GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	12/1/2023	Eduardo Rodrigues	

Client

Instructions

Submit these 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
Eduardo Rodrigues

1. Algorithm Cipher

We will be implementing two cryptographic algorithms here, RSA and SHA3-244.

I. Our TLS connection requires us to have our endpoint authenticated using a self-signed X.509 certificate employing Rivest Shamir Adleman Digital Signature Algorithm with a key length of 2048 bits.

Overview: A digital signature is formed by the generation of private and public keys. It is an analog of a written signature; it aims to prevent anyone but the private key owner from being authenticated by the public key pair. (FIPS186, 2023) The certificates create a system whereby a public key is mathematically matched to a public key. (Akram,2020) NIST defines security strengths with respect to the effort in bit operations. An RSA key of 2048 bits will have a security strength of at most 112 bits where the probability of a collision occurring is roughly half or 2^{1024}.This can be more likely if the random number generation within the cryptosystem is not made random enough (Barker,2020) (Cook,2017)).

Mechanism: We will be using Oracle's key tool to generate a self-signed certificate via command line tools and IDE extensions. (We can keytool that ships with the Java SDK)

RSA Methodology: The RSA algorithm is a digital signature system that encrypts message blocks to cipher text using a mathematical function that raises an integer corresponding to the message to the power of a very large random prime number. (Rivest et al, 1978)

State of the Art: Research Suggests efforts are being made to thwart code breaking by quantum computing; NIST should have more information on acceptable standards via lattice-based cryptography. (Concover,2023) Current standards indicate that Ed25519 or Ed448 are safe and show speed improvements to RSA. (RFC8032, 2017) (Bernstein, 2011); but current TLS specifications can only implement the use of RSA and ECDSA certificates. (CA Forum, 2023) ECDSA certificates decrypt faster than RSA as key size increases and may be an option to consider in future implementations. (RFC 4050)

References

Akram et al. June 2020. SP 1800-16, Securing Web Transactions: TLS Server Certificate Management | CSRC (nist.gov) http://doi.org/10.6028/NIST.SP.1800-16. Gaithersburg, MD. NIST.

Barker, Elaine. May 2020. Recommendation for Key Management. SP800-57. https://csrc.nist.gov/csrc/media/projects/key-management/documents/transitions/transitioning_cryptoalgos_070209.pdf.

Bernstein et al. High-speed high-security signatures. 2011. https://ed25519.cr.yp.to/ed25519-20110926.pdf.

CA Browser Forum. Baseline Requirements for the Issuance and Management of Publicly Trusted Certificates. 11 April 2023. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-v2.0.0.pdf

Conover, Emily. June 28, 2017. Quantum computers could break the internet. https://www.sciencenews.org/article/quantum-computers-break-internet-save#:~:text=Public%2Dkey%20cryptography%20keeps%20our,vulnerable%20to%20future%20quantum%20computers.

FIPS186. Digital Signature Standard. February 3, 2023. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf. NIST.

John Cook. 2017. Probability of secure hash collisions https://www.johndcook.com/blog/2017/01/10/probability-of-secure-hash-collisions/#:~:text=As%20a%20rule%20of%20thumb,or%20about%204%20billion%20items.

Rivest et al. 1978. A method for obtaining digital signatures and public key cryptosystems.

RFC4050 Using the Elliptic Curve Signature Algorithm for XML Digital Signatures. https://www.rfc-editor.org/rfc/rfc4050

RFC8032. 2015. RSA Cryptography Specifications Version 2.2 https://datatracker.ietf.org/doc/html/rfc8017

II. A message digest will be used to produce a checksum for the data in question here; so, we will be using a one-way hashing function that research has regarded as being the most secure. NIST recommends that we move to SHA-3 for this kind of work; so, we choose to generate SHA3-224 message digests to verify information assurance.

Overview: The Secure Hash Algorithm-3 family of functions is based on the KECCAK algorithm that won a competition hosted by NIST. The methods employed prove to add state of the art mechanisms to resist collision, pre-image and second pre-image attacks more so than their predecessors in SHA-2. (NIST PUB 202. 2015)

Mechanism: We will use Oracle Java SDK Api to generate message digest programmatically.

SHA3-224 Methodology: SHA3-224 hash function creates a 224-bit digest using the KECCAK algorithm. (NIST PUB 202. 2015) KECCAK takes a binary input of variable length and produces a fixed length binary output that is not feasibly reproducible until proven otherwise. (Bertoni et al. 2012)

State of the Art: NIST suggests a migration from SHA-1 to SHA-2 or SHA-3 as soon as possible. (NIST, 2022)

References

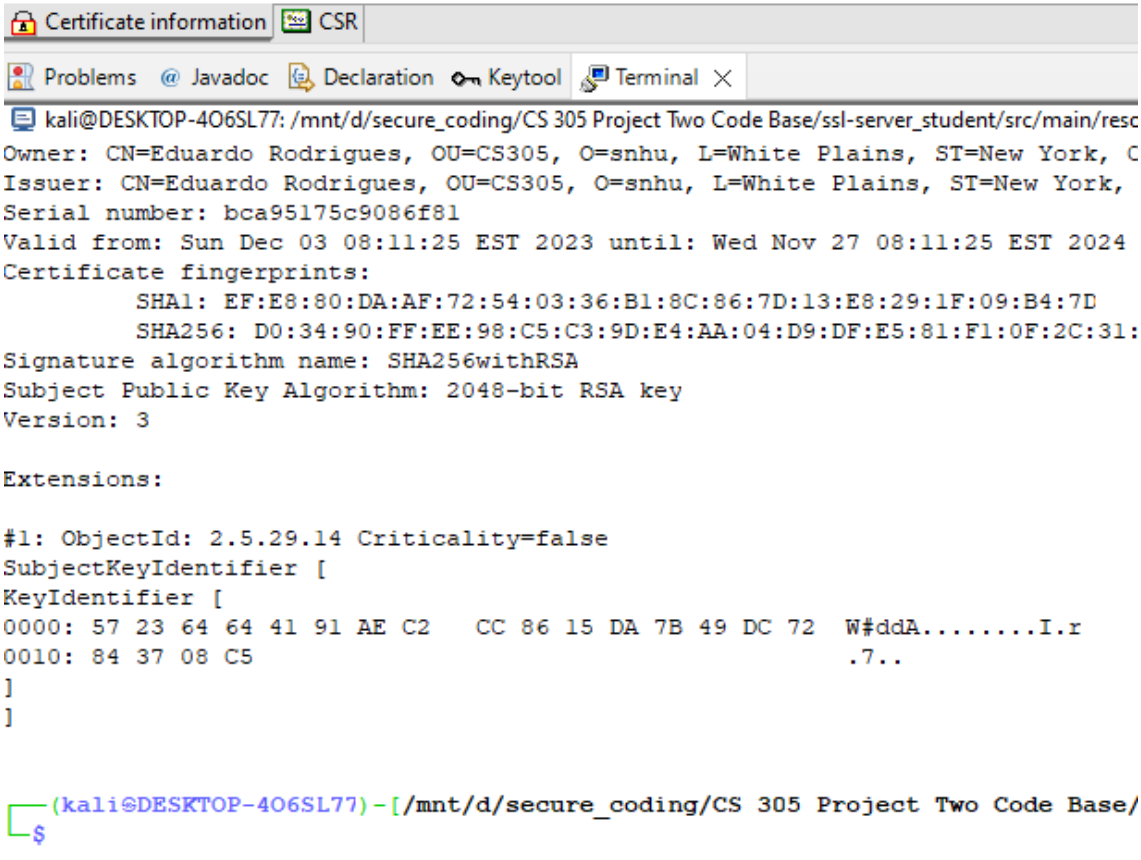
Bertoni et al. 2012. https://keccak.team/files/MakingOfKeccak.pdf
NIST. 2015. FIPS PUB 202. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

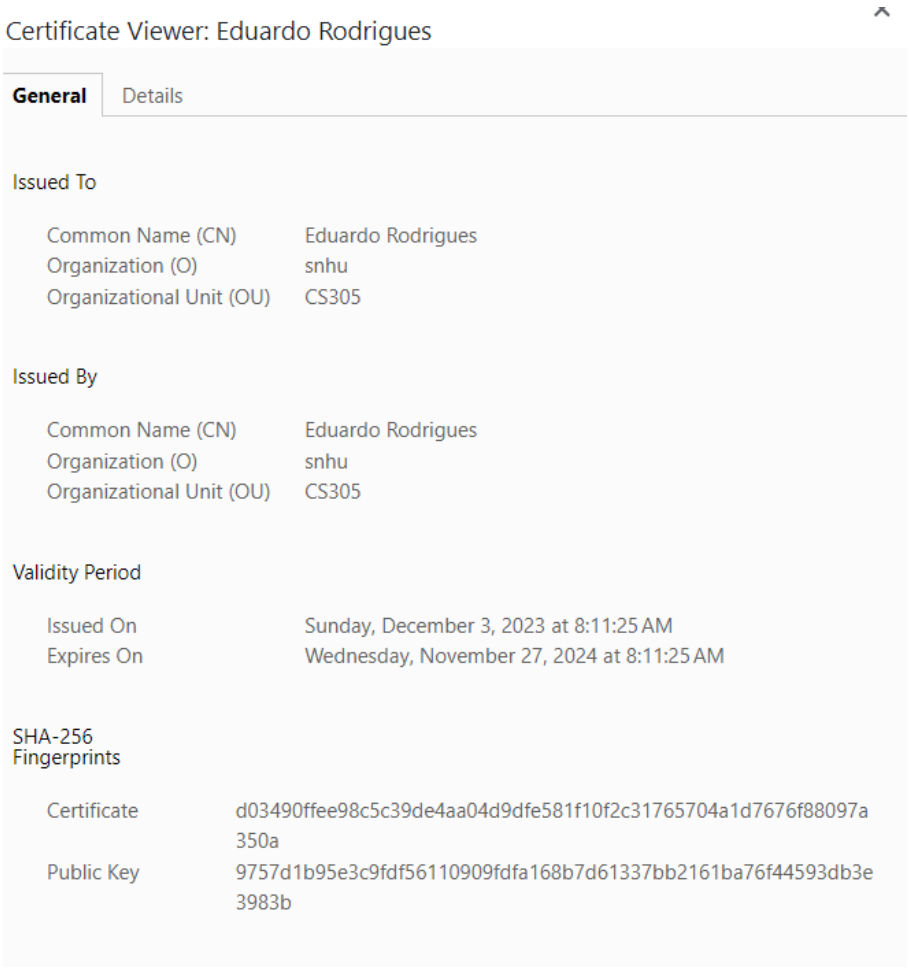
NIST. December 15, 2022. NIST Retires SHA-1 Cryptography Algorithm.
https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm.

2. Certificate Generation

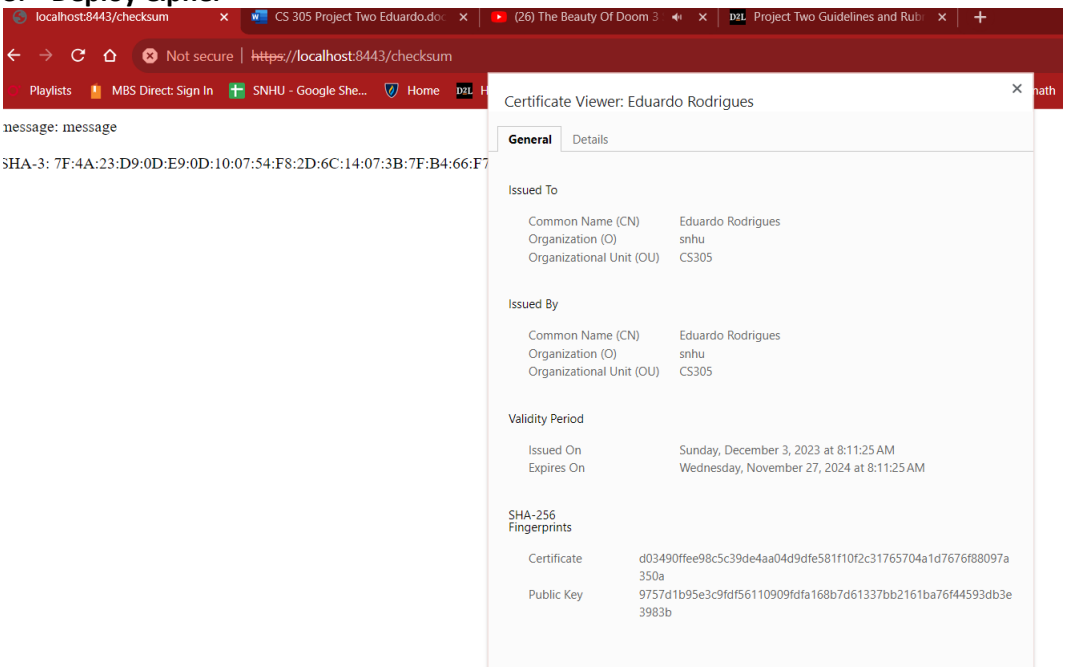
Certificate

Owner	CN=Eduardo Rodrigues,OU=CS305,O=snhu,L=White Plains,ST=New York,C=US
Issuer	CN=Eduardo Rodrigues,OU=CS305,O=snhu,L=White Plains,ST=New York,C=US
Valid from	Sun Dec 03 08:11:25 EST 2023
Valid to	Wed Nov 27 08:11:25 EST 2024
Serial Number	bca95175c9086f81
MD5 Fingerprint	C0:12:27:01:0E:A6:34:CA:C2:B2:2D:A7:DC:BC:36:F2
SHA1 Fingerprint	EF:E8:80:DA:AF:72:54:03:36:B1:8C:86:7D:13:E8:29:1F:09:B4:7D

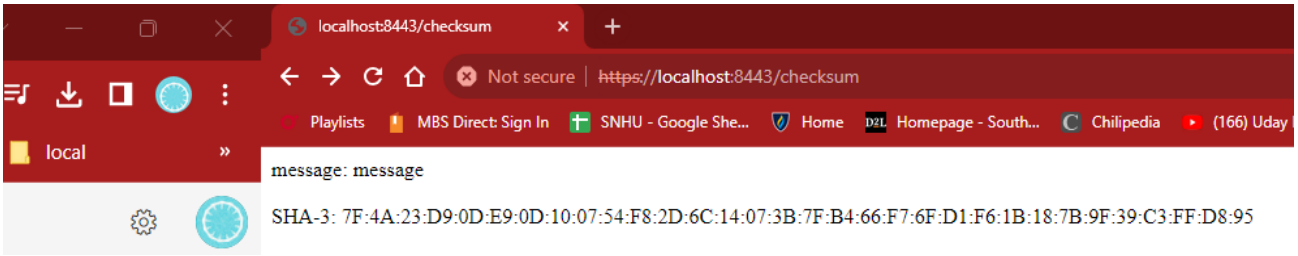




3. Deploy Cipher



4. Secure Communications



5. Secondary Testing

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

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

2023-12-03 19:53:06.656  INFO 9000 --- [           main] com.snhu.sslserver.SslS
erverApplication  : Starting SslServerApplication on DESKTOP-4O6SL77 with PID 90
00 (started by erodr in D:\secure_coding\CS 305 Project Two Code Base\ssl-server
_student)
2023-12-03 19:53:06.661  INFO 9000 --- [           main] com.snhu.sslserver.SslS
erverApplication  : No active profile set, falling back to default profiles: def
ault
2023-12-03 19:53:08.214  INFO 9000 --- [           main] o.s.b.w.embedded.tomcat
.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2023-12-03 19:53:08.224  INFO 9000 --- [           main] o.apache.catalina.core.
StandardService   : Starting service [Tomcat]
2023-12-03 19:53:08.224  INFO 9000 --- [           main] org.apache.catalina.cor
e.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2023-12-03 19:53:08.322  INFO 9000 --- [           main] o.a.c.c.C.[Tomcat].[loc
alhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-12-03 19:53:08.322  INFO 9000 --- [           main] o.s.web.context.Context
Loader            : Root WebApplicationContext: initialization completed in 1583
 ms
2023-12-03 19:53:09.004  INFO 9000 --- [           main] o.s.s.concurrent.Thread
PoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2023-12-03 19:53:09.679  INFO 9000 --- [           main] o.s.b.w.embedded.tomcat
.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2023-12-03 19:53:09.683  INFO 9000 --- [           main] com.snhu.sslserver.SslS
erverApplication  : Started SslServerApplication in 3.5 seconds (JVM running for
 3.878)
```

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show less):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Sun, 3 Dec 2023 19:46:07 -0500
- *Dependencies Scanned*: 49 (34 unique)
- *Vulnerable Dependencies*: 17
- *Vulnerabilities Found*: 73
- *Vulnerabilities Suppressed*: 0
- *NVD CVE Checked*: 2023-12-03T19:45:58
- *NVD CVE Modified*: 2023-12-03T18:00:00
- *VersionCheckOn*: 2023-11-10T19:50:20

6. Functional Testing

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

```
[INFO] --- install:2.5.2:install (default-install) @ s
sl-server ---
[INFO] Installing D:\secure_coding\CS 305 Project Two Code Base\ssl-server_stude
nt\target\ssl-server-0.0.1-SNAPSHOT.jar to C:\Users\erodr\.m2\repository\com\snh
u\ssl-server\0.0.1-SNAPSHOT\ssl-server-0.0.1-SNAPSHOT.jar
[INFO] Installing D:\secure_coding\CS 305 Project Two Code Base\ssl-server_stude
nt\pom.xml to C:\Users\erodr\.m2\repository\com\snhu\ssl-server\0.0.1-SNAPSHOT\s
sl-server-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
---
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
---
[INFO] Total time:  25.681 s
[INFO] Finished at: 2023-12-03T19:46:08-05:00
[INFO] ------------------------------------------------------------------------
---
```

7. Summary

We have an api route that displays a message digest and an embedded Tomcat server configured to make a TLS connection using a self-signed certificate generated using a keystore.

8. Industry Standard Best Practices

Both message digest and keystore are using the most secure form of hashing and digital signing that their platform allows. Our static testing reveals that our dependencies need upgrading. On manual review I will also note that a self-signed certificate should only be used on systems with little risk of eavesdropping; and that an endpoint as we have here is subject to be targeted for memory exhaustion.