****

# Practices for Secure Software Report Project 2

Alexis Indick

Computer Science Department, Southern New Hampshire University

CS-305: Software Security

Dr. Vivian Lyon

October 16th, 2022

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions Error! Bookmark not defined.](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 5](#_Toc102040759)

[3. Deploy Cipher 6](#_Toc102040760)

[4. Secure Communications 7](#_Toc102040761)

[5. Secondary Testing 7](#_Toc102040762)

[6. Functional Testing 8](#_Toc102040763)

[7. Summary 20](#_Toc102040764)

[8. Industry Standard Best Practices 21](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/10/22** | **Alexis Indick** | **Worked on Algorithm Cipher and Certificate generation parts.** |
| **1.0** | **10/11/22** | **Alexis Indick** | **Finished Algorithm Cipher and Certificate generation parts. Started deploying the cipher.** |
| **1.0** | **10/12/22** | **Alexis Indick** | **Working on the Secure Communications part.** |
| **1.0** | **10/13/22** | **Alexis Indick** | **Worked on Summary and finished making a Secure connection, Secondary testing, and Functional testing.** |
| **1.0** | **10/14/22** | **Alexis Indick** | **Worked on the Summary.** |
| **1.0** | **10/15/22** | **Alexis Indick** | **Finished the project up.** |
| **1.0** | **10/16/22** | **Alexis Indick** | **Fixed some issues with citations.** |

## Client



## Developer

Alexis Indick

## Algorithm Cipher

The algorithm cipher best suited for Artemis Financial’s application is the AES-256-CBC. SHA-256 can also be used in conjunction with this algorithm for message digest. AES is used to this day by the United States government (Bernstein & Cobb, 2021). AES was developed in 1997 to replace DES which the government had used before until it became vulnerable to brute forcing (Bernstein & Cobb, 2021). It is a symmetric block cipher which means that the same key is used for encryption and decryption (Bernstein & Cobb, 2021). With non-symmetric, also known as asymmetric, ciphers, the algorithm uses two related keys where there is a public and a private key (Bernstein & Cobb, 2021). AES has three bit levels to choose from: 128, 192, and 256 (Bernstein & Cobb, 2021). The more bits we have, the more secure the algorithm will be (Bernstein & Cobb, 2021). AES also has random numbers it uses to keep it more difficult to crack.

AES-256-CBC is a form of AES which uses 256 characters and CBC mode aka cipher block chaining. CBC is a mode where plaintext is hidden away and block chaining every plaintext block (Mustafeez, n.d.). CBC is secure since it adds randomness to each block of plaintext (Mustafeez, n.d.).

SHA-256 is a hashing checksum algorithm consisting of 256 bits. It is used for message digesting which is just a term for hashing. Hashing is when data is made into a more secure format and is unreadable (N-able, 2019). It becomes readable only if the recipient has the key (N-able, 2019). SHA-256 is used by the United States government and requires that it be used on sensitive information from agencies (N-able, 2019). The algorithm is very secure since an attacker would need to spend 2256 tries to even try to crack it (N-able, 2019). Also, like with AES-256, SHA-256 has different bit levels and as you go up in bit levels, it gets more secure. SHA-256 is also good with preventing any collisions from happening (Callaghan, 2020). If a collision happens, a hacker could trick their way into getting the hash code the algorithm used and gain access through that way (Freeman, 2022).

## Certificate Generation

Text

Description automatically generated

Text

Description automatically generated

**Certificate File:**

**Graphical user interface, text, application, email

Description automatically generated**

## Deploy Cipher

Graphical user interface, text, application

Description automatically generated

## Secure Communications

Graphical user interface, text, application

Description automatically generated

## Secondary Testing

Graphical user interface, text, application

Description automatically generated

**Before Refactored:**

Graphical user interface, text, application, email

Description automatically generated

**After Refactored:**

**Graphical user interface, text, application, email

Description automatically generated**

## Functional Testing

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

The vulnerabilities that came up on the report were:

* **CVE-2022-27772:** This vulnerability comes from the spring-boot-starter-data-rest package and made it so an application could be susceptible to directory hijacking (NIST, 2022a). The lines of code where this vulnerability is found is in the pom.xml file and lines 422-426:

Text

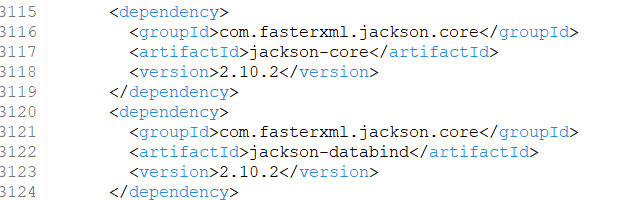
Description automatically generated

The fix to this vulnerability is to apply the patch that is addressed in recent versions of spring-boot (Leitschuh, 2022).

* **CVE-2021-22047:** This vulnerability affects the spring-data-rest-webmvc package and can allow for unauthorized access (NIST, 2021a). This vulnerability is under lines 422-426 like the previous one. To fix this vulnerability, we should upgrade to a most recent version of spring-boot (Tanzu VMware, n.d.-a).
* **CVE-2022-31679:** This vulnerability also affects spring-data-rest-webmvc and allows a hacker to create HTTP requests that could leak attributes (NIST, 2022b). The vulnerability shows up from lines 422-426 like with the previous vulnerabilities. To mitigate this vulnerability, we will need to upgrade to a more recent version of spring-boot (Tanzu VMware, n.d.-b).
* **CVE-2013-4152:** This vulnerability is a part of the spring-hateoas package and can cause possible DoS, CSRF attacks, and let attackers read files (NIST, 2014a). It is found in the pon.xml effective file which is lines 443-445.Text

  Description automatically generated

It can be mitigated by applying the patch (Red Hat, 2014a).

* **CVE-2013-7315:** This vulnerability is also found within the spring-hateoas package where it also allows for CSRF and DoS attacks along with allowing attackers to read files potentially, which is like the previous vulnerability but different since it affects different versions (NIST, 2014b). It can be found within line 443-445 like the previous one. This vulnerability can be fixed also by patching it with the latest version of spring-boot (Debian, n.d.-a).
* **CVE-2014-0054:** This vulnerability is also a part of the same package as the above two vulnerabilities and allows hackers to read files, perform DoS and CSRF attacks, and does not disable XXE resolution (NIST, 2014c). This vulnerability is also within lines 443-445 like the previous ones. The vulnerability requires an update to the spring-boot package also (Red Hat, 2014b).
* **CVE-2016-1000027:** This vulnerability could allow for RCE attacks and is from the spring-hateoas package also (NIST, 2020a). We can fix this bug by making sure we do not use Java serialization for any external end points (Baines, n.d.). I believe this vulnerability cannot be found anywhere in the code since it is mostly about us using correct practices with Java serialization.
* **CVE-2018-11039:** This vulnerability could allow for an attacker to perform an XST attack if the application has a pre-existing XSS issue (NIST, 2018a). This can be mitigated by upgrading the libspring-java packages (Beucler, 2021). This bug can be found within lines 443-445.
* **CVE-2018-11040:** This vulnerability could allow for cross-domain requests to happen (NIST, 2018b). To mitigate the issue, we must upgrade the libspring-java packages (Beucler, 2021). This flaw can be found within lines 443-445.
* **CVE-2018-1257:** This vulnerability makes it possible to expose STOMP and an attacker could just use a regular expression to DoS (NIST, 2018c). Another one that can be found within lines 443-445. To fix this issue, we must update the package (Red Hat, 2018).
* **CVE-2020-5421:** This vulnerability could make it possible for a hacker to use an RFD attack (NIST, 2020b). This flaw can be found within line 443-445. To fix this flaw, we must apply the patch (Tanzu VMware, n.d.-c).
* **CVE-2022-22950:** This vulnerability could allow a hacker to make a SpEL and cause a DoS attack (NIST, 2022c). The vulnerability can be found within lines 443-445. To mitigate this, upgrading the spring framework is ideal (Tanzu VMware, n.d.-d).
* **CVE-2022-22965:** This vulnerability could make an application vulnerable to RCE attacks (NIST, 2022d). The vulnerability can be found within lines 443-445. To solve this problem, we must upgrade versions (Tanzu VMware, n.d.-e).
* **CVE-2022-22968:** This vulnerability makes it so patterns are case sensitive for disallowedFields (NIST, 2022e). The vulnerability can be found within lines 443-445. To fix this vulnerability, we need to upgrade the spring framework and look at our DataBinder configuration (Tanzu VMware, n.d.-f).
* **CVE-2022-22970:** The last vulnerability of the spring-hateoas package is one that can make file uploading application vulnerable to DoS attacks if they need data binding (NIST, 2022f). The vulnerability can be found within lines 443-445. To mitigate it, we will need to upgrade our spring framework version (Tanzu VMware, n.d.-g).
* **CVE-2020-25649:** This vulnerability is a part of the Jackson-databind package. It can allow for XXE attacks (NIST, 2020c). This can be found in lines 3115-3124. ****

To mitigate it, we must upgrade to the latest Jackson-databind patch (Oracle, 2021).

* **CVE-2020-36518:** This vulnerability can cause an attacker to be able to use a DoS attack and a StackOverflow exception through nested objects of large depth (NIST, 2022g). This can be found along lines 3115-3124 like the previous one. To mitigate this issue, we must upgrade to the latest version of Jackson-databind (Koschany, 2022a).
* **CVE-2022-42003:** This vulnerability could cause resource exhaustion (NIST, 2022h). It can be found within the same lines 3115-3124 like the previous ones. To fix this problem, there seems to not be a patch out for it yet so we would have to wait until it is available (Nied, 2022).
* **CVE-2022-42004:** This last vulnerability for the jackson-databind package has the same issue as the previous vulnerability except only an application is vulnerable when there are certain deserialization choices chosen (NIST, 2022i). This vulnerability is within the same lines of code as before, 3115-3124. A patch is not out yet so we will have to wait (Nied, 2022).
* **CVE-2021-42550:** This vulnerability pops up for the logback-core module. This vulnerability can allow a hacker to edit configuration files if they have the right privileges (NIST, 2021b). This can be found within lines 597-601:**Text, letter

  Description automatically generated**

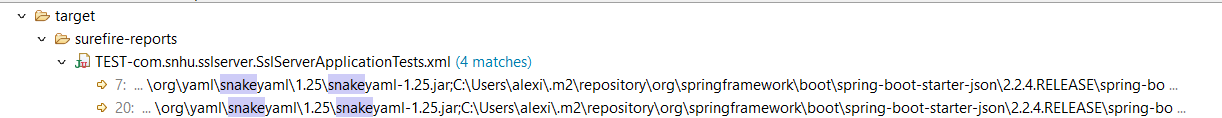
To fix this issue, we must update our logback package (Heiland, 2022).

* **CVE-2020-9488:** This vulnerability is a part of the Apache log4j2 package. It can allow for man-in-the-middle attacks (NIST, 2020d). It can be found on lines 3719-3723 since that is where the dependency is.

**Text

Description automatically generated**

To fix this, we will need to upgrade our Apache logback4j2 packages (Debian, n.d.-b).

* **CVE-2021-44228:** This vulnerability can cause not protection from attackers who can control log messages (NIST, 2021c). It can be found on lines 3719-3723 like the previous one. To mitigate this, we must upgrade our logback package (Heiland, 2022).
* **CVE-2021-44832:** This vulnerability makes it so an application is vulnerable to RCE attacks (NIST, 2021d). It can be found on lines 3719-3723 like the previous ones. To fix this issue, we must limit JNDI names (NIST, 2021d).
* **CVE-2021-45046:** This vulnerability can allow attackers who already have control of MDC input to create bad input data (NIST, 2021e). It can be found on lines 3719-3723 like the previous ones. To fix this, we must get rid of support for message look ups and not allow JNDI functionality (NIST, 2021e).
* **CVE-2021-45105:** This vulnerability can allow an attacker who has control over Thread Context Map information to make a DoS attack (NIST, 2021f). It can be found on lines 3719-3723 like the previous ones. We can mitigate this by upgrading the packages (Debian, n.d.-c).
* **CVE-2017-18640:** This vulnerability is a part of the snakeyaml package. It can allow entity expansion (NIST, 2019). I could not find exact lines of code for it, but the snakeyaml packages are in a folder on my computer. ****

To fix this issue, to update to the latest version (Fedora, 2020).

* **CVE-2022-25857:** This vulnerability makes it possible to be susceptible to DoS attacks (NIST, 2022j). This can be mitigated by upgrading the snakeyaml package (Snyk, 2022).
* **CVE-2022-38749:** This vulnerability can allow DoS attacks like the previous one, but the attacker can cause a stackoverflow (NIST, 2022k). To mitigate it, we need to upgrade the snakeyaml packages (Koschany, 2022b).
* **CVE-2022-38750:** This vulnerability is the same as the previous one since they both can allow an attack to perform a DoS attack and cause a crash via stackoverflow (NIST, 2022l). We can fix this issue by upgrading the snakeyaml package (Koschany, 2022b).
* **CVE-2022-38751:** This vulnerability is just like the previous two vulnerabilities involving potential DoS attacks and stackoverflow crashes (NIST, 2022m). It can be mitigated by upgrading the snakeyaml packages (Koschany, 2022b).
* **CVE-2022-38752:** This vulnerability has the same issues as the previous vulnerabilities (NIST, 2022n). It can be mitigated by updating the packages (Koschany, 2022b).
* **CVE-2020-10693:** This vulnerability belongs to the hibernate-validator package. It allows for invalid EL expressions (NIST, 2020e). Lines 2283-2287 is where the dependency is for hibernate.**Graphical user interface, text

  Description automatically generated**

To mitigate this vulnerability, we should make sure to patch the dependency with the latest version of hibernate (Oracle, 2022a).

* **CVE-2021-27568:** This vulnerability belongs to the json-smart package. The vulnerability could make programs crash since it has an exception that can be thrown that sometimes does not get caught (NIST, 2021g). It does not seem to appear in the pom.xml but I see that it is in a folder on my computer.

****

To fix this problem, we must update the json-smart package (Oracle, 2022a).

* **CVE-2021-31684:** Last vulnerability for the json-smart package and it can cause a DoS attack if the attacker makes a bad web request (NIST, 2021h). To mitigate it, we will have to upgrade our version of json-smart (Oracle, 2022b).
* **CVE-2021-22060:** This is a vulnerability that popped up for spring-boot. This vulnerability can allow for malicious input (NIST, 2021i). It can be found in lines 4810-4814.

Text, letter

Description automatically generated

To fix this vulnerability, we need to upgrade our spring version (Tanzu VMware, n.d.-h).

* **CVE-2021-22096:** This vulnerability allows for malicious input also (NIST, 2021j). It also is within lines 4810-4814. To mitigate this issue, we should upgrade our spring version (Tanzu VMware, n.d.-i).
* **CVE-2021-22118:** This vulnerability makes an application that uses WebFlux vulnerable to privilege escalation (NIST, 2021k). It can be found within lines 4810-4814 like the previous two. To solve this problem, we must upgrade our spring version (Tanzu VMware, n.d.-j).
* **CVE-2022-22971:** This vulnerability is the last for the spring framework issues. This vulnerability makes it possible for a DoS attack to happen (NIST, 2022o). This bug can be found also in lines 4810-4814. To mitigate it, we need to upgrade the spring version (Tanzu VMware, n.d.-k).
* **CVE-2019-17569:** This vulnerability is from the Apache tomcat package. HTTP Request Smuggling is possible with this one (NIST, 2020f). In lines 1861-1865 is where the tomcat core is and where are issues are going to be with the tomcat vulnerabilities. Graphical user interface, text

  Description automatically generated

To mitigate this problem, we must upgrade our tomcat packages (Debian, n.d.-d).

* **CVE-2020-11996:** With this vulnerability, it may spark a high CPU usage response for a couple of seconds (NIST, 2020g). This can be found also within lines 1861-1865. To fix this, we will have to upgrade (Debian, n.d.-e).
* **CVE-2020-13934:** This vulnerability can allow an OutOfMemoryException to happen if enough requests happen and leads to a DoS attack (NIST, 2020h). This can be found also within lines 1861-1865. To fix this issue, we need to upgrade our tomcat packages (Debian, n.d.-e).
* **CVE-2020-13935:** This vulnerability could cause an infinite loop and lead to a DoS attack (NIST, 2020i). This can be found also within lines 1861-1865. We must upgrade the tomcat packages to mitigate this issue (Debian, n.d.-e).
* **CVE-2020-13943:** This vulnerability could lead users to see responses from resources they were not expecting (NIST, 2020j). This can be found also within lines 1861-1865. We must upgrade the tomcat packages (Debian, n.d.-f).
* **CVE-2020-17527:** This vulnerability can lead to a leak between requests of information (NIST, 2020k). This can be found also within lines 1861-1865. We must upgrade the tomcat packages (Debian, n.d.-f).
* **CVE-2020-1935:** This vulnerability allows certain invalid HTTP headers to be valid and can cause a possibility of HTTP Request Smuggling (NIST, 2020l). This can be found also within lines 1861-1865. We can fix this issue by upgrading our packages for tomcat (Debian, n.d.-d).
* **CVE-2020-1938:** This vulnerability allows attackers to be able to exploit an HTTP connection since tomcat treats AJP as if it has a higher trust than a HTTP connection (NIST, 2020m). This can be found also within lines 1861-1865. We can fix this issue by upgrading our packages for tomcat (Debian, n.d.-d).
* **CVE-2020-8022:** This vulnerability can cause a root exploit due to improper permissions (NIST, 2020n). This can be found also within lines 1861-1865. To fix this, we just need to apply a patch (openSUSE, 2020).
* **CVE-2020-9484:** This vulnerability a hacker can be able to control the contents of a file and make an RCE attack, but it takes a bit for the hacker to be able to get to that point (NIST, 2020o). This can be found also within lines 1861-1865. To fix this, we will have to upgrade the package (Debian, n.d.-e).
* **CVE-2021-24122:** This vulnerability makes it so some JSP source can be disclosed (NIST, 2021l). This can be found also within lines 1861-1865. To mitigate it, we must upgrade our tomcat packages (Gladky, 2021).
* **CVE-2021-25122:** This vulnerability can cause duplicate headers and make user B able to see the result of user A’s request (NIST, 2021m). This can be found also within lines 1861-1865. To mitigate it, we must upgrade our tomcat packages (Gladky, 2021).
* **CVE-2021-25329:** This vulnerability is an incomplete fix from CVE-2020-9484 (NIST, 2021n). This can be found also within lines 1861-1865. We can mitigate this issue by applying the patch for it (Debian, n.d.-g).
* **CVE-2021-30640:** This vulnerability allows a hacker to authenticate using different variations from a username that are valid (NIST, 2021o). This can be found also within lines 1861-1865. To fix this, we must upgrade the tomcat packages (Debian, n.d.-h).
* **CVE-2021-33037:** This vulnerability makes request smuggling an issue since HTTP headers are not being parsed correctly (NIST, 2021p). This can be found also within lines 1861-1865. To fix this, we must upgrade the tomcat packages (Debian, n.d.-h).
* **CVE-2021-41079:** This vulnerability can incorrectly validate incoming TLS packets (NIST, 2021q). This can be found also within lines 1861-1865. To mitigate this, we must update the tomcat packages (Debian, n.d.-i).
* **CVE-2021-43980:** This vulnerability could make it possible for client connections to share a HTTP11Processor instance (NIST, 2021r). This can be found also within lines 1861-1865. This can be mitigated by upgrading tomcat (Thomas, 2022a).
* **CVE-2022-29885:** This was a documentation mistake where it was incorrectly said that the EncryptInterceptor enabled clustering on an untrusted network (NIST, 2022p). This can be found also within lines 1861-1865. To fix this mistake, it is advised to use a VPN when running a cluster on an untrusted network (Thomas, 2022b).
* **CVE-2022-34305:** This vulnerability allows for a web application to display user data (NIST, 2022q). This can be found also within lines 1861-1865. To fix this issue, we need to update the tomcat packages (Gentoo Linux, n.d.).

## Summary

Cryptography is one of the processes of the Vulnerability Assessment Process Diagram that was implemented for this project. I used SHA-256 for the cryptography portion since encryption is an essential element in keeping data safe. Especially for a financial institution like Artemis Financial since they will be dealing with sensitive user data. Lines 30-60 is where the hash function was created.

Another essential element to keep in mind from the diagram is client/server. The client/server element can be seen through the testing part with the checksum and how the server responded to our request to display it. The information we made code for sent it from the client to the server to be seen by the user. By setting up the ServerController class starting on line 30, I was able to make a connection between the client and server.

The next element that was a focus for the project was code quality. The quality of the code does matter here since it is important to use comments and make easy-to-understand function names for a person reviewing the code to understand. If we do not have readability, it could make revising the code harder later.

APIs were utilized in this project also through HTTPS and having a viewable interface. The RESTful API for this project made a secure connection when we generated a self-signed certificate, so we were able to have a secure browsing session which is important if anyone were to use our application. They will be protected from their information being leaked thanks to HTTPS.

Lastly, code error was an important element also since we would not be able to run the application if there were errors. We would also potentially be allowing users to use an application that has bugs, and this could in turn provide an effortless way in for an attacker to steal information through our errors. Having exceptions that can catch issues in code for example, on line 32, I put an “Exception” keyword so that an exception is thrown if there was an issue with the myHash() method. That way I can figure out where the problem is quick, and the application will stop.

When I was adding security components to the application, the first layer was adding the certificate. This way, we have the certificate immediately ready for when we connect so the platform stays secure. Then, I added the SHA-256 cipher to make sure the company will have their data encrypted correctly. Lastly, I ran a vulnerability test to see what vulnerabilities were lurking on the application. There were a lot, but I was able to research the best ways to mitigate them easily and it involved mostly updating dependencies to the latest versions.

## Industry Standard Best Practices

The standard best practices I implemented in this project were: using SHA-256, using a HTTPS secure connection, and identifying vulnerabilities along with solutions to them. Using a hash function such as SHA-256 protects from attackers trying to steal information. An HTTPS connection is essential if we want to protect user data also. A HTTP connection will not cut it since it is not secure. So, generating a secure certificate will whether it is from a trusted third party or self-signed, will benefit Artemis Financial a lot. Lastly, I suggest to Artemis Financial to keep up to date with any packages that the web application runs. I identified the vulnerabilities and packages that need to be updated along with solutions to these issues. This will help Artemis Financial stay up to date with potentially dangerous issues and be able to mitigate them quickly. I also suggest giving the least amount of privilege possible since it will help with mitigating potential attacks from people who would abuse the power. Users, for example, would not have the ability to delete other users account or check other user’s accounts. That would be giving them too much privilege.

**Reference**

Baines, J. (n.d.). *[R2] Pivotal spring framework httpinvokerserviceexporter readremoteinvocation method untrusted Java deserialization*. Tenable. Retrieved October 15, 2022, from https://www.tenable.com/security/research/tra-2016-20

Bernstein, C., & Cobb, M. (2021, September 24). What is the advanced encryption standard (aes). TechTarget. Retrieved October 10, 2022,  
from <https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard>

Beucler, S. (2021, April 23). *[Security] [DLA 2635-1] libspring-java security update*. Debian. Retrieved October 15, 2022, from https://lists.debian.org/debian-lts-announce/2021/04/msg00022.html

Callaghan, P. (2020, August 19). *Why you should use SHA-256 in evidence authentication*. Pagefreezer Blog. Retrieved October 11, 2022, from <https://blog.pagefreezer.com/sha-256-benefits-evidence-authentication#:~:text=Collisions%20are%20incredibly%20unlikely%3A%20There,the%20exact%20same%20hash%20value>

Debian. (n.d.-a). *DSA-2842-1 libspring-java – denial of service*. Debian Security Advisory. Retrieved October 15, 2022, from <https://www.debian.org/security/2014/dsa-2842>

Debian. (n.d.-b). *DSA-5020-1 apache-log4j2 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2021/dsa-5020>

Debian. (n.d.-c). *DSA-5024-1 apache-log4j2 – security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2021/dsa-5024>

Debian. (n.d.-d). *DSA-4673-1 tomcat8 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2020/dsa-4673>

Debian. (n.d.-e). *DSA-4727-1 tomcat9 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2020/dsa-4727>

Debian. (n.d.-f). *DSA-4835-1 tomcat9 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2021/dsa-4835>

Debian. (n.d.-g). *DSA-4891-1 tomcat9 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2021/dsa-4891>

Debian. (n.d.-h). *DSA-4952-1 tomcat9 -- security update*. Debian. Retrieved October 16, 2022, from <https://www.debian.org/security/2021/dsa-4952>

Debian. (n.d.-i). *DSA-4986-1 tomcat9 -- security update*. NVD. Retrieved October 16, 2022, from https://www.debian.org/security/2021/dsa-4986

Fedora. (2020, April 24). *[SECURITY] Fedora 32 update: snakeyaml-1.26-1.fc32*. Fedora. Retrieved October 16, 2022, from https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/CKN7VGIKTYBCAKYBRG55QHXAY5UDZ7HA/

Freeman, J. B. (2022, January 25). *Hash collisions explained*. Freeman Law. Retrieved October 11, 2022, from <https://freemanlaw.com/hash-collisions-explained/>

Gentoo Linux. (n.d.). *Apache tomcat: Multiple vulnerabilities — glsa 202208-34.* Gentoo Linux. Retrieved from https://security.gentoo.org/glsa/202208-34

Gladky, A. (2021, March 16). *[SECURITY] [DLA 2596-1] tomcat8 security update*. Debian. Retrieved October 16, 2022, from https://lists.debian.org/debian-lts-announce/2021/03/msg00018.html

Heiland, M. (2022, July 21). *Open-Xchange security advisory 2022-07-21*. Seclists. Retrieved October 16, 2022, from https://seclists.org/fulldisclosure/2022/Jul/11

Koschany, M. (2022a, May 2). *[Security] [DLA 2990-1] Jackson-databind security update*. Debian. Retrieved October 15, 2022, from <https://lists.debian.org/debian-lts-announce/2022/05/msg00001.html>

Koschany, M. (2022b, October 3). *[Security] [DLA 3132-1] snakeyaml security update*. Debian. Retrieved October 3, 2022, from https://lists.debian.org/debian-lts-announce/2022/10/msg00001.html

Leitschuh, J. (2022, February 15). *Temporary directory hijacking to local privilege escalation vulnerability in org.springframework.boot:spring-boot*. GitHub. Retrieved October 14, 2022, from <https://github.com/JLLeitschuh/security-research/security/advisories/GHSA-cm59-pr5q-cw85>

Mustafeez, A. Z. (n.d.). *What is CBC?* Educative. Retrieved October 10, 2022, from <https://www.educative.io/answers/what-is-cbc>

N-able. (2019, September 12). *SHA-256 algorithm overview.* N-able. Retrieved October 11, 2022, from <https://www.n-able.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20a%20patented,as%20long%20as%20when%20unencrypted>

Nied, P. (2022, October 6). *CVE-2022-42003 (high) Detected in jackson-databind-2.13.4.jar #2146*. GitHub. Retrieved October 16, 2022, from https://github.com/opensearch-project/security/issues/2146

NIST. (2014a, January 23). *CVE-2013-4152*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2013-4152>

NIST. (2014b, January 23). *CVE-2013-7315*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2013-7315>

NIST. (2014c, April 17). *CVE-2014-0054*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2014-0054>

NIST. (2018a, June 25). *CVE-2018-11039*. NVD. Retrieved October 15, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2018-11039

NIST. (2018b, June 25). *CVE-2018-11040*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2018-11040>

NIST. (2018c, May 11). *CVE-2018-1257*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2018-1257>

NIST. (2019, December 11). *CVE-2017-18640*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2017-18640>

NIST. (2020a, January 2). *CVE-2016-1000027*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2016-1000027>

NIST. (2020b, September 19). *CVE-2020-5421*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-5421>

NIST. (2020c, December 3). *CVE-2020-25649*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-25649>

NIST. (2020d, April 27). *CVE-2020-9488*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-9488>

NIST. (2020e, May 6). *CVE-2020-10693*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-10693>

NIST. (2020f, February 24). *CVE-2019-17569*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2019-17569>

NIST. (2020g, June 26). *CVE-2020-11996*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-11996>

NIST. (2020h, July 14). *CVE-2020-13934*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-13934>

NIST. (2020i, July 14). *CVE-2020-13935*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-13935>

NIST. (2020j, October 12). *CVE-2020-13943*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-13943>

NIST. (2020k, December 3). *CVE-2020-17527*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-17527>

NIST. (2020l, February 24). *CVE-2020-1935*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-1935>

NIST. (2020m, February 24). *CVE-2020-1938*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-1938>

NIST. (2020n, June 29). *CVE-2020-8022*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-8022>

NIST. (2020o, May 20). *CVE-2020--9484*. NVD. Retrieved October 16, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2020-9484

NIST. (2021a, October 28). *CVE-2021-22047*.NVD. Retrieved October 14, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-22047>

NIST. (2021b, December 16). *CVE-2021-42550*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-42550>

NIST. (2021c, December 10). *CVE-2021-44228*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-44228>

NIST. (2021d, December 28). *CVE-2021-44832*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-44832>

NIST. (2021e, December 14). *CVE-2021-45046*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-45046>

NIST. (2021f, December 18). *CVE-2021-45105*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-45105>

NIST. (2021g, February 22). *CVE-2021-27568*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/cve-2021-27568>

NIST. (2021h, June 1). *CVE-2021-31684*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-31684>

NIST. (2021i, January 10). *CVE-2021-22060*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-22060>

NIST. (2021j, October 28). *CVE-2021-22096*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-22096>

NIST. (2021k, May 27). *CVE-2021-22118*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-22118>

NIST. (2021l, January 14). *CVE-2021-24122*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-24122>

NIST. (2021m, March 1). *CVE-2021-25122*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-25122>

NIST. (2021n, March 1). *CVE-2021-25329*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-25329>

NIST. (2021o, July 12). *CVE-2021-30640*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-30640>

NIST. (2021p, July 12). *CVE-2021-33037*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-33037>

NIST. (2021q, September 16). *CVE-2021-41079*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2021-41079>

NIST. (2021r, September 28). *CVE-2021-43980*. NVD. Retrieved October 16, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2021-43980

NIST. (2022a, March 30). *CVE-2022-27772*.NVD. Retrieved October 14, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-27772>

NIST. (2022b, September 21). *CVE-2022-31679*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-31679>

NIST. (2022c, April 1). *CVE-2022-22950*. NVD. Retrieved October 15, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2022-22950

NIST. (2022d, April 1). *CVE-2022-22965*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-22965>

NIST. (2022e, April 14). *CVE-2022-22968*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-22968>

NIST. (2022f, May 12). *CVE-2022-22970*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-22970>

NIST. (2022g, March 11). *CVE-2022-36518*. NVD. Retrieved October 15, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2020-36518>

NIST. (2022h, October 2). *CVE-2022-42003*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-42003>

NIST. (2022i, October 2). *CVE-2022-42004*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-42004>

NIST. (2022j, August 30). *CVE-2022-25857*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-25857>

NIST. (2022k, September 5). *CVE-2022-38749*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-38749>

NIST. (2022l, September 5). *CVE-2022-38750*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-38750>

NIST. (2022m, September 5). *CVE-2022-38751*. NVD. Retrieved October 16, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2022-38751

NIST. (2022n, September 5). *CVE-2022-38752*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-38752>

NIST. (2022o, May 12). *CVE-2022-22971*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-22971>

NIST. (2022p, May 12). *CVE-2022-29885*. NVD. Retrieved October 16, 2022, from <https://nvd.nist.gov/vuln/detail/CVE-2022-29885>

NIST. (2022q, June 23). *CVE-2022-34305*. NVD. Retrieved October 16, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2022-34305

openSUSE. (2020, June 29). *[security-announce] openSUSE-SU-2020:0911-1: important: Security update for tomcat*. openSUSE. Retrieved October 16, 2022, from https://lists.opensuse.org/archives/list/security-announce@lists.opensuse.org/message/UKH3XHJOALCLYY5SDDGU4G3ZG5TYAXXI/

Oracle. (2021). *Oracle critical patch update advisory – July 2021*. Oracle. Retrieved October 15, 2022, from <https://www.oracle.com/security-alerts/cpujul2021.html>

Oracle. (2022a). *Oracle critical patch update advisory – April 2022*. Oracle. Retrieved October 16, 2022, from <https://www.oracle.com/security-alerts/cpuapr2022.html>

Oracle. (2022b). *Oracle critical patch update advisory – July 2022*. Oracle. Retrieved October 16, 2022, from <https://www.oracle.com/security-alerts/cpujul2022.html>

Red Hat. (2014a, February 25). *RHSA-2014: 0212 – Security advisory*. Red Hat. Retrieved October 15, 2022, from <https://access.redhat.com/errata/RHSA-2014:0212.html>

Red Hat. (2014b, April 14). *RHSA-2014:0400 – Security advisory*. Red Hat. Retrieved October 15, 2022, from <https://access.redhat.com/errata/RHSA-2014:0400.html>

Red Hat. (2018, June 7). *RHSA-2018:1809 – Security advisory*. Red Hat. Retrieved October 15, 2022, from <https://access.redhat.com/errata/RHSA-2018:1809>

Snyk. (2022, August 29). *Denial of service (DoS)*. Snyk Vulnerability Database. Retrieved October 16, 2022, from https://security.snyk.io/vuln/SNYK-JAVA-ORGYAML-2806360

Tanzu VMware. (n.d.-a). *CVE-2021-22047: Potential security bypass for customized spring data rest resource*. Tanzu VMware. Retrieved October 14, 2022, from <https://tanzu.vmware.com/security/cve-2021-22047>

Tanzu VMware. (n.d.-b). *CVE-2022-31679: Potential unintended data exposure for resource exposed by spring data rest*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2022-31679>

Tanzu VMware. (n.d.-c). *CVE-2020-5421: RFD protection bypass via jessionid*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2020-5421>

Tanzu VMware. (n.d.-d). *CVE-2022-22950: Spring expression dos vulnerability*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2022-22950>

Tanzu VMware. (n.d.-e). *CVE-2022-22965: Spring framework rce via data binding on jdk 9+*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2022-22965>

Tanzu VMware. (n.d.-f). *CVE-2022-22968: Spring framework data binding rules vulnerability*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2022-22968>

Tanzu VMware. (n.d.-g). *CVE-2022-22970: Spring framework dos via data binding to multipartfile or servlet part*. Tanzu VMware. Retrieved October 15, 2022, from <https://tanzu.vmware.com/security/cve-2022-22970>

Tanzu VMware. (n.d.-h). *CVE-2021-22060: Additional log injection in spring framework (follow-up to CVE-2021-22096)*. Tanzu VMware. Retrieved October 16, 2022, from <https://tanzu.vmware.com/security/cve-2021-22060>

Tanzu VMware. (n.d.-i). *CVE-2021-22096: Log injection in spring framework*. Tanzu VMware. Retrieved October 16, 2022, from https://tanzu.vmware.com/security/cve-2021-22096

Tanzu VMware. (n.d.-j). *CVE-2021-22118*: *Local privilege escalation within spring webflux multipart request handling*. Tanzu VMware. Retrieved October 16, 2022, from <https://tanzu.vmware.com/security/cve-2021-22118>

Tanzu VMware. (n.d.-k). *CVE-2022-22972: Spring framework dos with stomp over websocket*. Tanzu VMware. Retrieved October 16, 2022, from <https://tanzu.vmware.com/security/cve-2022-22971>

Thomas, M. (2022a, September 28). *[SECURITY] CVE-2021-43980 Apache tomcat - information disclosure*. Apache. Retrieved October 16, 2022, from <https://lists.apache.org/thread/3jjqbsp6j88b198x5rmg99b1qr8ht3g3>

Thomas, M. (2022b, May 10). *[SECURITY] CVE-2022-29885 Apache tomcat encryptinterceptor dos*. Apache. Retrieved October 16, 2022, from https://lists.apache.org/thread/2b4qmhbcyqvc7dyfpjyx54c03x65vhcv