****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.lr4levy0t04)

[**Client 3**](#_heading=h.n6013pfwmzjh)

[**Instructions 3**](#_heading=h.sojreor1iu0i)

[**Developer 4**](#_heading=h.oq39ikva9bjn)

[**1. Algorithm Cipher 4**](#_heading=h.bv3azzratba2)

[**2. Certificate Generation 4**](#_heading=h.bs3v3yfu4kmz)

[**3. Deploy Cipher 4**](#_heading=h.9ox1f6vazh17)

[**4. Secure Communications 4**](#_heading=h.t2uyv1rn97am)

[**5. Secondary Testing 4**](#_heading=h.2nyjpjfi8ym7)

[**6. Functional Testing 4**](#_heading=h.zbdkftxgo4zh)

[**7. Summary 4**](#_heading=h.ieb9pi6haqz0)

[**8. Industry Standard Best Practices 4**](#_heading=h.5tkzk9xxjq4q)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/17/2025** | **Huan Ai** |  |

## 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

Huan Ai

## Algorithm Cipher

For Artemis Financial's web application, implementing SHA-256 ensures secure file verification through checksums. SHA-256 is a type of cryptographic hash function designed by the NSA and standardized by NIST. It produces a 256-bit (32-byte) hash value, typically rendered as a 64-digit hexadecimal number. SHA-256 is widely used in security applications and protocols, including TLS and SSL, PGP, SSH, and Bitcoin. It is ideal for verifying data integrity in financial applications because it provides collision resistance, making it extremely difficult for attackers to forge different inputs that produce the same hash.

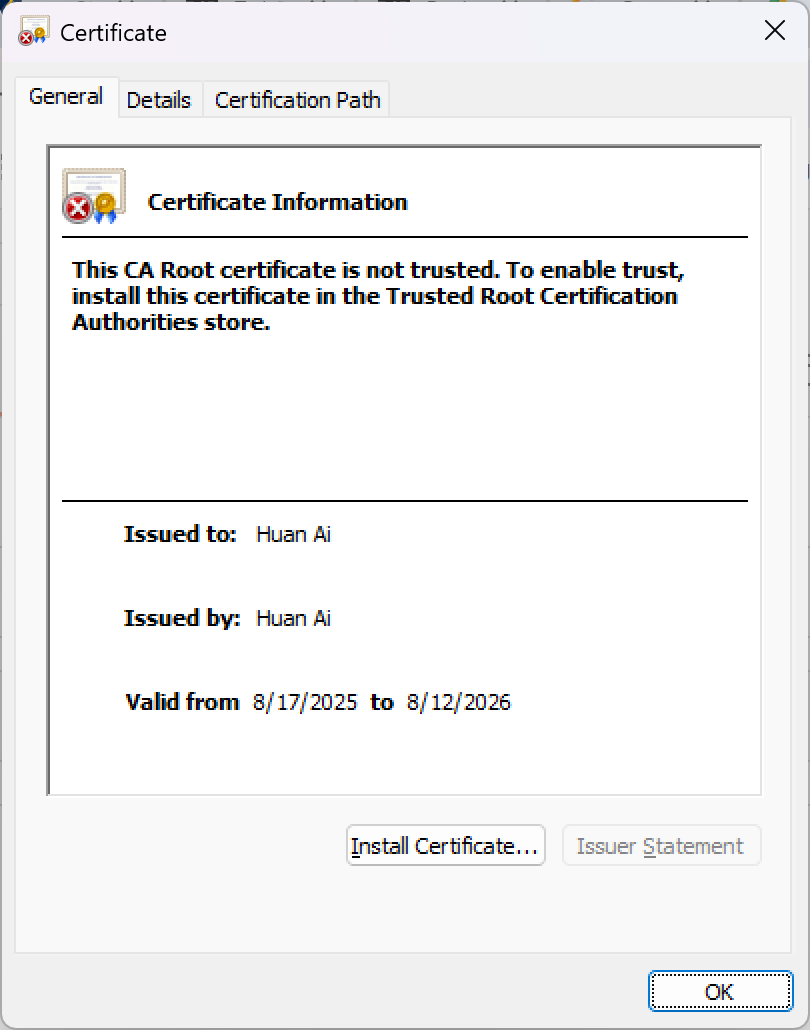
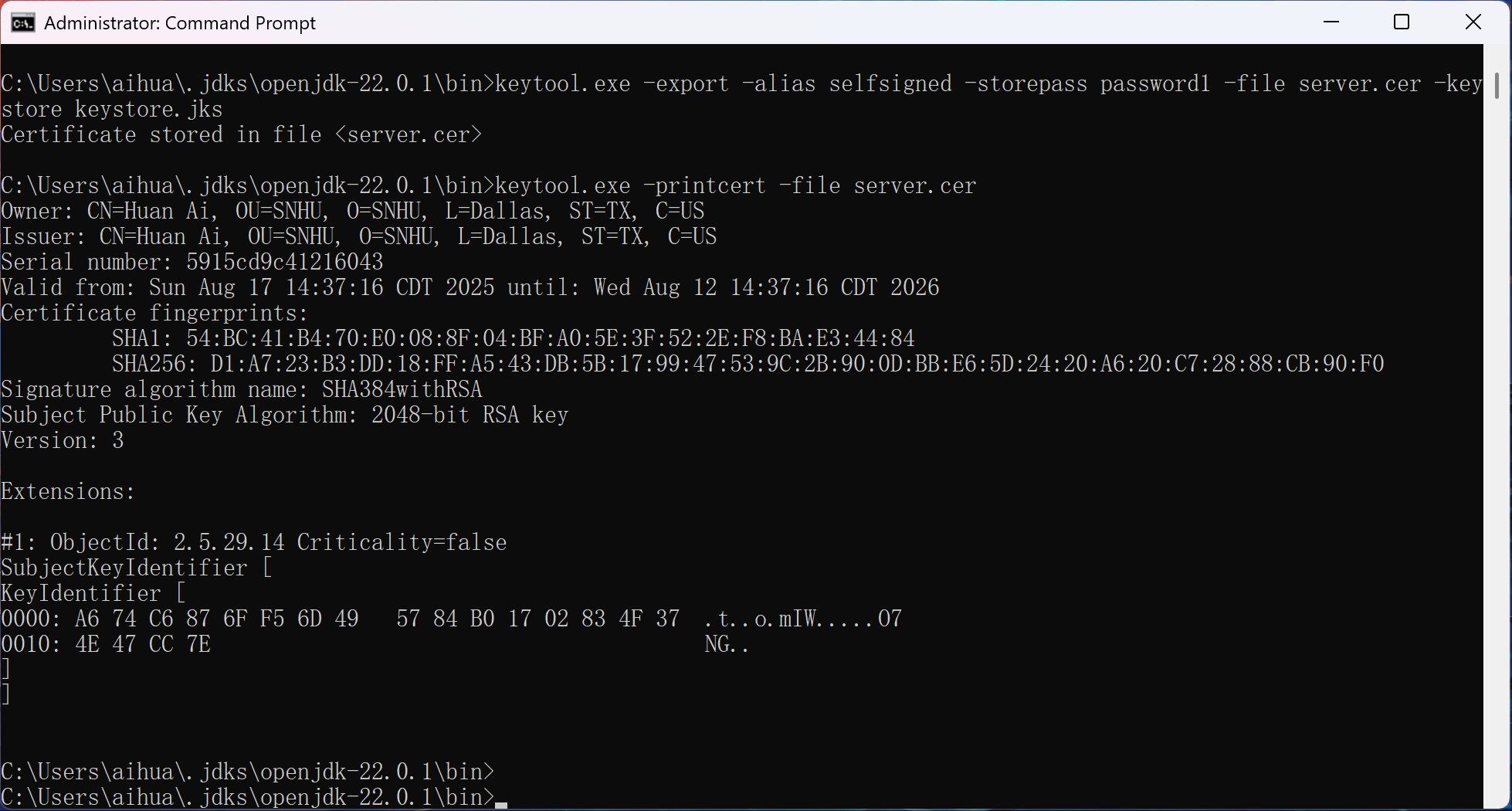
A hash function is like a digital fingerprint machine—it takes any input (a file, password, or message) and scrambles it into a fixed-length code called a hash value (NIST, 2015). For SHA-256, the input could be any data, and the output will always be a 64-character string like d7a8fbb3..., no matter the input size (Paar & Pelzl, 2010). The bit level refers to the hash size. SHA-256 produces a 256-bit hash (32 bytes). Think of it as a longer, more unique fingerprint than shorter hashes (e.g., 128-bit). This is going to be 2^256 possible hash combinations for SHA-256 (Stallings, 2017). Higher bit levels = harder to crack.

Randomness is critical in cryptography, but SHA-256 itself does not generate randomness. Instead it can enhance security by adding a random value (salt) to file inputs before hashing, which prevents rainbow table attacks (Eastlake & Jones, 2001). Symmetric keys use one secret key to lock/unlock data—fast for big files, but you must share the key securely. Asymmetric keys use two keys (public + private). They are safer for sharing, but slower, so they're mainly used to send symmetric keys securely. SHA-256 by itself doesn’t use keys, but we can use it in combinations with keys. For example, SHA-256 can be paired with HMAC for symmetric key verification (Paar & Pelzl, 2010), ensuring only authorized parties validate checksums. For asymmetric security, Artemis Financial could use digital signatures (e.g., RSA + SHA-256) to verify file authenticity (Stallings, 2017).

Encryption dates back thousands of years. The Caesar Cipher (used by Julius Caesar around 50 BCE) shifted letters by a fixed number to hide messages (Kahn, 1996). Modern encryption algorithms - AES and SHA-2 family, were approved in 2001, which still protect top-secret data today.

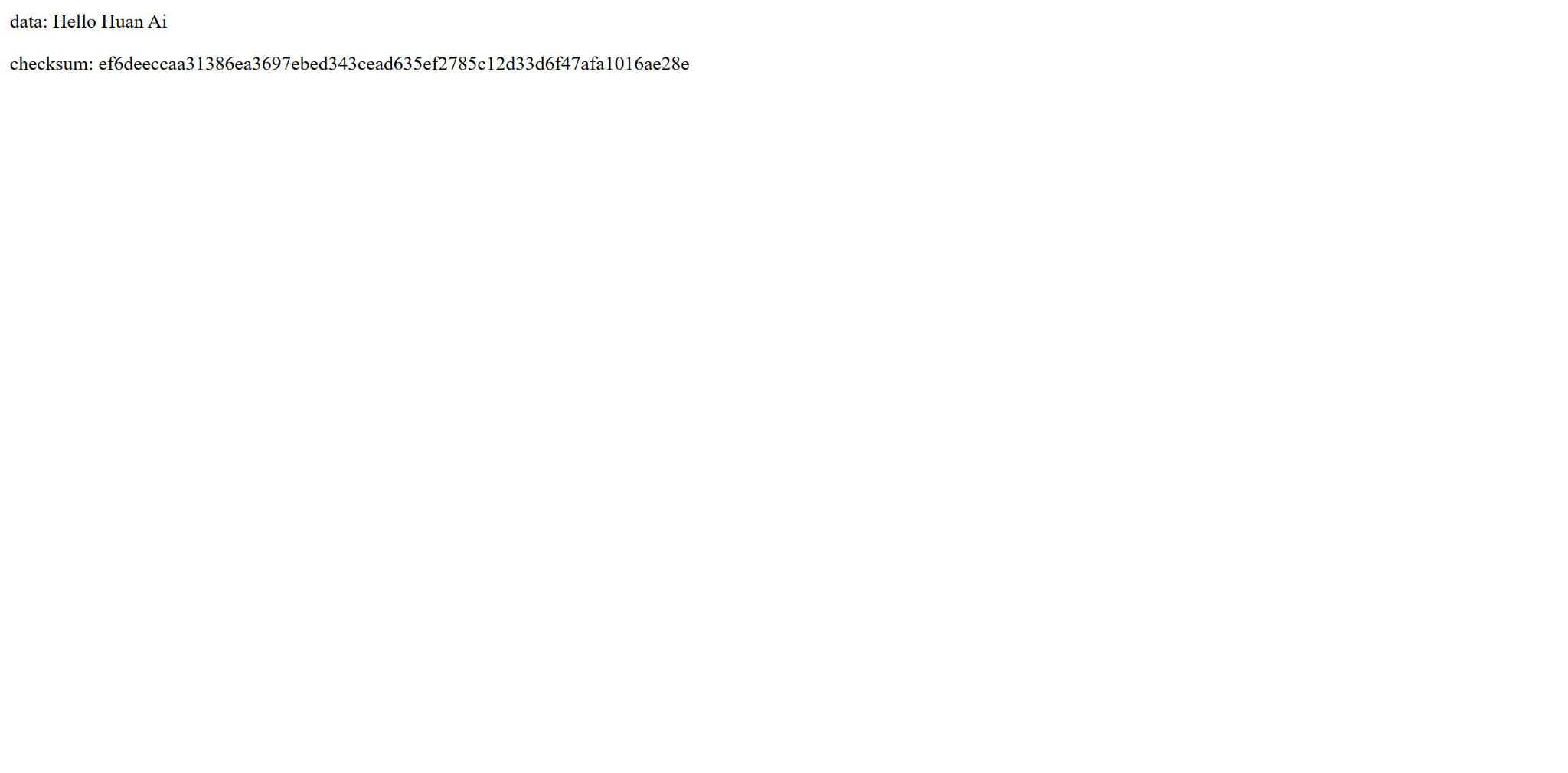
## Certificate Generation

Insert a screenshot below of the CER file.



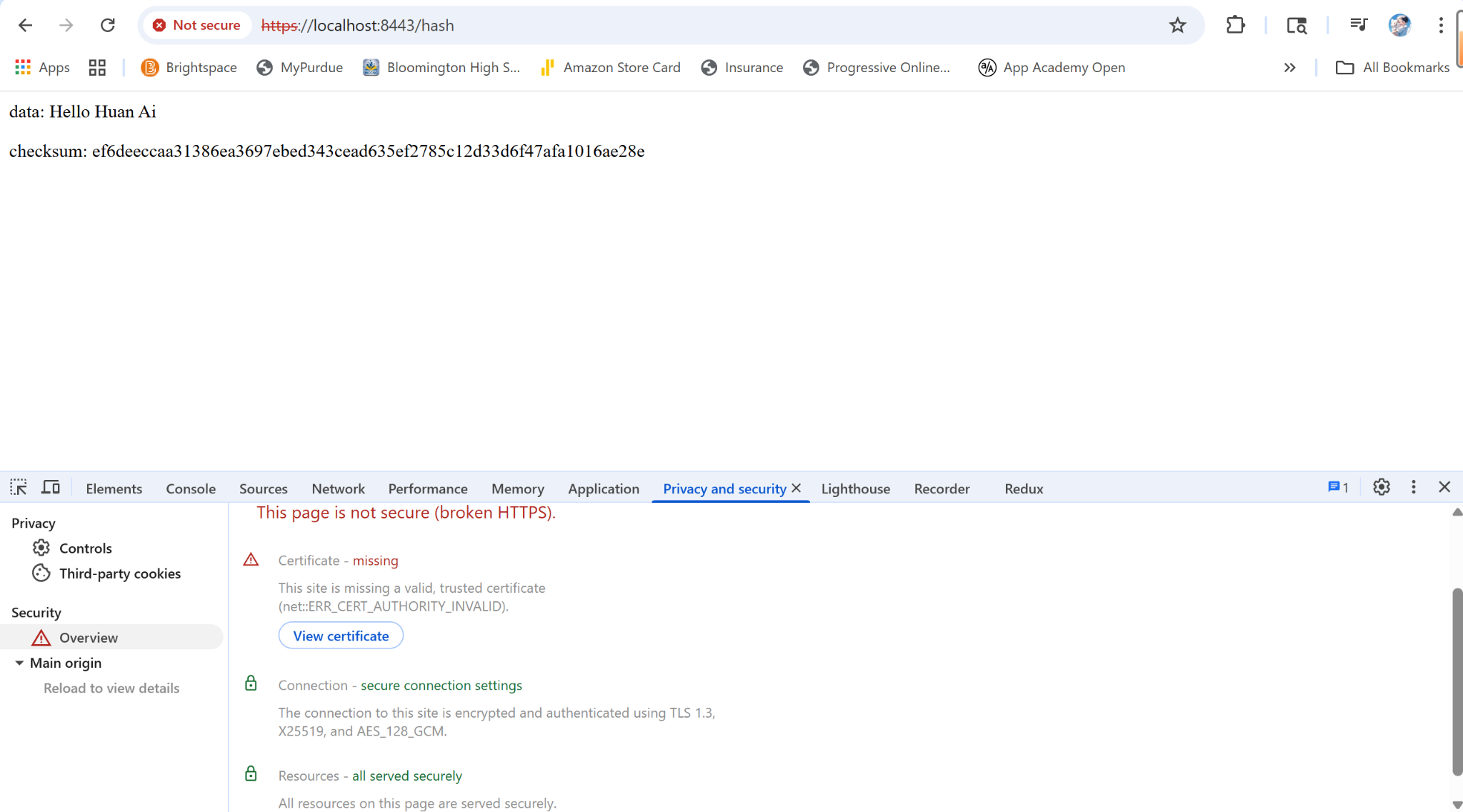
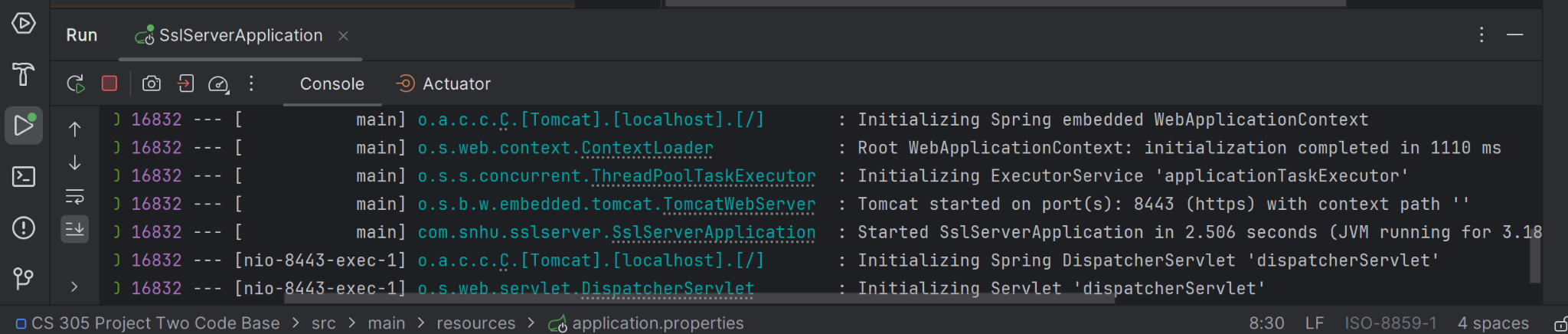
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

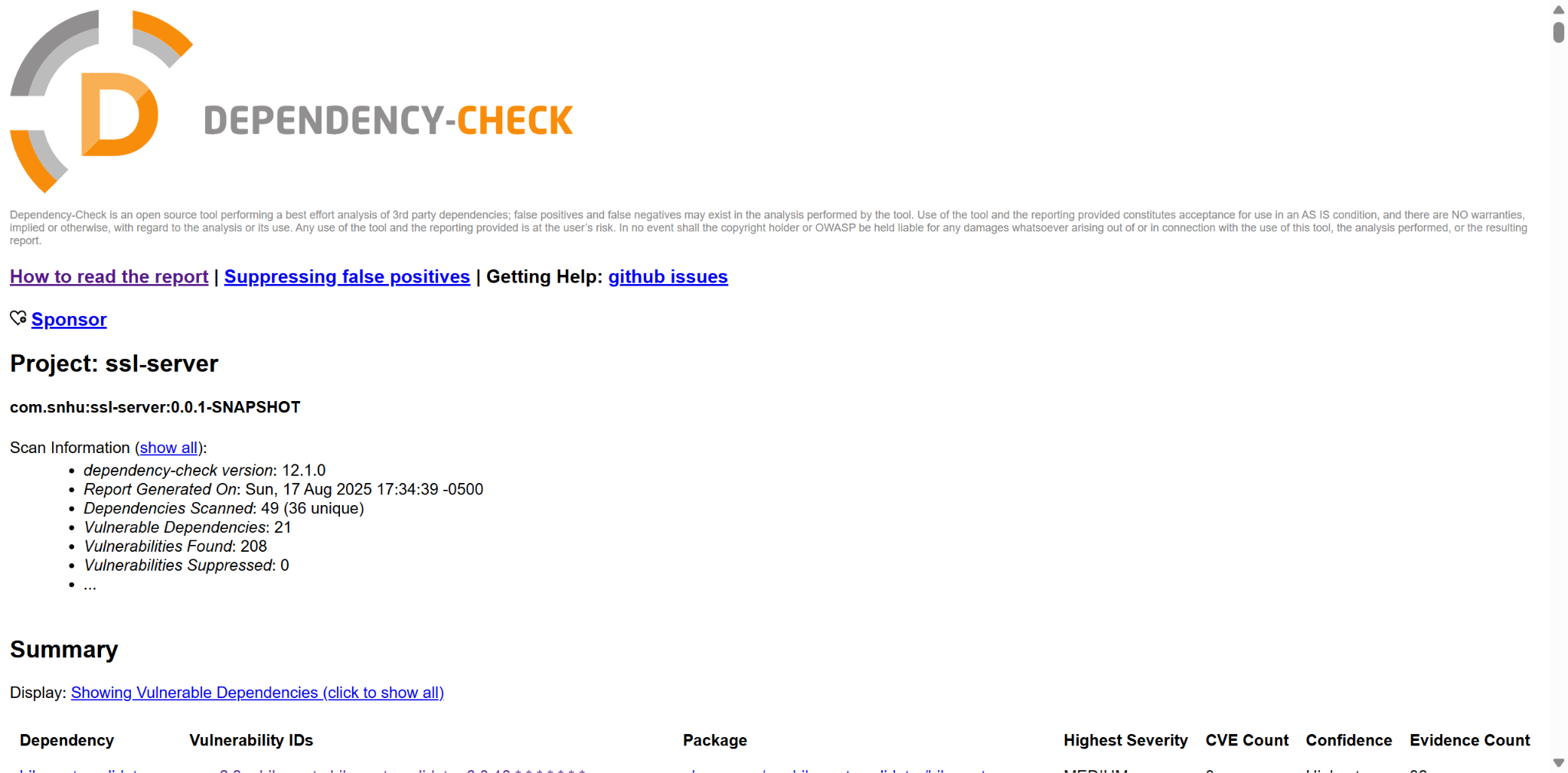
Insert a screenshot below of the web browser that shows a secure webpage.

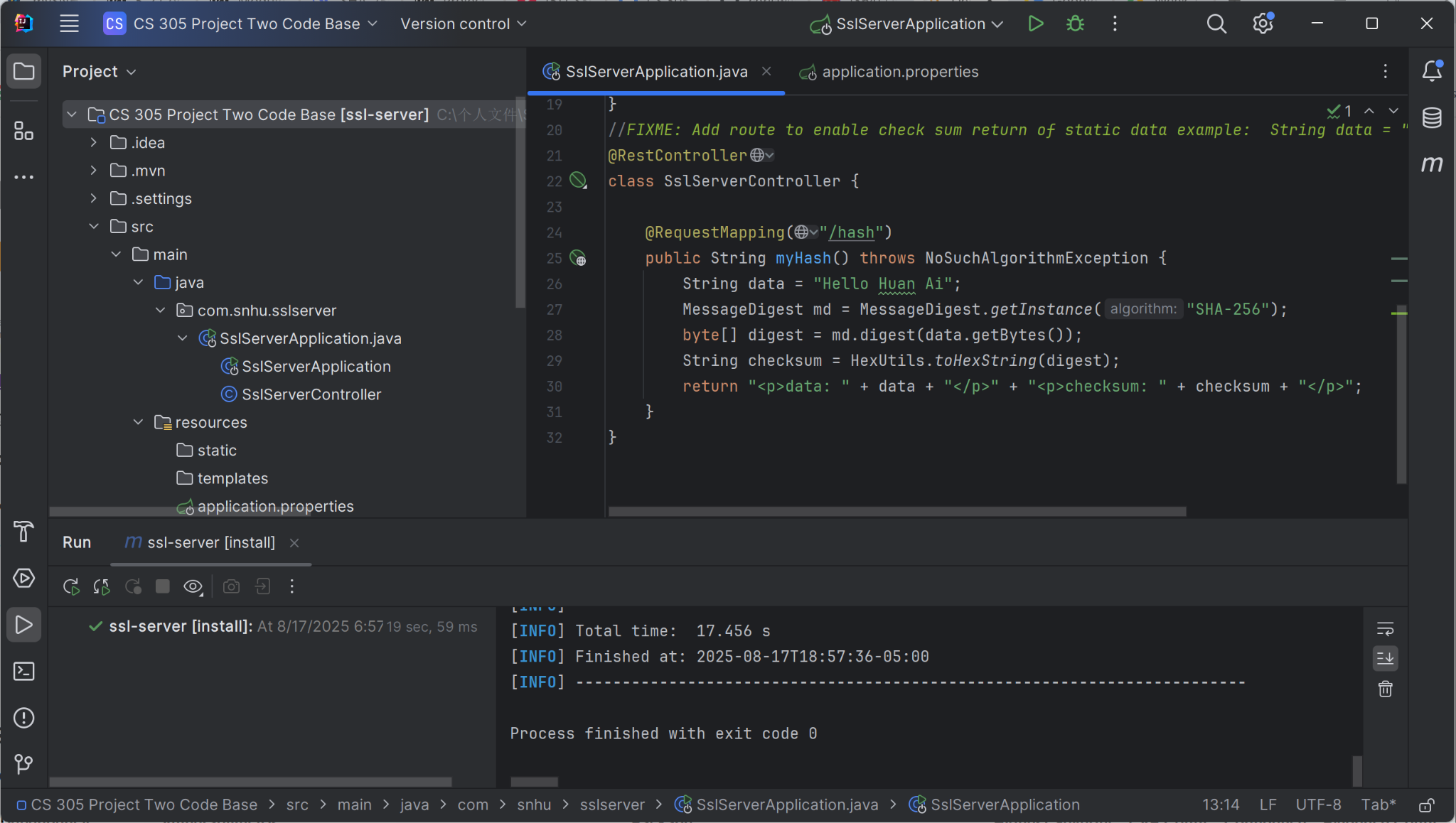


Springboot is loading on HTTPS, however due to a self-signed certificate, the browser is still giving the warning.

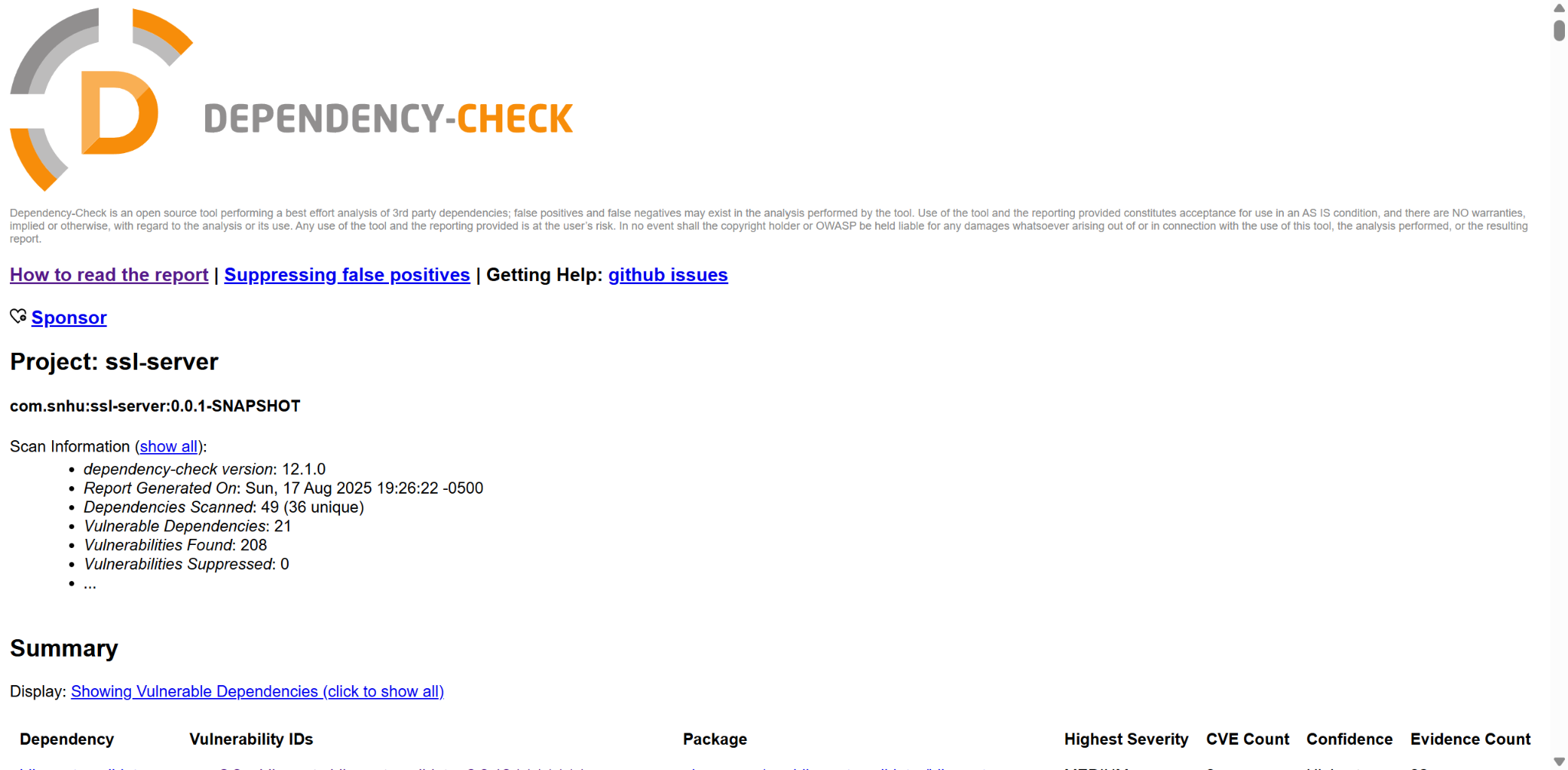
## Secondary Testing

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





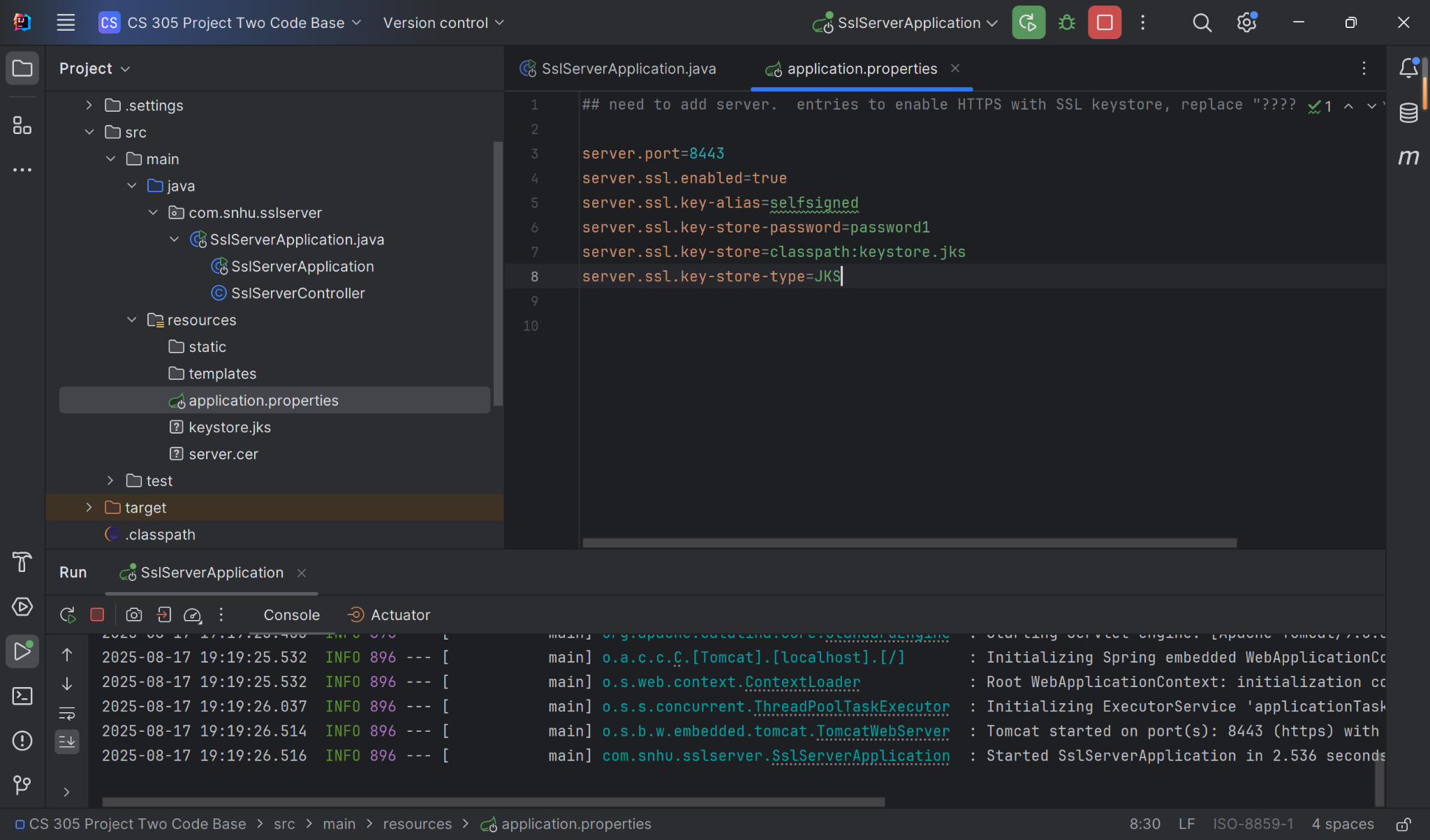
Post refactored:



The refactored code didn’t introduce additional vulnerabilities.

## Functional Testing

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



## Summary

I implemented SHA-256 checksums in SslServerApplicaiton.java to verify data integrity, which corresponds to Cryptography in the vulnerability assessment process flow diagram. Hashing "Hello Huan Ai" ensures tampering is detectable via the /hash endpoint. I also refactored application.properties to enforce HTTPS by filling out missing information with certification information, which prevents eavesdropping. This corresponds to Client/Server Security in the vulnerability assessment process flow diagram. Lastly, I improved Code Quality in the vulnerability assessment process flow diagram by using NoSuchAlgorithmException to handle possible missing crypto providers.

My approach to add layers of security to the software application starts with generating a self-signed certificate (keystore.jks) to enable HTTPS. Then I would add checksum verification for static data (SHA-256) to achieve data integrity. Lastly, I would test the application by verifying HTTPS worked via <https://localhost:8443/hash> and the application runs without any errors.

## Industry Standard Best Practices

To uphold Artemis Financial’s security standards, I applied the following best practices: cryptographic integrity, secure communication, minimal exposure, and error handling. For cryptographic integrity, I used SHA-256 (NIST-approved) for checksums in the /hash endpoint, ensuring data tampering is detectable. For secure communication, I enforced HTTPS-only connections in application.properties, disabling HTTP to prevent downgrade attacks. The self-signed certificate (keystore.jks) was configured with a strong password and 2048-bit RSA keys. To ensure minimal exposure, I limited the /hash endpoint to checksum verification only, avoiding unnecessary data exposure. No sensitive data is processed without encryption. For error handling, I implemented NoSuchAlgorithmException to fail securely if SHA-256 is unavailable, preventing silent crypto failures.

Applying these practices directly benefits Artemis Financial in terms of risk reduction. HTTPS and checksums mitigate risks like man-in-the-middle attacks and data tampering, critical for financial data. There will also be cost savings. According to IBM, proactive vulnerability prevention (e.g., dependency checks) reduces breach-related costs, which average $4.45M per incident (IBM, 2023).

**APA citation**

Eastlake, D., & Jones, P. (2001). \*US Secure Hash Algorithm 256 (SHA-256)\*. IETF RFC 4634. https://tools.ietf.org/html/rfc4634

National Institute of Standards and Technology. (2015). Secure Hash Standard (SHS) (FIPS PUB 180-4). https://doi.org/10.6028/NIST.FIPS.180-4

Stallings, W. (2017). Cryptography and Network Security (7th ed.). Pearson.

https://dl.hiva-network.com/Library/security/Cryptography-and-network-security-principles-and-practice.pdf

Paar, C., & Pelzl, J. (2010). Understanding Cryptography: A Textbook for Students and Practitioners. Springer.

http://ndl.ethernet.edu.et/bitstream/123456789/89369/1/Understanding%20cryptography%20a%20textbook%20for%20students%20and%20practitioners%20by%20Christof%20Paar%2C%20Jan%20Pelzl.pdf

Kahn, D. (1996). The Codebreakers: The Comprehensive History of Secret Communication. Scribner.

https://archive.org/details/2.thecodebreakersthecomprehensivehistoryofsecretcommunicationfromancienttimestotheinternet

IBM. (2023). Cost of a Data Breach Report.

https://d110erj175o600.cloudfront.net/wp-content/uploads/2023/07/25111651/Cost-of-a-Data-Breach-Report-2023.pdf