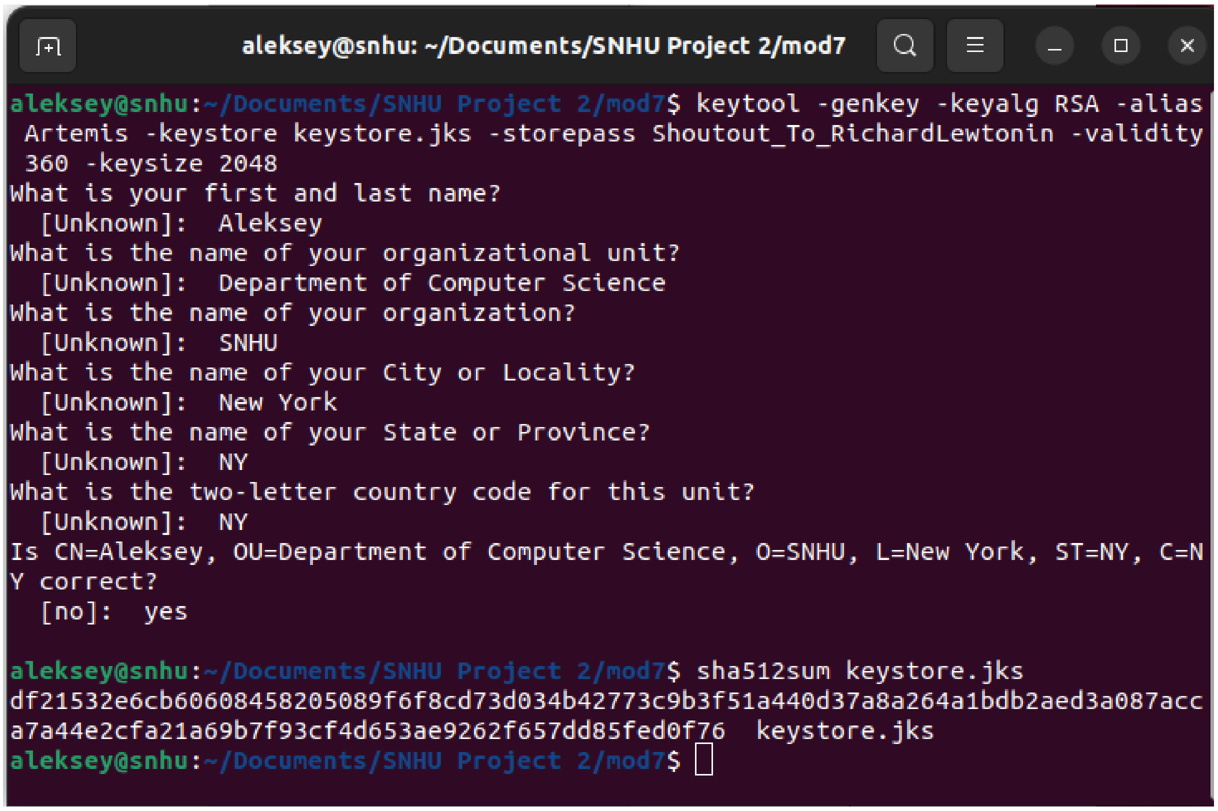
[4] For a detailed description, see Gueron, S., Johnson, S., Walker, J. (2011). SHA-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*, *Las Vegas, NV, USA, 2011, (pp. 354-358)*. https://doi.org/10.1109/ITNG.2011.69

[5] Specifically, *Advanced Encryption Standard* is a symmetric encryption procedure, where the same key is used to both encrypt and decrypt the data structure of interest. This contrasts with an asymmetric encryption procedure, where a public key is used to encrypt data and a private key is used to decrypt data.

[6] Another element that contributes to the security of a cryptographic procedure is the level of randomness of numbers produced by a *pseudo-random number generator* (PRNG). A procedure to generate random numbers that is easy to predict would lessen the security of the encryption procedure. The inverse is also the case: a procedure that generates random numbers that is hard to predict would increase the security of the encryption procedure.

2. **Certificate Generation**[7]

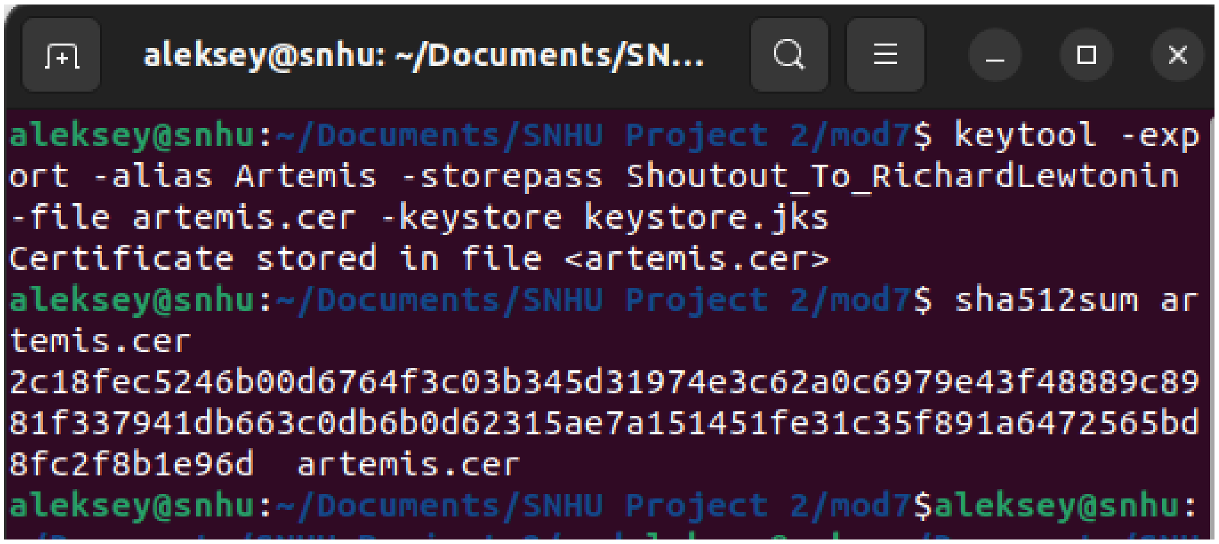
The figures below depicts my procedure for generating an RSA[8] certificate:



```
aleksey@snhu:~/Documents/SNHU Project 2/mod7$ keytool -genkey -keyalg RSA -alias
 Artemis -keystore keystore.jks -storepass Shoutout_To_RichardLewtonin -validity
 360 -keysize 2048
What is your first and last name?
  [Unknown]:  Aleksey
What is the name of your organizational unit?
  [Unknown]:  Department of Computer Science
What is the name of your organization?
  [Unknown]:  SNHU
What is the name of your City or Locality?
  [Unknown]:  New York
What is the name of your State or Province?
  [Unknown]:  NY
What is the two-letter country code for this unit?
  [Unknown]:  NY
Is CN=Aleksey, OU=Department of Computer Science, O=SNHU, L=New York, ST=NY, C=N
Y correct?
  [no]:  yes

aleksey@snhu:~/Documents/SNHU Project 2/mod7$ sha512sum keystore.jks
df21532e6cb60608458205089f6f8cd73d034b42773c9b3f51a440d37a8a264a1bdb2aed3a087acc
a7a44e2cfa21a69b7f93cf4d653ae9262f657dd85fed0f76  keystore.jks
aleksey@snhu:~/Documents/SNHU Project 2/mod7$ 
```

[7] I consulted the following resource for the usage of keytool: SSL Shopper (n.d.). *How to Create a Self Signed Certificate using Java Keytool*. Retrieved on Aug. 14, 2023 from: https://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html

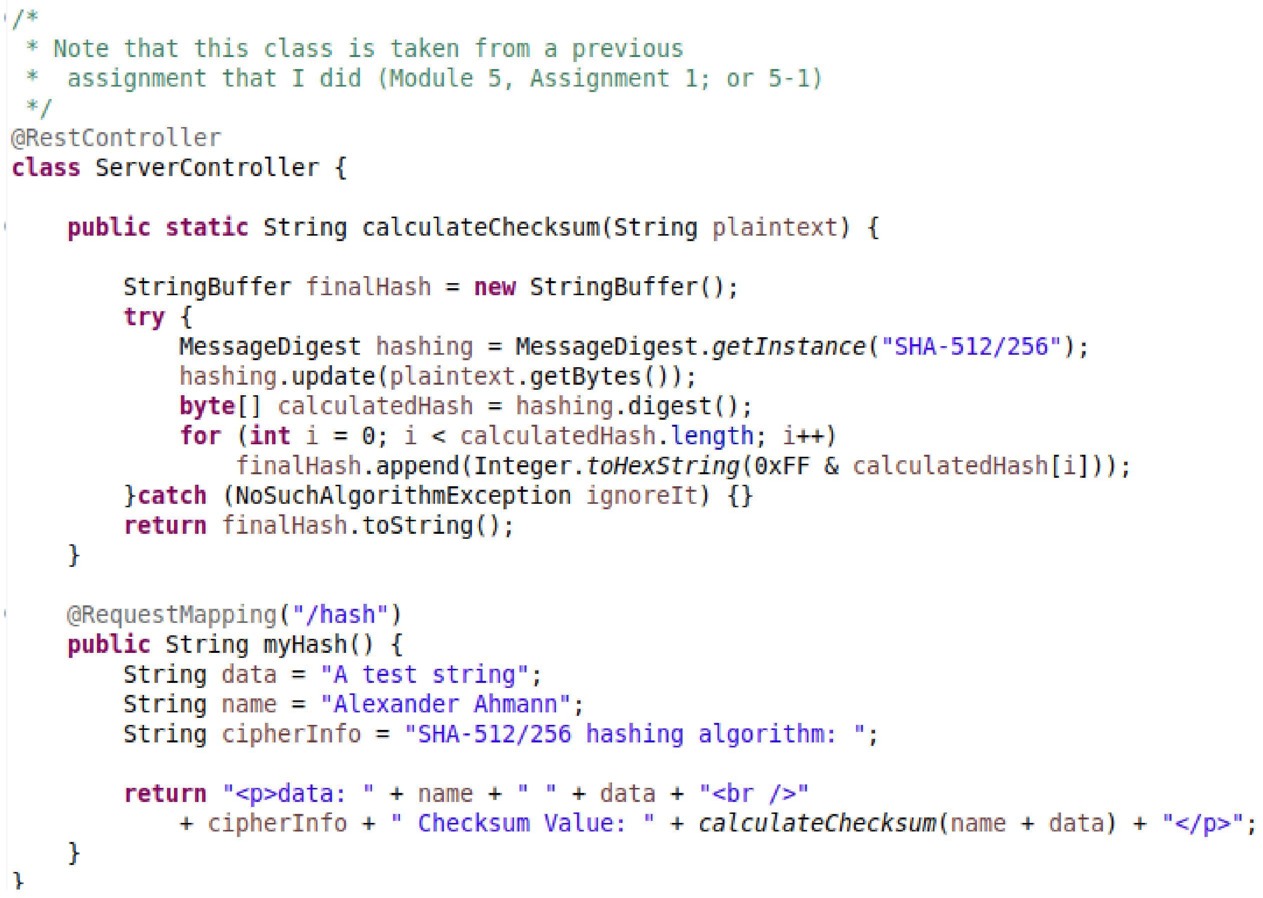
[8] This is something of the "asymmetric" equivalent of the Advanced Encryption Standard. Also do note that I increased the keysize from 256 bits to 2048 bits.



```
aleksey@snhu:~/Documents/SNHU Project 2/mod7$ keytool -exp
ort -alias Artemis -storepass Shoutout_To_RichardLewtonin
-file artemis.cer -keystore keystore.jks
Certificate stored in file <artemis.cer>
aleksey@snhu:~/Documents/SNHU Project 2/mod7$ sha512sum ar
temis.cer
2c18fec5246b00d6764f3c03b345d31974e3c62a0c6979e43f48889c89
81f337941db663c0db6b0d62315ae7a151451fe31c35f891a6472565bd
8fc2f8b1e96d  artemis.cer
aleksey@snhu:~/Documents/SNHU Project 2/mod7$aleksey@snhu:
```

## 3. Deploy Cipher

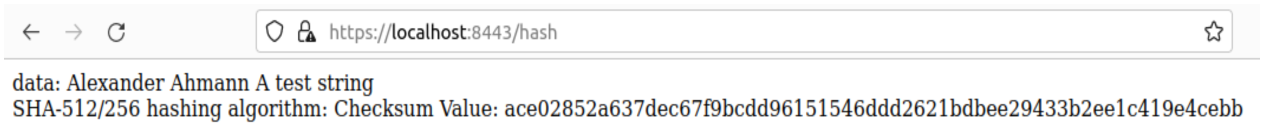
This is the source code that I have reapplied from a previous assignment to calculate SHA-512/256 hashes from plaintext string data:



```java
/*
 * Note that this class is taken from a previous
 *  assignment that I did (Module 5, Assignment 1; or 5-1)
 */
@RestController
class ServerController {

    public static String calculateChecksum(String plaintext) {

        StringBuffer finalHash = new StringBuffer();
        try {
            MessageDigest hashing = MessageDigest.getInstance("SHA-512/256");
            hashing.update(plaintext.getBytes());
            byte[] calculatedHash = hashing.digest();
            for (int i = 0; i < calculatedHash.length; i++)
                finalHash.append(Integer.toHexString(0xFF & calculatedHash[i]));
        }catch (NoSuchAlgorithmException ignoreIt) {}
        return finalHash.toString();
    }

    @RequestMapping("/hash")
    public String myHash() {
        String data = "A test string";
        String name = "Alexander Ahmann";
        String cipherInfo = "SHA-512/256 hashing algorithm: ";

        return "<p>data: " + name + " " + data + "<br />"
            + cipherInfo + " Checksum Value: " + calculateChecksum(name + data) + "</p>";
    }
}
```

## 4. Secure Communications[9]

After refactoring the *application.properties* configuration file, I managed to get the web service running with an HTTPS/SSL/TLS certificate.[10]



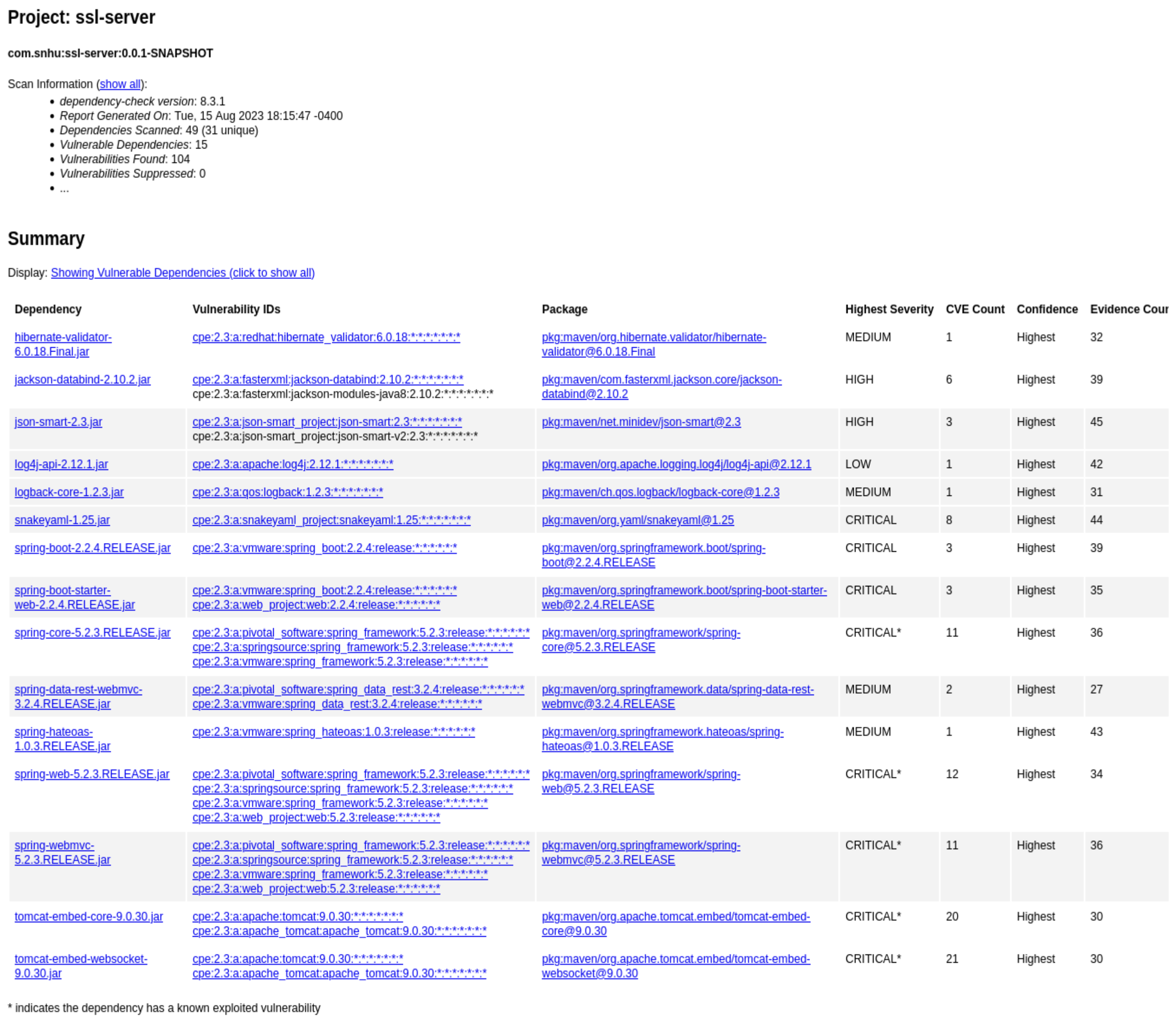
```
data: Alexander Ahmann A test string
SHA-512/256 hashing algorithm: Checksum Value: ace02852a637dec67f9bcdd96151546ddd2621bdbee29433b2ee1c419e4cebb
```

[9] I referenced the following article when doing this section: Vitale, T. (2017). How to enable HTTPS in a Spring Boot Java application. Retrieved on Aug. 15, 2023 from: https://www.thomasvitale.com/https-spring-boot-ssl-certificate/

[10] The web browser reports that the HTTPS connection is "Not Secure." I think that the reason for this being the case is that this certificate has not been registered with a formal certificate authority.

## 5. Secondary Testing

I proceeded to use the *Apache Maven* tool to do a preliminary assessment of software vulnerabilities; the screenshot below is an excerpt of its output.



### Project: ssl-server

**com.snhu:ssl-server:0.0.1-SNAPSHOT**

Scan Information (show all):

- *dependency-check version*: 8.3.1
- *Report Generated On*: Tue, 15 Aug 2023 18:15:47 -0400
- *Dependencies Scanned*: 49 (31 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 104
- *Vulnerabilities Suppressed*: 0
- ...

### Summary

Display: Showing Vulnerable Dependencies (click to show all)

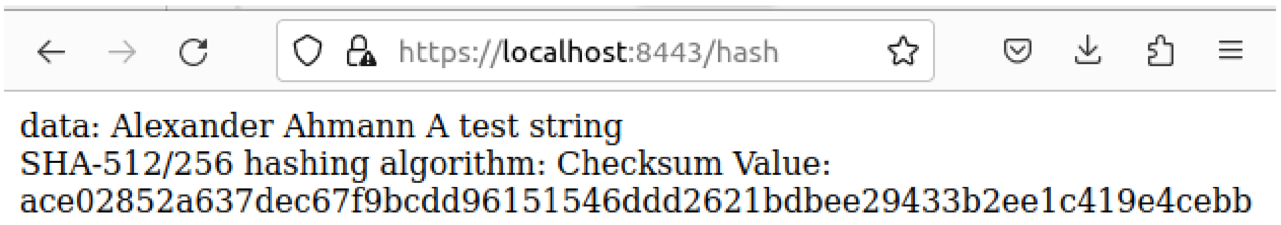
| Dependency | Vulnerability IDs | Package | Highest Severity | CVE Count | Confidence | Evidence Cour |
|---|---|---|---|---|---|---|
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:* | pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final | MEDIUM | 1 | Highest | 32 |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:*<br>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 |
| json-smart-2.3.jar | cpe:2.3:a:json-smart_project:json-smart:2.3:*:*:*:*:*:*:*<br>cpe:2.3:a:json-smart_project:json-smart-v2:2.3:*:*:*:*:*:*:* | pkg:maven/net.minidev/json-smart@2.3 | HIGH | 3 | Highest | 45 |
| 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_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_boot:2.2.4:release:*:*:*:*:*:*<br>cpe:2.3:a:web_project:web:2.2.4:release:*:*:*:*:*:* | pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE | CRITICAL | 3 | Highest | 35 |
| spring-core-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:* | pkg:maven/org.springframework/spring-core@5.2.3.RELEASE | CRITICAL* | 11 | Highest | 36 |
| spring-data-rest-webmvc-3.2.4.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_data_rest:3.2.4:release:*:*:*:*:*:*<br>cpe:2.3:a:vmware:spring_data_rest:3.2.4:release:*:*:*:*:*:* | pkg:maven/org.springframework.data/spring-data-rest-webmvc@3.2.4.RELEASE | MEDIUM | 2 | Highest | 27 |
| spring-hateoas-1.0.3.RELEASE.jar | cpe:2.3:a:vmware:spring_hateoas:1.0.3:release:*:*:*:*:*:* | pkg:maven/org.springframework.hateoas/spring-hateoas@1.0.3.RELEASE | MEDIUM | 1 | Highest | 43 |
| spring-web-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:* | pkg:maven/org.springframework/spring-web@5.2.3.RELEASE | CRITICAL* | 12 | Highest | 34 |
| spring-webmvc-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:*<br>cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:* | pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE | CRITICAL* | 11 | Highest | 36 |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:*<br>cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:* | pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30 | CRITICAL* | 20 | Highest | 30 |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:*<br>cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:* | pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30 | CRITICAL* | 21 | Highest | 30 |

* indicates the dependency has a known exploited vulnerability

## 6. Functional Testing

From my purview, there is not much significant errors in the code that warrant more refactorings of it. The codebase was rather small, but nonetheless I may have missed a programming error and recommend sending this to a more experienced programmer for their input. Vulnerable packages from the *Apache Maven* report may need to be updated or upgraded.

A screenshot of the web application running post-refactoring is provided here:



```
data: Alexander Ahmann A test string
SHA-512/256 hashing algorithm: Checksum Value:
ace02852a637dec67f9bcdd96151546ddd2621bdbee29433b2ee1c419e4cebb
```

## 7. Summary

For this project, I addressed many issues with regards to the security of a web application. I addressed cryptography, code error handling, and code quality control.[11] I created a cryptographic certificate that can be used to access the website securely, implemented a procedure to calculate SHA-512/256 checksums, and worked out what dependencies are vulnerable so that other developers can upgrade them.

## 8. Industry Standard Best Practices

In this report, I was able to implement standard tools like Apache Maven to detect security flaws in dependencies, and encryption and hashing algorithms to strengthen the confidentiality and integrity of the web application. This will lessen, though not totally eliminate, vulnerability vectors that a malicious hacker could exploit to gain access to sensitive information or to modify sensitive information. This means that customers will feel safer when making transactions and *Artemis Financial* can better service its clients without having to fear for malicious hackers.[12]

[11] After the Vulnerability Assessment Process Flow Diagram.

[12] This is not totally true because security is an ongoing process. No ”magic bullet” solution exists to totally keep out malicious hackers, and it is up to technicians and systems administrators to constantly monitor their networks and kick out malicious hackers.