GLOBALRAIN

CS 305 Project Two
Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	10/16/2022	Kennedy Uzoho	Dev and Security

Client

Developer
Kennedy Uzoho

1. Algorithm Cipher
Encryption Algorithm Recommendation: AES-256
Advanced Encryption Standards (AES) is one of the Secure Encryption Algorithms available. AES-256 is a very patented cryptographic encryption function that can encrypt a digital value or data that is 256 bits long. The most important feature of AES-256 is the resilience it has in protecting data from unintended access or hackers. AES-256 has been used in some of the most popular authentication and encryption protocols, including bank transaction encryptions, classified data, online shopping data, and social media app data. AES-256 is a symmetric cryptographic function that allows the use of the same key to decrypt encrypted data, unlike RSA which is asymmetric. RSA is asymmetric it uses two different but linked keys (public and private keys) to encrypt.

Hashing Data Algorithm Recommendation: SHA-1 OR SHA-256
SHA-256 does not belong to the encryption algorithm category because it is a hash function. it belongs to the family of Hash algorithms. However, most of them turn data into random ciphertext that will be extremely hard to decrypt or read without confusion.

Secure Hash Algorithm 256 (SHA-256) is a secure cipher hash algorithm that uses a cryptographic hash (digest) function to verify the integrity of data. The hash function is designed to produce a unique collision-free value from various data types. SHA-256 is among the most secure hashing algorithms in production. It has about a 0.01% probability of having collisions. Collision-free means that the hash algorithm will not produce the same hash value for two different data. SHA-256 returns characters of either lowercase or numerals, starting from zero through nine.

2. Certificate Generation- print out of local-cert.crt

```
Kenyk@bigscreen MINGW64 ~/Desktop/SSL/self-signed
$ ls
local_ssl.p12  local-cert.crt

Kenyk@bigscreen MINGW64 ~/Desktop/SSL/self-signed
$ keytool -printcert -file local-cert.crt
Owner: CN=kennedy uzoho, OU=snhu, O=cs 305, L=philly, ST=PA, C=US
Issuer: CN=kennedy uzoho, OU=snhu, O=cs 305, L=philly, ST=PA, C=US
Serial number: 63c705f4533babe6
Valid from: Sun Oct 16 02:32:03 EDT 2022 until: Mon Oct 16 02:32:03 EDT 2023
Certificate fingerprints:
         SHA1: 00:7F:76:67:FD:8F:EE:EA:52:36:57:E8:AE:43:F3:33:55:45:E7:DE
         SHA256: 21:D6:FD:0B:7D:F2:4E:03:BA:A8:44:62:9F:95:0F:58:CD:04:A8:5C:22:18:36:82:79:36:77:74:80:18:39:75
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.17 Criticality=false
SubjectAlternativeName [
  DNSName: localhost
]

#2: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 77 27 68 EF DD AA 98 18   A3 C8 8C 09 FE F4 8E BA  w'h.............
0010: 8F 1C F6 BE                                        ....
]
]
```

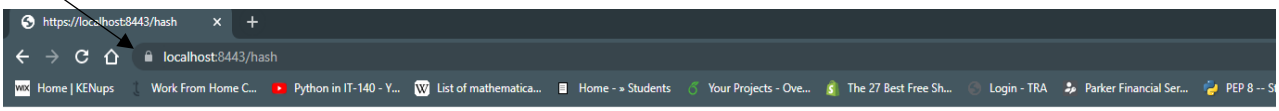
3. Deploy Cipher

@RestController held a source code for cryptographic hash function SHA-256, which will in return generate a hashed value for the provided data string “Kennedy Uzoho” and a return message “Welcome to a safe browsing environment”

```java
@RestController
class SslServerApplicationController {
    public static String calculateHash(String name) throws NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance("SHA-256"); // initialize object using SHA-256 and
        byte[] hash = md.digest(name.getBytes(StandardCharsets.UTF_8)); // compute messageDigest
        BigInteger number = new BigInteger(1, hash); // create hash value
        StringBuilder hexString = new StringBuilder(number.toString(16));
        while (hexString.length() < 32) {
            hexString.insert(0, '0');
        }
        return hexString.toString(); // return hexadecimal string/hashed data
    }
    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException {
        String data = "Kennedy Uzoho"; // declare messageDigest object (data to be encrypted)
        String hash = calculateHash(data); // instruction to calculate-hash data
        return "<p>Welcome to a safe browsing environment, " + data + " : SHA-256 " + " : " + hash; // return data, SHA-256, and Hashed
    }
}
```

Deploy Cipher

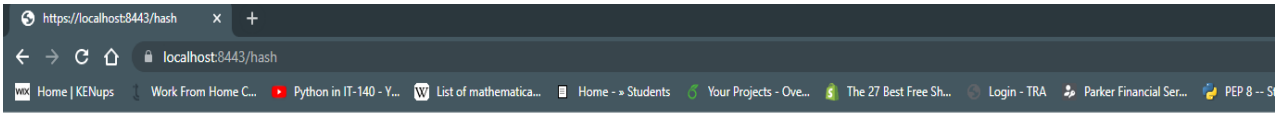
Trusted HTTPS WEB Server connection with Hashed string data type (“Kennedy Uzoho”) displayed



4. Secure Communications

HTTP to the HTTPS protocol **https://localhost:8443/hash**

I installed my own self-signed cert generated from CA into my computer and I trusted the key, that is why it is showing a secure connection.



5. Secondary Testing

Code base was analyzed, refactored, and executed as maven verify, maven install and as a stand-alone java web application listening on tomcat server port 8443.

Run type: Maven verify and install {Dependency report and a jar file was generated after this successful build}

```
[INFO] In DISPOSE, [CENTRAL] put 0 into auxiliary CENTRAL
[INFO] No longer waiting for event queue to finish: Pooled Cache Event Queue
Working = true
Alive = false
Empty = true
Queue Size = 0
Queue Capacity = 2147483647
Pool Size = 0
Maximum Pool Size = 150
[INFO] In dispose, destroying event queue.
[INFO] Region [CENTRAL] Saving keys to: CENTRAL, key count: 0
[INFO] Region [CENTRAL] Finished saving keys.
[INFO] Region [CENTRAL] Shutdown complete.
[INFO] In DISPOSE, [CENTRAL] disposing of memory cache.
[INFO] Memory Cache dispose called.
[INFO] In DISPOSE, [POM] fromRemote [false]
[INFO] In DISPOSE, [POM] auxiliary [POM]
[INFO] In DISPOSE, [POM] put 0 into auxiliary POM
[INFO] No longer waiting for event queue to finish: Pooled Cache Event Queue
Working = true
Alive = false
Empty = true
Queue Size = 0
Queue Capacity = 2147483647
Pool Size = 0
Maximum Pool Size = 150
[INFO] In dispose, destroying event queue.
[INFO] Region [POM] Saving keys to: POM, key count: 0
[INFO] Region [POM] Finished saving keys.
[INFO] Region [POM] Shutdown complete.
[INFO] In DISPOSE, [POM] disposing of memory cache.
[INFO] Memory Cache dispose called.
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ ssl-server ---
[INFO] Installing C:\Users\Kenyk\eclipse-workspace\CS 305 Project Two Code Base.zip_expanded\ssl-server_student\target\ssl-server-0.0.1-SNAPSHOT.jar to C:\Users
[INFO] Installing C:\Users\Kenyk\eclipse-workspace\CS 305 Project Two Code Base.zip_expanded\ssl-server_student\pom.xml to C:\Users\Kenyk\.m2\repository\com\snh
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  44.568 s
[INFO] Finished at: 2022-10-16T18:46:57-04:00
[INFO] ------------------------------------------------------------------------
```

Run type: Java web app listening on tomcat server, port 8443. https://localhost:8443/hash

```
Problems  Javadoc  Declaration  Console  Terminal  Keytool
SslServerApplication [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe  (Oct 16, 2022, 7:09:04 PM) [pid: 19588]

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2022-10-16 19:09:07.200  INFO 19588 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on bigscreen with PID 19588 (
2022-10-16 19:09:07.206  INFO 19588 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: de
2022-10-16 19:09:11.179  INFO 19588 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2022-10-16 19:09:11.212  INFO 19588 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-16 19:09:11.213  INFO 19588 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2022-10-16 19:09:11.427  INFO 19588 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-16 19:09:11.428  INFO 19588 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 410
2022-10-16 19:09:12.924  INFO 19588 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2022-10-16 19:09:14.560  INFO 19588 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path '
2022-10-16 19:09:14.570  INFO 19588 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 8.491 seconds (JVM running
```

Dependency check report

Report without suppressing false positives

DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at your own risk. In no event shall any party owning or maintaining this tool or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.2.1
- *Report Generated On*: Sun, 16 Oct 2022 14:07:27 -0400
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 14
- *Vulnerabilities Found*: 76
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

First iteration to suppress known false positives

DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | **Getting Help:** github issues

Sponsor

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 7.2.1
- *Report Generated On*: Sun, 16 Oct 2022 18:16:24 -0400
- *Dependencies Scanned*: 49 (31 unique)
- *Vulnerable Dependencies*: 11
- *Vulnerabilities Found*: 44
- *Vulnerabilities Suppressed*: 40
- ...

Summary

Second iteration to suppress known false positives

DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting pro or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | **Getting Help:** github issues

Sponsor

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

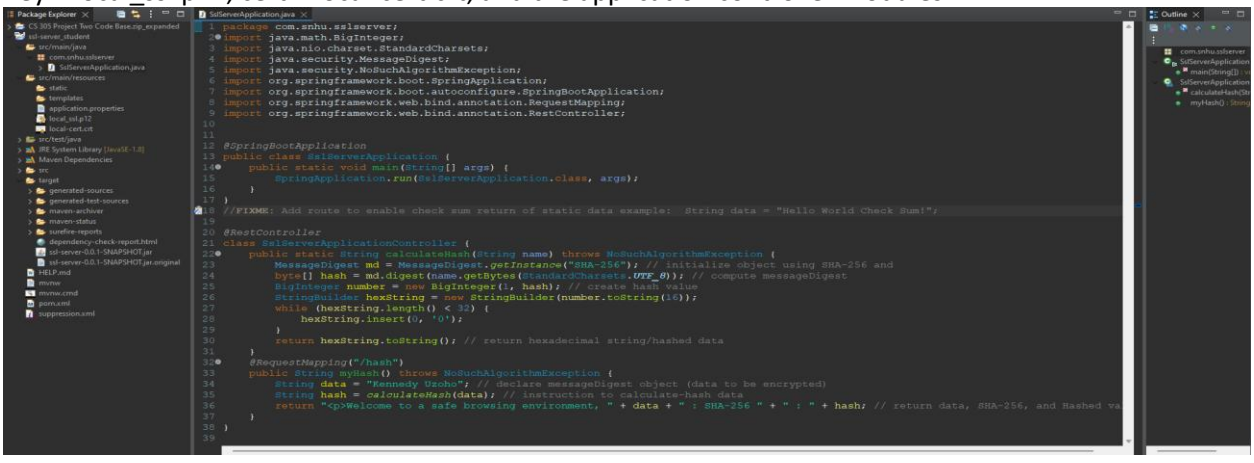
Scan Information (show less):

- *dependency-check version*: 7.2.1
- *Report Generated On*: Sun, 16 Oct 2022 18:28:30 -0400
- *Dependencies Scanned*: 49 (32 unique)
- *Vulnerable Dependencies*: 10
- *Vulnerabilities Found*: 38
- *Vulnerabilities Suppressed*: 55
- *NVD CVE Checked*: 2022-10-16T18:16:01
- *NVD CVE Modified*: 2022-10-16T18:00:01
- *VersionCheckOn*: 2022-10-12T13:18:18

6. Functional Testing

Reviewed and refactored code base with all parts completed as required for this milestone.

Key = local_ssl.p12, cert = local-cert.crt, and the application controller modules.



7. Summary

Summary and process for adding layers of security to the software application.

After an initial review of the code base, a controller java class was created, and a hash algorithm function was added to the controller class. In-code comments were provided for easy readability and analysis and to comply with the industry standard best practices. The code was debugged with no known errors. The pom.xml file was reviewed and refactored by updating the associated APIs and nested apps linked in the application. The maven dependency check was updated for the best result. There was a key “local_ssl.p12” and keystore “local_ssl.cer” generated in the code resource directory. These keys and their cert as published from CA allowed the web application to have a trusted connection after installing and trusting the self-signed certificate from CA into my computer.

The areas of the Vulnerability Assessment Process that were addressed after the code was refactored included:

API interaction > pom.xml file
Cryptography > Hash functions/ AES-256
Code Error > Debugging
Code quality > in-code comments
Encapsulation > nested APIs and secure data structure
Code review (controllers) and
Secure coding practice pattern

References

Compliance with cybersecurity and privacy laws and regulations. NIST. (2021, August 4). Retrieved July 23, 2022, from https://www.nist.gov/mep/cybersecurity-resources-manufacturers/compliance-cybersecurity-and-privacy-laws-and-regulations

https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

https://www.baeldung.com/spring-boot-https-self-signed-certificate

Symmetric vs asymmetric encryption: A guide for non-techies. HackerNoon. (2021). Retrieved August 10, 2022, from https://hackernoon.com/symmetric-vs-asymmetric-encryption-a-guide-for-non-techies-p03c316t