****

# 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** | **04-19-2024** | **Nicholas Rusinski** |  |

## Client



## Developer

Nicholas Rusinski

## Algorithm Cipher

Artemis Financial is a consulting company that develops individualized financial plans for their customers, where protecting their clients' information from attacks both at rest and in transit is of the utmost importance. This data can take the shape of accounting information, personal identification information, and various other liabilities. As the Global Rain Service Developer at Artemis Financial, I was tasked with recommending a solution to help protect this data from attackers.

Cipher algorithms can be used to encrypt sensitive data from attackers attempting to access it leaving the data illegible for even the most advanced computers. These ciphers use an encryption key and an algorithm together to “scramble ordinary plain text into a cipher text that is impossible to read without the proper encryption key” (Marget, 2024). In actual use, a system will use an encryption method to disguise data as it is sent between server and client as well as protect data that is temporarily stored in system memory. Understanding this is crucial for web server management and development.

Another key aspect of cipher algorithms is the encryption key style such as symmetric and asymmetric keys. Symmetric algorithms use a single key to both decrypt and encrypt data to prevent unwanted access. This style is “one of the most widely used techniques” (Daniel, 2021.) as well as “one of the oldest, dating back to the days of the Roman Empire” (Daniel, 2021.). What symmetric ciphers lack in complexity they make up for with ease of use and are less resource-intensive. The other type is asymmetric, which requires the use of public and private keys to provide encryption and decryption services. Generally, these ciphers are also exponentially greater because the key size is typically larger. This also means that the encryption and decryption require the allocation of more system resources making them much slower to use. A general rule of thumb according to Daniels (2021) is that “symmetric encryption is faster and simpler but is often viewed as less secure than asymmetric encryption” (Daniel, 2021). Both of these algorithms have their uses and benefits and it's up to the developer to interpret the client's need and implement the right cipher for the job.

The Rivest Shamir Adleman (RSA) algorithm cipher is a recognized encryption method that has been adopted widely for its prominent level of security. RSA is an asymmetric cipher that requires the use of both encryption and decryption keys that range from 2048 bits to 4096 bits (Lake, 2024) that will be used to protect the data communications at Artemis Financial. The large bit keys help by providing an exponentially complex cipher to secure the data from potential brute force and integer factorization attack methods that are often used against asymmetric encryption methods like RSA (Lake, 2024). Another benefit of the RSA encryption algorithm is the use of random prime numbers to generate secure keys ultimately leading to large keys that are always odd. These large keys give RSA an exponentially broad encryption power to break.

A hash function can also be used alongside RSA to provide a digital signature of data that would notify the system of any modifications that occurred while the data was in transit (Lake, 2024). Another benefit to the RSA cipher is its asymmetric nature that requires the use of an encryption key to provide the encryption requirements and oppositely the use of a decryption key to decrypt the data. The decryption key is also known as the private key and is derived from the encryption key requiring another difficult mathematical problem to crack, adding another factor of security for sensitive data (2024). Overall, RSA provides a strong toolkit for protecting your data.

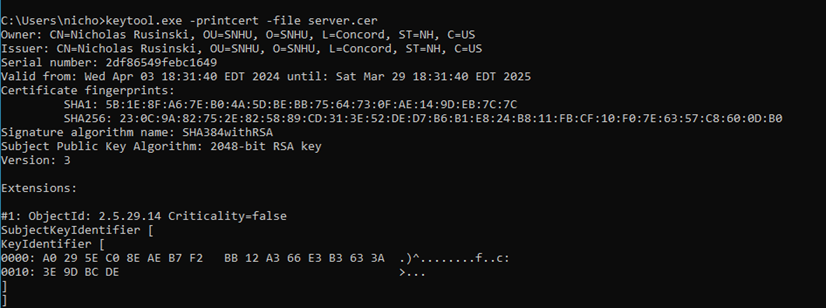
Choosing a cipher algorithm comes down to the level of security needed for the data you are defending. Protecting sensitive data that if in the wrong hands could be potentially damaging requires more complex ciphers and the implantation of additional filters to be used alongside them. With these additions, the system is becoming increasingly resource-intensive, which should be a high consideration when choosing a cipher. The National Institute of Standards and Technology offers standards for the use of cipher algorithms and implementation guidelines for software cyber security. NIST should be considered a good standard for any decision regarding secure system practices.

When it comes to what is the best algorithm cipher there is not just one answer. The application of ciphers is complex and all types of attacks and usages need to be considered. Encryption methods often work best when used in tandem with other security toolkits and in some cases even with other encryption methods. The highest chance of a secure application comes from understanding all the vulnerabilities, advantages, and weaknesses of your cipher. In the case of the RSA cipher, its wide adoption makes its strengths and weaknesses known and allows for solutions for these shortcomings to be easily found and implemented across the board.

In conclusion, RSA offers a complex algorithm that can be scaled to correctly protect the system and client's sensitive data, while not being too demanding. RSA also offers great constructive interaction with other security features and would be a great cipher algorithm choice for Artemis Finacial's new platform.

## Certificate Generation

## Certificate generation example.



The server.cer file used to install the certificate.

A computer screen with a certificate

Description automatically generatedA screenshot of a computer

Description automatically generated

## Deploy Cipher

## Screenshot shows the refactored code.

## A screenshot of a computer program Description automatically generated

## Checksum verification example with name.

## A screenshot of a computer Description automatically generated

## Secure Communications

## Refactored server properties file

## A screenshot of a computer Description automatically generated

## Eclipse project workspace showing the keystore located in the main resources directory.

## A screenshot of a computer Description automatically generated

## The screenshot showing HTTPS was successful. The red strike is because the browser is flagging that the certificate was self-signed and not from a valid CA.

## A screenshot of a phone Description automatically generated

## Screenshot verifying that the browser can successfully see the cert.

## A screenshot of a computer Description automatically generated

## Secondary Testing

## The initial test showed 18 dependencies with a total of 77 known vulnerabilities.

A screenshot of a computer error

Description automatically generated

To correct these vulnerabilities I updated the spring boot application to version 3.2.5. At first, this caused a build failure, to fix that I updated the Maven plugin to match a more recent version of the Java JDK. This allowed Maven to correctly install the most recent version of Spring Boot which fixed 76 out of the 77 known vulnerabilities. The final vulnerability was CVE-2023-35116 and was despised by the author with the NVD as not possible. The vulnerability was suppressed and declared as a false report.

Screenshot showing the spring boot update

A screen shot of a computer code

Description automatically generated

A screenshot shows the final Dependency check.

A screenshot of a computer

Description automatically generated

Screenshot of the NVD for suppressed vulnerability

A screenshot of a computer

Description automatically generated

## Functional Testing

## Error-free code screenshot.

A screenshot of a computer program

Description automatically generated

## Summary

To summarize the path taken to refractor the Rest application to be free of dependency vulnerabilities as well syntax and logical security risks.

The first step that was taken was to run a static dependency check to scan the architecture for potential vulnerabilities. This step was critical considering the outdated state that the provided architecture was supplied in specifically the Spring Boot application that provides the restful functionality. The original test came back with seventy-seven known vulnerabilities and was easily corrected by updating the Spring Boot application to the most recent version which removed all but one of the vulnerabilities. The last vulnerability has a history of being disputed with the NVD and declared as an unrealistic problem. I have added logic to the POM file to suppress this vulnerability as it is a false report.

Fixing the known vulnerability was the first step in the report because it ensured that the architecture of the application provided a secure foundation. This is also highlighted by the fact that architecture is the first stop on the Vulnerability Assessment Flow Diagram. Essentially, it is better to catch dependency vulnerabilities before the code is designed around a dependency that provides an unsecured environment for the application.

The next step was to install encryption and cryptography covering the next relevant topic of the Vulnerability Assessment Flow Diagram. The first action taken was to add the Message Digest packaged to provide one-way hash checksum verification. At this stage, we just introduced the hashing algorithm to create the checksum hashes. These hashes use the SHA-256 hash algorithm and can be used with further functionality to check data that has been sent for modification as well as to protect data like passwords from attackers.

Another example we implemented was adding a valid keystore certificate to the application. This allows the use of HTTPS communication between the server and the browser ensuring that data is properly protected when being sent back and forth between the client and host. One of the challenges encountered was understanding why the browser was flagging the connection as unsecured.

This was because modern browsers expect third party certificates that have been signed by certificate authorities. The use of these third parties is essential because without them anyone could create a certificate and trick you into believing the connection is secure when it is not.

These three examples summarize the direct improvements that were made to the client with the Vulnerability Assessment Flow Diagram. The application in the current version is simple and lacks functionality, but the three improvements we have made provide a secure foundation for the application moving forward. Future improvements to the application will require additional vulnerability assessments. Such as input validation to filter unwanted input for taking advantage of system design in the form of denial of service attacks or injection to steal sensitive data. Another aspect that will need to be implemented is secure error handling to catch logical errors and prevent system crashes causing denial of service.

Overall, the application is off to a good start and will provide a safe and secure environment to expand on with future projects and improvements.

## Industry Standard Best Practices

Several of the improvements made are considered to be industry-standard best practices. Some examples would be using a secure and trusted hashing algorithm such as SHA-256 to hide sensitive information from attackers. Another example is the use of a catch statement to handle any potential problems when generating the hash sum. The use of HTTPS protocol for transferring sensitive information is also considered to be best practice. In the case of the application, it provides a second aspect of protection for the data that is transmitted to and processed by the browser.

The value of applying these best practices is that it provides a secure foundation for the application from the start. By incorporating security best practices from the start you ensure the tools exist to create a secure process and it forces the secure mindset to be adopted by the project developers, essentially forcing them to design a secure application and not applying security as an afterthought. The use of best practices also ensures trust with your customers and helps build your reputation. Lastly, it helps to avoid costly mistakes and the unrecoverable loss of sensitive data.

References

Lake, J. (2024, March 5). What is RSA encryption and how does it work?. Comparitech. <https://www.comparitech.com/blog/information-security/rsa-encryption/>

Marget, A. (2024, March 14). Data Encryption: How It Works & Methods used. Unitrends. https://www.unitrends.com/blog/data-encryption

Types of encryption algorithms + pros and cons for each. Keyfactor. (2024, February 15). https://www.keyfactor.com/education-center/types-of-encryption-algorithms/ website 3

Daniel, B. (2021). *Symmetric vs. asymmetric encryption: What's the difference?* Trusted Computing Innovator | Trenton Systems. <https://www.trentonsystems.com/en-us/resource-hub/blog/symmetric-vs-asymmetric-encryption>