****

# 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** | **March 25, 2023** | **Maurice Wesley** | **Created Rest Control** |
| **1.1** | **April 13, 2023** | **Maurice Wesley** | **Recommend Cipher** |
| **1.2** | **April 14, 2023** | **Maurice Wesley** | **Functional Testing** |
| **1.3** | **April 15, 2023** | **Maurice Wesley** | **Summary** |

## Client



## Developer

Maurice Wesley

## Algorithm Cipher

## Recommend an appropriate encryption algorithm cipher:

SHA-256: Is a secure hash algorithm that produces a 256-bit final hash value. The algorithm satisfies three constraints.

* 1. **Collision resistance**: It is not feasible to find two distinct bit strings such that H(x) = H(x’).
  2. **Preimage resistance**: Given the hash value t in the range of H, it is not feasible to find a string x such that hash(x) = t.
  3. **Second preimage resistance**: Given a string x, it is not feasible to find another string x’, such that hash(x) = hash(x’).

**Provide a brief, high-level overview of the encryption algorithm cipher.**

I am guaranteed to get a 256-bit AES key by taking the SHA-256 hash of the greeting provided in the code. I am also able to encrypt the hash string using AES encryption. This adds another layer of protection. However, I am not completely sure that I am implementing combination correctly.

SHA-256 is part of the SHA family. It requires the plaintext to be less than 264 bits. This maximizes the randomness. The final hash value is irreversible and one should not get plaintext if you have the digest nor can the digest provide the original value.

**Discuss the hash functions and bit levels of the cipher.**

The hashing algorithm implements plaintext padding. Using a 1 followed by zeros, it adds extra bits until the length of the string is precisely 64 bits less than a multiple of 512. Afterwards, some date is added to make the total length a multiple of 512.

Next the algorithm initializes default values for eight different buffers that are used in the rounds and store 64 different keys in an array. The complete message is decomposed into multiple 512-bit blocks. Each block is put through 64 rounds of computation. The preceding output serves as input for the next block. The process is repeat until the last 512-bit block is reached. As the name implies, the final hash value will be 256 bits (Jena, 2023).

**Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.**

**Random Numbers**: The developer may have code that requires the use of random numbers. Unfortunately, the base random generators in that are high quality but limited. When the available random numbers have been used, then the generators have to wait until more hardware-related events occur. The developers code may be blocked while system waits for more events (Manico & Detlefsen, 2014).

The cryptographic pseudorandom number generator algorithm mitigates the blocking issue by producing numbers that appear to be random. Special effort is needed to periodically remove seeds from the GPRNG or the algorithm will produce the same random numbers. Implementation of the former will supply a reliable pool of pseudorandom data (Manico & Detlefsen, 2014).

**Symmetric key cipher**: The kind of cryptography that is used for encryption. Uses the same key to encrypt and decrypt data. This requires the sender and the receiver to have the same secret key. Subsequently, the key must remain secret for the overall system to be secure. Symmetric keys are better for Artemis Financial since it will use the cipher to secure its archived data or internal transfers.

**Asymmetric key cipher**: Better for external transfers. The cipher uses two different keys (public and private). One is kept secret (private), only known by the sender, and used to sign the data. The other one is open (public), known by anyone, and used to validate the signature. Currently, modern computing power cannot derive the private key using the public key.

**Describe the history and current state of encryption algorithms.**

Before the 70s, the rules and formulas needed to securely encrypt data were closely held military secrets. Commercial entities did not have formulaic way to properly ensure their data. In 1972, the NSB started a project to develop an algorithm would be released to the public. However, the secret key would remain private. The purpose of the project was to create an algorithm to encrypt commercial and unclassified government data. Eventually in 1977, the Data Encryption Standard became the Federal Information Processing Standard.

DES is symmetric algorithm that encrypts blocks, 64 bits of plaintext, using a 56-bit key that produces a one-to-one relationship between plaintext and cyphertext. Cryptanalyst predicted the DES lifespan to be 15 years. In retrospective, the standard outperformed expectations for five more years. Unfortunately, in 1977 a DES encrypted message was cracked in 5 months by using key exhaustion and a network of computers.

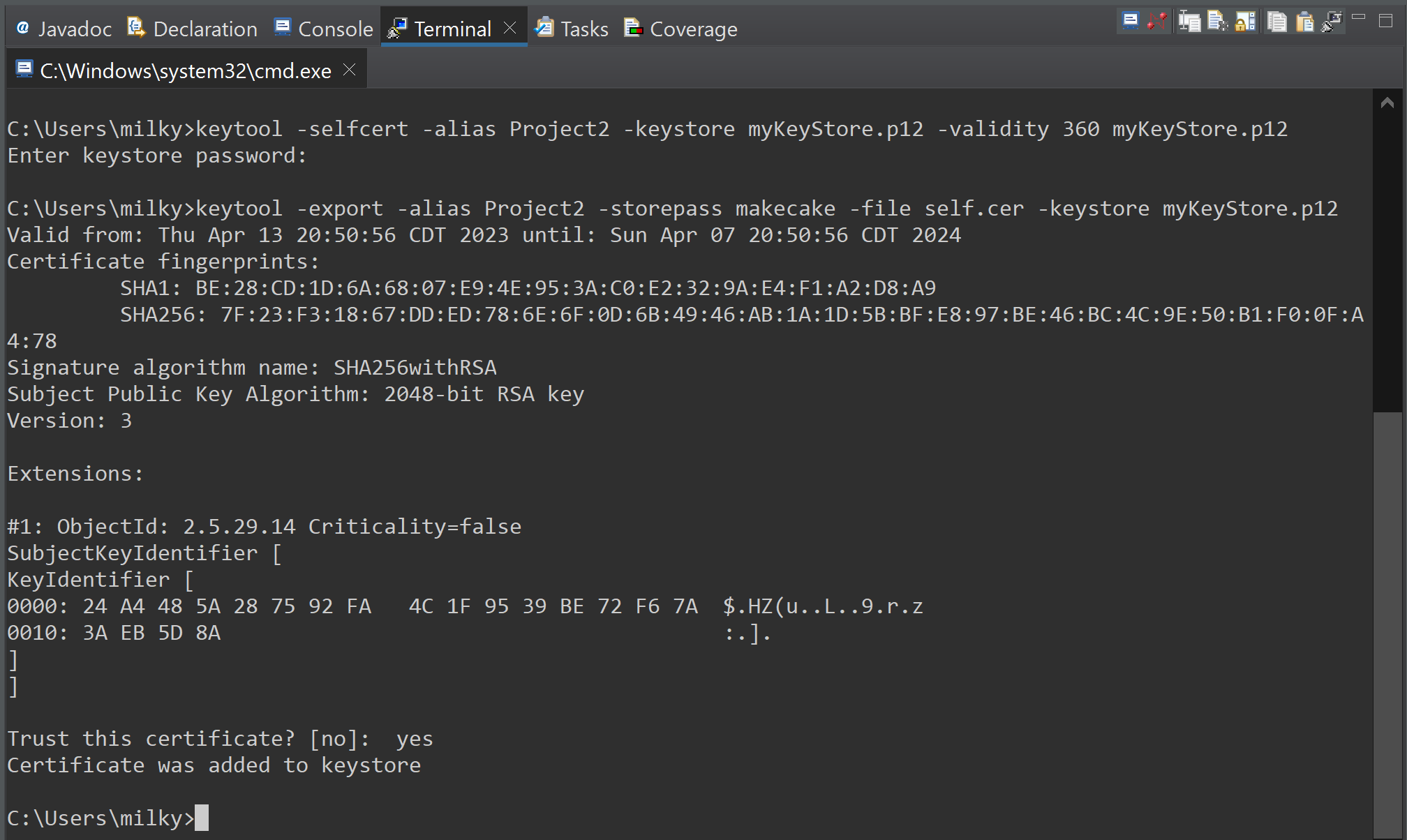
The NIST started researching the development of AES in 1997 when Data Encryption Standard became vulnerable to brute-force-attacks. AES is an open standard that is available to the public, private, commercial, and non-commercial applications. Currently, modern computing processing power is not advanced enough to crack AES 128, 192, and 256.

The categories of encryption ciphers were discussed earlier. As previously mentioned, solely relying on a cipher for data security is an inadequate risk management approach. Microsoft had a mobile application that leaked two of their private keys. Irrespective of the key method used (private and/or public), attackers can compromise the cipher if the key is leaked, obtained through social engineering, or created without security best practices.

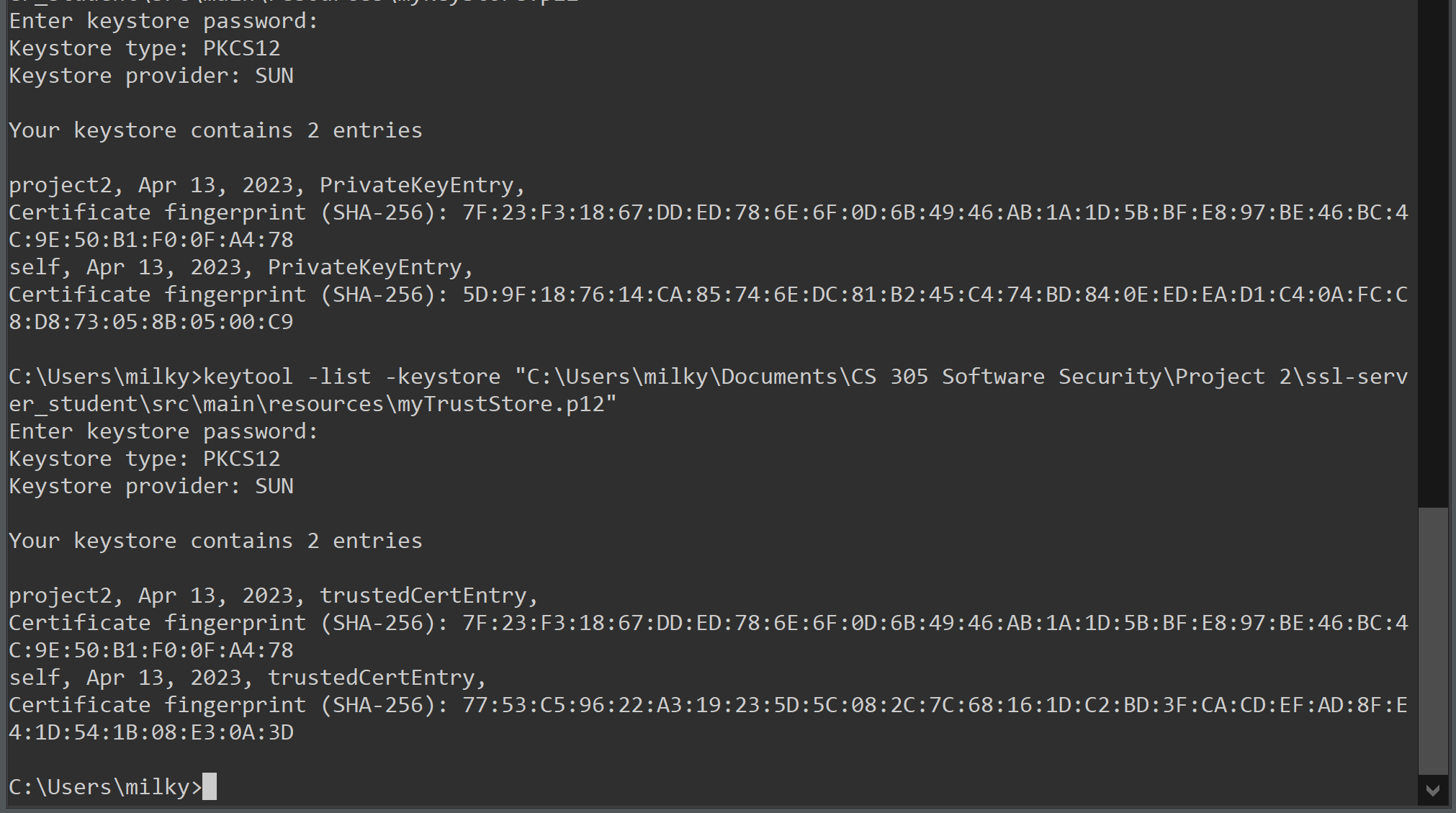
**Note**: The feedback on project 1 mentioned that I should expand on governmental regulations. However, governmental regulations were not mentioned in the prompt.

## Certificate Generation

**Export**



**Print Certificate**

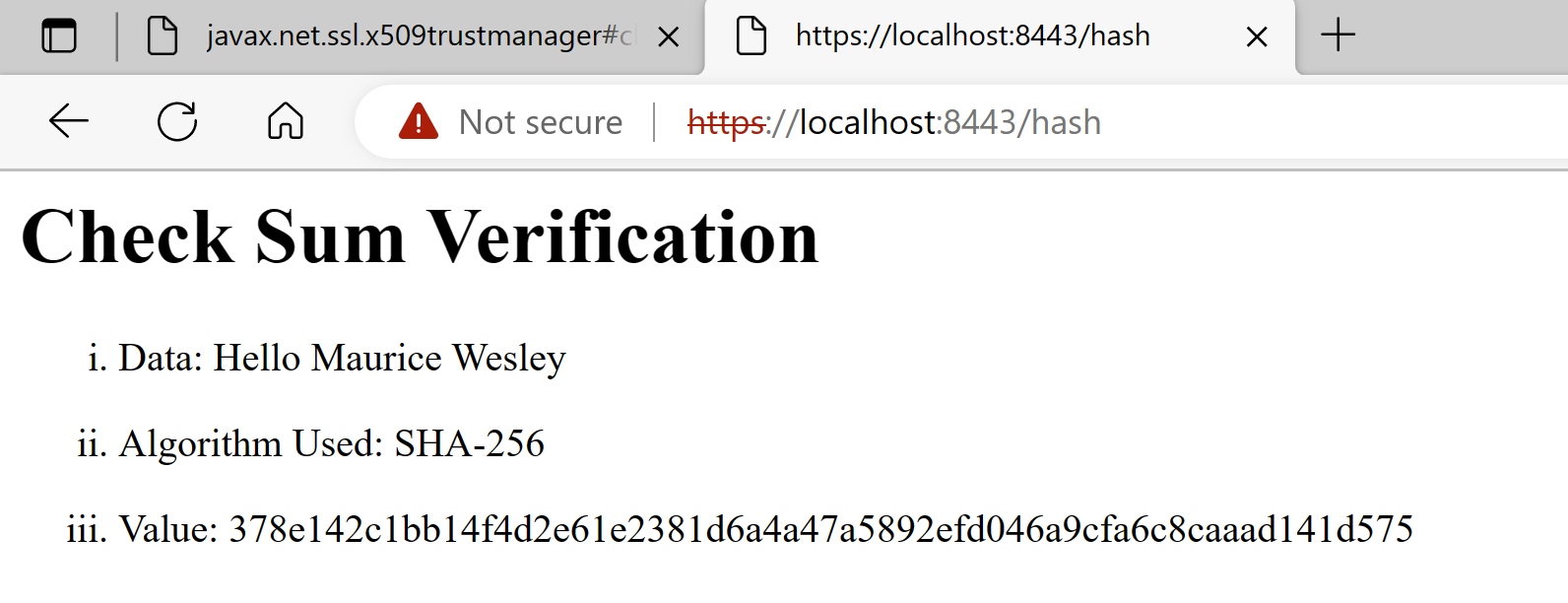


## Deploy Cipher

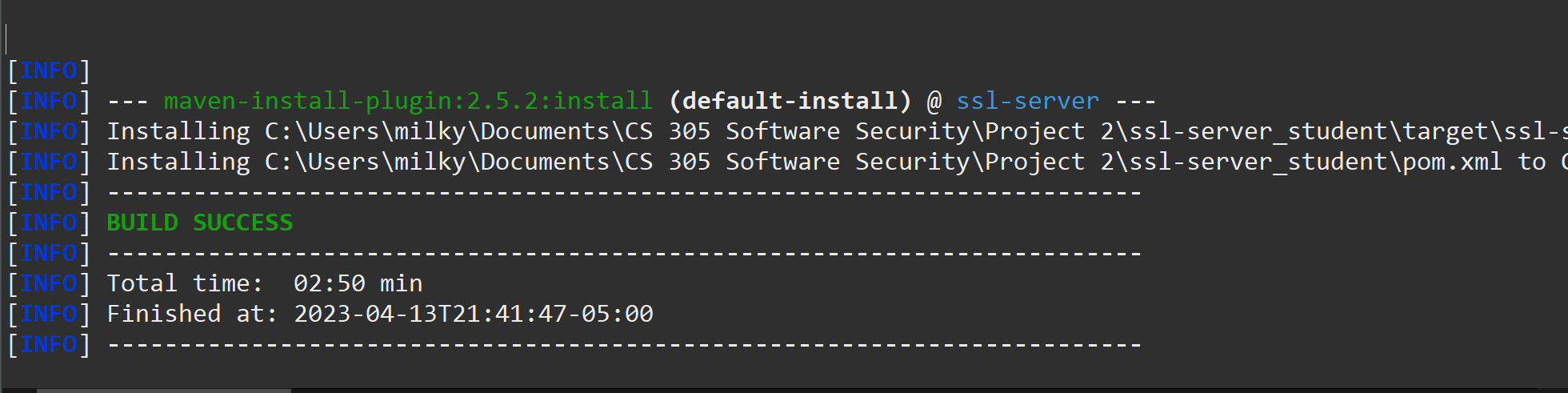


## Secure Communications

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



## Secondary Testing



**Dependency Report**



Before [Dependency-Check Report](file:///C:\Users\milky\Documents\CS%20305%20Software%20Security\Project%202\ssl-server_student\before-change-dependency.html)

## Functional Testing

**SSL Server Application Class**

**SQL Injection:** @RequestMapping

The Spring model view controller automatically bind request parameters to beans declared as arguments of methods if they are annotated with @RequestMapping. Subsequently, it is possible to insert malicious input in the arguments of the @RequestMapping annotated methods. Conversely, @Entity or @Document objects are linked to the underlying database and updated automatically by the framework that Artemis Financials implements (Spring MongoDB).

The above conditions can lead to a malicious attack: if the persistent objects are used as an argument in a method annotated with @RequestMapping, it is possible to construct user input injections, to change the content in the database fields. “The use of @Entity or @Document objects as arguments in methods annotated with @RequestMapping should be avoided. In addition to @RequestMapping, this rule also considers the annotations introduced in Spring Framework 4.3: @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping” (Sonarsource, 2023).

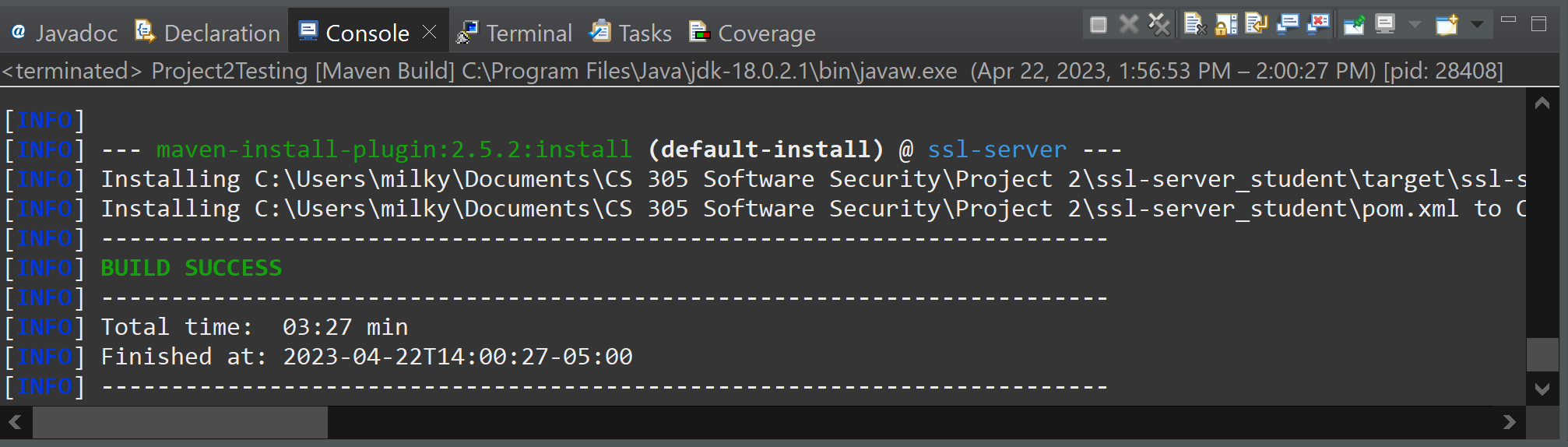
To mitigate the @RequestMapping vulnerabilities, it is recommended declare a string variable and initialize it to parameterized request containing a request.getParameter() statement with “user” or “pass” passed as arguments. The string variables are passed into a query statement. The statement would not be vulnerable to SQL injection.

**SSL Server Application Tests Class**

The current application implements a J Unit test class. However, the tests are incomplete. The at spring boot test annotations needed to be commented out to prevent compilation errors. The annotation would cause the class to try an execute the test annotation and subsequent test method that is empty.

The prompt mentioned that I should concentrate on the code that I added. The code does not have any syntactical or decision logic errors and I was able to successfully compile the code.

I am unsure if I am supposed to repost the picture from above showing that the refactored code executes without errors. I was able to update the maven denpendancies. The code base has the dependency check before the update and the dependency check after the update. Below is the output of the successful build.



After: [Dependency-Check Report](file:///C:\Users\milky\Documents\CS%20305%20Software%20Security\Project%202\ssl-server_student\after-change-dependency.html)

## Summary

**Highlight the areas of security that you addressed by refactoring the code.**

I was recently promoted to Global Rain’s new agile scrum team. I have been tasked with examining Artemis Financials’ web-based software application to identify any security vulnerabilities. As the agile security officer for the financial company, I need to ensure that the organization’s use of secure communications channels to domestic and international customers meets government export regulations.

**Cryptography**: According to the resource material, I am inclined to investigate cryptography vulnerabilities because it is an export item. Artemis Financial conducts international transactions. The Gramm-Leach-Bliley Act was an act passed by congress that requires companies, like Artemis Financial, that provides financial, investment, and insurance to explain the company’s information-sharing practices to their customers and to safeguard sensitive data (FTC, n.d.). Similarly, the Act establishes strict data access policies and regulates the way financial institutions handles customers’ private data.

**Discuss your process for adding layers of security to the software application.**

The main method in the Server Application Class invokes the Spring Application. Also, the class has two methods with Request Mapping and Rest Controller annotations. First, the layered approach that I took involves calling the method after the Request Mapping annotation.

The method declares a string variable and assigns it to a string literal. Another string variable is declared and assigned to the return value of a method call passed with an argument.

Next, the method called creates a SHA-256 message digest instance. The message digest instance is updated after transposing the argument into a byte array. Then, the byte array is transformed into a hash byte array using the digest method. The hash byte array is passed to a method that converts the hash array into a unique hexadecimal string. The string is returned.

A server-side key, keystore, password, alias, and certificate were created. The certificate was self-signed and imported into the keystore. Additionally, a client-side trust store and password were created. The certificate was imported into the trust store.

The applications properties file was changed to execute on a secure port (8443). A secure socket layer was enabled to true. Also, the file contained the key alias, key/keystore password, key type, and trust store password. Class paths to the key store and trust were added. The data was encoded into a hash and transformed into a hexadecimal string. I was able to output data using html that was displayed on a HTTPS site.

## Industry Standard Best Practices

A one size fits all security measure does not exist. The security officer/developer must implement security measures that address a specific threat. However, there are standard practices that can be implemented. The team can start with using a cryptographic protocol (secure socket layer, transport layer security). It is best not to rely on the web server or client front end. Test the SSL/TLS with expired certificates and certificates with incorrect hostnames (Trust Manager).

The General rules for selecting an appropriate cryptographic suite are:

* Do not use suites that list ANON, because they do not provide authentication
* Do not use suites that contain NULL and avoid suites that contain EXPORT
* Avoid RC4, DES, and 3DES
* Use ECDHE and DHE for key agreement. Provides protection when private keys get compromised
* Use key sizes that are 128 bits are larger
* Choose suites that support forward secrecy.

A developer has to be aware of an impersonation attack. Two key techniques to implement are verifying the hostname and the certificate. The default configuration of JSSE only verifies the certificate (Manico & Detlefsen, 2014).

**Avoiding Collisions**

**Authentication**: In the legal realm, an individual may sign a document using a public key. If an attacker was able to produce to documents with the same hash, then the attacker to impersonate the individual and have them claim a false document.

**Man in the Middle**: In a distributed environment, companies share file, computers receive updates, and/or web host providers may initiate platform maintenance/upgrades. Senders and receivers analyze cryptographic hashes of files to ensure the data integrity. If an attacker is able to produce the different document with the same hash, then it could compromise the data integrity which may lead to system penetration.

**Source**

Jena, Baivab Kumar. (2023). A definitve Guide to Learn The SHA-256 (Secure Hash Algorithms). Simplilearn. [Blog Post]. [What Is SHA-256 Algorithm: How it Works and Applications [2022 Edition] | Simplilearn](https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm)

Dang, Quynh. (2012). Recommendation for Applications Using Approved Hash Algorithms. NIST. [White Papers, p. 6]. [NIST Special Publication 800-107 Revision 1, Recommendation for Applications Using Approved Hash Algorithms](https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf)

Manico, Detlefsen. (2014). Iron-Clad Java: Building Secure Web Applications. Oracle. [White Papers]. [Iron-Clad Java (oreilly.com)](https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso_link=yes&sso_link_from=SNHU)

*Data Encryption Standard*, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977). [/nist3:/usr/kjw/SP958-Lide/Pvi191](https://nvlpubs.nist.gov/nistpubs/sp958-lide/250-253.pdf)

Electronic Frontier Foundation. (1998). *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*, O’Reilly & Associates, Inc., Sebastopol, CA. [/nist3:/usr/kjw/SP958-Lide/Pvi191](https://nvlpubs.nist.gov/nistpubs/sp958-lide/250-253.pdf)

SonarSource. (2023). Java static code analysis. Sonar Rules. [White Papers]. [Java static code analysis | spring: Persistent entities should not be used as arguments of "@RequestMapping" methods (sonarsource.com)](https://rules.sonarsource.com/java/tag/spring/RSPEC-4684)