****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **14-12-2024** | **Alibi Shamat** |  |

## 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

Alibi Shamat

## Algorithm Cipher

Recommended Encryption Algorithm: AES (Advanced Encryption Standard)

AES is a symmetric encryption algorithm known for its security and efficiency. It encrypts data in 128-bit blocks and can handle key sizes of 128, 192, or 256 bits, with higher bit levels providing greater resistance to brute-force attacks. AES uses secure random number generators for initialization vectors (IVs) to provide unpredictability and defend against replay or pattern attacks. Unlike asymmetric encryption (e.g., RSA or ECC), AES employs the same key for encryption and decryption, making it excellent for large-scale data encryption.

Historically, encryption progressed from simple ciphers to advanced standards such as DES, which AES supplanted due to security flaws. Today, AES is the global standard, utilized in protocols such as TLS/SSL and VPNs, and provides excellent resilience to cryptanalytic attacks. AES-256 offers strong protection against upcoming threats, including quantum computing, making it the best option for secure, efficient encryption in modern applications.

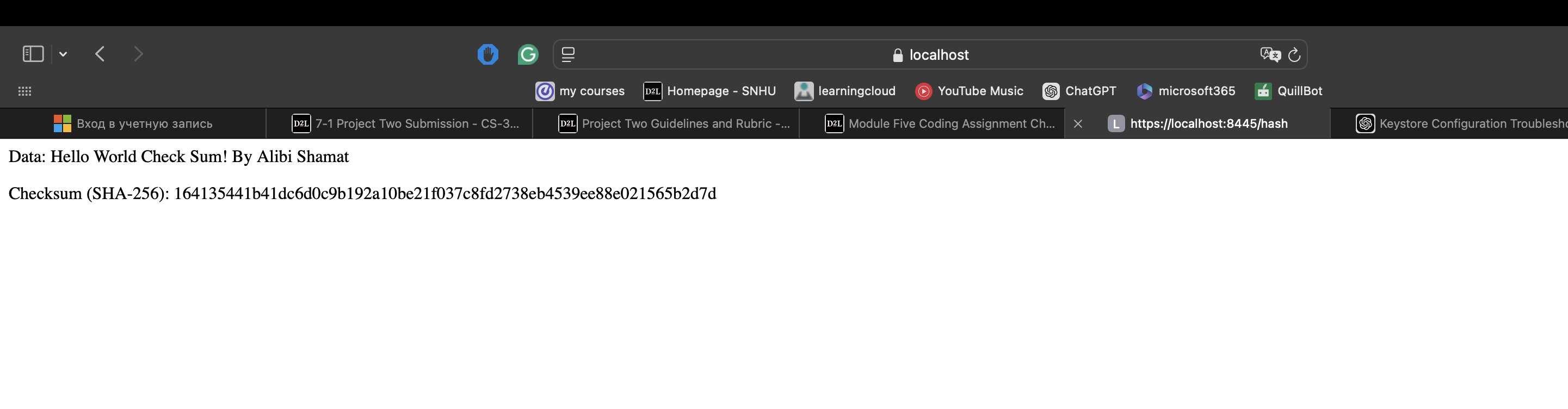
## Certificate Generation

Insert a screenshot below of the CER file.



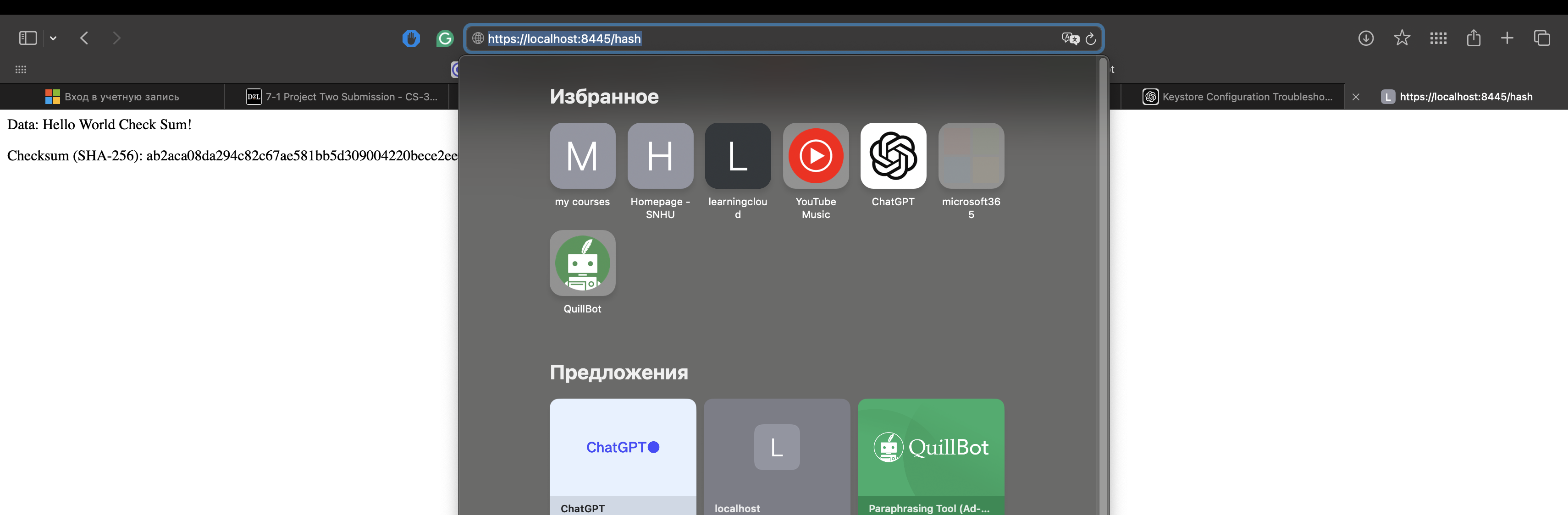
## Deploy Cipher

Insert a screenshot below of the checksum verification.



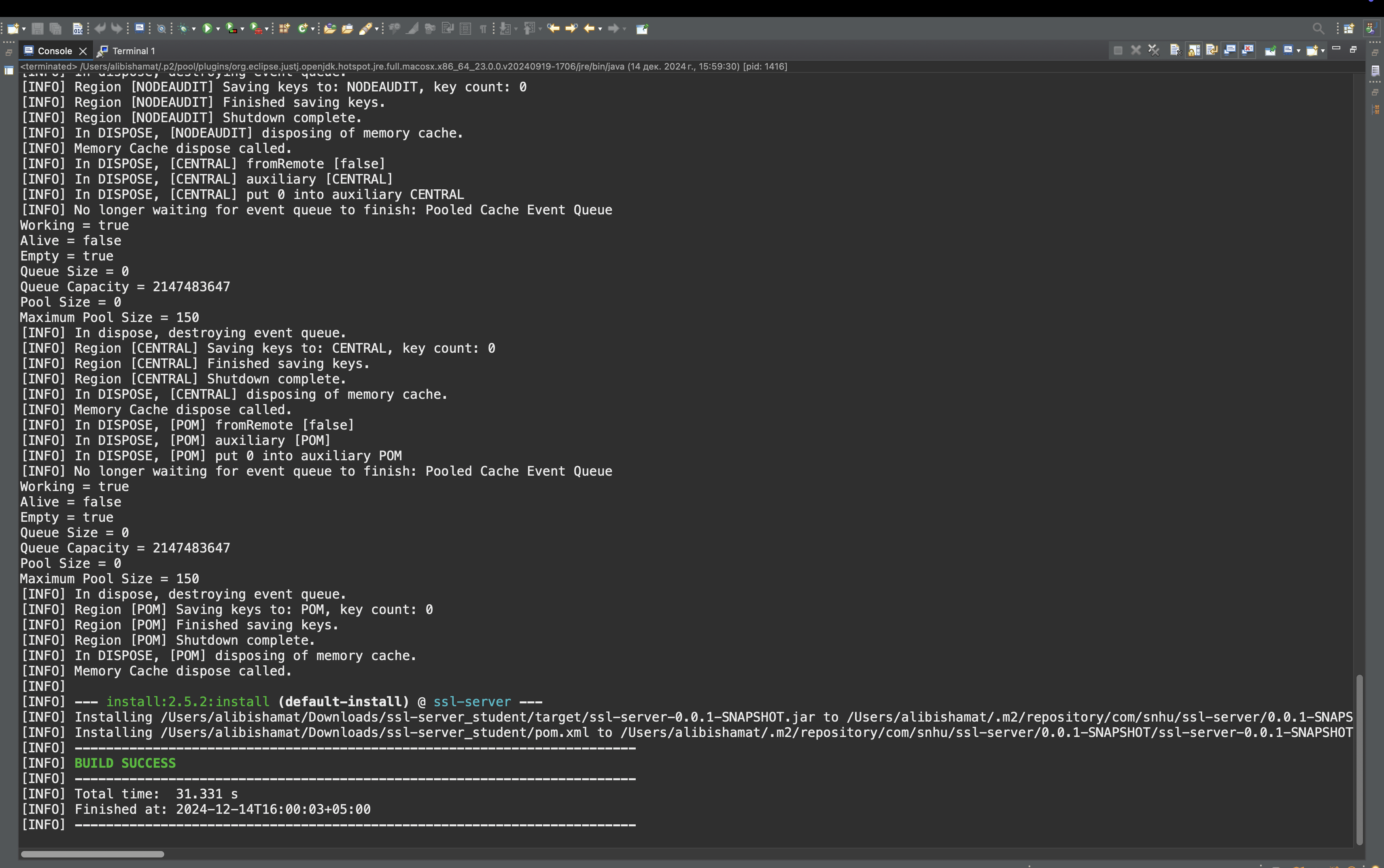
## Secure Communications

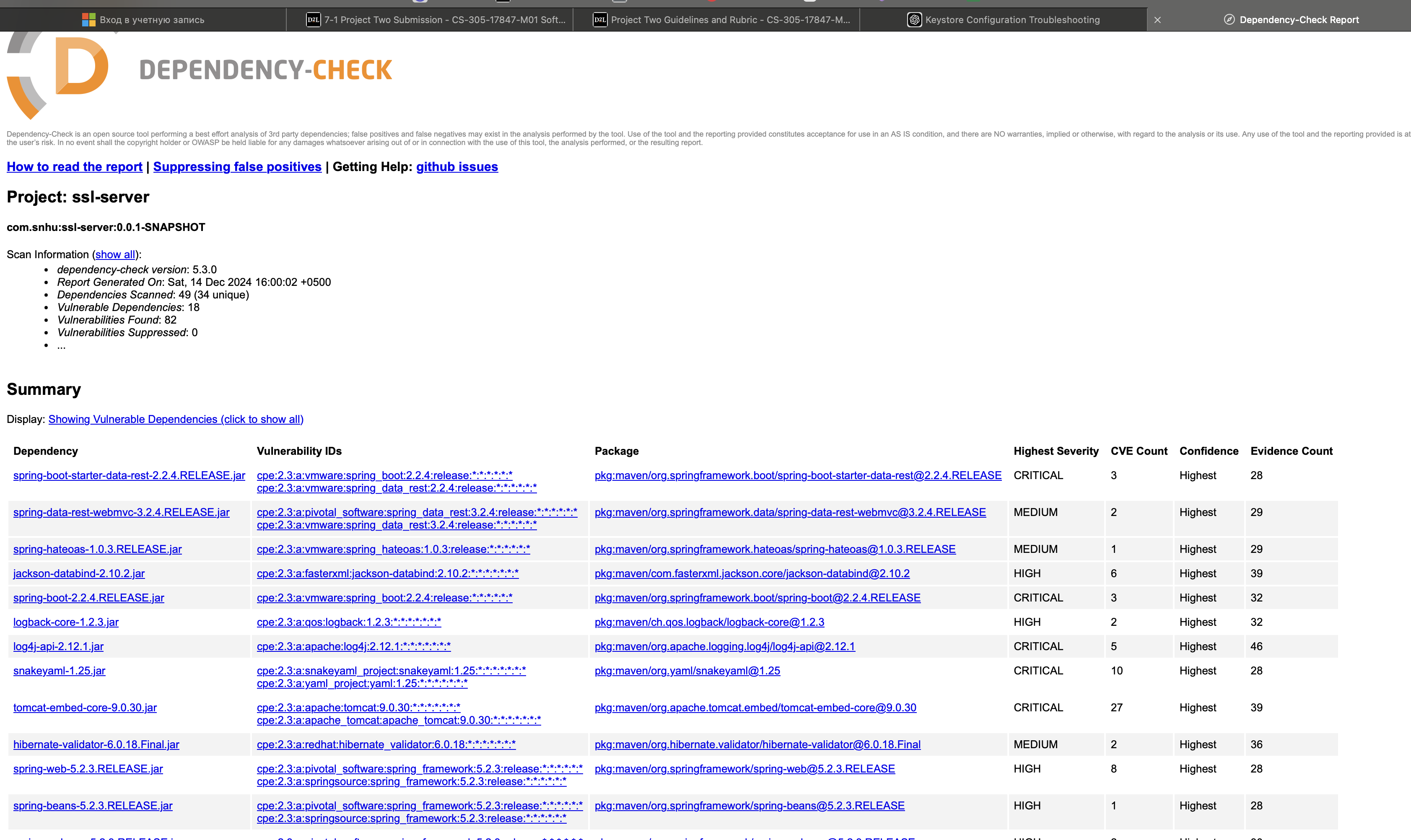
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

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





## Functional Testing

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

Hi professor, I reviewed the code but did not find any issues, what should I insert here ?

Sorry for not finding anything

## Summary

Refactoring to Address Security Vulnerabilities

Several important security weaknesses were addressed when reworking the code for secure communication in order to meet industry standards and best practices. The primary goal was to secure the communication channel by moving from HTTP to HTTPS, which ensures that data is secured during transmission. In addition, we confirmed that the keystore required for SSL settings was properly established and saved. This update protects the integrity and confidentiality of the data being transferred.

The vulnerability assessment process flow diagram highlights areas that were addressed through this refactoring:

* SSL/TLS Encryption: By activating HTTPS with a valid keystore and applying SSL settings, we secured communication between the client and server. This is critical for securing sensitive information from eavesdropping and man-in-the-middle attacks.
* Key Management: Securing cryptographic keys, such as using a strong password for the keystore and key alias setup, reduced the danger of unwanted access to critical data.
* Dependency Vulnerabilities: Following the refactoring, we performed a static code analysis with the dependency-check tool to assess the security of the project's libraries. Several dependencies were found to have serious vulnerabilities, including logging (e.g., Log4j), data binding (e.g., Jackson), and web framework (e.g., Spring). Addressing these dependencies and updating to more secure versions is a crucial step towards lowering the application's attack surface.

Process for Adding Layers of Security

The process of adding layers of security in the application consists of many major steps:

1. SSL/TLS Encryption: We enabled SSL/TLS encryption to verify that data in transit was encrypted. This stops attackers from intercepting or interfering with sensitive data.

2. Strong Authentication and Authorization: Although it is not specifically stated in the code, we would ensure that correct user authentication and authorization are enforced to limit access to sensitive resources.

3. Input Validation: Input validation techniques should be used across the application to protect against common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). These were not explicitly addressed in this restructuring, but they should be part of a regular security practice.

1. **Industry Standard Best Practices**

To mitigate known security vulnerabilities and maintain a secure codebase, we followed the following industry standard best practices:

1. Secure Communication (HTTPS)
2. Secure Key Management
3. Dependency Management
4. Security Headers
5. Secure Configuration

Value of Applying Best Practices

Implementing industry-standard best practices for secure coding has a substantial impact on the company's security and well-being.

* Risk Reduction: By resolving security vulnerabilities early on and consistently maintaining a safe codebase, the chance of security breaches is reduced. This lowers the risk of data leaks, unauthorized access, and other security events.
* Compliance with Regulations: Following security best practices ensures that the organization satisfies industry compliance standards such as GDPR, HIPAA, and PCI-DSS, which need strong data protection and secure communication channels.
* Trust and Reputation: A company that prioritizes security builds trust with its customers, partners, and stakeholders. Protecting sensitive information and implementing security measures improves the company's reputation and customer happiness.
* Cost Savings: Early detection and mitigation of security vulnerabilities avoids costly data breaches and security events. By consistently adhering to best practices, the organization can avoid the enormous financial costs and legal consequences connected with security breaches.

Finally, the refactored code tackles significant security issues, with a focus on HTTPS implementation and safe key management. Static security testing with tools like Maven Dependency-Check helped detect potential problems in third-party dependencies, which are now being addressed. By using industry-standard best practices, we improved the application's security posture, laying the groundwork for secure software development and ensuring the company's overall health.