GLOBALRAIN

Practices for Secure Software Report

Table of Contents

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

Document Revision History

Version	Date	Author	Comments
1.0	10/14/2025	Gianna Screen	

Client

ARTEMIS
FINANCIAL

Developer
Gianna Screen

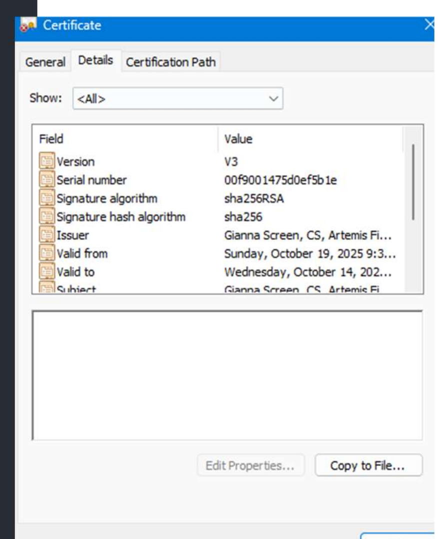
1. Algorithm Cipher

After reviewing the specific needs of Artemis Financial, I recommend deploying the Advanced Encryption Standard (AES) algorithm cipher. AES was developed in 2001 in response to attackers exploiting the limitations of the Data Encryption Standard (DES) and its 64-bit key size. Due to this restricted length, DES became increasingly vulnerable to brute-force attacks, allowing hackers to eventually guess the key after numerous attempts (Haldar, 2024). AES resolves this vulnerability by supporting a fixed block size of 128 bits and key sizes of 128, 192, or 256 bits, making encryption both more secure and efficient.

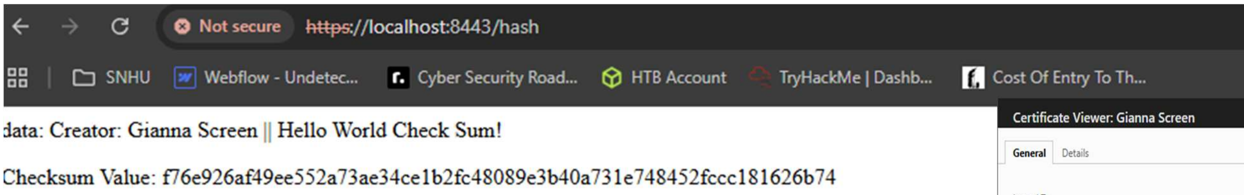
AES works by converting readable plaintext data into ciphertext, which appears as a randomized sequence of numbers and characters. It is a symmetric encryption algorithm, meaning the same key is used to both encrypt and decrypt data (Kumar, 2020). Since Artemis Financial manages not only sensitive information but also large volumes of it, this is an ideal choice because it allows for faster encryption and decryption. These attributes are especially important because, even if attackers were to exploit a system with weak input validation or other structural flaws, the data they obtained would remain unreadable and ultimately useless without the correct decryption key. This further highlights the importance of implementing cryptographic protections such as AES-128, particularly when handling the sensitive data Artemis Financial oversees. In addition to its larger block size and symmetric key implementation, I recommend AES over the other cipher algorithms listed by Oracle because it has consistently proven secure over time and has been widely adopted by top organizations, including the United States government (Haldar, 2024).

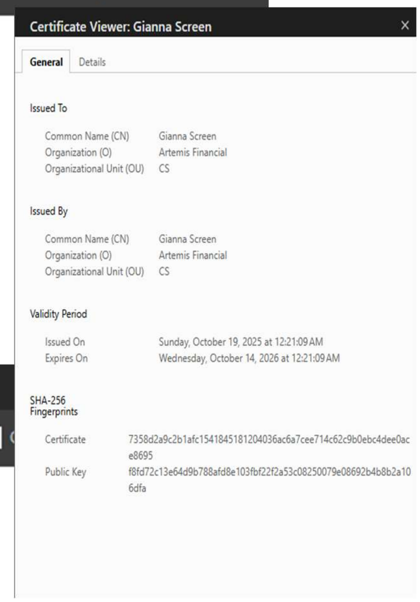
2. Certificate Generation

```
PS C:\Users\giann> keytool.exe -export -alias selfsigned -storepass GNSPSW@@ -file server.cer -keystore keystore.jks
Certificate stored in file <server.cer>
PS C:\Users\giann> keytool.exe -printcert -file server.cer
Owner: CN=Gianna Screen, OU=CS, O=Artemis Financial, L=Providence, ST=RI, C=US
Issuer: CN=Gianna Screen, OU=CS, O=Artemis Financial, L=Providence, ST=RI, C=US
Serial number: f9001475d0ef5b1e
Valid from: Sun Oct 19 21:32:57 EDT 2025 until: Wed Oct 14 21:32:57 EDT 2026
Certificate fingerprints:
         SHA1: 9C:5C:2A:E1:12:23:1F:F7:16:CE:02:BE:11:EA:1F:26:D9:A9:52:B0
         SHA256: D5:4C:55:96:26:3E:E2:F8:35:BB:0B:3C:30:1C:16:1E:3F:EB:FA:10:A6:B4:C7:30:70:35:D9:B6:38:BE:57:DB
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: 40 BD A8 55 1D 70 2F CD   7E 47 33 1F 7A 72 5F FA  @..U.p/..G3.zr_.
0010: 39 CD CB 13                                        9...
]
]

PS C:\Users\giann> |
```



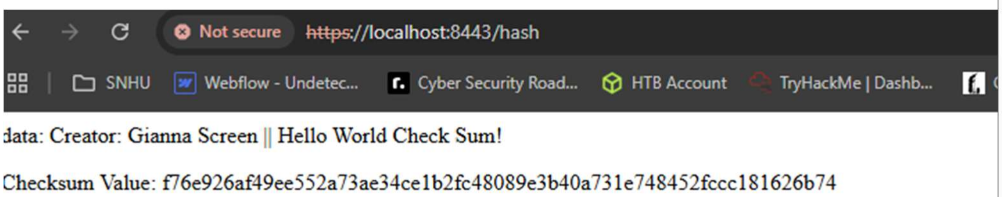
3. Deploy Cipher





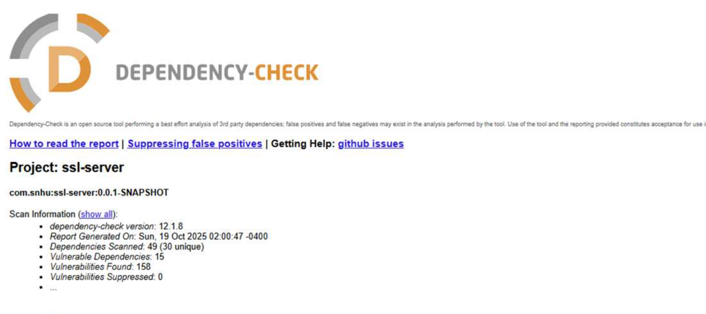
4. Secure Communications

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

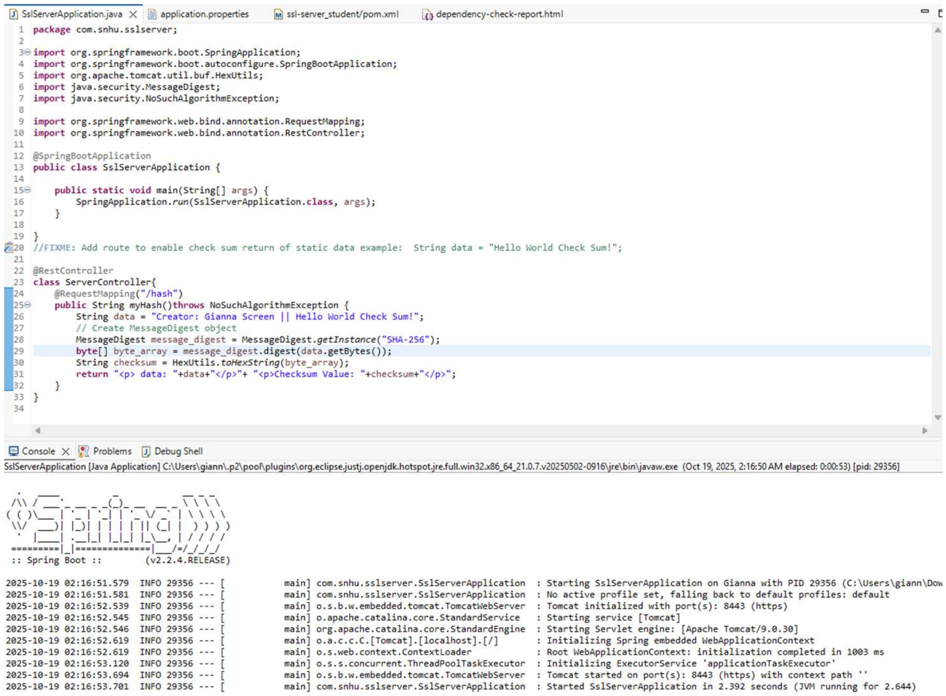


5. Secondary Testing

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



6. Functional Testing



7. Summary

The provided code base was refactored to comply with security testing protocols primarily through the implementation of cryptography using the SHA-256 hashing algorithm. Upon reviewing Artemis Financial's existing program, the vulnerability assessment process flow diagram was used to identify security gaps, with the most significant issue being the absence of cryptographic hashing for data protection. In the refactored version, sensitive data processed by the program is now cryptographically hashed, converting original plaintext into an unreadable and randomized sequence of characters.

In addition to implementing hashing, secure communication vulnerabilities were mitigated through the implementation of HTTPS protocol in place of HTTP, which ensures that communication between the browser and server is encrypted using self-signed certificates. Additionally, security vulnerabilities were assessed through secondary and functional testing. Secondary testing utilized the OWASP Dependency Check, which identifies known vulnerabilities within a program's dependencies, and functional testing involved manual code review to ensure there were no syntax errors and the program functioned as intended. For this code base, in addition to the refactored security features, I updated the Java and OWASP Dependency Check plugin versions to the most recent releases in the pom.xml file, which was essential for successfully building and running the program.

When thinking about the process for how each layer of security was added to the program, each step depended on how the previous one performed. For example, the first step was generating a digital certificate, stored as a CER file, to enable HTTPS encryption. This was followed by implementing the hash algorithm so that sensitive data displayed on the website appeared as a hashed sequence. With that complete and running, the application was configured to use the HTTPS protocol, which is dependent on the certificate already being generated and implemented appropriately within the codebase. Once those aspects were complete, the final steps were to verify that security requirements and protocols were met through testing, which was accomplished via manual review and OWASP dependency checks.

8. Industry Standard Best Practices

In order to mitigate known security vulnerabilities, following industry-standard best practices is most important. For this project, this was accomplished by following the outline provided in the Vulnerability Assessment Process Flow Diagram and applying those checkpoints to the project workflow. Examples of implementing industry standard best practices include ensuring clean, maintainable code through manual code review and adding comments where necessary for clarity. Additionally, implementing the hashing algorithm, switching from HTTP to HTTPS protocol for secure communications, updating dependencies to current versions, and testing for known vulnerabilities using the OWASP Dependency Check tool all classify as following industry standard best practices. These practices are important because these guidelines and recommendations were established based on previous experience with cyberattacks and the severe consequences that occur when they are not followed.

Following industry standards also ensures system compatibility and reliability, as most dependencies and APIs are designed with the expectation that developers will adhere to these security practices. When best practices are ignored, it not only creates security vulnerabilities but can also lead to integration failures and broken functionality.

Citations

Haldar, M. (2024, March 4). What is AES? How does it work? Encryption Consulting. https://www.encryptionconsulting.com/education-center/what-is-aes/