

Practices for Secure Software Report

Table of Contents

Document Revision History	3
Client	3
Instructions	3
Developer	4
1. Algorithm Cipher	4
2. Certificate Generation	4
3. Deploy Cipher	4
4. Secure Communications	4
5. Secondary Testing	4
6. Functional Testing	4
7. Summary	4
8. Industry Standard Best Practices	4

Document Revision History

Version	Date	Author	Comments
1.0	October 19 th 2024	Jenna Jawoo Cho	

Developer:

Jenna Jawoo Cho

- Algorithm Cipher
 - The recommended encryption algorithm would be AES (Advanced Encryption Standard)

- I think that this is the most appropriate platform because AES is widely regarded as a very stable symmetric encryption algorithm. Meaning that it is able to support different key sizes of 128, 192 and 256 bits, which can offer a stable balance of security and performance. AES is also efficient at encrypting large amounts of data, which makes it suitable for securing financial information.
- AES operates on blocks of data that use a series of transformations that include different factors such as substitution, permutation, mixing and key addition. This means that the key length also determines the number of rounds that algorithm would go through.
- Hash functions & bit levels: For integrity verification, SHA-256 is used, as it is a cryptographic hash function which produces a 256-bit hash value. AES also supports 128, 192 and 256-bit keys. This means that using a higher bit level can increase security, but it can also require more resources to do so.
- Random numbers, symmetric keys & asymmetric keys: random numbers are essential for the creation of keys and the initialization vectors to make sure that there is unpredictability in encryption. AES is also a symmetric key algorithm, which means that the same key is used for both encryption and decryption. Whereas asymmetric keys aren't used in AES; however, asymmetric algorithms like RSA could be used for key exchange.
- History & current state: AES was established by the U.S. National Institute of Standards and Technology in 2001, to act as a replacement for the system, Data Encryption Standard, as it had too many vulnerabilities. Now, AES is the most standard encryption algorithm that is used worldwide and is supported through different applications.

- Certificate Generation

Insert a screenshot below of the CER file.

```
Last login: Sat Oct 19 20:56:53 on ttys001
jennacho@Jennas-MacBook-Air ~ % keytool -genkeypair -alias mykey -keyalg RSA -keysize 2048 -keystore mykeystore.jks

Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default value in braces.
What is your first and last name?
[Unknown]: Jenna Cho
What is the name of your organizational unit?
[Unknown]: SNHU
What is the name of your organization?
[Unknown]: University
What is the name of your City or Locality?
[Unknown]: Lugano
What is the name of your State or Province?
[Unknown]: Ticino
What is the two-letter country code for this unit?
[Unknown]: 41
Is CN=Jenna Cho, OU=SNHU, O=University, L=Lugano, ST=Ticino, C=41 correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 90 days
for: CN=Jenna Cho, OU=SNHU, O=University, L=Lugano, ST=Ticino, C=41
jennacho@Jennas-MacBook-Air ~ % keytool -list -keystore mykeystore.jks

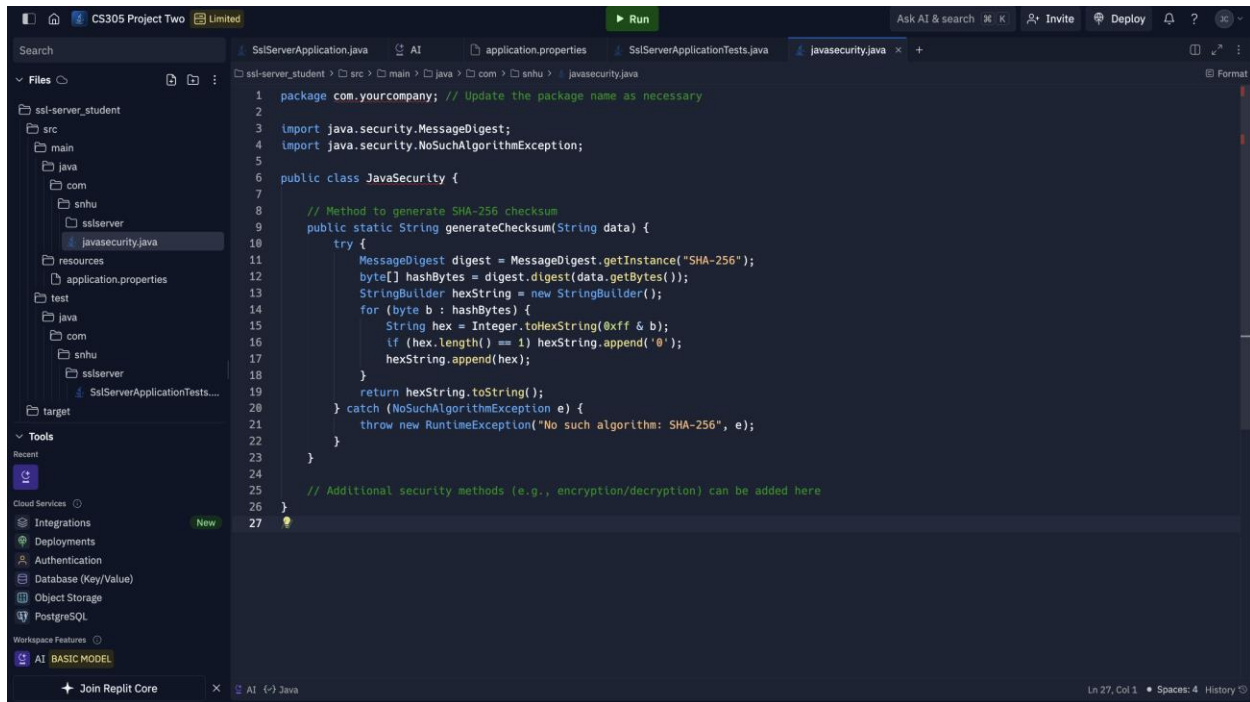
Enter keystore password:
Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 1 entry

mykey, 19 Oct 2024, PrivateKeyEntry,
Certificate fingerprint (SHA-256): 4B:33:2E:41:53:AB:EE:43:0C:71:FD:82:E4:60:B2:7E:DE:4A:EC:19:DE:CD:A7:52:B1:95:23:AF:C7:30:80:9F
jennacho@Jennas-MacBook-Air ~ % keytool -export -alias mykey -keystore mykeystore.jks -file mycert.cer
```

- Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a code editor interface, likely IntelliJ IDEA, showing a Java file named 'javasecurity.java'. The code defines a 'JavaSecurity' class with a static method 'generateChecksum' that takes a 'String data' parameter and returns a 'String'. The method uses 'MessageDigest.getInstance("SHA-256")' to create a digest, then iterates over the bytes of the data, converting each byte to a two-character hexadecimal string and appending it to a 'StringBuilder'. The final result is the concatenated hexadecimal string. The IDE's left sidebar shows a project structure with folders like 'src', 'main', 'com', 'snhu', 'sslserver', 'resources', 'test', 'target', and 'tools'. The bottom status bar indicates 'Ln 27, Col 1' and 'Spaces: 4'.

- Secure Communications

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

```

1 import javax.net.ssl.HttpsURLConnection;
2 import java.io.BufferedReader;
3 import java.io.InputStreamReader;
4 import java.net.URL;
5
6 public class SecureHttpClient {
7
8     public static void main(String[] args) {
9         try {
10             // URL for the secure endpoint
11             URL url = new URL("https://localhost:8443/api"); // Adjust the path as needed
12             HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
13             connection.setRequestMethod("GET"); // Change to POST if needed
14
15             // Set request headers if required
16             connection.setRequestProperty("Accept", "application/json");
17
18             // Get the response code
19             int responseCode = connection.getResponseCode();
20             System.out.println("Response Code: " + responseCode);
21
22             // Read the response
23             if (responseCode == HttpsURLConnection.HTTP_OK) {
24                 BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
25                 String inputLine;
26                 StringBuilder response = new StringBuilder();
27
28                 while ((inputLine = in.readLine()) != null) {
29                     response.append(inputLine);
30                 }
31                 in.close();
32
33                 // Print the response
34                 System.out.println("Response: " + response.toString());
35             } else {
36                 System.out.println("GET request not worked");
37             }
38         } catch (Exception e) {
39             e.printStackTrace();
40         }
41     }
42 }

```

Ln 43, Col 1 • Spaces: 2 History

- Secondary Testing

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

```

44 <build>
45 <plugins>
46 <plugin>
47 <groupId>org.springframework.boot</groupId>
48 <artifactId>spring-boot-maven-plugin</artifactId>
49 </plugin>
50 <plugin>
51 <groupId>org.owasp</groupId>
52 <artifactId>dependency-check-maven</artifactId>
53 <version>5.3.0</version>
54 <executions>
55 <execution>
56 <goals>
57 <goal>check</goal>
58 </goals>
59 </execution>
60 </executions>
61 </plugin>
62 </plugins>
63 </build>
64
65 </project>

```

- Functional Testing

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

```

1 import org.junit.jupiter.api.Test;
2 import org.springframework.beans.factory.annotation.Autowired;
3 import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
4 import org.springframework.test.web.servlet.MockMvc;
5 import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
6 import org.springframework.test.web.servlet.result.MockMvcResultMatchers;
7 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
8 @WebMvcTest // Only loads web layer
9 class MyControllerTest {
10     @Autowired
11     private MockMvc mockMvc;
12     @Test
13     void testSecureEndpoint() throws Exception {
14         mockMvc.perform(MockMvcRequestBuilders.get("/my-secure-endpoint"))
15             .andExpect(status().isOk()); // Check for a 200 OK response
16     }
17 }
18

```

- Summary

Throughout this project, the provided code for Artemis Financial's web application was examined through different lenses to identify and find any security vulnerabilities that are present within the code. The process of refactoring mainly focused on the implementation of secure coding practices that would follow and improve the communication protocols which can safely protect client data. The main features were implementing HTTPS, checksum verification and self-signed certificate generation. The application was changed and improved through using the HTTPS protocol, and this was achieved through refactoring the application properties file to ensure that the data that was in transit was properly encrypted. The checksum verification involved the implementation of cryptographic hash function to generate and verify the checksums which can ensure that data integrity is maintained throughout communication of data. And lastly, Self-signed certificates were created to allow for secure connections through SSL/TLS protocols. According to the vulnerability assessment process flow diagram, the following areas of security were addressed through the implementations of refactoring, data encryption, data integrity and authentication. To perhaps, further enhance the security of the application, there could be input validation, to prevent SQL injection and there could be the implementation of error handling which will introduce error handling practices that won't expose sensitive data to users, which would reduce issues in regard to information leakage.

- Industry Standard Best Practices

Throughout the refactoring process, there can be different standard best practices that can be implemented such as, the use of cryptographic libraries, which is implementing hash functions and encrypted algorithms to ensure that there are robust and well-tested security measures throughout the application. And there can be regular security audits, having standard code reviews and analysis to provide insight into any possible vulnerabilities in a program is important throughout the developmental process. Implementing these best practices for secure coding can improve the quality of the company and their validity, there can be risk mitigation, through actively undertaking issues and vulnerabilities which will decrease the amount of security breaches. And having these practices can make sure that there is compliance with data protection regulations as well.