



G L O B A L R A I N

**CS 305 Project Two
Practices for Secure Software Report**

Table of Contents

CS 305 Project Two	1
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	5
5. SECONDARY TESTING	6
Refactored code running without errors in Tomcat	6
Dependency Check report	7
6. FUNCTIONAL TESTING	7
7. SUMMARY	8

Document Revision History

Version	Date	Author	Comments
1.0	June 18, 2022	Lawrence Artl	

Client



Report

Every financial institution must consider data transfer from themselves to their customers and back again. Likewise, this data's security must also be considered, both in transit and while at rest. Artemis Financial has tasked Global Rain developers with designing the foundations of a RESTful web service that will employ the latest in cryptography and best-practice-security to help facilitate this transmission of sensitive data over potentially unsafe parts of the internet. Below is a report summarizing the process that developers at Global Rain have taken in order to achieve this goal.

Developer

Lawrence Artl

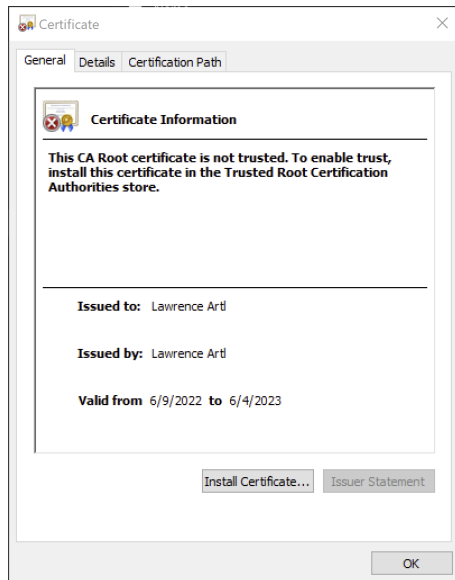
1. Algorithm Cipher

To begin, every data transmission should be encrypted with a strong hash algorithm. There are many to choose from, so determining which is best comes down to speed of the algorithm as well as security. In regards to the second option, collisions, or "how often two different pieces of data equate to the same hashed output", must be considered.

The SHA512 hashing algorithm was chosen for the security of data transmission from all endpoints in this server application. The SHA512 algorithm provides a greater amount of security than its slightly-lesser counterpart in SHA2, the SHA256 hashing algorithm. While SHA256 is 31% faster when hashing shorter inputs, the SHA512 algorithm is much less prone to "collisions" than others. Additionally, SHA512 has a message schedule array of 80 sixty-four-bit words, increasing the bit count from thirty-two, as found in SHA256. While SHA512 is not shown to be much greater security than SHA256, it is somewhat more secure, giving customers that little extra "piece of mind".

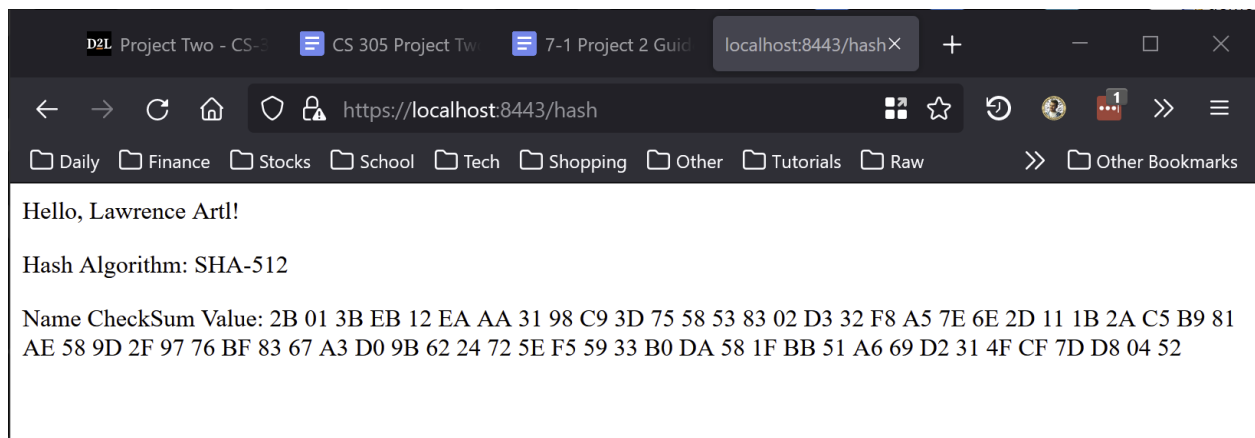
2. Certificate Generation

The Java keytool was used to generate the following certificate. This certificate is stored in the file “keystore” which will allow the website to access the web browser via the TLS and HTTPS protocols.



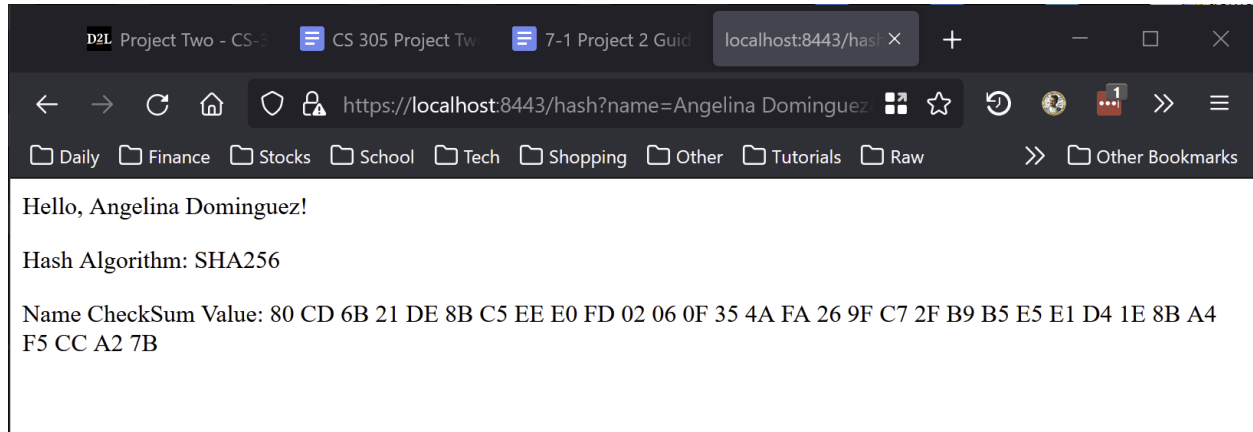
3. Deploy Cipher

As can be seen below, the /hash @RequestMapping takes advantage of the SHA512 hashing algorithm to take the input data (in this case, a user’s name) and convert it to a hash value. The hash value is then displayed in the web browser along with the original data string.



4. Secure Communications

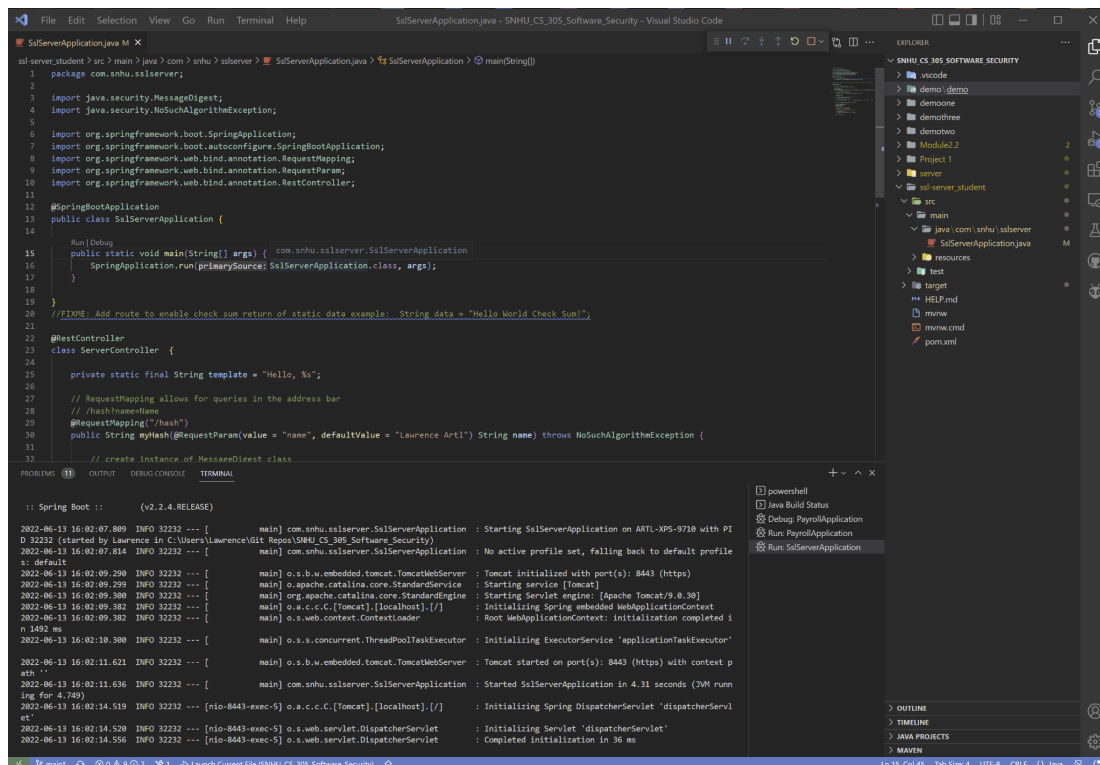
As indicated by the screenshot below, the refactored code uses the HTTPS protocol to send secure communications using a TLS certificate generated with the Java keytool. The certificate view can be found in [section 2](#).



5. Secondary Testing

A secondary static test was completed on the refactored code and a dependency check was run within the Eclipse IDE. All false-positives (items that are unlikely to apply or items that cannot be resolved at this time) were removed. The remaining dependency errors can be mitigated through an upgrade of the various dependencies within the application. Below is the refactored code running in VS Code, and the resulting Dependency Check output.

Refactored code running without errors through Tomcat (in VS Code)



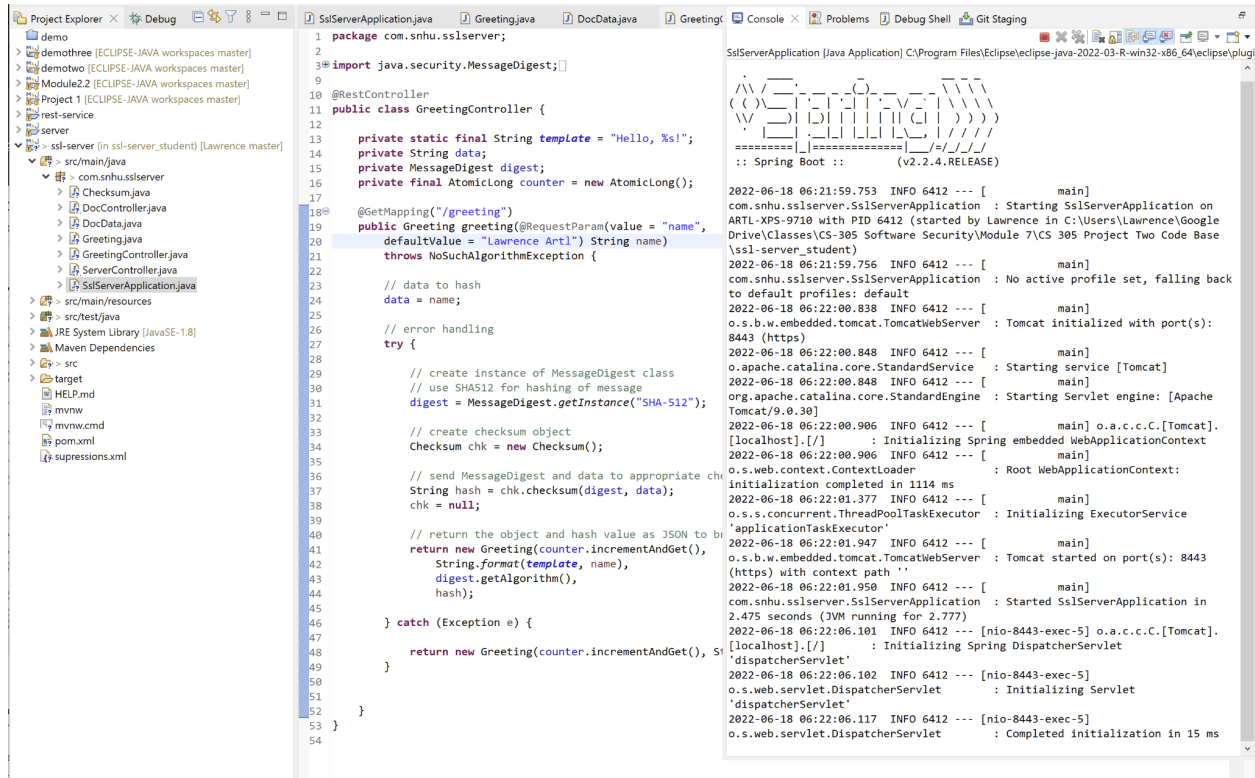
Dependency Check report

Dependency-Check Report							
Users/Lawrence/Google Drive/Classes/CS-305 Software Security/Module 7/CS 305 Project Two Code Base/ssl-ser							
Daily Finance Stocks School Tech Shopping Other Tutorials Raw Kali Links Web Dev							
Other Bookmarks							
Summary							
Display: Showing Vulnerable Dependencies (click to show all)							
Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count	
accessors-smart-1.2.jar	cpe:2.3:a:json-smart_project:json-smart-v1.1.2:*:*:*:*:*	pkg:maven/net.minidev/accessors-smart@1.2	CRITICAL	1	Low	37	
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	1	Highest	34	
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:* cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	2	Highest	41	
json-smart-2.3.jar	cpe:2.3:a:ini-parser_project:ini-parser:2.3:*:*:*:* cpe:2.3:a:json-smart_project:json-smart-v2.2.3:*:*:*:*	pkg:maven/net.minidev/json-smart@2.3	CRITICAL	2	Low	47	
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	1	Highest	44	
logback-core-1.2.3.jar	cpe:2.3:a:qos:logback:1.2.3:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.3	MEDIUM	1	Highest	33	
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:* cpe:2.3:a:yaml_project:yaml:1.25:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	HIGH	1	Highest	46	
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	HIGH	1	Highest	39	
spring-boot-starter-data-rest-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:* cpe:2.3:a:vmware:spring_data_rest:2.2.4:release:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-data-rest@2.2.4.RELEASE	HIGH	1	Highest	35	
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	CRITICAL	10	Highest	36	
spring-data-rest-webmvc-3.2.4.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_data_rest:3.2.4:release:*:*:* cpe:2.3:a:vmware:spring_data_rest:3.2.4:release:*:*:*	pkg:maven/org.springframework.data/spring-data-rest-webmvc@3.2.4.RELEASE	MEDIUM	1	Highest	27	
spring-tx-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:*:*:*	pkg:maven/org.springframework/spring-tx@5.2.3.RELEASE	CRITICAL	10	Highest	33	
tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache: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	16	Highest	33	
tomcat-embed-websocket-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30	CRITICAL	17	Highest	32	

6. Functional Testing

Syntactical, logical, and security vulnerabilities are reviewed and eliminated through manual review of the code. Below the code is shown running in Java Eclipse IDE with no errors thrown.

Refactored code running without errors (in Eclipse IDE)



```
1 package com.snhu.ssslserver;
2
3 import java.security.MessageDigest;
4
5
6 @RestController
7 public class GreetingController {
8
9     private static final String template = "Hello, %s!";
10    private String data;
11    private MessageDigest digest;
12    private final AtomicLong counter = new AtomicLong();
13
14    @GetMapping("/greeting")
15    public Greeting greeting(@RequestParam(value = "name",
16        defaultValue = "Lawrence Arti") String name)
17        throws NoSuchAlgorithmException {
18
19        // data to hash
20        data = name;
21
22        // error handling
23        try {
24
25            // create instance of MessageDigest class
26            // use SHA512 for hashing of message
27            digest = MessageDigest.getInstance("SHA-512");
28
29            // create checksum object
30            Checksum chk = new Checksum();
31
32            // send MessageDigest and data to appropriate checksum method
33            String hash = chk.checksum(digest, data);
34            chk = null;
35
36            // return the object and hash value as JSON to browser
37            return new Greeting(counter.incrementAndGet(),
38                String.format(template,
39                    digest.getAlgorithm(),
40                    hash));
41        } catch (Exception e) {
42
43            return new Greeting(counter.incrementAndGet(),
44                String.format(template,
45                    "Error: " + e.getMessage(),
46                    hash));
47        }
48    }
49 }
50
51
52
53
54
```

```
2022-06-18 06:21:59.753 INFO 6412 --- [main]
com.snhu.ssslserver.SslServerApplication : Starting SslServerApplication on
ARTL-XPS-9710 with PID 6412 (started by Lawrence in C:\Users\Lawrence\Google
Drive\Classes\CS-305 Software Security\Module 7\CS 305 Project Two Code Base
\ssl-server_student)
2022-06-18 06:21:59.756 INFO 6412 --- [main]
com.snhu.ssslserver.SslServerApplication : No active profile set, falling back
to default profiles: default
2022-06-18 06:22:00.838 INFO 6412 --- [main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
8443 (https)
2022-06-18 06:22:00.848 INFO 6412 --- [main]
o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-18 06:22:00.848 INFO 6412 --- [main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
Tomcat/9.0.30]
2022-06-18 06:22:00.906 INFO 6412 --- [main] o.a.c.c.c.[Tomcat].
[localhost.localhost] : Initializing Spring embedded WebApplicationContext
2022-06-18 06:22:00.906 INFO 6412 --- [main]
o.s.web.context.support.AnnotationConfigWebApplicationContext : Root WebApplicationContext:
initialization completed in 1114 ms
2022-06-18 06:22:01.377 INFO 6412 --- [main]
o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
'applicationTaskExecutor'
2022-06-18 06:22:01.947 INFO 6412 --- [main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8443
(https) with context path ''
2022-06-18 06:22:01.950 INFO 6412 --- [main]
com.snhu.ssslserver.SslServerApplication : Started SslServerApplication in
2.475 seconds (JVM running for 2.777)
2022-06-18 06:22:06.101 INFO 6412 --- [nio-8443-exec-5] o.a.c.c.c.[Tomcat].
[localhost.localhost] : Initializing Spring DispatcherServlet
'dispatcherServlet'
2022-06-18 06:22:06.102 INFO 6412 --- [nio-8443-exec-5]
o.s.web.servlet.DispatcherServlet : Initializing Servlet
'dispatcherServlet'
2022-06-18 06:22:06.117 INFO 6412 --- [nio-8443-exec-5]
o.s.web.servlet.DispatcherServlet : Completed initialization in 15 ms
```

7. Summary

The original code as delivered included a very basic rest server application that returned no results, no strings, and no objects. The first task was to implement several methods to illustrate the potential endpoints that would be available to the end user. An endpoint called “greeting” made use of the “Greeting” object. This object is created in the “GreetingController” at the endpoint “greeting” when the user inputs their name as a query, or if they use the default value. The user’s name is passed to the appropriate checksum method in the “Checksum” class to be hashed and converted to a string. When the string returns, the new Greeting object with an ID that is automatically incremented in the session, the user name value, the default hash algorithm name, and the hashed value of the name. This information is displayed in the web browser in JSON format.

The hash chosen for the default value is SHA-512 (for reasons discussed previously in this document). This partially addresses the customer’s need for strong cryptography. However, an additional step was taken to run the server through TLS (sometimes referred to as SSL) through an https port. This provides the data that is transferred with an extra layer of cryptographic protection. The TLS protocol makes use of a TLS certificate that is stored in a Java keystore file. This certificate contains both a public and private key that are used to authenticate the server and allow the server to encrypt and decrypt data. This adds a second layer of security to the data that is transferred from the server to the end user.

There are several other endpoints for users to interact with, each making use of the appropriate Checksum method depending on the type of data passed to be hashed. Each endpoint (found within a controller for its given data type) makes use of a try-catch block to handle errors during application execution. Should the data fail to hash properly, an error is thrown to the browser window to inform the user of failure to hash. The /hash endpoint makes use of multiple queries so the user can try various hashing algorithms for comparison. Inputting a bad query for the algorithm type will also throw an error informing the user of the failure to hash.

The code itself has been refactored and written to provide maximum information to any developers that may review the code later. This includes comments throughout to help with debugging. Additionally, the code makes extensive use of encapsulation, passing data to the external Checksum class for hashing, allowing the various hashing algorithms to be reused by other controllers. The data that is passed to the checksum method is passed as an object, and the object is then hashed using the algorithm found in the MessageDigest object. This allows the same method to be used for all controllers in the application, regardless of the type of object passed.

In order to add layers of security to the application, variables within classes are private, only accessible via get and set methods. Likewise, applicable methods for hashing the objects of the classes are separate from the classes through encapsulation. Hashing the objects is done with the SHA512 algorithm as the default, and while this can be modified in the /hash endpoint, this endpoint is not intended for sensitive information.

Finally, the application has been scanned for vulnerabilities, and while some will remain until patches are found, many can be mitigated with updates to the latest dependencies. The specific dependencies needing updates can be found in the included Dependency Check Report; all other “false-positives” (items that do not have a patch yet, or do not affect this application directly) have been suppressed in the “suppressions.xml” file.