****

# 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** | **4/15/2023** | **Alex Casanova** |  |

## 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

Alex Casanova

1. **Algorithm Cipher**: My recommendation for Artemis Financial is the use of AES-256 for their encryption algorithm.
   1. Consider security protection best practices to defend against various types of security attacks.
      1. Evaluating and Monitoring of 3rd Party Vendors – Ensuring that application developers and other 3rd party vendors follow rigorous security standards (at least those being used within your organization) ensures that the risk of attacks being carried out through systems
      2. Continuous Threat Monitoring – When developing applications for business customers, it’s important to build in controls for internal access to threat logs, monitoring and incident response plans. This allows for customers to ensure their applications are and remain secure over time, while allowing our team to make changes as necessary when threats arise, or as assessments of customers’ security require changes to be made.
      3. User Education – Ensuring that users know how to use an application, whether it be from the internal business team or its customers, ensures that the functionality of the application is maximized for customers. This training also ensures that users are in the know about security practices, even if it’s at a cursory level for end users. The development of a help page with instructions on how to use the application in a secure way can help to reduce threats such as social engineering.
      4. Incident Response Plans – The development of Incident Response plans can allow business customers to adaptively react to threats as they come up. This can include plans for identifying, eliminating and recovery from threats, as well as actual attacks. These response plans can help customers feel more secure in their knowledge that their application is safe to use in a business setting, and will likely be required for financial institutions of Artemis’ nature.
   2. Consider and identify all the risks in your recommendation.
      1. Data breaches from insecure application
         * If our application is not developed in a secure way, it could allow attackers to access customer data. Obviously, this would be a disaster for Artemis Financial, as the perception of insecurity can be a death sentence for financial institutions. It’s also important to consider the rapport Artemis has with our development team. If Artemis chooses to run security audits of our software (which any sensible financial institution should want to do anyway) and detects security vulnerabilities, they will expect us to fix them, and can damage our team’s reputation as a company worth doing business with in this setting.
      2. Access to Internal Systems
         * If an error in our software allows attackers to access internal systems at Artemis, irreparable damage could be done to the institution and its internal tools. This could also allow attackers to access customer data in a number of ways, which all need to be accounted for when developing software in a secure environment like this one.
      3. Insider threats
         * When developing software for use in any business setting, it’s also important to incorporate security controls for insiders, as they are often more capable of extracting sensitive information from internal systems than outside attackers. Controls for data exfiltration, including user and role-based access to systems, least-privilege access to systems, is crucial in ensuring the utmost security for institutions.
   3. Consider the most current government regulations and how they will be met.
      1. Security protection is of the utmost importance for financial institutions. The threat of attacks from outsiders is ever-present, and steps must be taken to protect sensitive information from these attacks. The US Government uses FIPS 140-2 to assess levels of security taken by organizations, on a scale from 1-4.
         * 1. No physical security, but the use of encryption algorithms.
         * 2. The system provides evidence of tampering if it occurs. This includes things like physical seals and logs.
         * 3. The system actively thwarts attackers, by modifying or deleting the cryptographic keys if an attack is detected.
         * 4. The system detects physical attributes such as voltage, temperature of a physical system, and can take action to destroy cryptographic information if a breach is detected.

These security levels are great for physical systems, and for financial systems, at least level 3 should be taken to secure physical systems, such as servers, banking terminals, and storage media. The use of full disk encryption, along with level 3 security controls, especially the use of encryption algorithms in the case of a digital application, need to be incorporated as a rule for financial systems.

* 1. How will this algorithm cipher be used?
     + - Algorithm Ciphers are designed to be used to encrypt data, through the use of encryption keys. These keys are meant to be long strings of characters, in order to be nearly impossible to guess without prior knowledge of the key. This ensures that only authorized parties are able to decrypt sensitive data.
  2. What is the best cipher, and why?
     + - While there is never a one-size-fits-all cipher, developers need to consider the “best” cipher for the task at hand. Larger ciphers take longer to decrypt, as ciphers rely on computational work to decrypt data. Smaller ciphers are technically less secure, but may allow developers to scale applications more efficiently. A decryption cipher taking a second to decrypt a personal database for a small project infrequently may not be much of an issue, but when companies expect to run millions of decryption operations as their business scales to global use, this is not feasible, as millions of seconds to run through a set of decryptions will lead to incredibly slow access to data for customers. Instead, the use of smaller ciphers exponentially reduce the computational workload required for decryption.
  3. What are the reasons why you might not choose the most secure cipher?
     + - As mentioned previously, the theoretical most secure cipher would be infinite in length, and would take forever to decrypt, even with the key. Instead, the use of smaller, more manageable ciphers allows for faster decryption of data, at the cost of reduced security. Even the use of AES 256-bit encryption would take longer than the age of the universe to brute-force without having the key, and stronger ciphers exist. Sometimes it is not necessary to add additional cipher security, such as in this case, since the chances of finding the key by brute force would already likely see the attacker die of old age before it was cracked.

1. **Justification**: You have been asked to give a brief overview of the encryption algorithm cipher to justify your reasoning for recommendation to Artemis Financial. Consider the following points:
   1. What is the purpose of the cipher's hash functions and bit levels?
      1. A cipher’s hash function is the algorithm used to generate encrypted data from raw, unencrypted data. Various hash functions exist, with differing levels of security. The current most used standard is AES-256, which takes raw data and encrypts it using a 256-bit cipher key. This bit level is the number of bits long the key is, and higher bit levels mean exponentially more security. Since bits are either a 1 or a 0, each bit used doubles the complexity of the key. One bit has 2 possible states, two bits has 4, then 8, and so on. 256 bits of complexity leads to 2^256, or 1.16x10^77 possible values. This is what gives ciphers their strength, as guessing that many combinations would take a absurdly long time, while having the key requires only one computation.
   2. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
      1. Random numbers are often used in ciphers, as they allow for the highest number of possible combinations in the smallest key.
      2. Symmetric keys use the same key to encrypt data as to decrypt it. This means that one key is responsible for the “lock” and “key”. On the other hand, systems with non-symmetric keys use different keys to encrypt and decrypt data.
      3. Performance – Symmetric keys are often smaller and less complex than asymmetric keys. This means lower resource use, and faster encryption than on Asymmetric systems. Common Symmetric Encryption methods include AES, the method recommended for this application. Common Asymmetric methods include RSA, a 2048-bit encryption method. Since the bits allow for exponential increases in computational requirements, RSA-2048 takes much longer to compute, and should not be used in real-time systems where time performance is important, such as in mobile banking.

## Certificate Generation

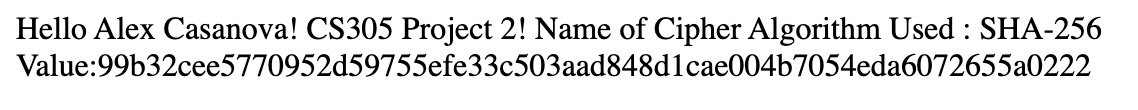
Insert a screenshot below of the CER file.

Text

Description automatically generated

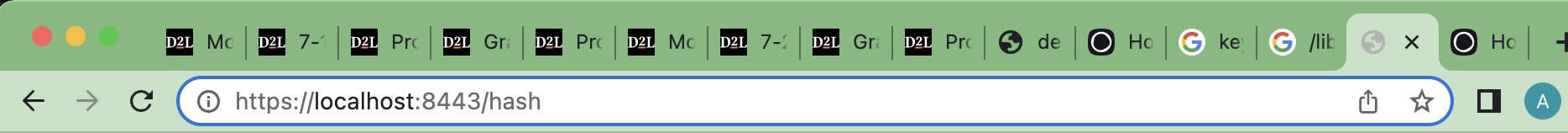
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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



## Secondary Testing

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

Graphical user interface, text

Description automatically generated

## Functional Testing

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

Text

Description automatically generated

## Summary: Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:

## Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.

## APIs – We used the Keytool API to generate certificates

## Cryptography – We used the SHA hashing algorithm to secure our certificates and used a checksum for verification.

## Client/Server – We used a web client to access the Web Service, which allowed us to pull the checksum for certificate verification.

## Discuss your process for adding layers of security to the software application.

## Using the Keytool API allows us to quickly and easily generate secure certificates with specified parameters.

## Using cryptography, we can ensure that our site is secure, by transmitting encrypted data rather than unsecured data.

## Using the client/server infrastructure allows us to store data securely on the server side, without unencrypted access to the data on client machines.

## Industry Standard Best Practices: Explain how you applied industry standard best practices for secure coding to mitigate against known security vulnerabilities. Be sure to address the following:

## Explain how you used industry standard best practices to maintain the software application's current security.

## Using SHA, a commonly-used encryption algorithm in web development, we can ensure compatibility with modern web browsers, many of which flag users for non-HTTPS traffic.

## Explain the value of applying industry standard best practices for secure coding to the company's overall wellbeing.

## Using industry best practices for secure coding ensured compatibility with other services who may require HTTPS. This includes many modern web browsers, so choosing not to use HTTPS can negatively affect the user experience. In addition, non-HTTPS websites are less secure, and many browsers let users know that sites that don’t support HTTPS are a security risk, and should be used with caution. The use of software security also means the data customers send and receive from the site is transferred in an encrypted fashion, ensuring that the data cannot be intercepted by attackers in a usable form.