



G L O B A L R A I N

Practices for Secure Software Report

Table of Contents

DOCUMENT REVISION HISTORY	3
CLIENT.....	3
INSTRUCTIONS	ERROR! BOOKMARK NOT DEFINED.
DEVELOPER	3
1. ALGORITHM CIPHER	3
2. CERTIFICATE GENERATION	4
3. DEPLOY CIPHER	4
4. SECURE COMMUNICATIONS	5
5. SECONDARY TESTING	5
6. FUNCTIONAL TESTING	6
7. SUMMARY	6
8. INDUSTRY STANDARD BEST PRACTICES	7

Document Revision History

Version	Date	Author	Comments
1.0	4/16/23	Secally Barbosa	

Client



Developer

Secally Barbosa

1. Algorithm Cipher

Artemis Financial is looking for more security in their application to make sure communications are secure for all users. A hacker is more likely to attack a financial institution for their own gain with money because they would be able to access personal banking account information from each Artemis user, thus access to their funds. The best way to prevent this would be to incorporate encryption. This would mean that it would be difficult for the hacker to get into said information without a key/access code. Asymmetric Communication would be the best way to succeed in terms of Artemis wanting secure communications. This would essentially mean that

the key to encryption would be public, but decryption would be private. External information transferring and the highest level of security would be achievable with SHA-256. This would mean that there would be 256-bit keys to encrypt. This also means that the key combination would be 256 bits in length with many possible combinations. This also uses a generator with Java to create a random key, making it far less vulnerable. A checksum also verifies the trustworthiness of the file itself. The cipher, SHA-256, will use a hash function to create the checksum which will then provide a message stating its validity.

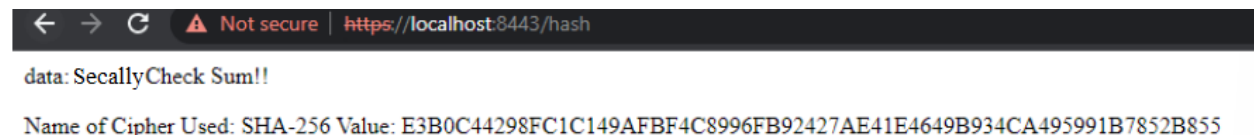
2. Certificate Generation

```
Certificate fingerprints:
  SHA1: 85:F7:8C:33:31:FC:90:C4:F4:8E:02:60:91:F0:F6:3E:C4:68:F8:45
  SHA256: 00:02:23:BD:20:D3:EB:A4:C9:56:7E:07:D4:01:8D:43:4F:6D:F6:90:1E:6A:EA:68:A8:E5:F6:58:9C:7B:A3:F1
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: A8 20 FD 6C 9B 6A D2 4C   95 29 68 ED 76 F5 CC C9   . .l.j.L.)h.v...
0010: 4C 2C 95 0E               L,..
]
]
```

3. Deploy Cipher



data: SecallyCheck Sum!!

Name of Cipher Used: SHA-256 Value: E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855

4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

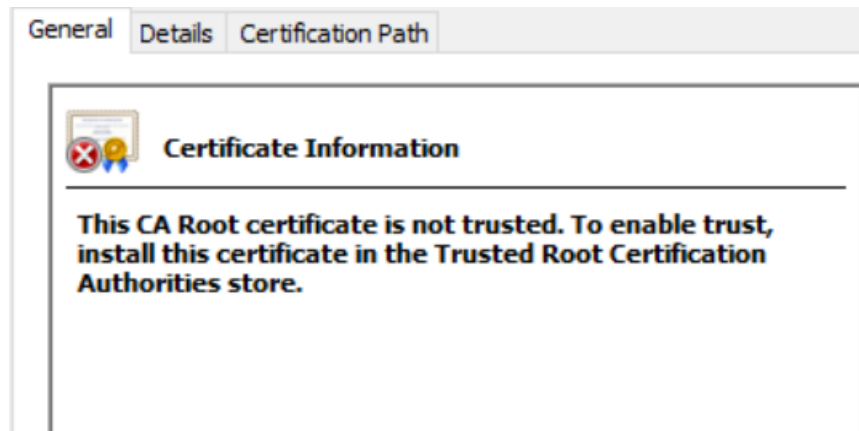
- Showing the HTTPS is working but that my Cert isn't official because it's self signed.

← → ↻ ⚠ Not secure | https://localhost:8443/hash

data:SecallyCheck Sum!!

Name of Cipher Used: SHA-256 Value: E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855

4. Secure Communications



5. Secondary Testing

```
@RestController
class ServerController {
    private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();

    private String getHash(String input) {
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
            byte[] messageDigestMD5 = messageDigest.digest();
            return bytesToHex(messageDigestMD5);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return input;
    }

    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }
}
```

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severit
jackson-databind-2.10.2.jar	cpe 2.3 a: fasterxml:jackson-databind:2.10.2 *****	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH
log4j-api-2.12.1.jar	cpe 2.3 a: apache:log4j:2.12.1 *****	pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1	LOW
snakeyaml-1.25.jar	cpe 2.3 a: snakeyaml:project:snakeyaml:1.25 *****	pkg:maven/org.yaml/snakeyaml@1.25	HIGH
tomcat-embed-core-9.0.30.jar	cpe 2.3 a: apache:tomcat:9.0.30 ***** cpe 2.3 a: apache:software_foundation:tomcat:9.0.30 ***** cpe 2.3 a: apache:tomcat:apache_tomcat:9.0.30 *****	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30	CRITICAL
hibernate-validator-6.0.18.Final.jar	cpe 2.3 a: redhat:hibernate_validator:6.0.18 *****	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM
json-smart-2.3.jar	cpe 2.3 a: json_smart:project:json_smart:2.3 *****	pkg:maven/net.minidev/json-smart@2.3	CRITICAL
spring-core-5.2.3.RELEASE.jar	cpe 2.3 a: pivotal_software:spring_framework:5.2.3.release ***** cpe 2.3 a: springsource:spring_framework:5.2.3.release ***** cpe 2.3 a: vmware:spring_framework:5.2.3.release ***** cpe 2.3 a: vmware:springsource_spring_framework:5.2.3.release *****	pkg:maven/org.springframework/spring-core@5.2.3.RELEASE	HIGH
spring-jcl-5.2.3.RELEASE.jar	cpe 2.3 a: pivotal_software:spring_framework:5.2.3.release ***** cpe 2.3 a: springsource:spring_framework:5.2.3.release ***** cpe 2.3 a: vmware:springsource_spring_framework:5.2.3.release *****	pkg:maven/org.springframework/spring-jcl@5.2.3.RELEASE	MEDIUM

6. Functional Testing

```

1 package com.snhu.sslserver;
2
3 @import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RestController;
7 import java.security.MessageDigest;
8 import java.security.NoSuchAlgorithmException;
9
10 @SpringBootApplication
11 public class SslServerApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(SslServerApplication.class, args);
15     }
16 }
17
18 //FIXME: Add route to enable check sum return of static data example: String data = "Hello World Check Sum!";
19
20 @RestController
21 class ServerController {
22     private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
23
24     private String getHash(String input) {
25         try {
26             MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
27             byte[] messageDigestMD5 = messageDigest.digest();
28             return bytesToHex(messageDigestMD5);
29         } catch (NoSuchAlgorithmException e) {
30             e.printStackTrace();
31         }
32         return input;
33     }
34 }
35
36 public static String bytesToHex(byte[] bytes) {
37     char[] hexChars = new char[bytes.length * 2];
38     for (int j = 0; j < bytes.length; j++) {
39         int v = bytes[j] & 0xFF;
40         hexChars[j * 2] = HEX_ARRAY[v >> 4];
41         hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
42     }
43     return new String(hexChars);
44 }

```

7. Summary

Adding a RestController incorporated a hash RESTful stop. Also, the ServerController class was chosen by looking at the vulnerability assessment diagram provided. The SHA-256 cipher is always very secure and has a very small chance of getting attacked because it isn't vulnerable. I felt that cipher would be the best choice in terms of what the client was looking for.

8. Industry Standard Best Practices

Maintaining the level of security means that there must be updates in a timely manner. Doing said updates will avoid new vulnerabilities and will help keep the data of each use protected. We could ensure the highest amount of security by making sure that the plugins running within the pom.xml are always at the latest iteration. Running dependencies would be helpful as well because this would also help find new vulnerabilities before it is too late. I also feel that always using a different set of eyes on the code could help find issues that maybe someone didn't see prior to running. This will always ensure that we find any issues within our code.