****

# 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** | **8/11/23** | **Steven Anderson** |  |

## 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

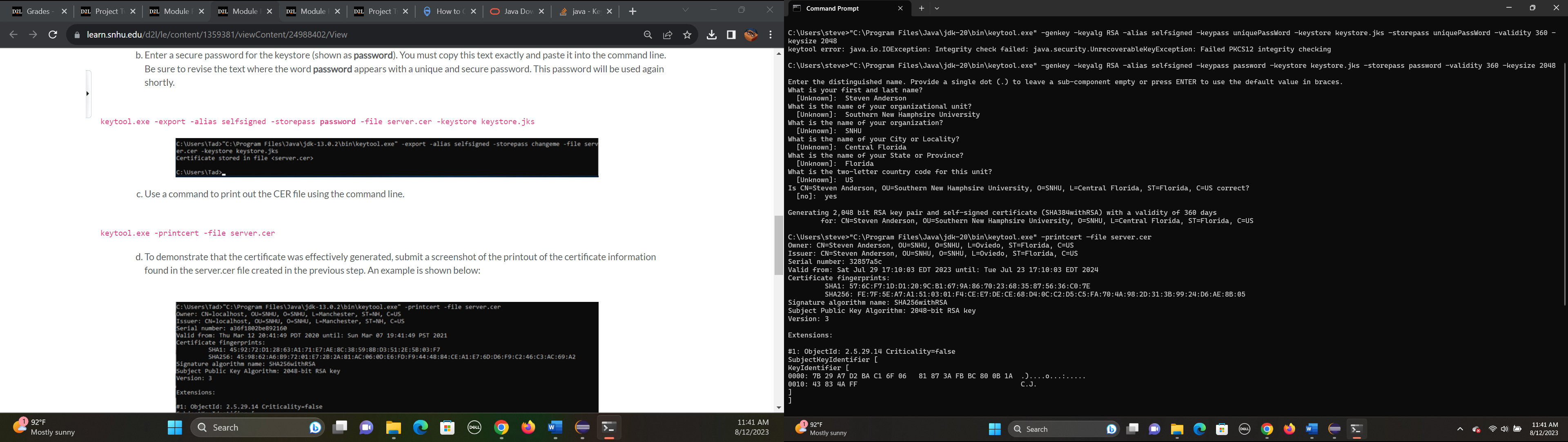
Steven Anderson

## Algorithm Cipher

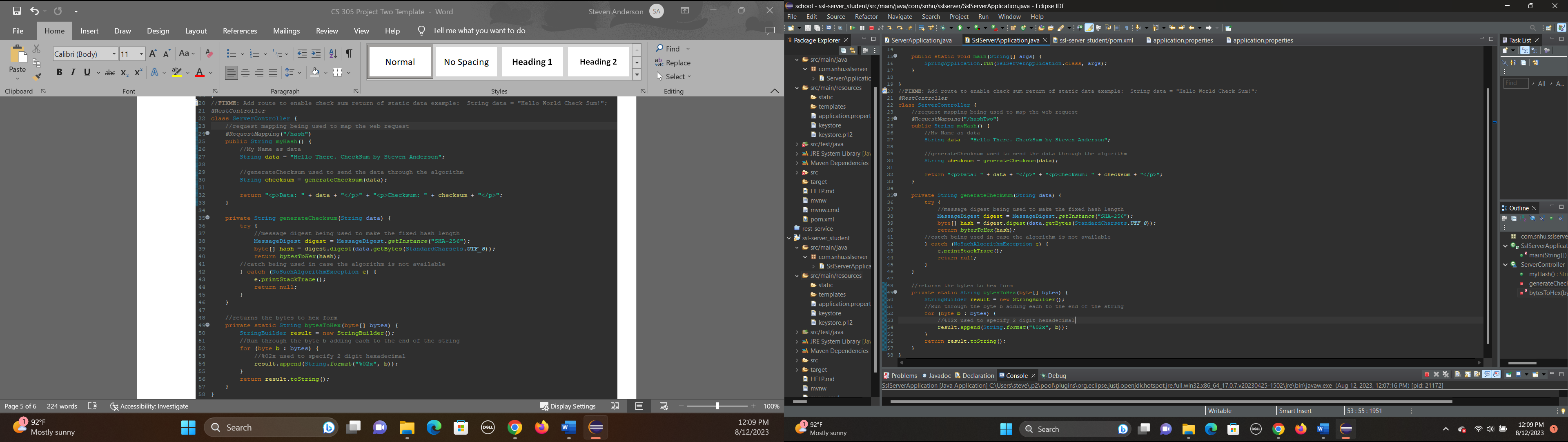
The algorithmic cipher that works best for Artemis Financial is an SHA-512 cipher. It is the industry standard in regard to banking and financial software security. It is a cipher that is so far unbroken and has no known security weaknesses. It also creates hash that in order to break would require checking all possible hashes, with which there are way, way more than even trillions of possibilities. It also creates these hashes by taking every character and converting them to their binary signature. Then taking those binary codes and breaking them down into 512 bits each with padding if needed. Then taking that and breaking it down into 32 bits each, then comes 64 rounds of additional compressions. Finally, what is outputted from that is the new hash values.

Though the client could also use SHA-256 but 512 is better for modern 64-bit systems to use. And also, they could choose to use an Asymmetric encryption method but I would not recommend it since it would slow down the system with the increased time. The Symmetric encryption method relies on one key for sending and receiving and works better when you have more data being sent in bulkier numbers. The Asymmetric encryption since it uses its more secure but complicated 2 key system where there is private and public keys is great for security but each data thread sent would take longer and with more data being sent it could bog down with bulk data sending.

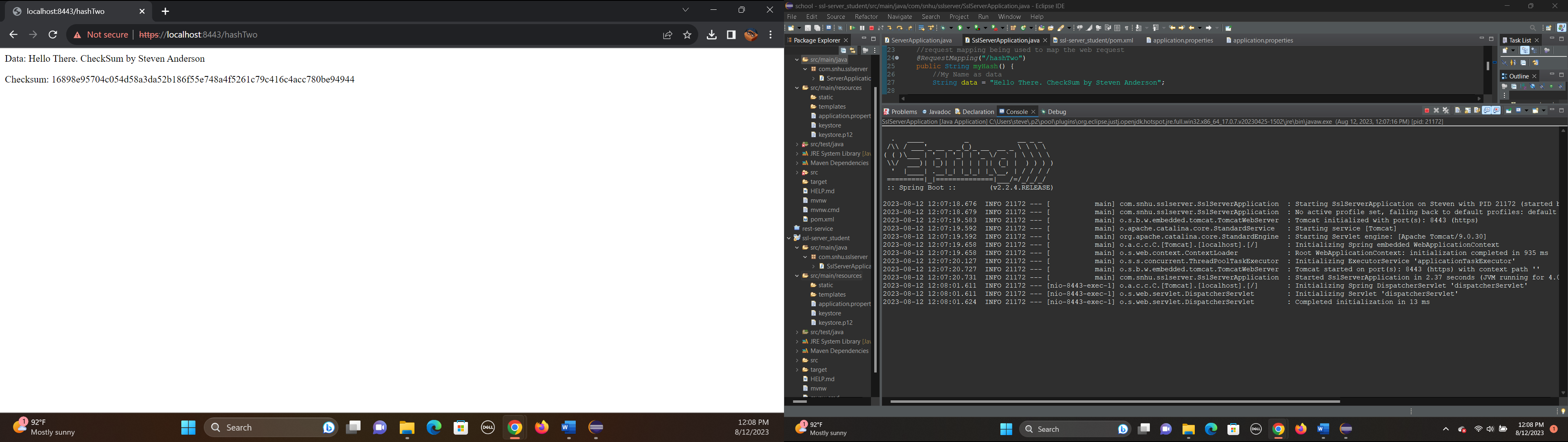
## Certificate Generation



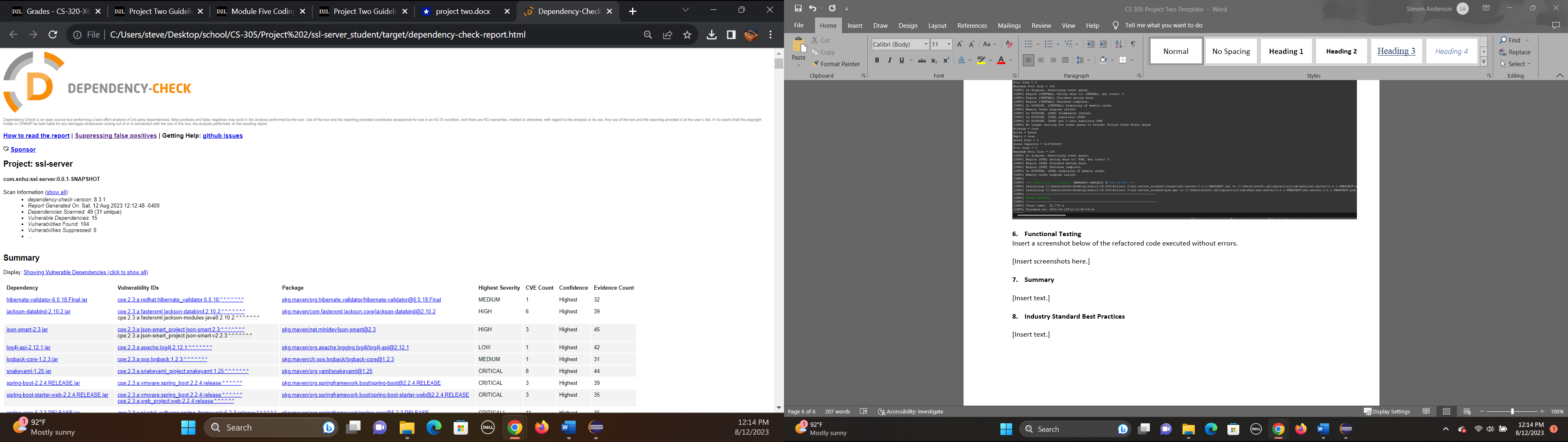
## Deploy Cipher

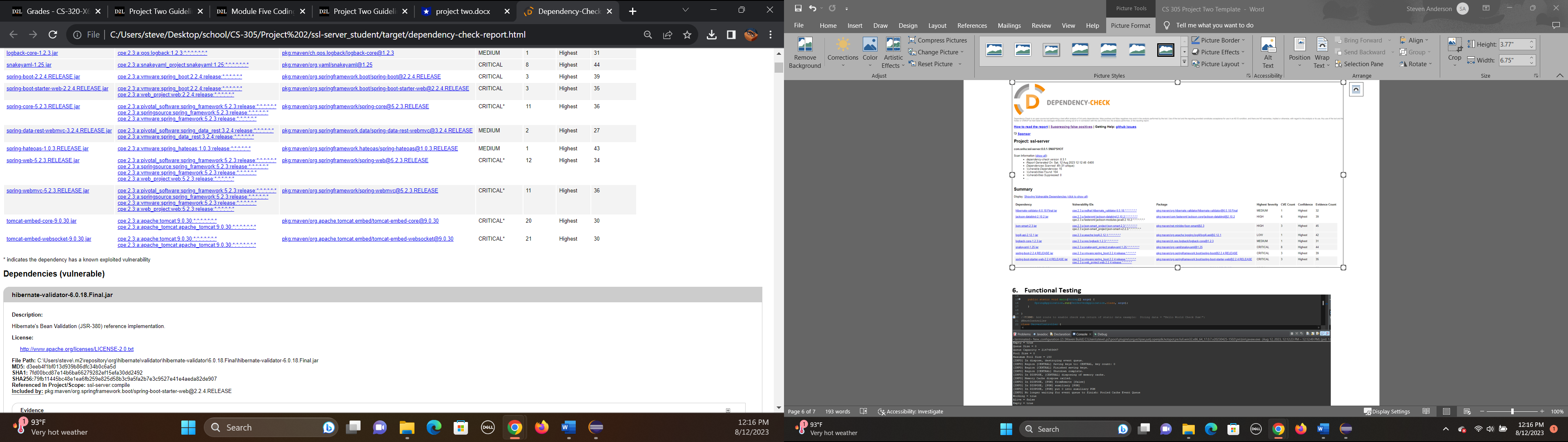


## Secure Communications



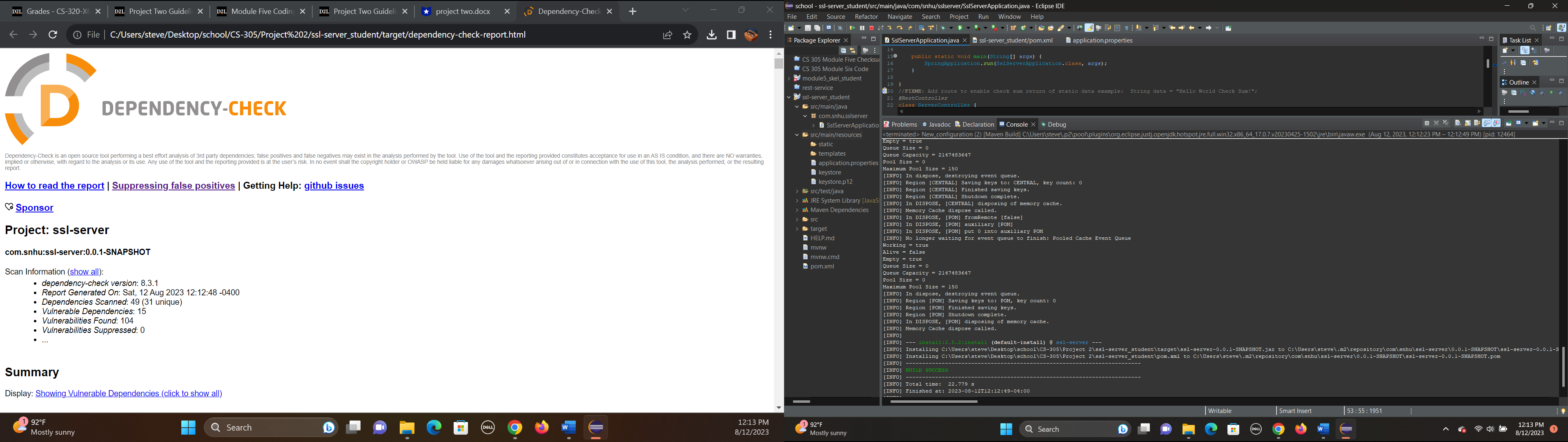
## Secondary Testing





­­

## Functional Testing



## Summary

The code has been adjusted to make sure it was more secure and followed the vulnerability flow chart. Such as following the cryptography process by using the encryption to encrypt the message in hash. This hashing step also applying to the next client/server step along with the web accessed secure connection. Then checking the code over to make sure it was clear of errors is the next part followed by making sure what I added or changed to the code followed the industry standards with in line comments and proper spacing.

Additionally, when I was adding the certificate that was used as another form of tightened security and helped cover the vulnerability of the client/server communications. Also, the hashing of the values being used by the program applied to the input validation step by helping keep all inputs secure.

And finally, the Dependency report can be used to do the manual review when it comes to plug-ins, services, and API’s. Since it can help get a good idea for anything being outdated or bogging the system down.

## Industry Standard Best Practices

When writing and changing the program I made sure to use the common inline comments and spacing to keep up with industry standards and also reviewed my code to make sure object and value names followed common naming forms. I also made sure that additional files had appropriate names and locations within the file system. This is important for anyone who wants to review the program after me. I also saved frequently and had the final program saved in two locations until submitted to prevent possibly losing it.

Another key industry standard when pertaining to securing software is to use the encryption on the data to prevent outside access being easier.

I also used some of the industry standards I’ve learned over time like not to repeat myself, meaning not to write code that just does the same thing as another section. Always keep a focus on readability and make sure your code can be read and used by others later. And don’t overuse a single variable too much.

The company Artemis Financial will need to follow some of the common standards moving forward to keep everything running well. This will include following security protocols to prevent access to the system from things like viruses or phishing attacks. Also making sure that the program goes to routine maintenance and upgrading to ensure that the plug-ins and security are kept up to date in case of new threats.