****

# 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** | **10-11-22** | **Kai Glass** |  |

## Client



## Instructions

Submit these completed practices for a 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

Kai Glass

## Algorithm Cipher

When considering which encryption cipher to use for Artemis Financial we must first consider security protection best practices. In chapter 8 of Iron-Clad Java, the book goes over some of the best practices to use when determining a cipher, the next 6 rules are listed in the book. “Don’t use suites that list ANON for authentication. They don’t provide authentication. Don’t use suites that contain NULL. Avoid the use of suites that contain EXPORT. Stick to bulk ciphers with key sizes of 128 bits or larger (note that 3DES provides no more than 112 bits of security). Try to avoid suites using RC4, DES, and 3DES. Prefer ECDHE and DHE for key agreement. While they are slower (and DHE is slower than ECDHE), they provide stronger protection even if the private keys are later compromised, a property known as forwarding secrecy. (Manico et al., 2015)” With those rules in mind are our decision being from the Java standard name document, I recommend that Artemis Financial use AES. The rules say to avoid RC4 and DES, so 7 options can be canceled from that list. The blowfish encryption algorithm can utilize bits as low as 32 and the rule states that we should stick to encryptions with 128 bits or higher. Therefore, we can cross blowfish off the list as well. That leaves us with only a few options left. AES looks the most promising because it is a 128-bit block cipher that can support keys of higher bits. The current government regulations for AES are an advanced encryption standard that must be used with modes of operation designed specifically for block cipher algorithms. The AES encryption has also been approved by the U.S. government to protect sensitive information (Computer Security Division). This shows that we are using the encryption cipher for the right purpose which is protecting financial information. One reason to not choose the most secure cipher is that it may not be user-friendly with the company or customers it is supposed to be operating with. Therefore, AES encryption is the smartest choice for Artemis Financial.

A hash value is a numerical capacity that changes over mathematical info and is compacted into another mathematical worth. Hash values are the most useful in password storage, to protect passwords. The purpose of a bit level of a cipher is that shows the level of efficiency of a cipher. A 128- bit level cipher is secure and difficult to discern the original code. A 256- bit cipher is a few levels harder to get back to the original code. For the case of Artemis Financial, a 128- bit encryption is sufficient because it is more than adequate for the level of security that is needed for the company.

In cryptography, the use of random number generation is important because it removes all predictability and reasoning of the codes. If there are no patterns, are cipher is harder to crack. A symmetric cipher utilizes a private key to encode and decode messages. It requires only one key for each process. Asymmetric encryption utilizes a public key to scramble the message and a private key to unscramble it. In the AES case, it is a symmetric cipher and will be more user-friendly for Artemis Financial.

Encryptions go back to the spartan era when they used a device called a scytale to send secret messages during battle. A scytale is a leather strap wrapped around a wooden rod. The message would only reveal itself when the rod is the correct size. Fast forward a few hundred years, we have roman encryption in which the Caesar cipher was created. During the world war 2 era, the Enigma machine is created to decode ciphers. All of these encryption methods were effective for their time but as technology advances, so does encryption technology. Since the 2000s more and more encryption methods have hit the web. In 2000 the Advanced Encryption Standard was replaced by DES. PKI, public key infrastructure, also surfaced. In 2005 the Elliptic-cure cryptography surfaced with a more advanced public-key scheme that allows for shorter encryptions. As new technology and malware are developed, encryption methods must also start to adopt new techniques.

References

*A brief history of encryption (and cryptography)*. Thales Group. (2021, October 1). Retrieved September 30, 2022, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

Computer Security Division, I. T. L. (n.d.). *Block cipher techniques: CSRC*. CSRC. Retrieved September 30, 2022, from https://csrc.nist.gov/projects/block-cipher-techniques

Java security standard algorithm names. (n.d.). Retrieved September 30, 2022, from https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names

Manico, J., Detlefsen, A., Kenan, K., & Smith, M. (2015). *Iron-clad java: Building secure web applications*. McGraw Hill Education.

## Certificate Generation

Graphical user interface, text, application

Description automatically generated

## Deploy Cipher

Text

Description automatically generated

## Secure Communications

Graphical user interface, text, website

Description automatically generated

## Secondary Testing

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

## Functional Testing

Graphical user interface, text, application

Description automatically generated

## Summary

In this project one of the first steps in the vulnerability assessment that we address is cryptography. The first step that we take in this project is to deploy a secure cipher and do a checksum report to show validation. The next point we hit on is APIs, we create a self-signed certificate for our code base which will secure API interactions. The next security aspect we improve is the client/server by refactoring the code to a more secure web browser HTTPS. The next two steps we did in this project are secondary and functional testing, these two steps hit four different security points. It touches on cryptography again because we are suppressing false positive vulnerabilities. It also touches on code errors and code quality as we examine the code for any errors. Finally, it touches on encapsulation because we are confirming that we are using secure data structures.

## Industry Standard Best Practices

The way I used industry standard best practices to maintain the software’s current security is by first looking up what the coding standards are for java security. When researching I came across CWE’s top 25 most dangerous software weaknesses. I kept this list up when I was doing my project to make sure I wasn’t making any of the most common errors. By doing this I was able to help the company’s overall well-being by keeping the code base secure and free from simple security errors.