****

# 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** | **8/11/2023** | **William Neal** |  |

## 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

William Neal

## Algorithm Cipher

AES-256 Advanced Encryption Standard as I have mentioned in multiple assignments before this is the best option for a financial company. It is used by the US government, and there are no known successful attempts at brute force attacks working. The NIST has recognized it as the gold standard, and it should be used in this situation. AES-256 doesn’t have a built-in hash function, only when combined with HMACs would it have this potential. AES-256 is primarily an encryption algorithm. As mentioned before sizing comes in three with AES. 128-bit, 192-bit and 256-bit, with 256 being its strongest and most secure. Random numbers are a major factor in all encryptions, but with AES they specifically become crucial for producing keys. As mentioned in previous documents AES is symmetric and would be void if converted to asymmetric. With asymmetric keys I used the example of bitcoin and wallets, public and private keys. With AES the system is so robust that they use the same key, the downside would be cracking one gives you full access, however; there is no evidence of brute force attacks working on AES-256. Ciphers and encryption date back to the Egyptians and have had notoriety throughout history. Our standards have changed from DES in 1971 until around 2001. As mentioned before we are looking towards quantum computing to see if a new era of cyber security is upon us.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

I tried on both versions of Eclipse and could not get a functional KeyTool certificate. It would generate MD5’s, SHA1’s, and serial numbers, but the files themselves were corrupting somewhere. Wondering if there’s a problem with the certificate authority itself.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A computer screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Can not get anything to run properly. I even tried migrating my old certificate from previous assignment. No matter what format I use for the application.properties I either get ???? or Invalid keystore format

WAIT NEVERMIND AFTER 4 HOURS OF FIGHTING WITH THE APPLICATION!!!!!

A screenshot of a computer

Description automatically generated

## Secure Communications

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

A computer screen with a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with a white screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

I tried 8443 and 443. I also tried using Opera browser instead of Google Chrome, I also went into Windows Task Manager after looking up the PID on CMD (netstat -aon | find “8443”) to manually stop the processes involved. I could not get anything other than Whitelabel Error Pages. I did google to see what I was doing wrong but found no real resolve to my issues. Hopefully I can get some feedback on this.

## Secondary Testing

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

At least one thing worked.

A screenshot of a computer

Description automatically generated

## Functional Testing

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

1. Identify the software application's syntactical, logical, and security vulnerabilitiesby manually reviewing code.
   1. Complete this functional testing and include a screenshot of the refactored code, executed without errors, in your practices for secure software report.

I spent some time looking through the dependencies. I honestly do not know enough about tomcat-embed-core to be messing around with that code base. I wasn’t really sure what you were looking for with this question. I do know that when I go into here:

A screenshot of a computer

Description automatically generated

There are no static tests at all. So I built a couple static/unit tests in order to improve security

A screenshot of a computer

Description automatically generated

I can also say that there were updates done to the application.properties and the SslServerApplication.java as well as the pom.xml but the actual dependencies I was afraid to touch, even though I was hitting critical’s on my dependency check.

## 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:

1. Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.
2. Discuss your process for adding layers of security to the software application.

So actual refactoring of the code would be updating these dependencies. A screenshot of a computer

Description automatically generated

As you can see there are several critical severity dependencies. What should be done here is to develop documentation on each one of these dependencies and see if we can mitigate to better versions and updating them. A lot of them are just dated packages (I looked at them). We could also build out some POSTMAN or SOAPUI tests for API/server-side validation. We could do better with encapsulating our application.java, maybe switch it from public to private or protected. Lastly and most importantly, CODE REVIEW. None of this should be pushed directly to main/production. I want someone reviewing my code, picking it apart and telling me what an awful job I did. I won’t take it personally, I just want to make sure that what I am putting out there is secure, stable, manageable code. I do not want to send out floppy “spaghetti code”.

## Industry Standard Best Practices

Explain how you applied industry standard best practices for secure codingto mitigate against known security vulnerabilities.Be sure to address the following:

1. Explain how you used industry standard best practices to maintain the software application’s current security.
2. Explain the value of applying industry standard best practices for secure coding to the company’s overall wellbeing.

As I have mentioned in previous documents the best practices are done in architecture. Before a line is ever written there should be discussion with Product Owners, Stakeholders, Developers, Quality Assurance and most importantly Security DevOp’s. Security should not be second rate, or an afterthought, it should be built right in. Updating dependencies and checking for security flaws should be a full-time position on the team. There should be an ethical hacking team working to break down our wall’s day and night. Best industry practice is encapsulating code internally and externally. There are teams that do not need to know what information is in the API call, just that that they are receiving it. This protects client data and in a way that team, should there be a data leak nobody can look at the developers because they never saw the information themselves. The best security practices are having quality assurance build out unit and static tests to ensure things are working as expected. This is critical when working with customers/client facing applications. I also believe having a strong Security Dev Ops and Dev Ops team is critical. We need to ensure that the pipeline is sound and maintainable. Product Owners should not be pushing releases that can endanger security protocols. If we know a client facing feature say an input field is not secure and is at risk for let’s say an SQL injection (this shouldn’t happen in the real world, we should be sanitizing everything but let’s pretend here) then the Product Owner needs to understand that the release should not happen. That no matter what their deadline is it’s a vulnerability that is not worth their time crunch. And lastly no developer should be pushing directly to the main branch. You should be building a branch off main, working on it and then pushing remote. Once that remote branch has been working on and pushed to a remote repository then a code review can happen. There should be at least three people inspecting that code for quality and security. No one is above this, I do not care how many years of experience you have, or how senior a position you are. There is always room for improvement and humans are indeed fallible, me included. The last thing any company wants is a self-righteous developer pushing code with major risks and costing them. The reality of applying these practices is that nobody wants to be the next Equifax, it’s been six years since their breach, and they are currently the industry's standard of what not to do. Not only did millions of clients’ information become available to unscrupulous individuals, but their lives might also have been destroyed. Their bank accounts emptied, their credit ruined, and millions of dollars spent trying to rectify the situation. Aside from clients having their lives destroyed, Equifax’s reputation is also burnt. Should I be applying to positions I would avoid working for them because of their history. This is the type of publicity that no company wants, so not only do they save themselves money in lawsuits they also save themselves money in attracting the best talent by simply doing the right thing. As I mentioned before, maybe Stakeholders and Product Owners and truly non-technical individuals do not understand the importance of security, but they do understand the importance of money, and security breaches to sensitive client data means a huge loss in bottom line.