****

# CS 305 Project Two

**Practices for Secure Software Report**

**Dane Clark**

**Department of Computer Science, Southern New Hampshire University**

**CS-305: Software Security**

**Dr. Vivian Lyon**

**08/10/2022**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **08/10/22** | **Dane Clark** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Dane Clark

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

For the purposes of Artemis Financial’s needs, I would suggest using the SHA-256 encryption algorithm. An encryption algorithm, or cipher is used to convert a readable message into a coded message that can only be read by using the proper key. “Ciphers, also called encryption algorithms, are systems for encrypting and decrypting data. A cipher converts the original message, called plaintext, into ciphertext using a key to determine how it is done”. (*Cipher*, 2022).

The “256” part of the SHA-256 cipher means it has a 256-bit encryption, meaning there are 256 bits in the key. “A 256-bit key can have 2256 possible combinations” (Nohe, 2019). This makes it extremely unlikely that anyone would be able to crack the key using modern computing power. The only real way to crack this key would be through a brute force attack, which is trying to guess the key via trial and error. This could take an attacker thousands of years using current gen computer.

There two types of keys symmetric and asymmetric. SHA-256 is a symmetric key. A symmetric encryption uses the same key for both decryption and encryption. This would be better suited for data that is rarely moved from device to device. For Artemis Financial’s purposes, stored archives would be the best use case for this type of cipher. The difference between symmetric and asymmetric keys is that asymmetric keys use 2 keys. “A public key, which is interchanged between higher than one user. Data is decrypted by a private key, which is not transformed. It is slower but more secure. The public key used in this encryption technique is applicable to everyone, but the private key used in it is not revealed” (Ginni, 2022).

As early as ancient Sparta encryption or sending coded messages have been part of human history. “From indirect evidence, the scytale was first mentioned by the Greek poet Archilochus who lived in the 7th century B.C. The ancient Greeks used this cipher to communicate during military campaigns. Sender and recipient each had a cylinder of exactly the same radius. The sender wound a narrow ribbon of parchment around his cylinder. Then he wrote on it lengthwise. After the ribbon is unwound, the writing could be read only by a person who had a cylinder of exactly the same circumference” (Djekic, 2013). This type of technology continued to advance throughout human history. Famously during World War II, Alan Turing and the Enigma machine helped the allied forces win the war against Nazi Germany. As for today encryption algorithms are at a premium with the amount of data that is transmitted through the internet at any given time. Companies want user data, and attackers want to steal that data. Protection is paramount in our digital age. Currently we just don’t have the computing power readily available to us to crack some of the ciphers that are widely used, such as SHA-256. As mentioned, before it could take a modern computer thousands of years to crack the key. However, according to Moore’s Law, this might not always be the case. “Moore's Law refers to Gordon Moore's perception that the number of transistors on a microchip doubles every two years, though the cost of computers is halved. Moore's Law states that we can expect the speed and capability of our computers to increase every couple of years, and we will pay less for them. Another tenet of Moore's Law asserts that this growth is exponential” (Tardi, 2022). Eventually we will be facing an age where quantum computing is the norm. Work has already begun to help face this issue before we reach that point. “In response, the industry is advancing encryption on several fronts. Some efforts are focused on increasing key sizes to protect against brute-force decryption. Other efforts are looking at new cryptographic algorithms. For example, the National Institute of Standards and Technology is evaluating a next-generation public key algorithm intended to be quantum safe” (Orrin, 2021). As computing power become more and more sophisticated so too must our ability to protect our sensitive data.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

Text

Description automatically generated

Text

Description automatically generated

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

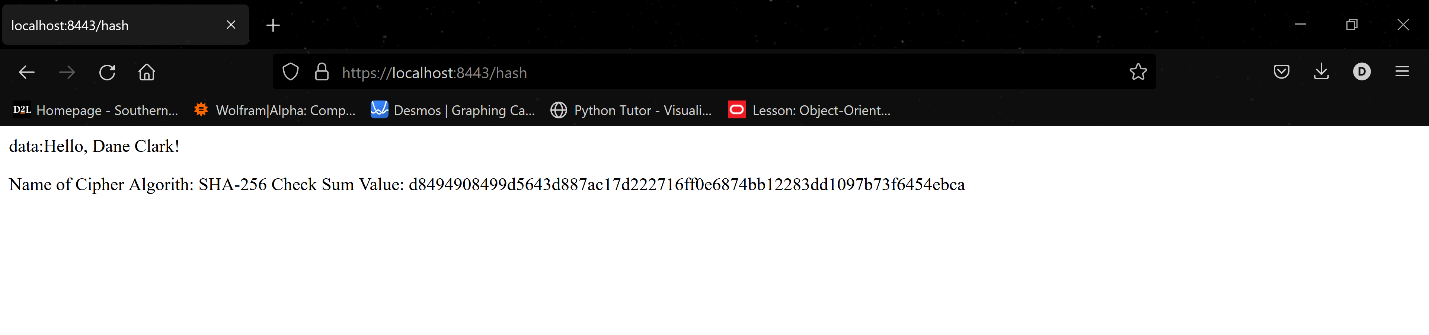
Graphical user interface, text, application, website

Description automatically generated

## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

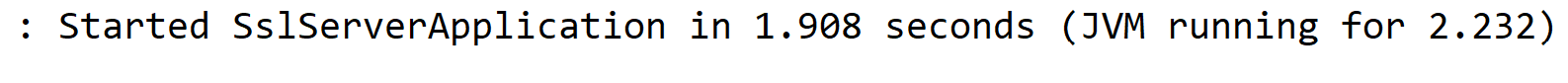
* Insert a screenshot below of the web browser that shows a secure webpage.



## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

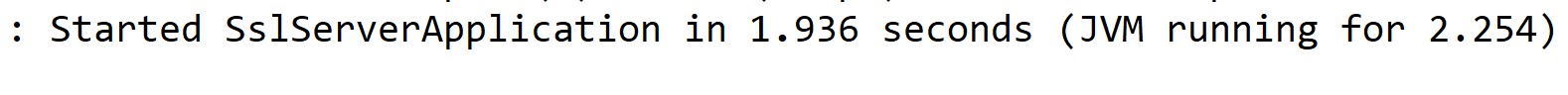
* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

When looking at the code itself there are no syntactical or logic errors to be shown. The code executes without errors. There are some areas that are hard coded that could potentially become problematic if the program were to become compromised.

Line 62: String data = "Hello, Dane Clark!"; could be changed to whatever the attacker may want if unwanted access were granted. The same issue rises in line 72’s return statement. This is the output to the HTTPS location that can be altered.

However, when looking at the security vulnerabilities we see that we run into a lot.

**accessors-smart-1.2.jar (CVE-2021-27568)**

This vulnerability may cause crashing or release of sensitive data due to no catching when an exception is thrown.

This vulnerability can be fixed by updating to newer version. “New version released 1.3.2 with [CVE-2021-27568](https://github.com/advisories/GHSA-v528-7hrm-frqp) fixed json-smart-mini had also been patched and released.”

**References:**

<https://github.com/netplex/json-smart-v1/issues/7>

<https://www.oracle.com/security-alerts/cpujan2022.html>

**hibernate-validator-6.0.18.Final.jar (CVE-2020-10693)**

This vulnerability could allow attackers the ability to enter blocked characters through "input sanitation". This issue can give attackers access to sensitive information or admin access to the system. The solution is to update to the latest version to remove the vulnerability.

"This issue has been addressed in the following products:

  Red Hat JBoss Enterprise Application Platform

Via RHSA-2020:3464"

**References:**

https[://access.redhat.com/errata/RHSA-2020:3464](https://access.redhat.com/errata/RHSA-2020:3464)

<https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2020-10693>

**json-smart-2.3.jar (CVE-2021-27568)**

This vulnerability causes the same issue as **accessors-smart-1.2.jar (CVE-2021-27568).** As such the fix is the same, updating the lates version.

**References:**

<https://github.com/netplex/json-smart-v1/issues/7>

<https://www.oracle.com/security-alerts/cpujan2022.html>

**jackson-databind-2.10.2.jar (CVE-2020-25649)**

“This vulnerability allows attackers to manipulate the way an application interacts with XML. XML handles data processing, a vulnerability to this could allow attackers access to company and client data. This can also allow attackers access to areas of the system that XML has access to. Protection of data is of the utmost importance. An update is available to patch this issue” (Clark, 2022).

This issue has been addressed in the following products:

  Red Hat JBoss Enterprise Application Platform

Via RHSA-2020:4402

**References:**

<https://access.redhat.com/errata/RHSA-2020:4402>

<https://bugzilla.redhat.com/show_bug.cgi?id=1887664>

**log4j-api-2.12.1.jar (CVE-2020-9488 )**

This vulnerability does not check to see if the certificate is coming from the host site. This means a certificate can be a valid certificate, but just not from the host site. To fix this issue updating to the latest version will ensure that checks for the host’s name are in place. Fixed in Apache Log4j 2.12.3 and 2.13.1, update to the latest version will solve the issue.

**References:**

<https://lists.apache.org/thread/p86m144mtr05httn2nw97y3nl1wj50xo>

<https://issues.apache.org/jira/browse/ZOOKEEPER-3817?page=com.atlassian.jira.plugin.system.issuetabpanels:all-tabpanel>

**logback-core-1.2.3.jar (CVE-2021-42550)**

This vulnerability could give attackers access to sensitive data and allow the modification of data. Additionally, this leaves the system open to denial of service (DoS). Fixing this issue is done through an update via cloud manager.

**References:**

<https://logback.qos.ch/news.html>

<https://security.netapp.com/advisory/ntap-20211229-0001/>

**snakeyaml-1.25.jar(CVE-2017-18640)**

This vulnerability could allow attackers to overload the memory of the system and cause a DoS or denial of service to occur. This can temporarily, or indefinitely, make the network resources unavailable to both users and hosts. According to GitHub user Stappe44, there is no solution to this issue outside of using a different component to handle this task.

**References:**

<https://github.com/prometheus/jmx_exporter/issues/458>

<https://lists.apache.org/thread/obr7cqy8lk0dk4gp018pplkqnqsxpkff>

**spring-boot-2.2.4.RELEASE.jar(CVE-2022-27772) / spring-boot-starter-data-rest-2.2.4.RELEASE.jar (**[**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)**)**

These vulnerabilities could allow access to the whole system regardless of privileges allowed to users. An attacker could gain permission to write to files in the directory thereby taking control of the entire system. A fix is available via update (Clark, 2022). “NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

This vulnerability exists because File.mkdir returns false when it fails to create a directory, it does not throw an exception.”

**References**:

<https://github.com/JLLeitschuh/security-research/security/advisories/GHSA-cm59-pr5q-cw85>

**spring-core-5.2.3.RELEASE.jar(CVE-2022-22965)**

This vulnerability could allow system access to remote attackers via the use of data binding. Data binding allows an attacker to change the data that is displayed on the UI of the system. This could allow attackers to place false data or false input options to gain the user’s sensitive data. An update to spring framework is the best solution to the issue as of now.

**References:**

<https://tanzu.vmware.com/security/cve-2022-22965>

<https://www.techtarget.com/whatis/definition/data-binding>

**spring-data-rest-webmvc-3.2.4.RELEASE.jar (CVE-2021-22047 (OSSINDEX))**

This vulnerability can allow unwanted access to the program depending on the Spring security. “HTTP resources implemented by custom controllers using a configured base API path and a controller type-level request mapping are additionally exposed under URIs”. Updating to the latest version will mitigate this problem.

**References:**

<https://tanzu.vmware.com/security/cve-2021-22047>

<https://ossindex.sonatype.org/vulnerability/CVE-2021-22047?component-type=maven&component-name=org.springframework.data%2Fspring-data-rest-webmvc&utm_source=dependency-check&utm_medium=integration&utm_content=7.1.1>

**spring-web-5.2.3.RELEASE.jar(CVE-2016-1000027)**

“This vulnerability could allow an attacker to gain control of the way data is distributed and stored. Essentially giving access to the flow of data to the attacker. This vulnerability does this by way of not having any checks on what data should be deserialized. The system will deserialize untrusted data without checking to see if it is valid. Unfortunately, there is no fix for this issue at this time other than to avoid the user of Java Serialization” (Clark, 2022).

**References:**

<https://security.snyk.io/vuln/SNYK-JAVA-COMGOOGLECODEGSON-1730327>

<https://github.com/spring-projects/spring-framework/issues/24434#issuecomment-582313417>

<https://github.com/gauravdeshmukh612>

**tomcat-embed-core-9.0.30.jar(CVE-2020-1938)**

“This vulnerability allows access to the Apache JServ Protocol (AJP). Since Tomcat give higher priority to AJP it treats AJP as having higher trust. This allows and attacker to use the AJP to gain access to the system by transferring malicious files without having it check. Access like this makes remote code access possible for attacker. This can put the whole system at risk. This would include data for clients and Artemis Financial. The way to fix this issue to update Apache Tomcat to the latest version” (Clark, 2022).

**References:**

<https://www.cvedetails.com/cve/CVE-2020-1938/>

<https://issues.apache.org/jira/browse/TOMEE-2789?page=com.atlassian.jira.plugin.system.issuetabpanels%3Aall-tabpanel>

**tomcat-embed-websocket-9.0.30.jar (CVE-2020-13935)**

“This vulnerability could allow and attacker to exploit how long the WebSocket stays open and cause overloading the memory of the program. This could cause memory leak, and DoS. Additionally, an attacker could send special requests through HTTP/2 that could overload the CPU causing the server to completely shut down. This would take the system offline or cause physical damage to the server. The recommended fix is to update Apache Tomcat to the latest version” (Clark, 2022).

**References:**

<https://nvd.nist.gov/vuln/detail/CVE-2020-11996>

<https://www.debian.org/security/2020/dsa-4727>

<https://lists.apache.org/thread/r7m5zthg1k9grytzqz0cwlnfb7wjfonz>

Text

Description automatically generated

A picture containing scatter chart

Description automatically generated

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

After reviewing my refactored code as well as the Vulnerability Assessment Process Flow Diagram, I addressed, API’s, cryptography, client/server, code error, and code quality.

The use of the Restful API within this code allowed for the rest controller to pass the HTTP information to the endpoint. “When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP” (“*What is a Rest API?”,* 2020). Cryptography was used to change the message, in this case “Hello, Dane Clark!” into a hash function that would be later displayed on the webpage. Addressing the client/server portion was done with the creation of a certificate that would be used to securely tie the HTTPS together. This allows for a more secure handshake between the client and the server side. Using this helps to ensure that the client side can trust the server side is actually coming from the trusted host. Code error was addressed by ensuring that the code was free of error. This means syntax, and logic were properly implemented to ensure that the code was able to execute without error. Finally, code quality was addressed by ensuring to use proper in code comments. This is important for future programmers within the company who may need to perform maintenance on this code. It allows the developer to easily understand what this code is doing.

I wanted to make sure that this code ran properly. As such my focus was on making sure that the code error was at a minimum, and code quality was a priority. Additionally, using the latest version of the dependency check report was also an area of importance. Having the most up-to-date version of the dependency check report ensures that we are getting the most accurate data on what vulnerabilities are found. It also ensures that if there is a solution to the vulnerabilities, we are able to find them. It also gives us the most accurate mitigation techniques available. This helps Artemis Financial keep their program as error free as possible. More importantly, having the latest version of dependencies ensures that Artemis Financial is taking the necessary steps to protect their data as well as the data of their clients. Finally, using the SHA-256 cipher will help to protect Artemis Finacial’s data and the data of their clients from brute force attacks. Since this cipher has such a high bit encryption their data will remain safe from attacks that don’t have access to the key.

Continuous use of the dependency checker will help to protect the company from any new vulnerabilities that may come up. Doing this often will also tell the company if new mitigation techniques are available. Additionally, any vulnerabilities that may have been discovered since the last time the checker was run will keep the most important information available to the company. This goes hand in hand with ensuring the plug-ins that are used are as up to date as possible. Sometime the dependency checker will address this however, sometimes this is not always the case. To maintain a high level of security updating these plug-ins is not only recommended but necessary. Code reviews are also an important are that should be focused on. If changes are made to the program, whether that be to the functionality of the code or updating the plug-ins, a code review should be executed. This is to make sure that issues were not introduced during the update period or when maintenance was being performed. Following these steps will help Artemis Financial maintain a high level of security for themselves and their clients.

Work Cited

*Cipher. (n.d.).* Security Encyclopedia. Retrieved August 10, 2022, from <https://www.hypr.com/cipher/>

Clark, D. (2022, July 18). *CS 305 Project One Artemis Financial Vulnerability Assessment Report.* [unpublished paper]. Computer Science Department, Southern New Hampshire University.

Djekic, M. (2013, November 25). *A Scytale – Cryptography of the Ancient Sparta. Australian Science.* Retrieved: Jul 22, 2022, from <http://ozscience.com/technology/a-scytale-cryptography-of-the-ancient-sparta/>

Ginni. (2022, March 10) *What is the difference between Symmetric Key Cryptographic and Asymmetric Key Cryptography?* <https://www.tutorialspoint.com/what-is-the-difference-between-symmetric-key-cryptographic-and-asymmetric-key-cryptography>

Nohe, P. (2019, May 2). *How strong is 256-bit Encryption?* <https://www.thesslstore.com/blog/what-is-256-bit-encryption/>

Orrin, S. (2021, July 12) *The Future of Data Encryption: What You Need to Know Now* <https://fedtechmagazine.com/article/2021/07/future-data-encryption-what-you-need-know-now>

Tardi, C. (2022, July 17). *Moore's Law.* What Is Moore's Law?. <https://www.investopedia.com/terms/m/mooreslaw.asp>

*What is a REST API?* (2020, May 8) retrieved August 11, 2022, from: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>