****

# CS-305 Project One

# Artemis Financial Vulnerability Assessment Report

**Table of Contents**

[Document Revision History 3](#_heading=h.gjdgxs)

[Client 3](#_heading=h.30j0zll)

[Instructions 3](#_heading=h.1fob9te)

[Developer 4](#_heading=h.3znysh7)

[1. Interpreting Client Needs 4](#_heading=h.2et92p0)

[2. Areas of Security 4](#_heading=h.tyjcwt)

[3. Manual Review 4](#_heading=h.3dy6vkm)

[4. Static Testing 4](#_heading=h.1t3h5sf)

[5. Mitigation Plan 4](#_heading=h.4d34og8)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **15NOV2022** | **Kyle Wucik** | **First Draft** |

## Client



## Developer

Kyle Wucik

## Interpreting Client Needs

The client, Artemis Financial, is a financial services company and requires top security in their application. Top security will be used to secure the sensitive information that the customers would give to Artemis Financial (AF). This sensitive information could include social security numbers, debit card numbers, or bank account information. In short, this information could be very valuable and is a high target.

AF might have customers with foreign money and this will require the ability to exchange different currencies securly. Depending on where the company is based, there might be restrictions on data and/or communication services from the Government. AF will have to read up on local government restrictions and guidelines to ensure they are familiar with them.

There will be many external threats now or in the future that are trying to get into the company’s system. They could do this in a variety of ways including an injection attack, a spoof attack, or even brute force. To make sure the client has the best security they will need to make sure all of their plugins are up to date, there could be a vulnerability that is being exploited but it’s patched in the new RELEASE, this includes all of the third party libraries.

## Areas of Security

* 1. **Input Validation -** Review the command input function used in the application and make improvements. It must be secure enough to prevent injection or another attack.
  2. **API (application program interface) -** Review and make improvements to the RESTful API Artemis Financial uses. This resource can be identified by the URL and protects the authentication credentials in transit.
  3. **Cryptography -** We will have to follow the requirements from the Federal Financial Institution Examination Council, you can read more from the reference cited.
  4. **Client-Server -** Review and make improvements in the mutually authenticated 5 client-side certificates this will help to provide protection. Their RESTful API has client-server as primary connector. REST service must only provide HTTPS endpoints where access control and JSON Web Tokens integrity can be confirmed.
  5. **Code Error -** Review all of the command inputs and make a function that can handle errors for input, system calls, database queries, and any other internal functions.
  6. **Code Quality -** Review the existing code and rewrite any of the software that impedes the security of our client’s data.
  7. **Encapsulation -** Review the existing code and rewrite the lines so that encapsulation is implemented properly.

## Manual Review

I only found a few vulnerabilities in my manual review. I found that the input validation has issues with the CRUDController which makes the input change into a string directly and inside the DocData there is a todo task comment that looks like it is to protect the code.

## Static Testing

Below are the dependencies, their common vulnerabilities and exposures, and how to fix them:

1) bcprov-jdk15on-1.46.jar - This .jar is a container for the JCE and lightweight API for the Bouncy Castle Crypto package (Java implementation of cryptographic algorithms).

1. CVE-2013-1624 - I don’t believe there is a fix for this yet.
2. CVE-2015-6644 (OSSINDEX) - I don’t believe there is a fix for this yet.
3. CVE-2015-7940 (OSSINDEX) - I don’t believe there is a fix for this yet.
4. CVE-2016-1000338 - Need to add length check for sequence in DSA signature (bcgit, n.d.)
5. CVE-2016-1000339 - Need to add better support for DH domain parameters (bcgit, n.d.)
6. CVE-2016-1000341 - Need to add randomizer to DSA signature generation (bcgit, n.d.)
7. CVE-2016-1000342 - Need to add header validation for INTEGER/ENUMERATED (bcgit, n.d.)
8. CVE-2016-1000343 - Need to update default DSA parameters to follow 186-4 (bcgit, n.d.)
9. CVE-2016-1000344 - Need to remove support for non-cbc mode ciphers in IES/ECIES (bcgit, n.d.)
10. CVE-2016-1000345 - Modify the IESEngine so that MAC check is the primary one; added general BadBlockException class for asymmetric ciphers. (bcgit, n.d.)
11. CVE-2016-1000346 - Need to add TLS validation check for DH keys and further agreement result checks (bcgit, n.d.)
12. CVE-2016-1000352 - Remove support for non-cbc mode cipers in IES/ECIES (bcgit, n.d.)
13. CVE-2017-13098 - Need to confirm size of decrypted PMS before using (bcgit, n.d.)
14. CVE-2018-5382 - Do not rely on the version one BKS keystore files, it isn’t cryptographically sound and it is recommended that a more robust keystore format be used (CMU, 2018)
15. CVE-2020-0187 (OSSINDEX) - I don’t believe there is a fix for this yet.
16. CVE-2020-26939 (OSSINDEX) - I don’t believe there is a fix for this yet.

2) spring-boot-starter-web-2.2.4.RELEASE.jar

1. CVE-2022-27772 - Update spring boot to version 2.2.11.RELEASE or later, it has been patched (JLLeitschuh, n.d.)

3) spring-boot-2.2.4.RELEASE.jar

1. CVE-2013-4152 - Need to add ‘processExternalEntities’ property to the JAXB2Marshaller (poutsma, 2013)
2. CVE-2013-7315 - Need to add ‘processExternalEntities’ property to the JAXB2Marshaller (poutsma, 2013)
3. CVE-2014-0054 - Hard code the property for resolving external entities to false (Spase Markovski, 2014)
4. CVE-2016-1000027 - “This vulnerability has been modified and is currently undergoing reanalysis” (NST, 2022)
5. CVE-2018-11039 - Update the version to the latest one available (Spring by Pivota, 2018l)
6. CVE-2018-11040 - Update the version to the latest one available (Spring by Pivotal, 2018)
7. CVE-2018-1257 - Update the version to the latest one available (Spring by Pivotal, 2018)
8. CVE-2020-5421 - No fix at this time
9. CVE-2022-22950 - Update the version to the latest one available (Spring by Pivotal, 2022)
10. CVE-2022-22965 - Update the version to the latest one available (Spring by VMware, 2022)
11. CVE-2022-22968 - Update the version to the latest one available (Spring by VMware, 2022)
12. CVE-2022-22970 - Update the version to the latest one available (Spring by VMware, 2022)
13. CVE-2022-27772 - Couldn’t find fix

4) logback-core-1.2.3.jar - Logback-core module

1. CVE-2021-42550 - Couldn’t find fix

5) log4j-api-2.12.1.jar - Apache’s Log4j API

1. CVE-2020-9488 - Update the version to the latest one available (Apache, 2021)
2. CVE-2021-44228 - Update the version to the latest one available (Apache, 2021)
3. CVE-2021-44832 - Update the version to the latest one available (Apache, 2021)
4. CVE-2021-45046 - Update the version to the latest one available (Apache, 2021)
5. CVE-2021-45105 - Update the version to the latest one available (Apache, 2021)

6) snakeyaml-1.25.jar - YAML 1.1 parer and emitter for Java

1. CVE-2017-18640 - Can’t find fix
2. CVE-2022-25857 - Restrict nested depth for collections to avoid DoS attacks (asomov, 2022)
3. CVE-2022-38749 - update software (bitbucket, n.d.)
4. CVE-2022-38750 - update software (bitbucket, n.d.)
5. CVE-2022-38751 - update software (bitbucket, n.d.)
6. CVE-2022-38752 - update software (bitbucket, n.d.)
7. CVE-2022-41854 - update software (bitbucket, n.d.)

7) jackson-databind-2.10.2.jar - General data-bind function for Jackson, this works on core streaming API

1. CVE-2020-25649 - Update to latest version (redhat, 2020)
2. CVE-2020-36518 - Update to latest version
3. CVE-2022-42003 - Update to latest version
4. CVE-2022-42004 - Update to latest version

8) tomcat-embed-core-9.0.30.jar - Core Tomcat implementation

1. CVE-2019-17569 - Update to latest version
2. CVE-2020-11996 - Update to latest version
3. CVE-2020-13934 - Update to latest version
4. CVE-2020-13935 - Update to latest version
5. CVE-2020-13943 - Update to latest version
6. CVE-2020-17527 - Update to latest version
7. CVE-2020-1935 - Update to latest version
8. CVE-2020-1938 - Update to latest version
9. CVE-2020-8022 - Update to latest version
10. CVE-2020-9484 - Update to latest version
11. CVE-2021-24122 - Update to latest version
12. CVE-2021-25122 - Update to latest version
13. CVE-2021-25329 - Update to latest version
14. CVE-2021-30640 - Update to latest version
15. CVE-2021-33037 - Update to latest version
16. CVE-2021-41079 - Update to latest version
17. CVE-2021-43980 - Update to latest version
18. CVE-2022-29885 - Update to latest version
19. CVE-2022-34305 - Update to latest version
20. CVE-2022-42252 - Update to latest version

9) hibernate-validator-6.0.18.Final.jar - Hibernate’s Bean Validation

1. CVE-2020-10693 - Update to the latest version available.

10) spring-core-5.2.3.RELEASE.jar - Spring Core

1. CVE-2016-1000027 - Update to latest version
2. CVE-2020-5421 - Update to latest version
3. CVE-2021-22060 - Update to latest version
4. CVE-2021-22096 - Update to latest version
5. CVE-2021-22118 - Update to latest version
6. CVE-2022-22950 - Update to latest version
7. CVE-2022-22965 - Update to latest version
8. CVE-2022-22968 - Update to latest version
9. CVE-2022-22970 - Update to latest version
10. CVE-2022-22971 - Update to latest version

I noticed that as I got further down the list the less specific the fixes became. Either this or they were just more and more difficult to actually find what they did besides updating the RELEASE version.

## Mitigation Plan

My mitigation plan is really quite simple. First, all of the spring framework dependencies must be updated, then the apache, TomCat, Jackson, and hibernator as well. Then we can recode anything in the software that is vulnerable, like the CrudeController file’s authentication.

After everything is solved and fixed, the best practice would be to run the dependency check again. When this is done it gives a better understanding of what is fixed, what stayed, and if there are anymore vulnerabilities that weren’t present prior, rinse and repeat.

**References**

Southern New Hampshire University. (n.d.). Software security: Module three. Brightspace.

Retrieved November 20, 2022, from

https://learn.snhu.edu/d2l/le/content/1196944/Home?itemIdentifier=D2L.LE.Content.Con

tentObject.ModuleCO-20833842

bcgit, bc-java, (n.d.), GitHub repository. Retrieved November 18, 2022, from

https://github.com/bcgit/bc-java

JLLeitschuh, security-research, (n.d.), GitHub repository. Retrieved November 19, 2022, from

https://github.com/JLLeitschuh/security-research

Carnegie Mellon University. VU#306792 - Bouncy Castle BKS-V1 keystore files vulnerable to

trivial hash collisions. (2018, March 19). Retrieved November 20, 2022, from

https://www.kb.cert.org/vuls/id/306792

poutsma, spring-projects (2013), GitHub repository. Retrieved November 22, 2022, from

https://github.com/spring-projects/spring-framework/pull/317

Federal Financial Institutions Examination Council. (2021, August). Authentication and Access

to Financial Institution Services and Systems. ffiec.gov. Retrieved November 22, 2022,

from

https://www.ffiec.gov/press/pdf/Authentication-and-Access-to-Financial-InstitutionServic

es-and-Systems.pdf

Spase Markovski, spring-projects (2014), GitHub repository. Retrieved November 22, 2022, from

https://github.com/spring-projects/spring-framework/issues/16003

asomov, snakeyaml (2022), GitHub repository, Retrieved November 22, 2022

https://github.com/snakeyaml/snakeyaml/commit/fc300780da21f4bb92c148bc902572012

20cf174