Southern New Hampshire University

7-1 Project Two Submission

David Vega

Dr. Vivian Lyon

CS-305 Software Security

02/19/2023

****

# Practices for Secure Software Report

Table of Contents

[Practices for Secure Software Report 2](#_Toc127720202)

[Document Revision History 4](#_Toc127720203)

[Client 4](#_Toc127720204)

[Developer 4](#_Toc127720205)

[1. Algorithm Cipher 4](#_Toc127720206)

[2. Certificate Generation 7](#_Toc127720207)

[3. Deploy Cipher 8](#_Toc127720208)

[4. Secure Communications 8](#_Toc127720209)

[5. Secondary Testing 8](#_Toc127720210)

[6. Functional Testing 10](#_Toc127720211)

[Refactored Code 11](#_Toc127720212)

[7. Summary 13](#_Toc127720213)

[8. Industry Standard Best Practices 14](#_Toc127720214)

[References 15](#_Toc127720215)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/19/23** | **David Vega** | **Initial draft** |

## Client



## Developer

David Vega

## Algorithm Cipher

The financial industry is one of the most regulated in the world. It is no surprise that governments would implement data security requirements because of the highly sensitive data that is being stored and accessed. One of these security requirements involve encryption around that data (Probasco, 2017).

The algorithm cipher that is recommended to meet the needs of Artemis Financial is the Advanced Encryption Standard, also known as, “AES” algorithm cipher. The bit level to implement is the 128-bit level. I chose this cipher for several reasons. One is because it is the most widely used cipher in the financial industry. Another is the cipher is still used in encrypting highly sensitive and classified government data. AES-128 uses the mode CBC (*“Cipher Block Chaining”*) which takes in input data and divides the data into a 4x4 “blocks” containing 16 bytes. A single byte contains 8-bits of data. The total number of bits in each block is 128 bits. AES is also a symmetric encryption algorithm which means that the same private key can be used to encrypt and decrypt data. In contrast, an asymmetric algorithm such as RSA named after its creators (Rivest-Shamir-Adleman) uses a private and public key for encryption and decryption. While RSA can be used to encrypt and decrypt data, it is much slower to process than AES. Therefore, RSA is commonly used in encrypting/decrypting keys and not large amounts of data (Devi & Harika, 2019).

Another encryption method would be to use a hashing algorithm. While it is certainly reasonable to use RSA to encrypt an AES key it is also reasonable to use a hashing algorithm instead. A hashing algorithm is a one-way encryption algorithm which is primarily used to validate the integrity of the data being transmitted rather than to encrypt or decrypt raw data. A hashing algorithm such as SHA-256 can be used to “mask” a key so that it would be virtually impossible to reverse engineer the hash to reveal the encrypted AES key. It is also important to note that SHA-256 hashing algorithm is collision-resistant. A collision occurs when two different pieces of data result in the same hash value. This is not ideal as a hash value should be unique and should change with any modification to the data. Attackers can use this exploit to fraudulently claim that data has been appropriately authenticated and verified. Additionally, you can implement a combination of the above referenced methods for added security. An organization such as Artemis Financial can benefit from using the AES-128 encryption cipher to encrypt/decrypt the data, a SHA-256 algorithm to hash the private key and RSA to encrypt the hashed key (N-able, 2019).

The AES encryption method was developed to replace the previous Data Encryption Standard or “DES” because it had been susceptible to brute force attacks. These attacks inevitably led to the DES algorithm being cracked by attackers in less than 24 hours. The U.S. government saw an opportunity to improve the standard. To do so, they announced a public contest to see who can develop the next improvement of the encryption standard and ultimately selected the Rijndael cipher named after its creator’s Belgian cryptographers Vincent Rijmen and Joan Daemen. It was renamed to Advanced Encryption Standard (AES) and adopted by the National Institute of Standards and Technology (NIST) in 2002. Although AES-256, a variant of AES, is an even stronger cipher, AES-128 has not been cracked and still provides sufficient protection against attacks. AES-256 also consumes more resources because of the size of the block cipher at 256 bits. Finally, a hacker would need to sift through 2128 different combinations to crack AES-128. By contrast, it would take a hacker 2256 different combinations to crack AES-256. These are astronomically large numbers and would take a lifetime to sift through. Hackers know that this takes an incredible amount of time and resources so they will most likely try to find other vulnerabilities to exploit. As computing devices become more sophisticated, it is critical to keep up with standards and ensure that security professionals are implementing these standards appropriately across applications. ­­­­­­­­­­­­ (Daniel, 2021)

## Certificate Generation

Insert a screenshot below of the CER file.

Figure 1: Key Generation

Graphical user interface, text

Description automatically generated

Figure 2: Key Export



Figure 3: Print server.cer

Graphical user interface, text, application

Description automatically generated

Figure 4: Copy of server.cer

A picture containing text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Figure 5: Secure HTTPS Checksum

Graphical user interface, application, website

Description automatically generated

## Secure Communications

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

Figure 6: Secure Connection

Graphical user interface, text, application, Excel

Description automatically generated

## Secondary Testing

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

Figure 7: Pre-refactor maven install – NO ERRORS

Text

Description automatically generated

Figure 8: Pre-refactor dependency check report.

Graphical user interface, text, application, email

Description automatically generated

Figure 9: Post re-factor maven install

Text

Description automatically generated

Figure 10: Post-refactor dependency check report.

Graphical user interface, text, application

Description automatically generated

## Functional Testing

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

The application’s initial code base was missing some libraries to implement a RESTful architecture, create hash (digest), and some error handling. The applications.properties file was also missing server configuration that would provide a secure connection (HTTPS).

Figure 11: SslServerApplication.java - Missing Libraries

Graphical user interface, text, application, email

Description automatically generated

Figure 12: application.properties - Missing server configurations

Graphical user interface, text, application, email

Description automatically generated

# Refactored Code

The below libraries support the necessary functions to implement RESTful routing as well as creating a message digest from pre-defined input. The final import is used in error handling to “throw” an exception when the hash algorithm is invalid.

Figure 13: SslServerApplication.java – Libraries imported.

Graphical user interface, text

Description automatically generated

The below ServerContoller class was added to the SslServerApplication.java file to handle the RESTful endpoint (“/hash”) and message digest (hash algorithm).

Figure 14: ServerController class code in SslServerApplication.java

Graphical user interface, text, application

Description automatically generated

The code below shows the configurations needed for the implementation of the HTTPS protocol in the application.

Figure : application.properties HTTPS server configurations

Graphical user interface, text, application, email

Description automatically generated

## Summary

In reviewing the initial code base there were some identified areas of security that were addressed post-refactoring. The areas of security that were addressed were Input Validation, APIs, Cryptography, Client/Server, Code Error, and Code Quality. Input validation was handled in the myHash() function that checked for a valid hash algorithm (“SHA-256”). This function also produced an exception when anything but “SHA-256” was detected which also addresses code error. The message digest method that is part of the java.security library provided a way to convert the pre-defined string from bytes into hash values. Regarding APIs and client/server, I’ve implemented a RESTful architecture. REST (Representational State Transfer) is an architectural style for streamlining communication between a client and server via the hypertext transfer protocol (HTTP). The idea behind RESTful web services is that they allow for client-server autonomy. Both client and server operate independently of one another which enables consumers and providers of APIs to manage their respective systems without it impacting each other. A uniformed interface also makes a RESTful architecture appealing as it doesn’t matter what stack the client or server are using, the information required and provided for the request and response is the same. Finally, the RESTful architecture has a layered structure allowing for each device to work independently of one another which makes it scalable. Theoretically, you can place a proxy server or load balancer between the client and server and the client will only need to worry about the proxy and the proxy only needs to worry about the server and vice versa. Implementing security as a separate layer will only enhance system safety (altexsoft, 2022). With secure coding, I made sure to initialize my variables to “null” as a way to identify that my code was returning what I was expecting. Also, having the server configuration as a separate file was a good way to organize my code and isolate server misconfigurations. In securing the connection to the server, I generated a self-signed certificate using the keytool command line interface. This certificate will help to verify that the server is a trusted source.

## Industry Standard Best Practices

Artemis Financial had some requirements that we had to consider since they are in the financial services industry which is highly regulated globally (Probasco, 2017). These considerations are regulatory laws, both domestic and foreign, that will need continued support to accommodate any changes in legislation. In addition, resources such as NIST (National Institute of Standards and Technology) are extremely useful as they provide a wealth of knowledge and guidance for implementing security across all technological systems. NIST provides tools like the National Vulnerabilities Database which allows you to search for and view known exploited vulnerabilities (NIST, 2021). In addition, OWASP (Open Web Application Security Project) is an organization that through community-led open source projects is devoted to improving the security of software. They provide a tool called the OWASP Dependency Check that a developer can install to analyze the dependencies of an application to generate a report of any known vulnerabilities (OWASP, 2023).

In conclusion, it goes without saying that software security should go hand in hand with software development. It is very difficult, time consuming, and potentially costly to consider security as an afterthought in software development. Security needs to be at the start of the software development lifecycle (SDLC).

# References

Devi, S., & Harika, D. K. (2019). AES encryption and decryption standards.*Journal of Physics: Conference Series, 1228*(1)https://doi.org/10.1088/1742-6596/1228/1/012006

altexsoft. (2022, November 19). *REST API: Key Concepts, Best Practices, and Benefits*. Retrieved from altexsoft.com: https://www.altexsoft.com/blog/rest-api-design/

Daniel, B. (2021, March 31). *What is AES Encryption? [The Definitive Q&A Guide]*. Retrieved from trentonsystems.com: https://www.trentonsystems.com/blog/aes-encryption-your-faqs-answered#:~:text=128%2Dbit%20AES%20encryption%20refers,not%20top%2Dsecret%20government%20information.

N-able. (2019, September 12). *N-ABLE*. Retrieved from n-able.com: https://www.n-able.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20a%20patented,as%20long%20as%20when%20unencrypted.

NIST. (2021, October 1). *National Institute of Standards and Technology*. Retrieved from nist.gov: https://www.nist.gov/

OWASP. (2023). *OWASP Dependency Check*. Retrieved from owasp.org: https://owasp.org/www-project-dependency-check/

Probasco, L. (2017, April 25). *Townsend Security Data Privacy Blog*. Retrieved from townsendsecurity.com: https://info.townsendsecurity.com/encryption-requirements-for-banks-financial-services