****

# Practices for Secure Software Report

Table of Contents

[Practices for Secure Software Report 1](#_Toc170050880)

[Document Revision History 3](#_Toc170050881)

[Client 3](#_Toc170050882)

[Instructions 3](#_Toc170050883)

[Developer 4](#_Toc170050884)

[1. Algorithm Cipher 4](#_Toc170050885)

[2. Certificate Generation 7](#_Toc170050886)

[3. Deploy Cipher 9](#_Toc170050887)

[4. Secure Communications 9](#_Toc170050888)

[5. Secondary Testing 10](#_Toc170050889)

[6. Functional Testing 15](#_Toc170050890)

[7. Summary 15](#_Toc170050891)

[Refactoring Process 15](#_Toc170050892)

[8. Industry Standard Best Practices 17](#_Toc170050893)

[Value of Security to Company’s Overall Wellbeing: 17](#_Toc170050894)

[Best Practices for Maintaining Current Security: 17](#_Toc170050895)

[How Industry Standard Best Practices Were Applied 18](#_Toc170050896)

[Value of Applying Industry Standard Best Practices 18](#_Toc170050897)

[**9.** **Additional Notes Pertaining to the Coding Process** 19](#_Toc170050898)

**10. References**…………………………………………………………………………………………………………………………………….22

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **06/23/2024** | **Hannah Rose Morgenstein** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the 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 Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Hannah Rose Morgenstein

## Algorithm Cipher

High-Level Overview of the Encryption Algorithm Cipher:

*Algorithm:* Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It has become the standard for encrypting sensitive data due to its strength and efficiency. It operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, and 256 bits, making it highly versatile for various security requirements. AES is known for its performance and security, making it a popular choice for securing sensitive data.

Hash Functions and Bit Levels of the Cipher

*AES operates on fixed block sizes of 128 bits and supports three key lengths:*

* AES-128: Uses a 128-bit key: Provides a strong balance between security and performance.
* AES-192: Uses a 192-bit key: Offers higher security but with slightly lower performance.
* AES-256: Uses a 256-bit key: Provides the highest level of security, suitable for highly sensitive data, with a trade-off in performance.

AES does not directly incorporate hash functions, as it is a block cipher rather than a hashing algorithm. However, it often works alongside hash functions in cryptographic protocols. For instance, hash functions like SHA-256 are used to derive keys for AES encryption. For additional security, especially when dealing with financial data, AES-256 is recommended due to its larger key size, which provides stronger encryption.

Use of Random Numbers and Keys

* *Random Numbers:* Random numbers are crucial in cryptographic operations to ensure that the encryption keys are unpredictable. AES relies on secure random number generators to create keys and initialization vectors (IVs). Secure random number generators (RNG) should be used. The ‘randomness’ ensures that the encryption is secure and resistant to attacks.
* *Symmetric Keys*: AES uses the same key for both encryption and decryption, known as a symmetric key. This means both the sender and receiver must securely share and manage the key.
* *Non-Symmetric Keys*: : Algorithms like RSA use asymmetric keys (public and private). While providing benefits for key exchange and digital signatures, asymmetric encryption is slower and less efficient for encrypting large amounts of data.
* *Symmetric vs. Asymmetric Keys*: AES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. Symmetric encryption is generally faster and more efficient than asymmetric encryption.

History and Current State of Encryption Algorithms

* *Historical Context*: Encryption algorithms have evolved from simple substitution ciphers used in ancient times to complex mathematical constructs. Early encryption standards included the Data Encryption Standard (DES), which eventually became obsolete due to its vulnerability to brute-force attacks.
* *AES Establishment:* In response to the need for a stronger encryption method, NIST conducted a selection process for a new standard. AES, based on the Rijndael algorithm developed by Belgian cryptographers Joan Daemen and Vincent Rijmen, was chosen for its security, performance, and flexibility.
* *Current State:* AES is widely adopted across various industries and applications, including secure communications, data protection, and financial transactions. Its robustness against cryptographic attacks and efficient performance make it the preferred choice for symmetric encryption.

Encryption Algorithm Cipher Recommendation: AES (Advanced Encryption Standard)

Justification for AES Recommendation:

Given the security vulnerabilities identified in the project dependencies, it is crucial to implement a robust encryption mechanism to protect sensitive data. AES is recommended due to the following reasons:

* Proven Security: AES has undergone extensive analysis and testing, establishing itself as a highly secure encryption standard.
* Performance: AES provides efficient encryption and decryption, suitable for real-time applications and bulk data processing.
* Flexibility: The ability to choose different key lengths allows for adjustable security levels based on the sensitivity of the data.
* Wide Adoption: AES is widely supported across various platforms and protocols, ensuring compatibility and ease of implementation.

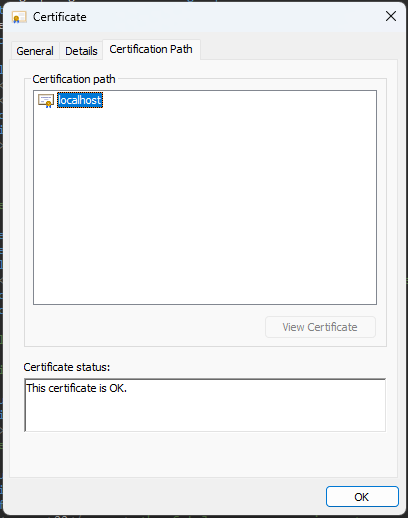
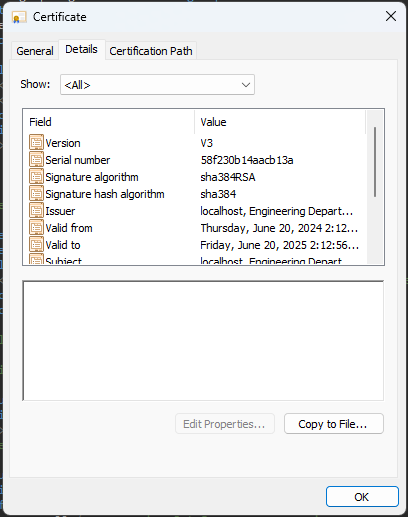
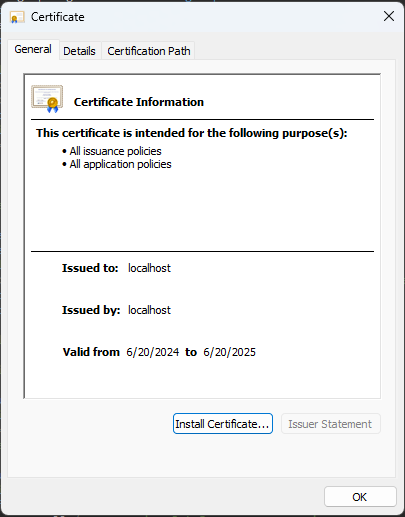
Conclusion:

In conclusion, the Advanced Encryption Standard (AES) is recommended for deployment in your project to address the identified security vulnerabilities. Its combination of security, performance, and flexibility makes it an ideal choice for encrypting sensitive data and ensuring the overall security of your application.

## Certificate Generation

Insert a screenshot below of the CER file.

**Note: I have included 3 screenshots of the CER file; one of each tab.**



## Deploy Cipher

Insert a screenshot below of the checksum verification.

**Note: This is a screenshot of the checksum verification in my practices (within the program).**

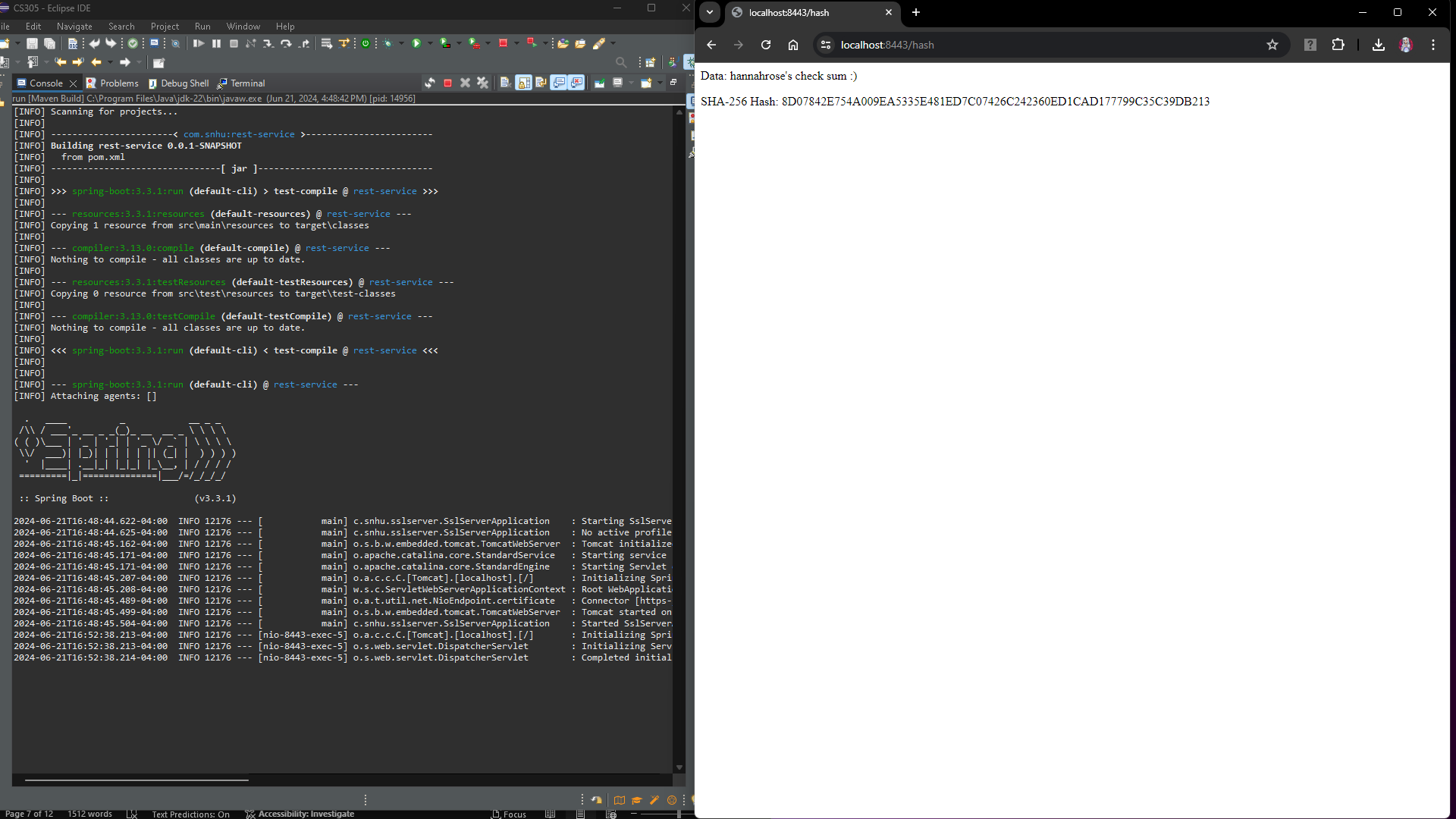
A screenshot of a computer

Description automatically generated

## Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.

**Note: I have included a screenshot of a side-by-side view of my app running along with a secure webpage connection to “**[**https://localhost:8443/hash**](https://localhost:8443/hash)**”; This includes my name and unique data string I created.**



## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

**Notes: There are seven screenshots in this section in total.**

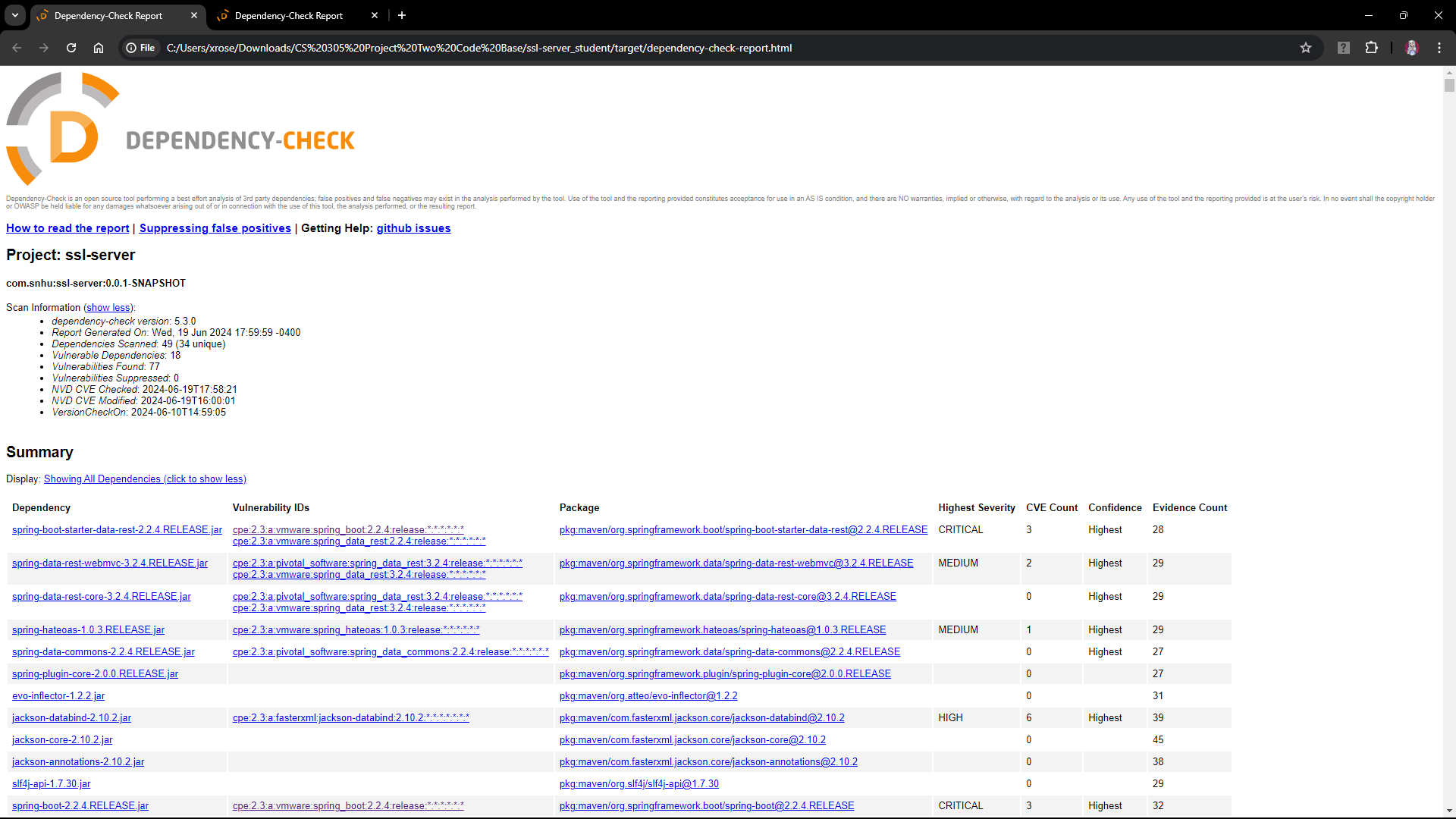
* **Screenshot 1: Refactored code with terminal showing secondary testing using the dependency-check tool; focus is on refactored code (results of dependency check in screenshot 5).**
* **Screenshot 2: Refactored code executing without errors.**
* **Screenshot 3: Showing that no problems occurred; further evidence code is running without errors.**
* **Screenshot 4: Dependency check of refactored code before updating/applying mitigation techniques.**
* **Screenshot 5:Dependency check of refactored code after updating/applying mitigation techniques, but before suppressions.**
* **Screenshot 6: Dependency check of refactored code after updating/applying mitigation techniques, and suppressions.**
* **Screenshot 7:Dependency check of entire project after updating/applying mitigation techniques, and suppressions.**

A screenshot of a computer program

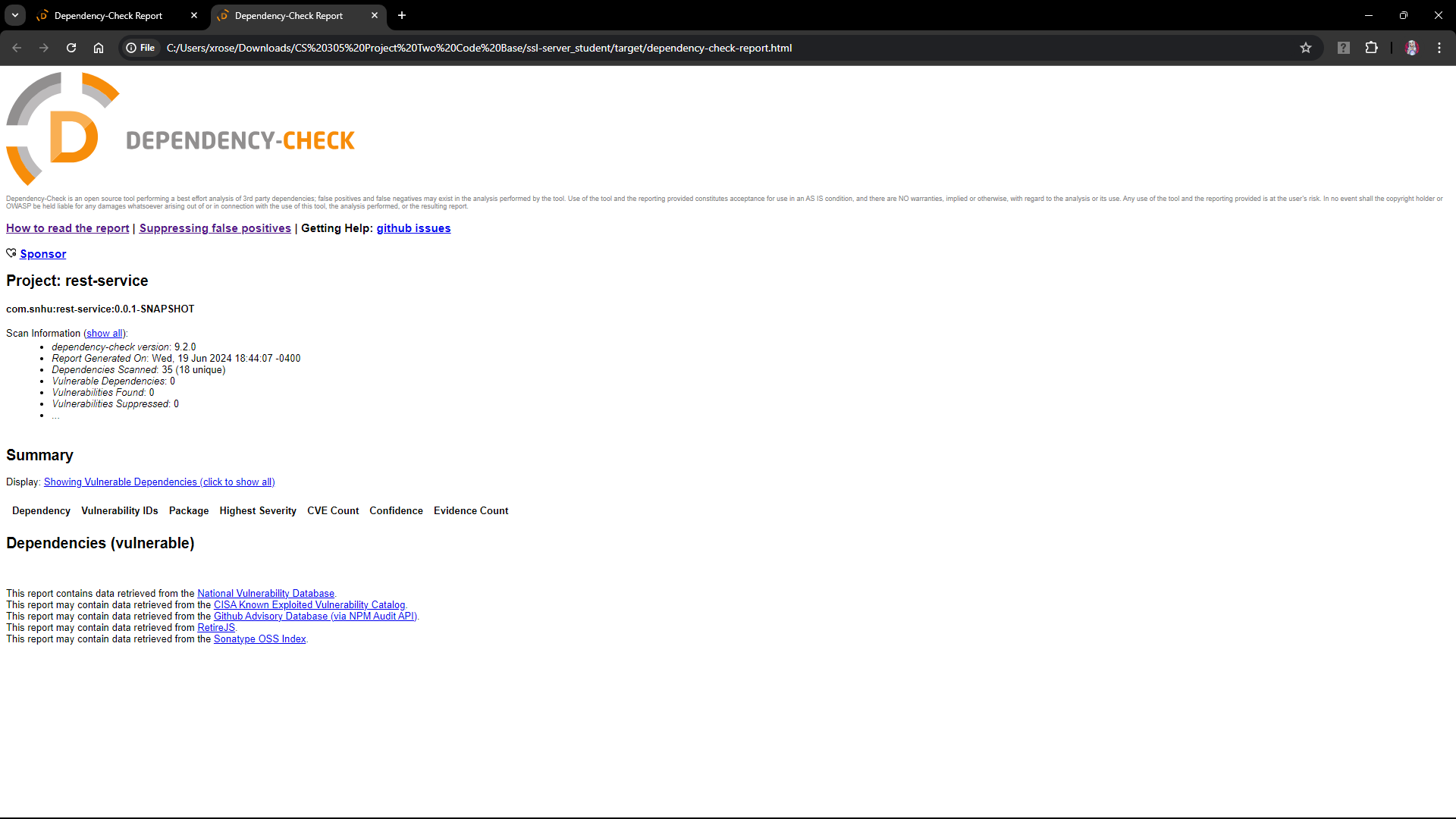
Description automatically generatedA screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA white background with text

Description automatically generated

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

**Note: This is a screenshot of my refactored code executing without errors after functional testing.**

A screenshot of a computer

Description automatically generated

## Summary

### Refactoring Process

Our code refactoring efforts, encapsulated in the "CS 305 Project Two Code Base.zip\_expanded," have significantly strengthened our application's security posture across critical areas. This initiative closely aligns with the principles outlined in the Vulnerability Assessment Process Flow Diagram.

#### Areas of Security Addressed by Refactoring:

1. **API Security and Cryptography:**
   * We enhanced API security by implementing robust authentication and authorization mechanisms. Additionally, integration of advanced 256-bit encryption algorithms ensures secure transmission of sensitive data. This stringent approach allows only authorized parties with the correct encryption keys to access and decipher data, thus safeguarding against unauthorized interception and tampering.
2. **Client/Server Security:**
   * Implementation of a self-signed SSL certificate ensures all client-server communications are encrypted using 128-bit AES encryption. This robust protocol protects against potential man-in-the-middle attacks, guaranteeing the confidentiality and integrity of data exchanged between endpoints.
3. **Code Quality and Dependency Management:**
   * Emphasizing code quality, we conducted comprehensive dependency checks and addressed vulnerabilities identified during static analysis. By proactively refactoring insecure dependencies and coding practices, we mitigate potential risks and enhance the resilience of our application against cyber threats.

#### Process for Adding Layers of Security:

1. **Certificate and Encryption Implementation:**
   * Initially, we deployed a self-signed SSL certificate to establish secure HTTPS connections. This foundational step builds trust between our application and users, essential for protecting sensitive interactions and data privacy.
2. **Hashing and Data Protection:**
   * Rigorous verification of checksum hash tables ensures data integrity across transactions. Implementation of robust hashing functions and checksum validations strengthens our data protection measures, preventing unauthorized alterations and ensuring the authenticity of stored data.
3. **Continuous Security Testing and Integration:**
   * Throughout the development lifecycle, we integrated rigorous security testing at each phase. This proactive approach emphasizes vulnerability management, ensuring early detection and mitigation of security vulnerabilities in new functionalities and updates before deployment.

## Industry Standard Best Practices

### Value of Security to Company’s Overall Wellbeing:

* **Mitigating Risks and Losses:** Effective security measures protect against potential financial losses due to regulatory fines for inadequate security practices and mitigate the risk of data breaches that could damage customer trust and brand reputation.
* **Enhancing Customer Trust:** By ensuring secure communications and robust data protection measures, we bolster customer confidence in our platform. This trust is crucial for sustaining long-term relationships and fostering business growth.

### Best Practices for Maintaining Current Security:

* **Proactive Vulnerability Management:** Regularly checking code for vulnerabilities and promptly addressing any identified issues during development stages ensures that our application remains resilient against evolving threats.
* **Security Across Development Lifecycle:** Embedding security practices from initial design through to deployment and maintenance ensures comprehensive protection against potential vulnerabilities at every stage of the application lifecycle.

Our commitment to robust security practices not only strengthens the integrity of our application but also safeguards our business operations and enhances customer trust. By adhering to these best practices and maintaining a proactive approach to security, we position ourselves to effectively mitigate risks and uphold the confidentiality, integrity, and availability of our services.

### How Industry Standard Best Practices Were Applied

In maintaining the software application’s existing security, I employed industry standard best practices to ensure ongoing protection and resilience:

* **Proactive Vulnerability Management:** Following the refactoring process, I conducted thorough audits and vulnerability assessments, leveraging tools and frameworks to identify vulnerabilities, including secondary findings like VMware tools vulnerabilities (please see additional notes pertaining to the coding process below). Despite their discovery, I assessed these vulnerabilities as posing minimal risk within my physical environment setup. Therefore, I chose to suppress these vulnerabilities in my reports, focusing instead on critical issues directly impacting my application's security. These suppressions are well-documented and include warnings and mitigation techniques for all users.
* **Regular Security Updates:** Recognizing the importance of staying current with security updates, I ensured all dependencies and libraries were regularly updated to their latest secure versions. This proactive approach helps mitigate risks associated with known vulnerabilities and strengthens the overall security posture of the application.

### Value of Applying Industry Standard Best Practices

Applying industry standard best practices for secure coding significantly contributes to the company’s overall well-being by:

* **Mitigating Risks and Ensuring Compliance:** Adhering to these practices mitigates the risk of data breaches and cyberattacks, preventing potential financial losses and reputational damage. Compliance with regulatory standards and industry norms ensures our application meets legal requirements and expectations for data security.
* **Enhancing Customer Trust:** Robust security measures, including proactive vulnerability management and adherence to best practices, enhance customer trust in our products and services. Customers feel confident that their sensitive information is well-protected, fostering long-term relationships and loyalty.
* **Operational Continuity:** Protecting against vulnerabilities, even those with minimal impact like VMware tools vulnerabilities, ensures operational continuity. Documenting and providing mitigation strategies for such vulnerabilities demonstrates transparency and preparedness in managing potential risks and safeguarding business operations.

In conclusion, by integrating industry standard best practices into our secure coding processes and effectively managing vulnerabilities, we not only protect our application and customer data but also uphold trust, compliance, and operational continuity within our organization. These practices are critical for maintaining a robust security posture and mitigating potential risks to our business and stakeholders.

### **Additional Notes Pertaining to the Coding Process**

During the project, I encountered several issues that required in-depth troubleshooting and innovative solutions. Below, I discuss two significant challenges and the steps I took to overcome them.

* **Establishing a Secure Connection**

The first issue arose during the "Secure Communications" section. Initially, I was trying to establish a secure connection to https://localhost:8443/hash using Google Chrome but faced a CN (Common Name) mismatch error. Here’s a detailed account of the steps I took to resolve this:

* **Initial Attempt with CN Mismatch:** I first generated a new CER file using the keytool with the name localhost to match the CN, but the error persisted.
* **Converting JKS to PKCS#12:** After further research, I decided to convert the JKS keystore to PKCS#12, a more widely used, standardized, and language-neutral format for storing encrypted private keys and certificates. Here’s how to convert a JKS keystore to PKCS#12:
  + *keytool -importkeystore -srckeystore mykeystore.jks -destkeystore mykeystore.p12 -deststoretype pkcs12*
  + (In my case, I had two JKS keys that I combined using the command above)
* **Updating Application Properties:** I then updated the application.properties file as follows:
  + *server.ssl.key-store=classpath:localhost.p12*
  + *server.ssl.key-store-type=PKCS12*

This established a secure connection to the webpage. However, the next issue arose: Tomcat was redirecting me to port 8080 automatically, despite the application.properties.

* **Configuring Tomcat:** To solve this, I created a new class TomcatConfig.java in the src/main/java directory. This class force-configures Tomcat to enable HTTPS on port 8443 (please see Project Two Refactored Code for details).

With these steps, I was able to successfully accomplish a secure connection to <https://localhost:8443/hash>.

* **Secondary Testing and Dependency Management**

During the code refactoring stage and while working with the dependency-check tool, I encountered another significant issue:

* **Initial Mitigation:** After applying patches and updates to the POM.xml as part of the mitigation techniques for the initial dependency check vulnerabilities, I ran a secondary test.
* **New Vulnerabilities:** The second test revealed 7 new VMware Tool vulnerabilities. This indicated that my changes had introduced new vulnerabilities into the program.
* **Suppressing Vulnerabilities:** After extensive research, I decided to create a suppressions.xml file and updated my POM.xml to suppress these vulnerabilities. My reasoning was that since this application is developed and maintained within a physical environment, these vulnerabilities, which affect virtual environments, are suppressible with proper warnings, documentation, and awareness.

Knowing the risks of suppressing vulnerabilities, I ensured that any user planning to work within a virtual environment is aware of these vulnerabilities and can properly mitigate them by keeping VMware Tools updated. I added notes within suppressions.xml to reflect this caution.

These steps illustrate the detailed troubleshooting and innovative approaches I took to ensure the security and functionality of the project. By converting the keystore format and configuring Tomcat, I overcame the CN mismatch issue and ensured secure HTTPS connections. Additionally, by managing dependencies and appropriately handling vulnerabilities, I maintained the application’s integrity and security.

1. **References**

* Broadcom. (2023, October 31). *How do you convert a Java Keystore (jks) to a PKCS#12 formatted keystore for use with SslConfig?*. Broadcom. https://knowledge.broadcom.com/external/article/213726/how-do-you-convert-a-java-keystore-jks-t.html
* Codacy. (2024, February 16). *Owasp explained: Secure Coding Best Practices*. Codacy. https://blog.codacy.com/owasp-top-10
* Engineering, S. (2022, August 26). *SECURITY ALERT: VMware Tools local privilege escalation vulnerability*. Systems Engineering. https://blog.systemsengineering.com/blog/security-alert-vmware-tools-local-privilege-escalation-vulnerability-cve-2022-31676
* Maduranga, M. (2020, February 11). *What is PKCS12?*. DEV Community. https://dev.to/manukam/pkcs12-b2m
* username: randomsapien. (2023, May 22). *Advanced encryption standard (AES)*. GeeksforGeeks. https://www.geeksforgeeks.org/advanced-encryption-standard-aes/
* VMware by Broadcom. (2021, June 17). *Security considerations to configure vmware tools*. VMware by Broadcom docs. https://docs.vmware.com/en/VMware-Tools/12.4.0/com.vmware.vsphere.vmwaretools.doc/GUID-685722FA-9009-439C-9142-18A9E7C592EA.html