GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	11 December 2022	Ethan Klukkert	

Client

ARTEMIS
FINANCIAL

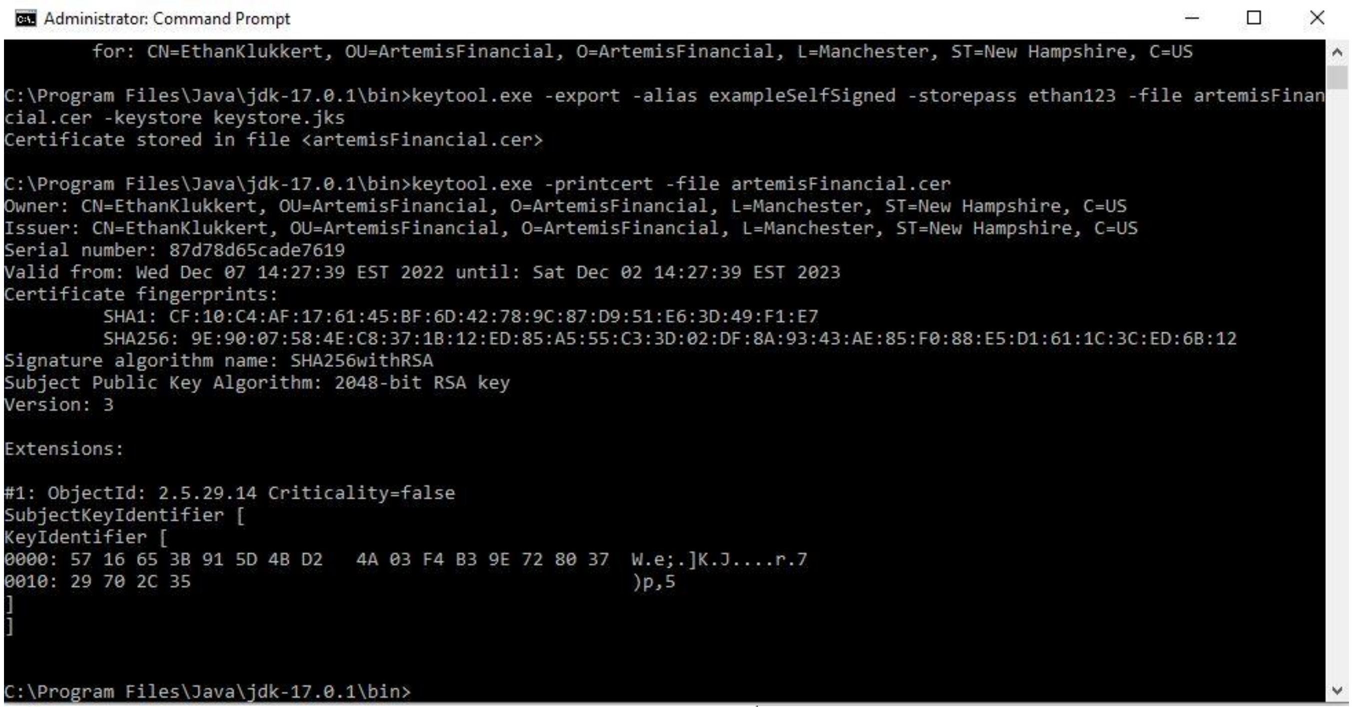
Developer
Ethan Klukkert

1. Algorithm Cipher

Artemis Financial will use the Advanced Encryption Standard, or AES, algorithm cipher to encrypt files and communications over their application services. AES is a symmetric algorithm cipher, so two communication ports need to have a pre-arranged encryption key (Technopedia 2022). AES will also be accompanied by a hash function called SHA-256, which is a 256-bit hash function that returns a unique hash value for a document. AES will be used for a communication channel, where encrypted data can be later decrypted by a receiver. The application will use the 256-bit level for maximum security, especially since the application handles sensitive personal financial information. The cipher also has no known exploits or vulnerabilities that may make it exposed to attacks (Dutta 2021). Once data is received, it must be verified to ensure that no data has been altered. This is where SHA-256 will be used for the application to ensure that data is not altered while in transit, even if it has been encrypted by AES. SHA-256 uses a 256-bit level hash function to return 64 hexadecimal characters. Since the algorithm returns a constant number of 64 characters, there is theoretically another document that can produce the same value. However, this means that there are 2 to the power of 256 different combinations to go through in order to find the original message or another message that produces the same result (Technopedia 2022). Even though it is technically possible, it is practically impossible, therefore making it the most secure hash algorithm in use today. So, a message can generate a unique string of characters that can be used to compare to the hexadecimal value received versus the value it should be. This "value that it should be" is obtained by a certificate authority (Vitale 2017). A self-signed certificate and cipher is implemented in this document to demonstrate how a unique hash value is generated and data is encrypted to ensure data is secure and untampered.

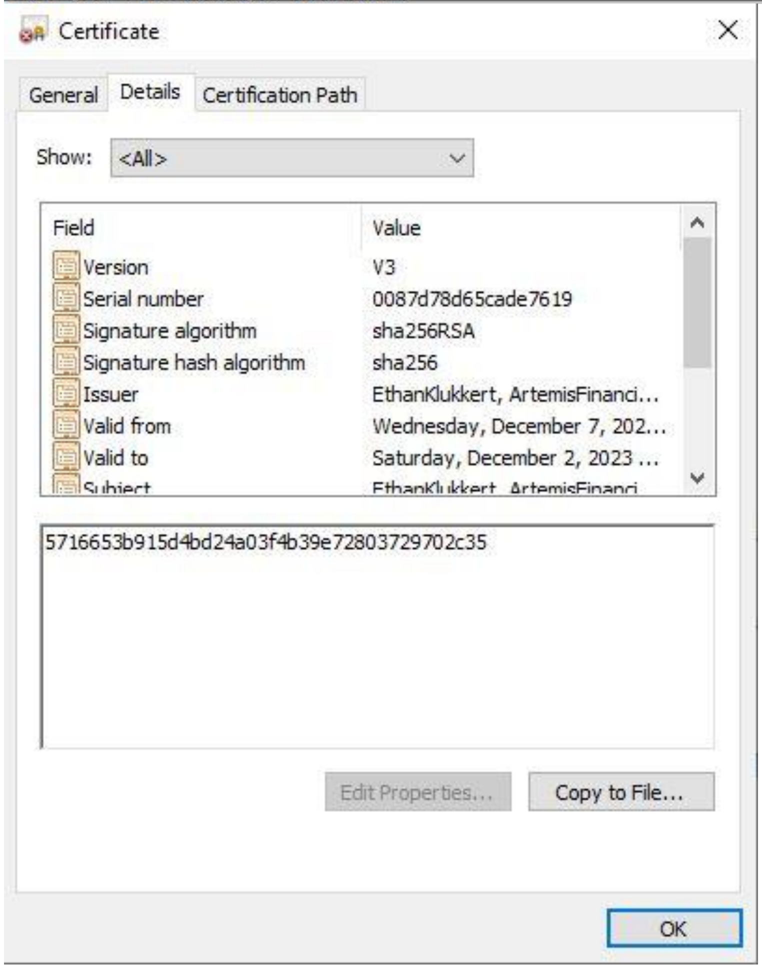
2. Certificate Generation

Here, this is an output of a self-signed certificate demonstrating how the SHA-256 algorithm is used to generate a self-signed certificate.



Administrator: Command Prompt

```
       for: CN=EthanKlukkert, OU=ArtemisFinancial, O=ArtemisFinancial, L=Manchester, ST=New Hampshire, C=US

C:\Program Files\Java\jdk-17.0.1\bin>keytool.exe -export -alias exampleSelfSigned -storepass ethan123 -file artemisFinan
cial.cer -keystore keystore.jks
Certificate stored in file <artemisFinancial.cer>

C:\Program Files\Java\jdk-17.0.1\bin>keytool.exe -printcert -file artemisFinancial.cer
Owner: CN=EthanKlukkert, OU=ArtemisFinancial, O=ArtemisFinancial, L=Manchester, ST=New Hampshire, C=US
Issuer: CN=EthanKlukkert, OU=ArtemisFinancial, O=ArtemisFinancial, L=Manchester, ST=New Hampshire, C=US
Serial number: 87d78d65cade7619
Valid from: Wed Dec 07 14:27:39 EST 2022 until: Sat Dec 02 14:27:39 EST 2023
Certificate fingerprints:
         SHA1: CF:10:C4:AF:17:61:45:BF:6D:42:78:9C:87:D9:51:E6:3D:49:F1:E7
         SHA256: 9E:90:07:58:4E:C8:37:1B:12:ED:85:A5:55:C3:3D:02:DF:8A:93:43:AE:85:F0:88:E5:D1:61:1C:3C:ED:6B:12
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 57 16 65 3B 91 5D 4B D2   4A 03 F4 B3 9E 72 80 37  W.e;.]K.J....r.7
0010: 29 70 2C 35                                        )p,5
]
]


C:\Program Files\Java\jdk-17.0.1\bin>
```



Certificate

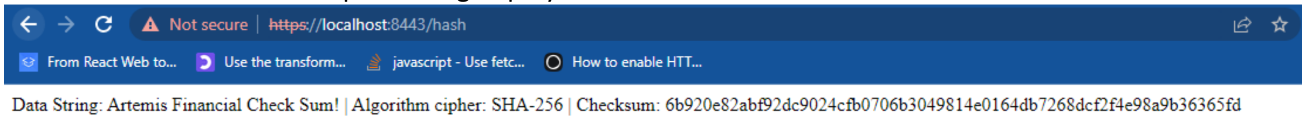
Field	Value
Version	V3
Serial number	0087d78d65cade7619
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	EthanKlukkert, ArtemisFinanci...
Valid from	Wednesday, December 7, 202...
Valid to	Saturday, December 2, 2023 ...
Subject	EthanKlukkert, ArtemisFinanci

5716653b915d4bd24a03f4b39e72803729702c35

The image above displays some of the details of the certificate generated. This certificate file is attached to this report in a zip file.

3. Deploy Cipher

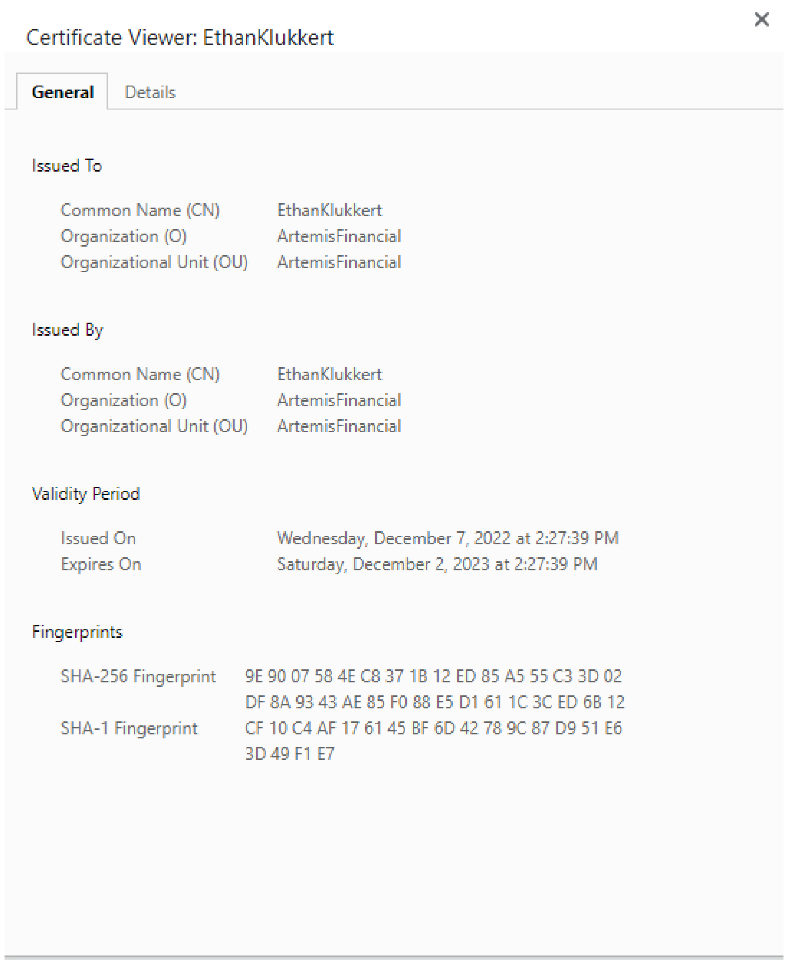
This screenshot shows the cipher being deployed with a checksum verification.



```
Data String: Artemis Financial Check Sum! | Algorithm cipher: SHA-256 | Checksum: 6b920e82abf92dc9024cfb0706b3049814e0164db7268dcf2f4e98a9b36365fd
```

4. Secure Communications

The connection itself must be secured using a valid certificate. The example above uses a self-signed certificate, which should not be used in production for a secure connection. A secure certificate can be obtained for free from Certbot (https://certbot.eff.org/instructions?ws=apache&os=windows) by installing it on the server's machine and running the installer. This needs to be done on a live server other than localhost servers (Vitale 2017).



Certificate Viewer: EthanKlukkert

Issued To

Common Name (CN)	EthanKlukkert
Organization (O)	ArtemisFinancial
Organizational Unit (OU)	ArtemisFinancial

Issued By

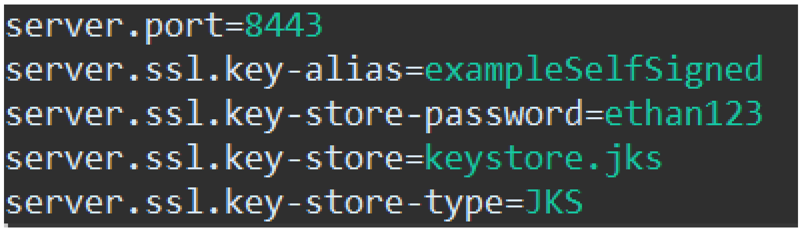
Common Name (CN)	EthanKlukkert
Organization (O)	ArtemisFinancial
Organizational Unit (OU)	ArtemisFinancial

Validity Period

Issued On	Wednesday, December 7, 2022 at 2:27:39 PM
Expires On	Saturday, December 2, 2023 at 2:27:39 PM

Fingerprints

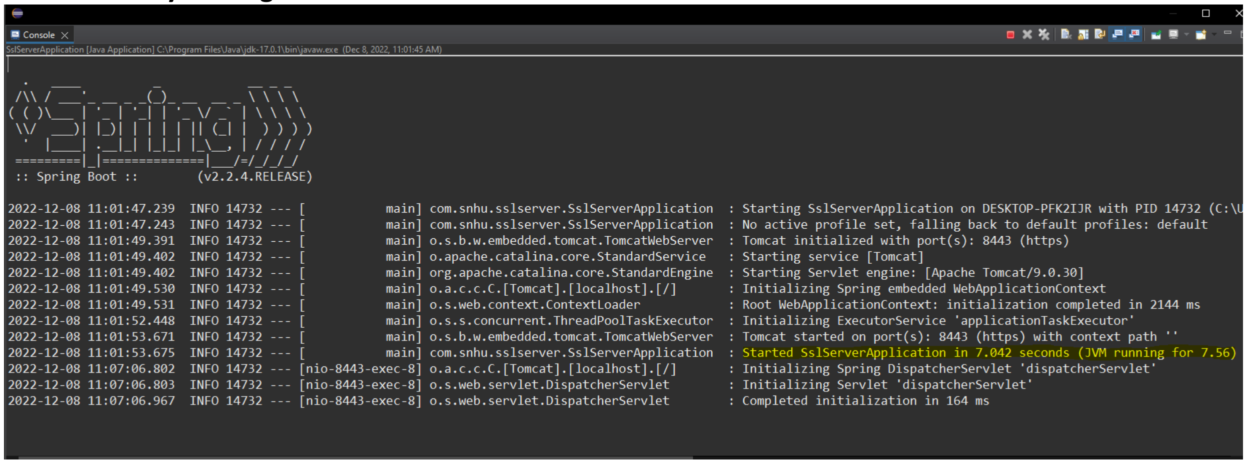
SHA-256 Fingerprint	9E 90 07 58 4E C8 37 1B 12 ED 85 A5 55 C3 3D 02 DF 8A 93 43 AE 85 F0 88 E5 D1 61 1C 3C ED 6B 12
SHA-1 Fingerprint	CF 10 C4 AF 17 61 45 BF 6D 42 78 9C 87 D9 51 E6 3D 49 F1 E7



```
server.port=8443
server.ssl.key-alias=exampleSelfSigned
server.ssl.key-store-password=ethan123
server.ssl.key-store=keystore.jks
server.ssl.key-store-type=JKS
```

This image above shows the settings for enabling HTTPS on a server, where the keystore.jks file is linked within the project files (see attached code).

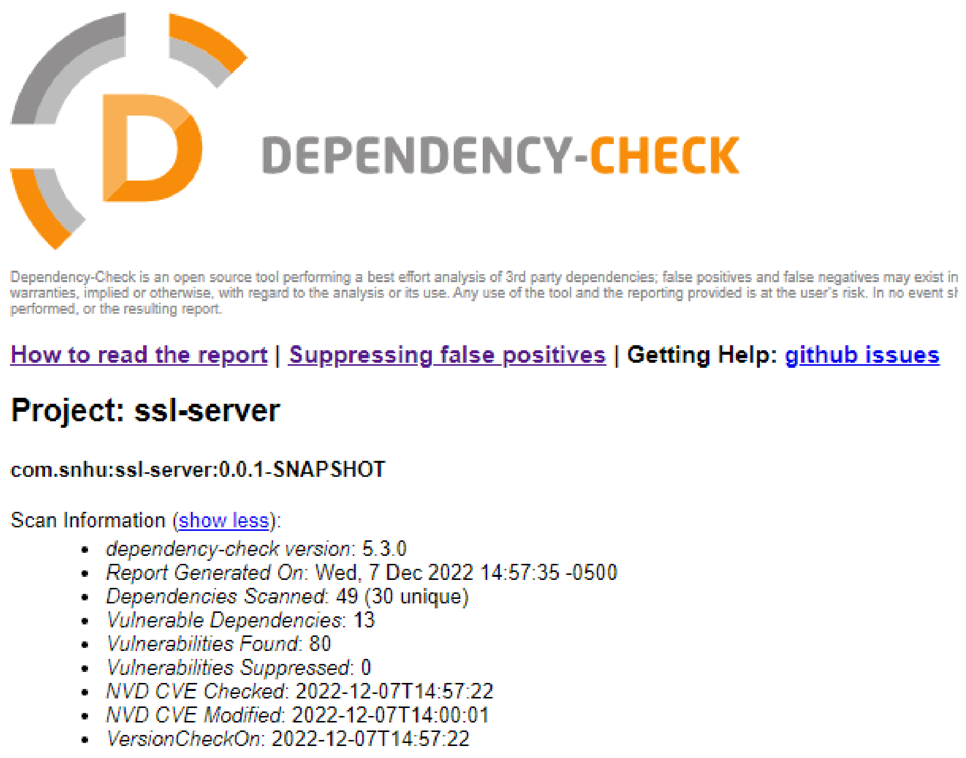
5. Secondary Testing



Console
SslServerApplication [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Dec 8, 2022, 11:01:45 AM)

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2022-12-08 11:01:47.239  INFO 14732 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on DESKTOP-PFK2IJR with PID 14732 (C:\U
2022-12-08 11:01:47.243  INFO 14732 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: default
2022-12-08 11:01:49.391  INFO 14732 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2022-12-08 11:01:49.402  INFO 14732 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-12-08 11:01:49.402  INFO 14732 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2022-12-08 11:01:49.530  INFO 14732 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-12-08 11:01:49.531  INFO 14732 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2144 ms
2022-12-08 11:01:52.448  INFO 14732 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2022-12-08 11:01:53.671  INFO 14732 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2022-12-08 11:01:53.675  INFO 14732 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 7.042 seconds (JVM running for 7.56)
2022-12-08 11:07:06.802  INFO 14732 --- [nio-8443-exec-8] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-12-08 11:07:06.803  INFO 14732 --- [nio-8443-exec-8] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-12-08 11:07:06.967  INFO 14732 --- [nio-8443-exec-8] o.s.web.servlet.DispatcherServlet        : Completed initialization in 164 ms
```

The original server running the operations above can be seen running without errors in this console window. Also, a preliminary dependency check was conducted to note the pre-existing vulnerabilities in the application (see attached for full file).



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event sh performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

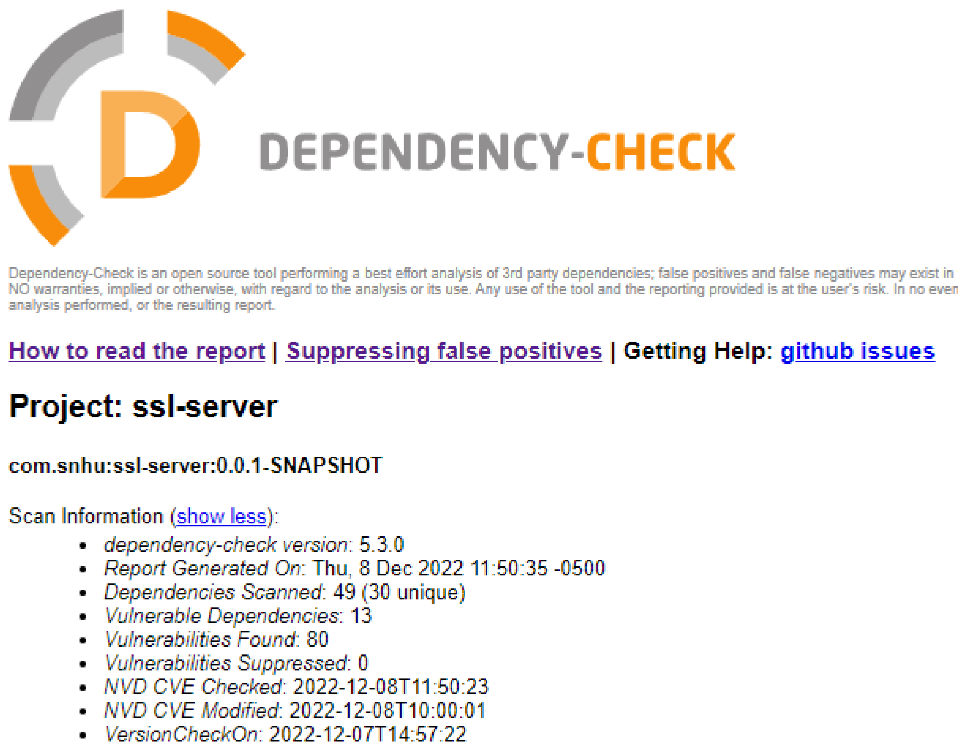
Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show less):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Wed, 7 Dec 2022 14:57:35 -0500
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 13
- *Vulnerabilities Found*: 80
- *Vulnerabilities Suppressed*: 0
- *NVD CVE Checked*: 2022-12-07T14:57:22
- *NVD CVE Modified*: 2022-12-07T14:00:01
- *VersionCheckOn*: 2022-12-07T14:57:22

Then, the dependency check was rerun after the code was refactored.



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no even analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

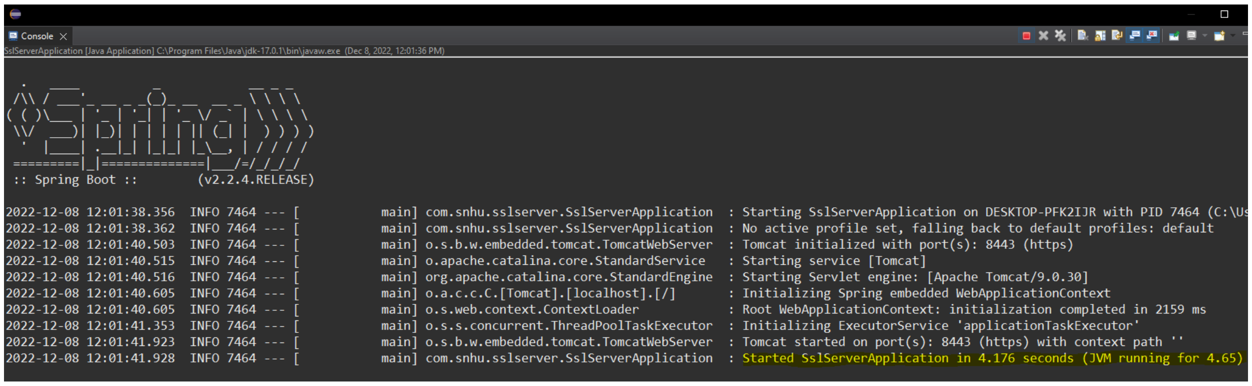
Scan Information (show less):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Thu, 8 Dec 2022 11:50:35 -0500
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 13
- *Vulnerabilities Found*: 80
- *Vulnerabilities Suppressed*: 0
- *NVD CVE Checked*: 2022-12-08T11:50:23
- *NVD CVE Modified*: 2022-12-08T10:00:01
- *VersionCheckOn*: 2022-12-07T14:57:22

As the report states, no "new" vulnerabilities have been introduced to the application. Of course, there are still vulnerabilities to be mitigated or suppressed, but the number of vulnerabilities has not been increased due to the code being refactored (see attached html reports).

6. Functional Testing

For functional testing, first the code is run without error as seen in the image below:

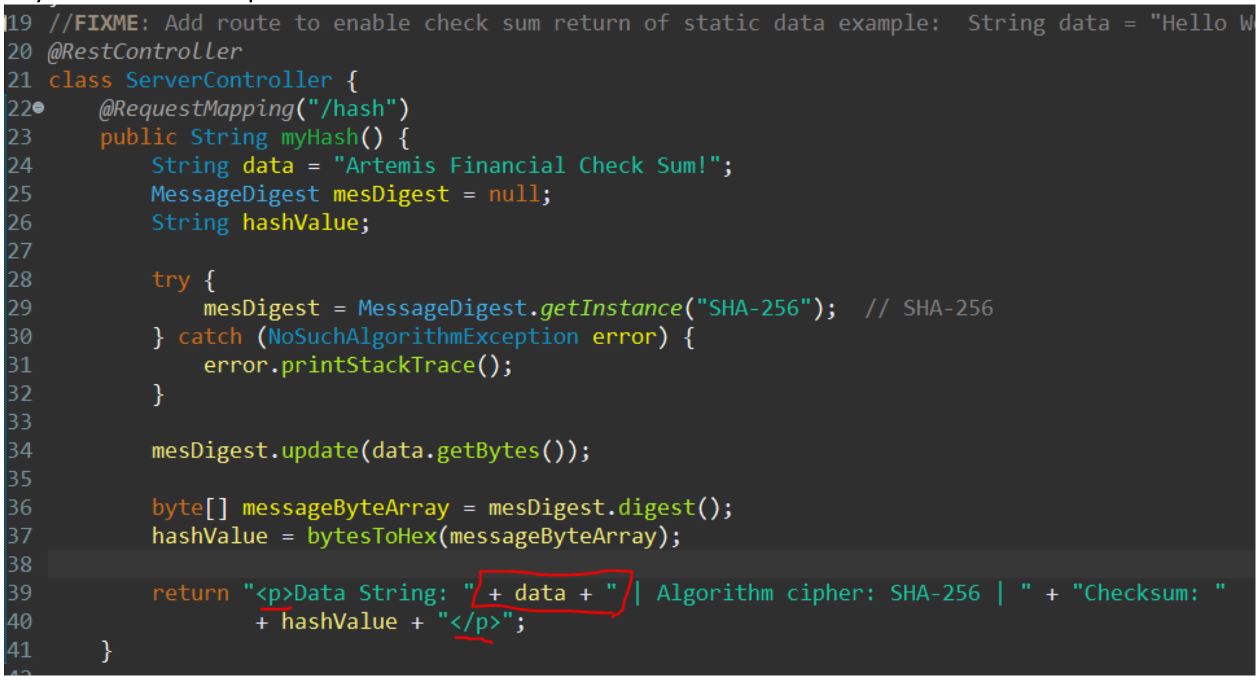


Console

SslServerApplication [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Dec 8, 2022, 12:01:36 PM)

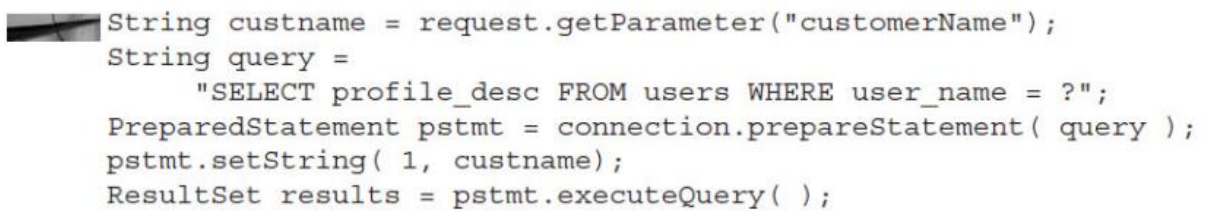
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2022-12-08 12:01:38.356  INFO 7464 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on DESKTOP-PFK2IJR with PID 7464 (C:\Us
2022-12-08 12:01:38.362  INFO 7464 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: default
2022-12-08 12:01:40.503  INFO 7464 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2022-12-08 12:01:40.515  INFO 7464 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-12-08 12:01:40.516  INFO 7464 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2022-12-08 12:01:40.605  INFO 7464 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-12-08 12:01:40.605  INFO 7464 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2159 ms
2022-12-08 12:01:41.353  INFO 7464 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2022-12-08 12:01:41.923  INFO 7464 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2022-12-08 12:01:41.928  INFO 7464 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 4.176 seconds (JVM running for 4.65)
```

Then the code is examined for any logical and security vulnerabilities. Since the application is small at this stage, it is not evident that there are many potential issues. However, there is one issue that stands out, and that is at line 39 of the SslServerApplication.java file, where data is directly appended to the html! This allows attackers to input raw html into the browser, and so this means the attacker can put any harmful JavaScript code into html.



```
19 //FIXME: Add route to enable check sum return of static data example:  String data = "Hello W
20 @RestController
21 class ServerController {
22     @RequestMapping("/hash")
23     public String myHash() {
24         String data = "Artemis Financial Check Sum!";
25         MessageDigest mesDigest = null;
26         String hashValue;
27 
28         try {
29             mesDigest = MessageDigest.getInstance("SHA-256");   // SHA-256
30         } catch (NoSuchAlgorithmException error) {
31             error.printStackTrace();
32         }
33 
34         mesDigest.update(data.getBytes());
35 
36         byte[] messageByteArray = mesDigest.digest();
37         hashValue = bytesToHex(messageByteArray);
38 
39         return "<p>Data String: " + data + " | Algorithm cipher: SHA-256 | " + "Checksum: "
40             + hashValue + "</p>";
41     }
```

Also, the attacker can even reroute the user to any untrusted website through JavaScript. Even SQL injection can be implemented here with server access. Clearly this can be a serious issue for the user and application, and so the solution to this problem is parameterization and input validation (Manico 2014). A prepared statement in Java can prevent any untrusted data from escaping the original intentions of the input handler code:



```
String custname = request.getParameter("customerName");
String query =
     "SELECT profile_desc FROM users WHERE user_name = ?";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

Credit: *Iron-Clad Java, Chapter 7,* Jim Manico et al., September 2014

7. Summary

In this project report, I refactored code to meet proper security protocols by securing connections, securing data reliability, and preventing code from becoming compromised in the application server. These particularly emphasized the importance of Input Validation, Cryptography, Client / Server security, and Code Quality. In the certificate generation section, we implemented a way for users to ensure that the connection to the application was legitimate and can be trusted. This makes sure that the user can send data to the right service. In the first section, we discussed how data can be transferred reliably across an unsecure network. This enables the user and application to send data to each other without fear of the data being intercepted and tampered with. Lastly, Input Validation and Code Quality were demonstrated using parameterization to ensure that untrusted data sent to the server cannot harm the server or users connected to the server. The data is sequestered in secure coding practices.

8. Industry Standard Best Practices

We achieved several industry standards for web security by implementing several key actions. First, a certificate is established from a third-party CA to ensure that users and the application are actually communicating with each other, and no one else. Second, cryptographic ciphers are used to make data unreadable from eavesdroppers. This is also paired with a checksum verification to ensure that the data is not changed even in the slightest by comparing the hexadecimal output of the document to another instance of the hexadecimal output. Lastly, server code is ensured to have parameterization for input data. This means that even when a user is verified through authentication and authorization, their data cannot be used to cause harm such as SQL injection and Cross-Site Scripting (Manico 2014). In summation, the server can connect to users securely, send data securely, and even if the user is malicious the server nor users should be damaged.

References

Technopedia (August 2022). *What is 256-Bit Encryption?* https://www.techopedia.com/definition/29703/256-bit-encryption#:~:text=256%2Dbit%20encryption%20is%20refers,by%20even%20the%20fastest%20computers.

Dutta, S. (Jan 2021). *This Unsolvable Problem is Worth Billions of Dollars* https://medium.com/the-wisest-friends/this-unsolvable-problem-is-worth-billions-of-dollars-4fc28b876e03#:~:text=SHA%2D256%2C%20which%20is%20a,how%20is%20it%20so%20secure%3F

Manico, J. et al. (September 2014) *Iron-Clad Java: Building Secure Web Applications.* McGraw Hill Computing.

Vitale, Thomas (July 2017). *How to enable HTTPS in a Spring Boot Java application* https://www.thomasvitale.com/https-spring-boot-ssl-certificate/