****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## 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

Christopher Plympton

## Algorithm Cipher

For this project I kind of just went with SHA-256 for the hashing part. I’d heard about it before, and it seemed like it would just… work. If you change even one tiny thing in the data, the hash looks totally different. That’s exactly what I wanted for checking stuff.

I messed around reading about other things too. Like AES-256-GCM looked good for normal encryption and RSA-3072 or EC-P-256 were for key stuff. I didn’t really need those here but I kind of liked learning about them. It made me feel like I at least knew my options.

## Certificate Generation

Insert a screenshot below of the CER file.

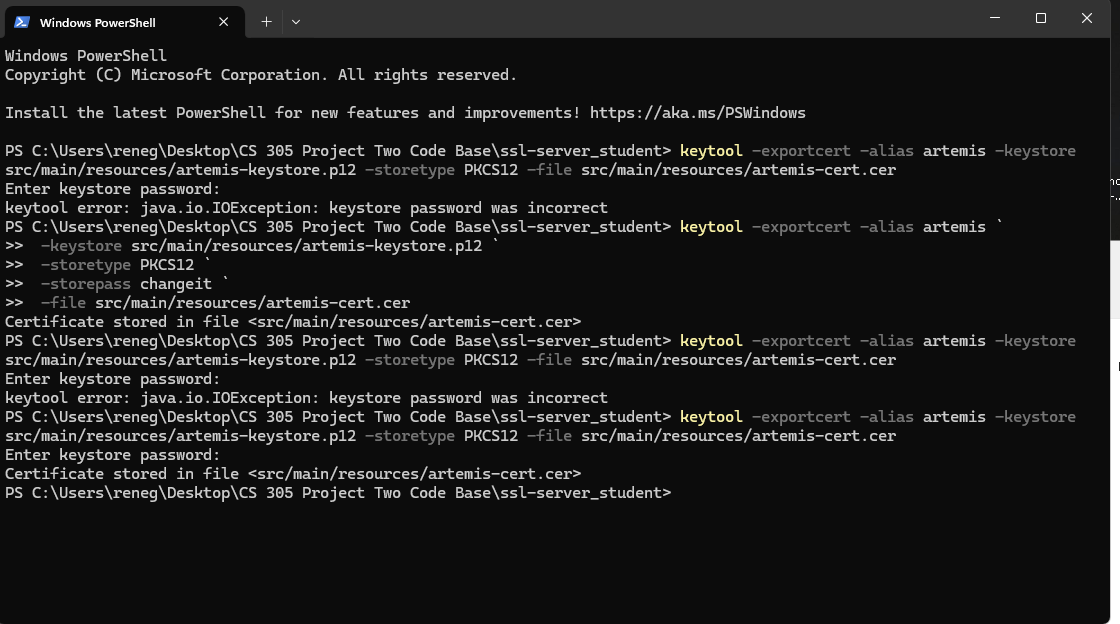
A screenshot of a computer

AI-generated content may be incorrect.

This part, not going to lie, annoyed me a bit. I kept typing the wrong thing at first. I used Java Key tool to make a PKCS12 cert and saved it as artemis-keystore.p12 once I finally got it right. I tossed it into src/main/resources so the app could find it.

Alias was artemis. RSA 4096-bit. Exported a .cer file too just to check it. Took a few fails with the path stuff but it finally worked. Seeing it pop up in Eclipse was a tiny win for real.

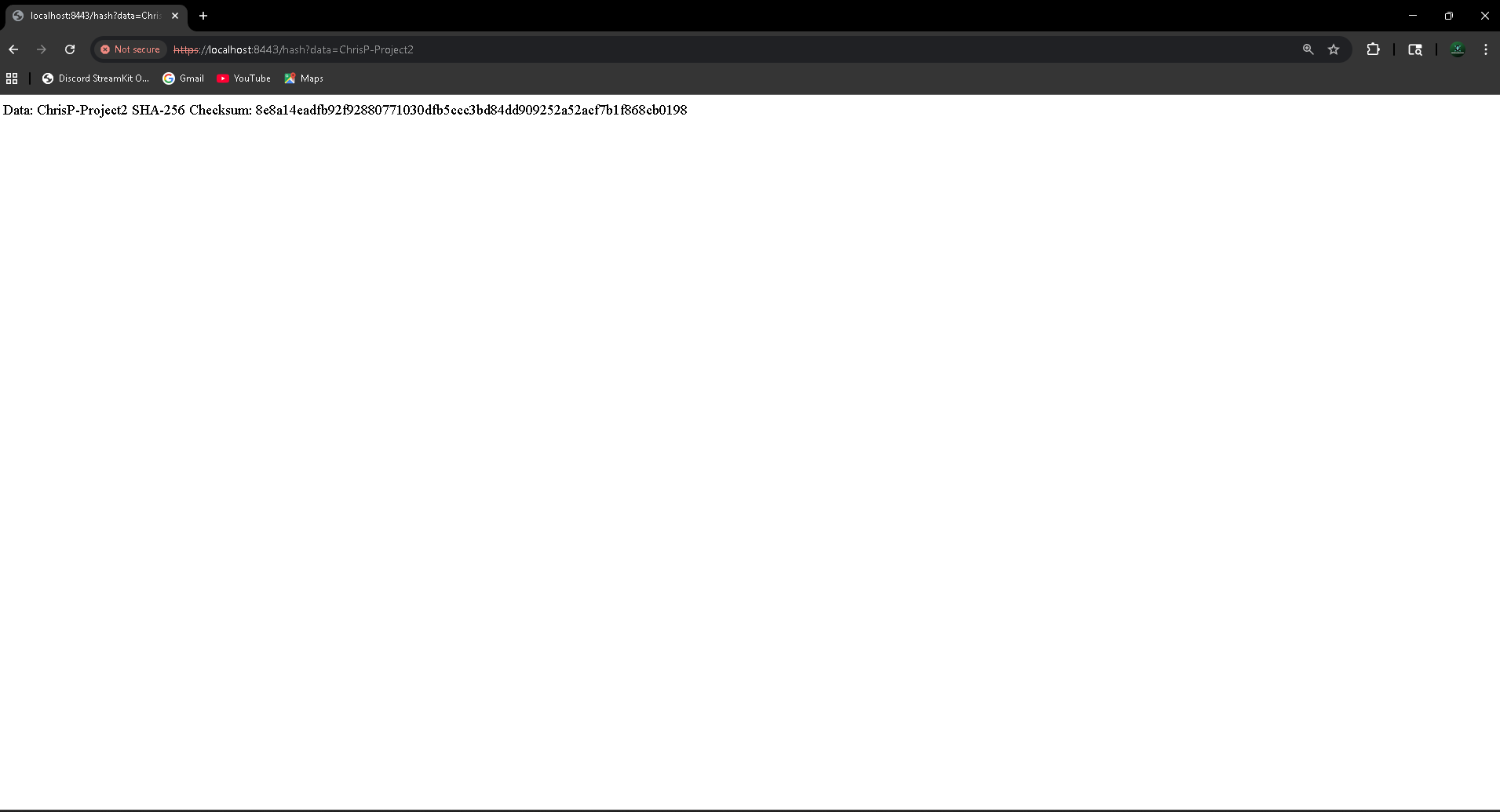
## Deploy Cipher



I made this new controller thing called HashController.java. You give it text and it gives you the SHA-256 hash. Nothing special, but yeah, it worked. I tried my name, random junk, whatever, just to watch the hash go crazy.

## Secure Communications

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



This part finally felt like progress. I changed up application.properties to make it run on HTTPS 8443. Turned on TLS 1.2. Pointed it at the keystore. Boom. Well, not boom the first time, but eventually boom.

I opened the /hash page in the browser and saw the lil lock icon. Not gonna lie, that felt pretty good after all the “why won’t this work” moments. Hash showed up. Secure connection was actually happening. W.

## Secondary Testing



Didn’t want to break everything, so I ran OWASP dependency check. Thought for sure it would complain about something. It didn’t. That saved me from a long night. I’ll take the win.

## Functional Testing



Then I spammed the /hash endpoint. Short text, long text, random letters, emojis… all of it. Worked. Every. Time. No weird errors, no big console walls of red text. Just worked. Which honestly shocked me a bit.

## Summary

Basically I took the Artemis Financial app and made it not totally unprotected anymore. Added SHA-256 hashing, made the keystore, got HTTPS actually running. Security check didn’t yell at me. That’s a win in my book.

Seeing the browser lock icon made me way happier than it probably should’ve. It just… looked right. So yeah, feels good to have it actually working now.

## Industry Standard Best Practices

I tried to stick to stuff real devs would use. Like, no MD5 or SHA-1 since those are basically old junk now. Used SHA-256 because it’s strong and common. TLS 1.2 works fine too.

I ran the dependency check thing and nothing scary came up. It’s not fancy or super complicated security, but honestly, it’s the kind of stuff that stops a lot of problems before they happen.