GLOBALRAIN

Practices for Secure Software Report
Paloma Rodriguez Project Two SNHU CS305

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	6/11/23	Paloma Rodriguez	Project Two

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
Paloma Rodriguez

1. Algorithm Cipher
Recommend an appropriate encryption algorithm cipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be sure to address the following:

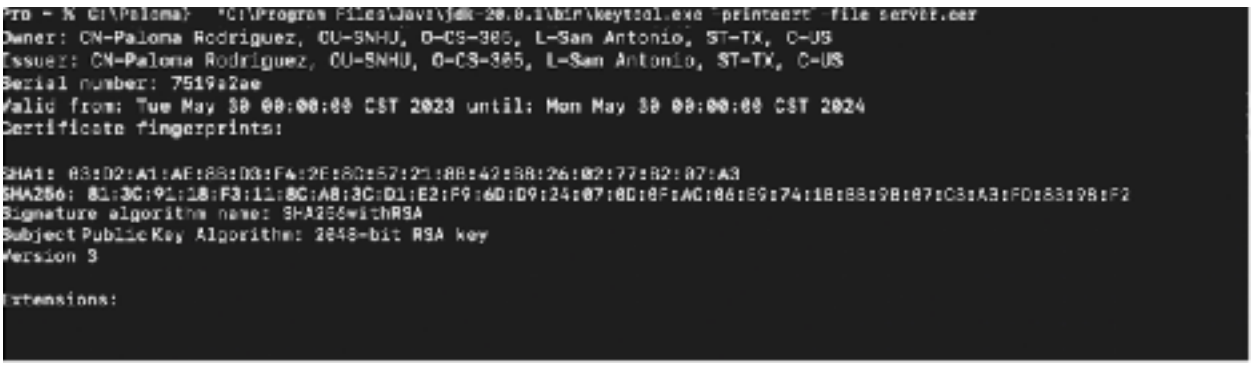
- Provide a brief, high-level overview of the encryption algorithm cipher.
- Discuss the hash functions and bit levels of the cipher.
- Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
- Describe the history and current state of encryption algorithms.

At Artemis Financial, the main goal is to provide financial solutions to clients worldwide. In order to achieve this objective, I highly recommend adopting the SHA-256 encryption algorithm cipher. This particular cipher ensures the utmost security for all information, protecting it from unauthorized access. SHA-256 is widely recognized as one of the most robust and virtually impenetrable encryption algorithms. Breaking this cipher would require an extensive amount of time and computational power through brute force attacks, making it highly secure. When it comes to communicating with financial institutions, SHA-256 is commonly recommended as the preferred cipher. Its hash function and bit levels are designed to incorporate randomness, resulting in a compressed and encrypted data known as the hash value. The length of the encryption depends on the selected bit levels, which determines the number of possible encryption combinations available. By incorporating random numbers in the encryption process, the likelihood of unauthorized access by hackers is significantly reduced. Randomness introduces unpredictability, thus enhancing the overall security of the system. For encryption techniques, symmetric keys are considered one of the simplest methods. They offer the advantage of being faster to implement, as well as requiring only a single key for encryption and decryption. Given these advantages, AES-256 commonly employs symmetric keys to encrypt plaintext using a single key. However, it is important to note that asymmetric keys, which require two keys, are generally considered more secure than symmetric keys. Asymmetric keys find extensive use in internet communication scenarios. The history of encryption algorithms dates back as far as 600 BC, reflecting humanity's enduring commitment to prioritizing security. Over time, the development of encryption techniques has enabled us to protect data from illegal access. As a result, we continue to prioritize and strive for enhanced security measures to safeguard sensitive information.

2. Certificate Generation
Generate appropriate self-signed certificates using the Java Keytool in Eclipse.

- To demonstrate that the certificate was correctly generated:
    - Submit a screenshot of the CER file in your practices for secure software report.
    - Insert a screenshot below of the CER file.

[Insert screenshots here.]



3. Deploy Cipher

Deploy and implement the cryptographic hash algorithm by refactoring code. Demonstrate functionality with a checksum verification.

- Submit a screenshot of the checksum verification in your practices for secure software report. The screenshot must show your name and a unique data string that has been created.
- Insert a screenshot below of the checksum verification.

[Insert screenshots here.]

4. Secure Communications

Verify secure communication. In the application.properties file, refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code. Then once the server is running, type https://localhost:8443/hash in a new browser to demonstrate that the secure communication works successfully.

- Create a screenshot of the web browser that shows a secure webpage and include it in your practices for secure software report.
- Insert a screenshot below of the web browser that shows a secure webpage.

[Insert screenshots here.]

5. Secondary Testing

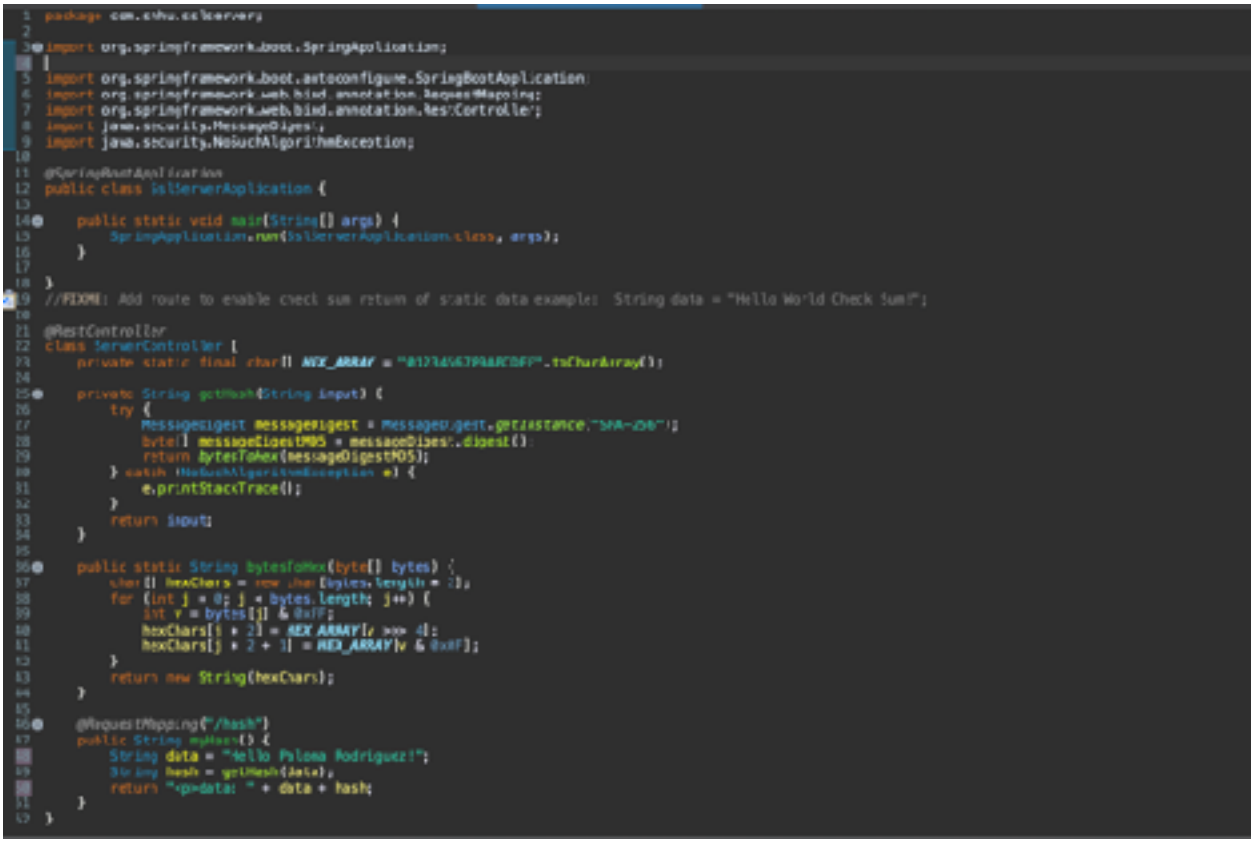
Run a secondary static testing of the refactored code using the OWASP Dependency-Check Maven (see Supporting Materials) to ensure code complies with software security enhancements. You need to focus on only the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities. Include the following in your practices for secure software report:

- A screenshot of the refactored code executed without errors
- A screenshot of the report of the output from the dependency-check static tester
- Insert screenshots below of the refactored code executed without errors and the dependency-check report.

6. Functional Testing

1. Identify the software application's syntactical, logical, and security vulnerabilities by manually reviewing code.
    - Complete this functional testing and include a screenshot of the refactored code, executed without errors, in your practices for secure software report.

[Insert screenshots here.]



7. Summary & Industry Standard Best Practices

Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:

- Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.
- Discuss your process for adding layers of security to the software application.

Explain how you applied industry standard best practices for secure coding to mitigate against known security vulnerabilities. Be sure to address the following:

- Explain how you used industry standard best practices to maintain the software application's current security.
- Explain the value of applying industry standard best practices for secure coding to the company's overall wellbeing.

One of the crucial security enhancements for our application was the implementation of self-signed certificates, which enabled the utilization of HTTPS. This step was paramount in ensuring the security of our website, establishing trust with users, and safeguarding our business. To ensure the proper functioning of HTTPS once our application was up and running, my initial focus was on creating and configuring the certificates correctly. This ensured that our website remained secure, providing users with confidence that they are interacting with our legitimate platform rather than a fraudulent one. By prioritizing this security aspect, we enhance the overall well-being of our business.

Next, we thoroughly examined our hashing function and validated its integrity through checksum verification. This step provided us with peace of mind, as it confirmed that our consumers' data is appropriately encrypted and extremely challenging to decrypt. By upholding this level of security, we further fortify the strength of our business. The final stage of our security measures involved addressing and fixing any vulnerabilities identified during the dependency check. By patching these vulnerabilities, we ensure that our business is well-prepared and that all internal components of our application are up-to-date and functioning as intended. This commitment to security instills confidence in our stakeholders and safeguards our business operations. In addition to these measures, it is essential to follow best practices for maintaining application security. Keeping our software and systems updated is crucial, as it protects against potential attacks targeting outdated components. Furthermore, enforcing the principle of least privilege is vital to mitigate internal threats by granting users only the necessary access, rather than providing universal access to all resources. By implementing these practices, we uphold the security standards of our application, thereby protecting our organization.