****

# CS 305 Project Two

**Practices for Secure Software Report**

**-Bradley Byard-**

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** |
| --- | --- | --- | --- |
| **1.2** | **12-10-20** | **Bradley Byard** | **Project 2 Submission** |

## 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

Bradley Byard

## 1. Algorithm Cipher

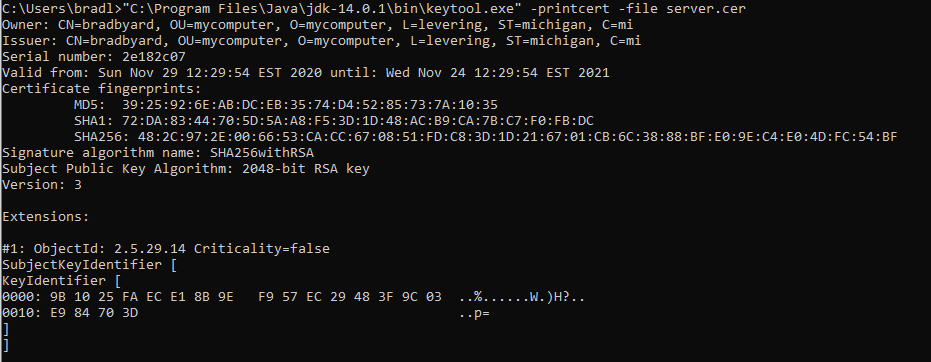
I would recommend using 128-bit Advanced Encryption Standard (AES). We can define AES as a modern **block cipher** that supports three key lengths of **128, 192, and 256-bit encryption**, and it provides excellent long-term security against brute-force attacks (Chernev, 2020). At its simplest, AES is a **cryptographic algorithm** used to protect electronic data. It’s a**symmetric block cipher** that can encrypt and decrypt information (Chernev, 2020). Like many other block ciphers, AES uses **rounds of encryption** that carry out the cipher transformations (Chernev, 2020). Each round typically consists of several building blocks designed jointly to create a function, which is then run multiple times (Chernev, 2020).  The number of rounds AES performs depends on the length of its key. **At 128 bits, it does 10 at 192 – 12, and at 256 – 14.**

**Here is what makes AES a perfect option for Artemis Financial**

* 128-bit encryption provides very high-level security
* Specific numbers about its adoption are hard to come by, Professor Christof Paar of the Ruhr University of Bochum, Germany – a world-renowned specialist in AES cryptology – has estimated that it is used to encrypt **over 50% of all data globally** (Chernev, 2020)**. The reason other people are using AES is because it is one of the most secure encryptions to protect data.**
* It is fast and compact on a wide range of platforms (Chernev, 2020).
* Outperforms is predecessor DES (Chernev, 2020).
* It’s not possible to carry out a successful **brute-force attack** on AES-256; any such attempt would require roughly as many combinations as **1,100 followed by 75 zeroes** (Chernev, 2020). Brute force attacks with 128-bit or 256-bit encryption are near impossible.

## 2. Certificate Generation

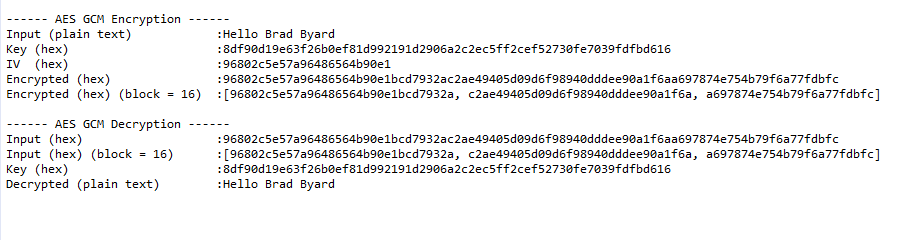
Below is appropriate self-signed certificates using the Java Keytool, which is used through the command line. Server.cer file will be included also.





## 3. Deploy Cipher

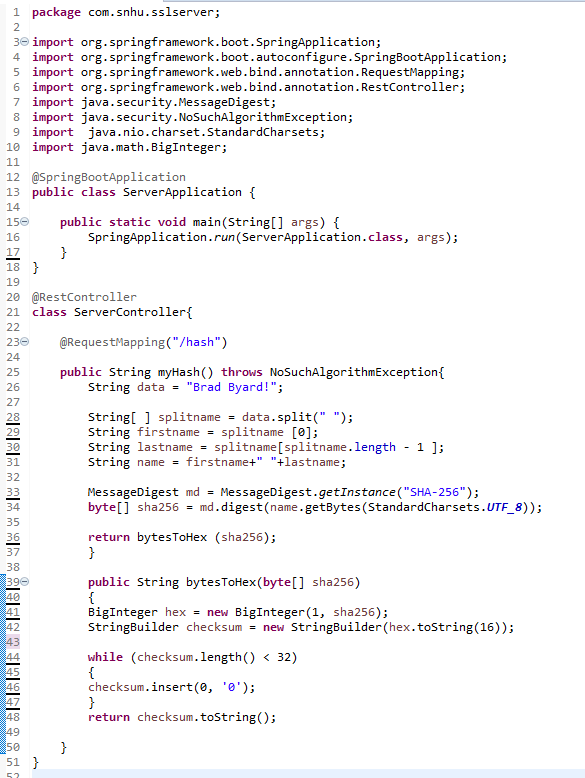
Below is output of test code that Encrypts and decrypts the phrase “Hello Brad Byard” in 256-bit AES.



AES code is somewhat lengthy so it will be attached to project.

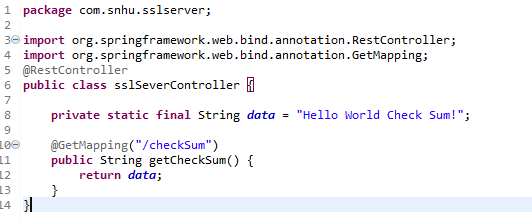
Also included some new code that used SHA-256 Cryptographic Hash Algorithm

That code is shown below

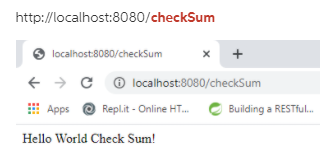


## 4. Secure Communications

To implement this code, I added a new class called sslSeverController to incorporate Checksum

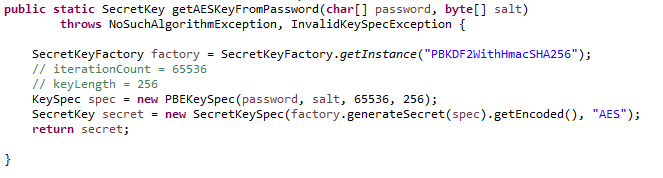
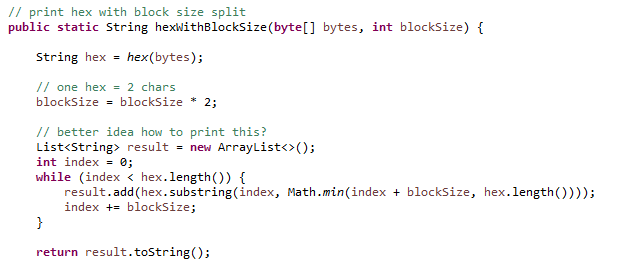


Below is a screenshot of the localhost page I was able to generate.

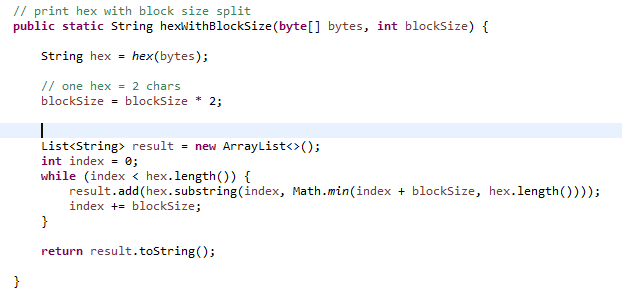


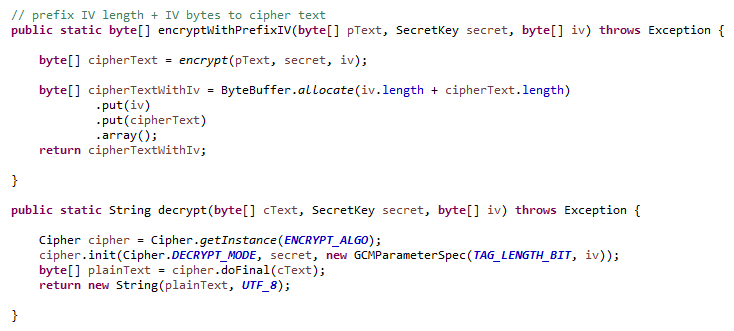
The above code didn’t factor in any hashing so I re-edited code to include hashing using Message Digest. That message digest code is shown on page 5.

That code will be shown below in section 5. The Below code is code used to encrypt in AES



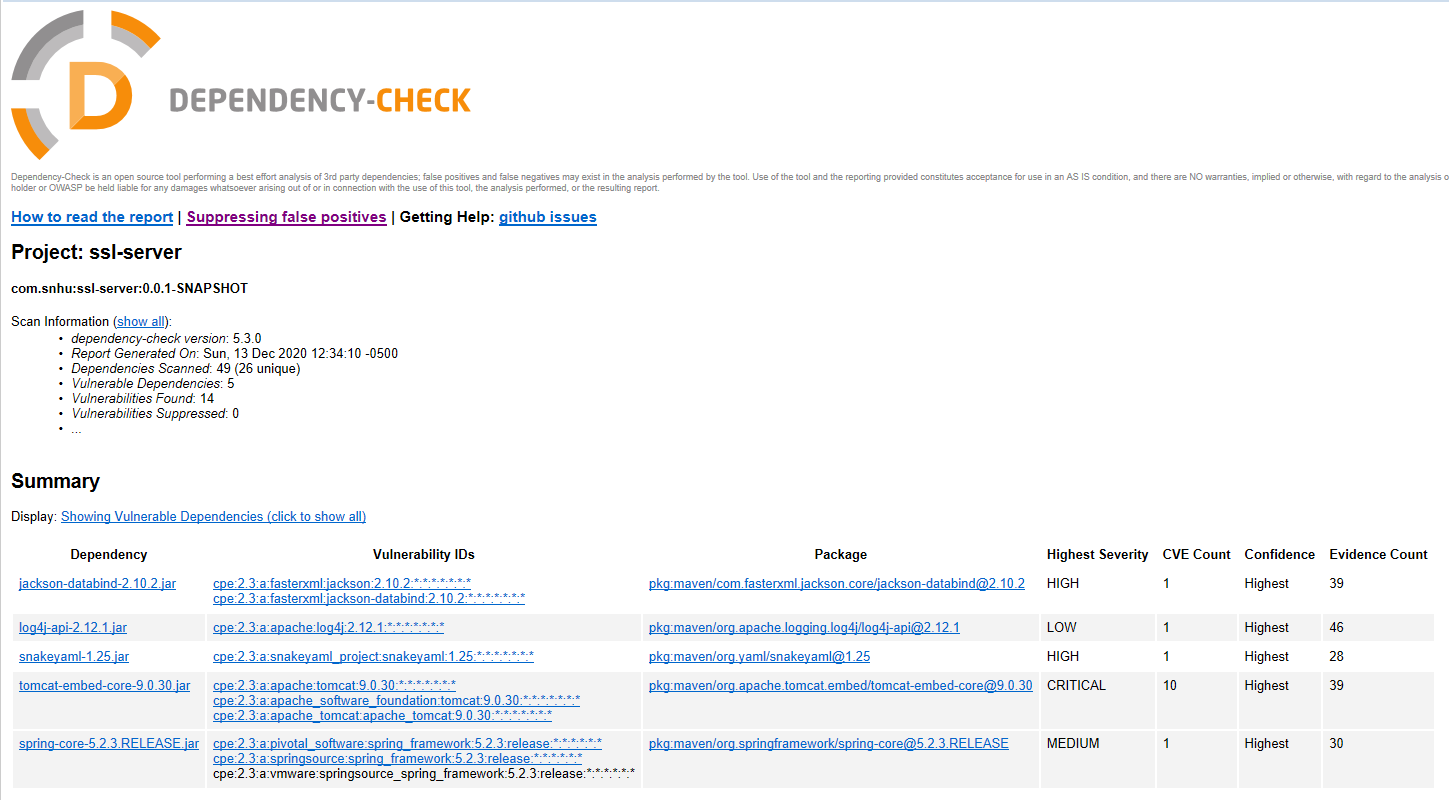
SOME AES Code





## 5. Secondary Testing

* The following are included here:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Original Dependency Check before code changes

[Insert screenshots here.]

## 

## Refactored Code with no errors while showing new Hash code

## 

## 6. Functional Testing

As shown in the dependency check above the main issue here with this code is the

“[pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30](https://ossindex.sonatype.org/component/pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30?utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0" \t "_blank)”

What can we do to secure this vulnerability?

First we can update the version of Tomcat to a version 10.0 that is current (dec 2020).

After updating this in the provided code Like so:

<dependency>

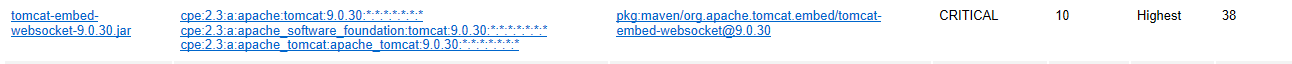
<groupId>org.apache.tomcat.embed</groupId>

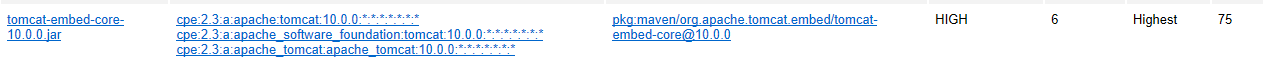
<artifactId>tomcat-embed-core</artifactId>

<version>10.0.0</version>

</dependency>

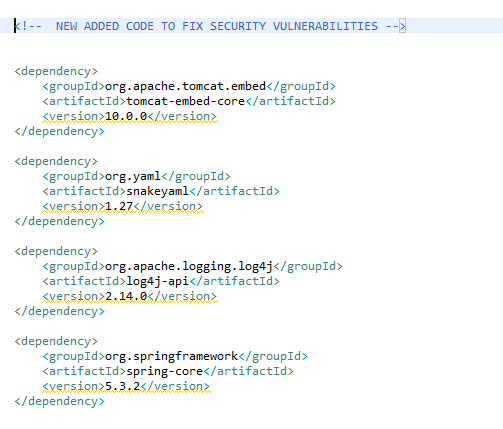
And then re-running the code, let’s take a look at a new dependency check.





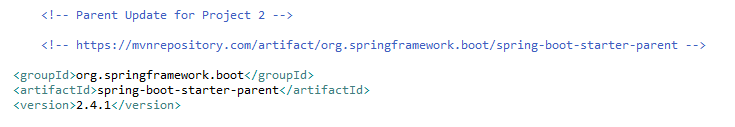
The first look shows the severity of the dependency as critical. When we updated to Tomcat 10.0 we see that it took severity of Critical to High, and also reduced the CVE count from 10 to 6. So, this did help.

Manually went through code and updated all of the following dependencies

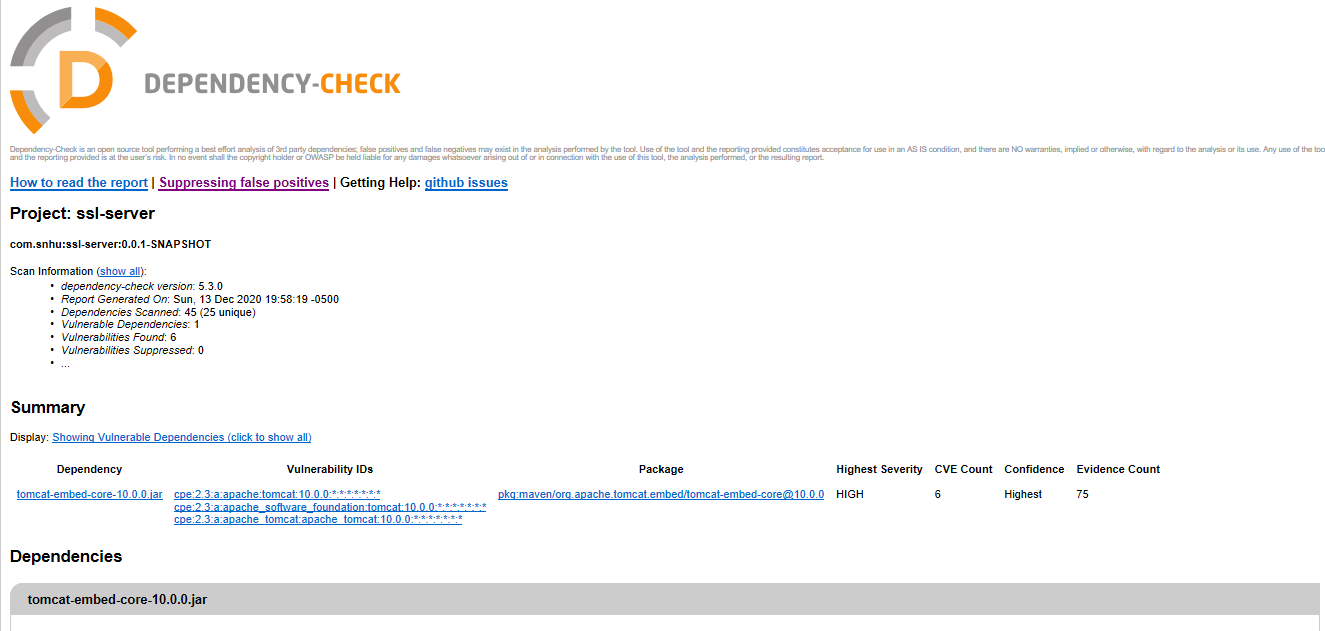


Also updated the Parent in the Spring boot Starter

Shown Below:



After all of the above changes (version updates to the most modern versions) and updated spring boot parent version and re-running the dependency check I got the following summary:



You see here, no more of critical severity and in fact only once dependency found with only 6 CVE counts for Tomcat, which is better than the original 10. All other problems were fixed completely and don’t show up anymore.

## 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.

Maven Repository was a very good resource here to update the code and the pom.xml file to incorporate the most up to date and secure versions we can run with our code.

I manually looked through code and made sure code quality was good, and also made sure that all cryptography algorithms were secure. One way to do that is make sure to avoid using code that can cause Collisions. There are no collisions possible in my AES code and also only a .01% chance of collisions in my hash code that I included.

By updating and patching all vulnerabilities, I made sure that all server/client and API code was up to date and secure.

Keeping everything up to date helps keep the company’s security modern and up to date with the latest problems and fixes in software security. These vulnerabilities and dependencies in code should be checked every so often and updated to help keep company secure.

The encryption packages should be monitored also and kept up to date.

By using secure encryption methods like AES, which is almost impossible to brute force attack, you stay safe against hackers.

Below is a great list of best practices for maintaining security of this software application via (Synopsys Editorial Team, 2020). It’s best the company has an IT department that stays on top of these tasks.

* Patch your software
  + Many attackers exploit known vulnerabilities associated with old or out-of-date software. To thwart common attacks, ensure that all your systems have up-to-date patches. Regular patching is one of the most effective software security practices. (Synopsys Editorial Team, 2020)
* Educate and train users
  + Make sure all employees using this software are staying on top of security threats and possible attacks and have proper training. (Synopsys Editorial Team, 2020)
* Automate Routine Tasks
  + Attackers use automation to detect open ports, security misconfigurations, and so on. So, you can’t defend your systems using only manual techniques. Instead, automate day-to-day security tasks, such as analyzing firewall changes and device security configurations. Automating frequent tasks allows your security staff to focus on more strategic security initiatives. (Synopsys Editorial Team, 2020)
* Enforce Least Privilege
  + Ensure that users and systems have the minimum access privileges required to perform their job functions. Enforcing the principle of least privilege significantly reduces your [attack surface](https://en.wikipedia.org/wiki/Attack_surface) by eliminating unnecessary access rights, which can cause a variety of compromises. (Synopsys Editorial Team, 2020)
* Document your security policies.
  + Maintain a knowledge repository that includes comprehensively documented [software security policies](https://www.synopsys.com/software-integrity/software-security-services/policy-standards.html). Security policies allow your employees, including network administrators, security staff, and so on, to understand what activities you’re performing and why. (Synopsys Editorial Team, 2020)
* Monitor user Activity
  + Trust, but verify. Monitoring user activities helps you ensure that users are following software security best practices. It also allows you to detect suspicious activities, such as privilege abuse and user impersonation. (Synopsys Editorial Team, 2020)

**Resources**

Chernev, B. (2020, July 25). What Is AES and Why You Already Love It. Retrieved November 22, 2020, from https://techjury.net/blog/what-is-aes/#gref

Synopsys Editorial Team. (2020, June 29). Are you following the top 10 software security best practices? Retrieved December 13, 2020, from https://www.synopsys.com/blogs/software-security/top-10-software-security-best-practices/