

# 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** | **April 20** | **Afshin E. Ahvazi** |  |

## 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

Afshin E. Ahvazi

## Algorithm Cipher

After considering multiple industry-standard encryption ciphers, we have concluded that AES (Advanced Encryption Standard) is the most appropriate to deploy for this project.

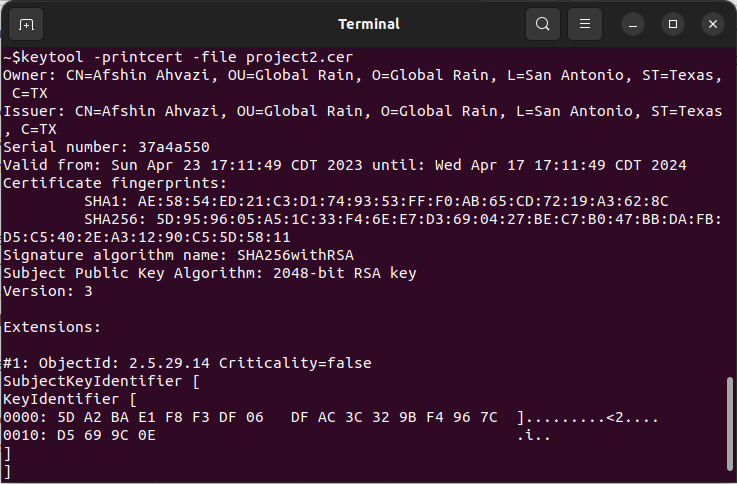
It started as a government project to protect sensitive data. Thus, since 1997, the NIST was actively holding competition among computer scientists to develop an efficient encryption algorithm that was royalty-free, implemented symmetric key cryptography, and supported various block and key sizes (such as 128-, 192-, and 256-bits). Finally, in 2001, AES, as proposed by a group of Belgian cryptographers, was finally approved and shortly after widely adopted across the software community.

AES borrows complex mathematical theories from the discrete mathematics and linear algebra (linear transformations and permutations) as well as from number theory (pseudo-random numbers). Specifically, a random number generator picks a random sequence of bits from a range of values defined by the encryption scheme used. A string representation of such a key could include letters, numbers, special characters, and other ASCII characters.

Furthermore, uses only one key to encrypt and decrypt the messages, which means it is a symmetric key algorithm. In contrast, non-symmetric encryption methods use two keys. A public key is used to encrypt, and a private key is used to decrypt.

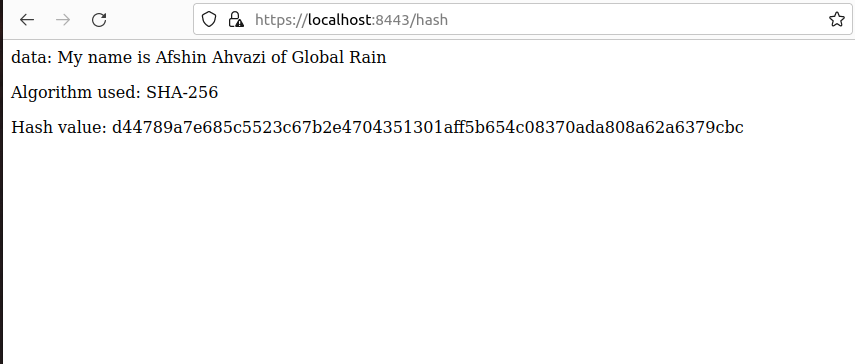
## Certificate Generation

The following screenshot shows the CER file containing the certificate created for this project.



## Deploy Cipher and Secure Communications

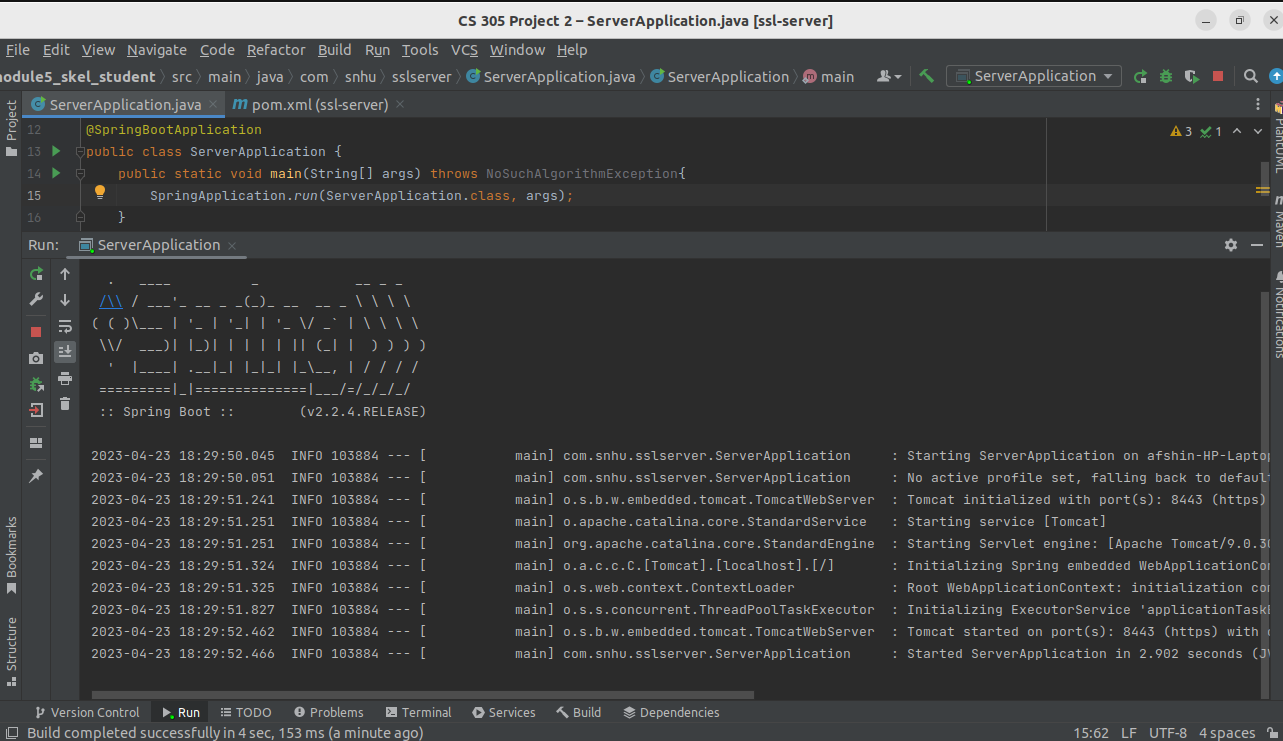
Here, we show the use of SHA-256 hashing algorithm used for checksum verification. In addition, we can see the use of secure communications (as evident by https in the address bar).



## 

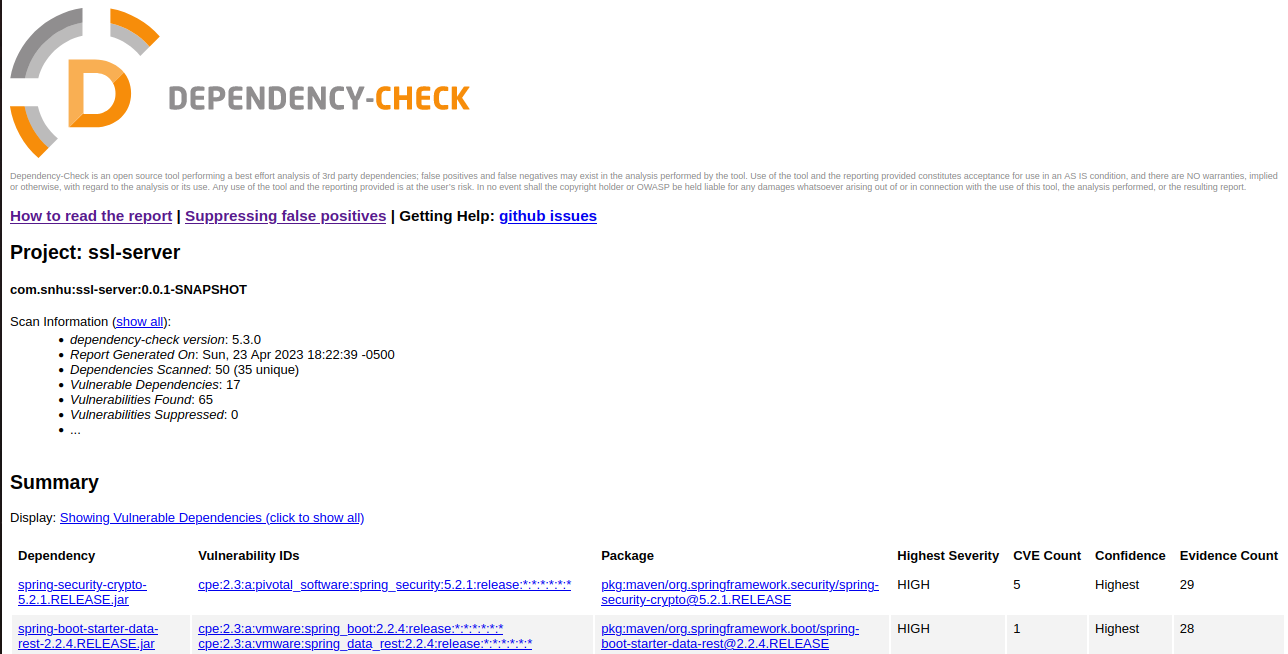
## Secondary Testing and Functional Testing

The following screenshot of the project code executed without any errors.

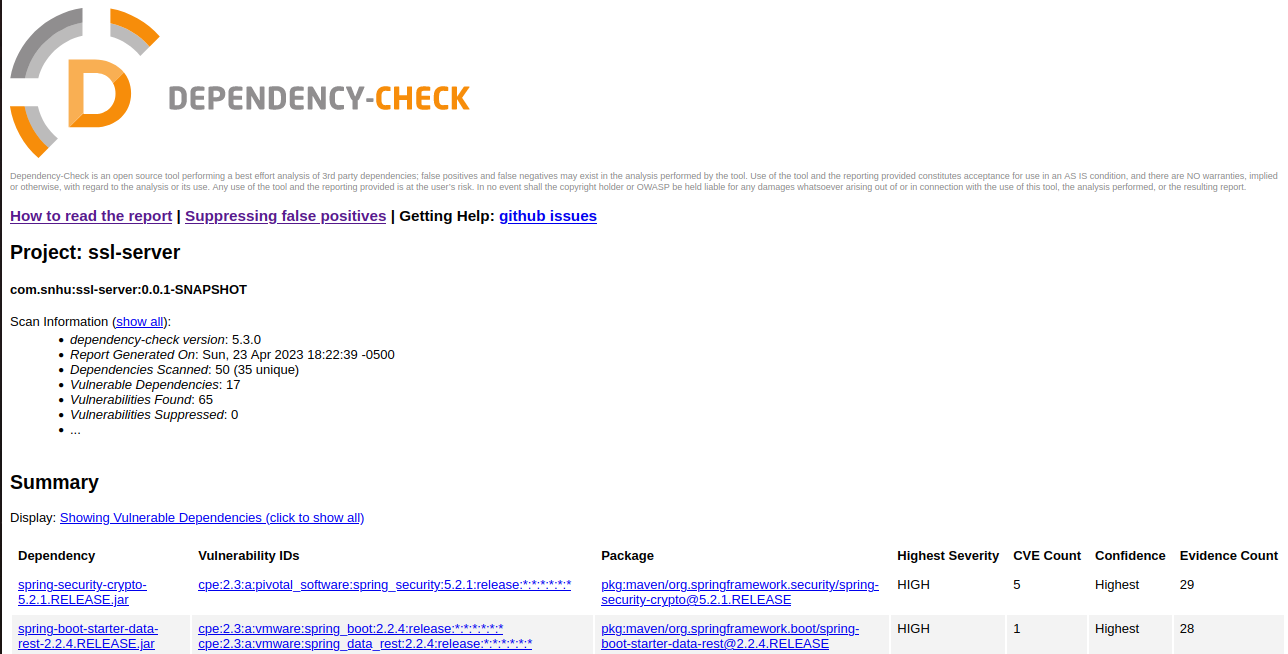


Furthermore, a comparison of the dependency check reports created before and after the code was refactored, shows that no additional vulnerabilities were added to the project. These reports are included in the submitted project files.

Before:



After:



## Summary

Refactoring the project code base aims to address several items from the Vulnerability Assessment Process Flow Diagram. For example, for the Cryptography category, we used the SHA-256 algorithm to we encrypt messages between the client and server. In addition, we created a hash value sent along the message that can attest to the security of the communication channel, and ergo the software. To further secure the client and server, we configured the properties of our Spring application to enable the use of Secure Hyper Text Transfer Protocol (HTTPS). Overall, these additions add much needed, extra measure of security to our application.

## Industry Standard Best Practices

The widespread contemporary use of online technology is directly proportional to the measure of security that software engineers should consider when developing their projects. Attackers would spare no opportunity to exploit a vulnerability that subserves their malicious goals, causing irreparable damage to application firms and their millions of users (customers).

It is important that developers follow the industry best practices to maintain the software application’s current security. Specifically, we should always employ input validation as the first layer of protection many dangerous attack types (such as denial and service and injection attacks). We should use error handling techniques in prospect of runtime errors to circumvent crashes and unexpected system behavior that may lead to data leaks or unauthorized access. It is important to control how external sources may access data via added encapsulation. Developers should also follow best coding practices and implement secure patterns to produce high quality code with low risk. And finally, developers should constantly test their code (as we did via the WASP dependency check tool) to find and correct dangerous vulnerabilities within their application, often by updating the vulnerable software to a more recent version.