****

# 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** | **12/15/2024** | **Ariel Baez** |  |

## Client



## Developer

Ariel Baez

## Algorithm Cipher

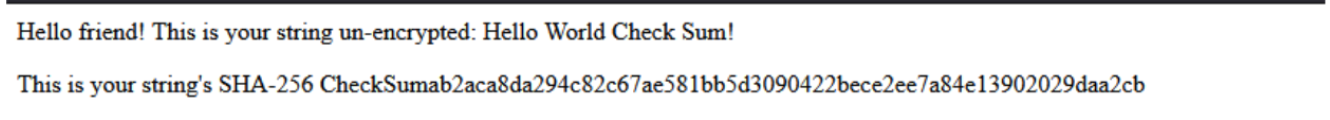
When looking at Artemis data we need to make sure to keep it as safe as possible. In order to keep the information safe using AES128 encryption is best and make sure to have CTR mode on it. Since the system is symmetric, we need to make sure the keys are as secure as possible. RSA is the best way to make sure that the keys are protected and do not get leaked.

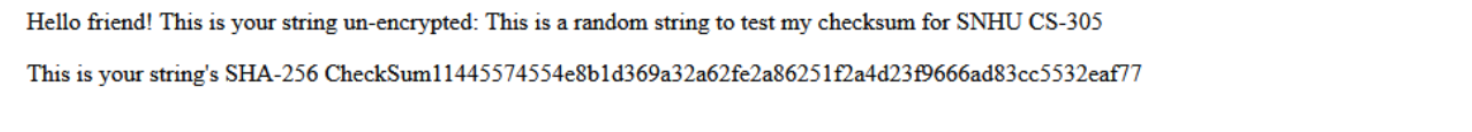
## Certificate Generation

In the screen sot I used Java Keystore tool in the SDK

## Deploy Cipher

We made sure to use the APIU to checksum for the strings, and is using SHA 256.



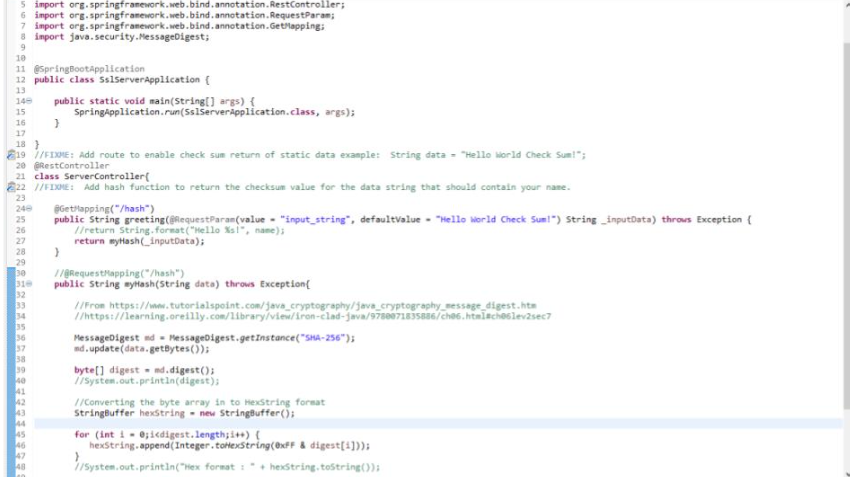


## 

## Secure Communications

As we look at the code we need to make HTTP to HTTPS. We need to run to code and verify that it is secure with localhost:8443/hash to make sure that we have good communication. There are many issues, but I was never able to make it secure.

## Secondary Testing



## 

## Functional Testing

## 

## The API is not sanitized. When the users put in a value it is passed to be hashed. Multiple API need to be slowed to prevent DOS. We also need to prevent buffer overruns.

## Summary

The code overall was to make sure that web traffic was sent to HTTPS. This makes sure that websites don’t get information leaked to anyone that may be trying to hack the network. We made sure it was secure with AES-128 using a personal certificate and it needs to be enforced with a root-type certificate. The API returned a checksum, this is to make sure that we get fingerprints of a digital object. When the hash is created correctly it needs to not have collisions, someone needs to validate the file and make sure it’s not manipulated. The check is SHA-256 which has a low accidental collide. Collisions would allow us to spot spoof of the check or reverse the original file. The code was passed through static analyzer to check for CVE. The input length is between 1-25 characters to prevent overrun attacks.