****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **03/16/2023** | **Philip Trinh** |  |

## Client



## Developer

Philip Trinh

## Interpreting Client Needs

The client Artemis Financial is a consulting company who providing financial plans for their customers.

* The client’s company will be collecting a lot of private information regarding to their customers include investment profiles, insurance, retirement, and saving accounts. Therefore, security of their website, software programming interface must be top notch and constantly review to adapt and update if necessary to prevent possible exploitation of any vulnerabilities that may cause a leak of client’s information (bank accounts, saving accounts, investments, credit card, etc.), delay in accessing content, or other technical problems. Any of these problems if occur and not addressed can cost the company to lose clients, financially cripple if substantial fund is stolen by hackers, etc.
* Artemis Financial working with transactions for customers from both international and domestic transactions.
* For U.S government specifically, there are cyber security law to protect customers information such as the Gramm–Leach–Bliley Act (GLBA) which requires financial institutions to protect customer data and honestly disclose all data-sharing practices with customers, then the Financial Industry Regulatory Authority (FINRA) is an organization that established a set of rules for protecting customer data from compromise if Artemis Financial is register with them. For international laws the Payment Services Directive (PSD 2) is applied for European Union countries that includes regulations for protecting online payments, enhancing customer data security, and strong customer authentication (e.g., multi-factor authentication). For Asia countries such as Vietnam and China, there are Cyber security laws that protect the customers data and private online information as well. In summary, Artemis Financial must ensure a secure communication and heighten security for online transactions, financial information and need to comply by domestic and international laws so that the customers’ information is fully protected.
* External threat that presenting now within the current Artemis Financial system is that “RESTful API is collection of URLs, in which HTTP calls to URI and in response, it serves JSON or XML data… REST APIs are vulnerable to common and well known OWASP attacks such as injection, CSRF, Cross site script, XMLExternalEntity, etc.” (Rapid7, 1-2).
* Future threat to the company could be other form of attack such as Phishing attacks, which are used by hackers that pretend to be the company website and trick people into clicking on a malicious link and fill out confidential information. SQL injection which hacker manipulates a standard SQL query. Denial-of-Service which attackers target systems, servers, or networks and flood them with traffic to exhaust their resources and bandwidth. Malware attacks such as virus, spyware, adware, etc. can get through network by letting users click a dangerous link and download the virus to their computer and allow the hackers access control to the customers’ information. All these attacks are aimed to cause problem to the network and try to steal the customers’ personal information to receive monetary gain.
* The modernization of the company website may include open-source libraries which are very useful for computer programmers as they provide access to reusable, pre-written, frequently used codes, which drastically reduces the workload as programmers can reference this code instead of writing everything from scratch. Open-source library get constantly updated by a talent group of individuals from the community who aim to provide better and quicker improvement to the technical community projects, troubleshoot any problem that come up. Therefore, Artemis Financial should check this library regularly for update to fix and patch any vulnerabilities that found.

Also, adding cloud technologies are very helpful and provide great solution to data storage. Cloud service company often provides a basic amount of website security for their customers and the advantage of not owning physical server with the choice of updating storage space base on usage will help the company expand as much as they need to. However, it is recommended that the Artemis Financial adding their own security for server, website, data transfer, storages as the cloud storage companies are not fully responsible for sensitive information using their service.

## Areas of Security

* Input validation – the website application does collect users’ input, so validation of such input is necessary to avoid SQL injection or potential error.
* APIs – Secure API interaction is very important for the company as the company has a RESTful web application programming interface (API) and is built on Spring Framework. The APIs allow the company and the customers to exchange information securely over the internet.
* Cryptography – Cryptography is required as the company will be dealing with secure communications domestic and international. Cryptography of data will help the company to meet with the government export regulations requirement.
* Code quality –Secure coding practice/pattern ensure that the code is good, it is a superset of input validation and APIs.

## Manual Review

Manual review of vulnerability of the provided code base, there are multiple concern:

* Outdated version of springframework. boot: In pom.xml, the version listed is out of date “2.2.4. RELEASE”. The current available version is 3.0. The latest version should be used to ensure the application using the most up to date dependency that may include more security feature than the older version.
* Absent of access control: There is no use of access control such as username and passwords while using API. This will lead to problem as users need to be limit on what can be see and features, they are allowed to use.
* Missing input validation: for example in Greeting controller and CRUDcontroller, the current code let the program take in whatever value input and use that as greeting names and business name. This is not right as it can cause complication and overload of data if customer is not limited in input control (possible DDOS attack).
* Fail to work API: The program giving too much information on API unsecured. This fail causes the missing link between users and the company, cause a lost in interaction interface.
* Absent of cryptography: there is no conversion type of data encryption appear anywhere in the provided code. This left the customers’ account number and deposit exposed. Artemis Financial would need to develop data encryption such as via encryption of JSON, XML data, SSL to store information and send international transactions.
* Database connection parameters are hard coded in DocData: meaning giving the web link and set username and password for accessing. This is very unsecure practice and can easily get exploit by hackers to gain access to the system.

## Static Testing

## bcprov-jdk15on-1.46.jar

**Description:**

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.

[**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338), [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342) , [**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343)**,** [**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352), [**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341)**,** [**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345) , [**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098)**,** [**CVE-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522), **CVE-2020-0187** (OSSINDEX), [**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339)**, CVE-2020-26939** (OSSINDEX), [**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940)**,** [**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382), [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)**,** [**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346)**, CVE-2015-6644** (OSSINDEX).

**Solution**: Update Bouncy Castle to version 1.60 or later:

**hibernate-validator-6.0.18.Final.jar**

**Description:**

Hibernate's Bean Validation (JSR-380) reference implementation. A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages

[**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693)

**Solution:** Update Fixed In Version hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, hibernate-validator 6.0.20.Final or later versions

.

**jackson-databind-2.10.2.jar**

**Description:**

General data-binding functionality for Jackson: works on core streaming API

[**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649) ,[**CVE-2020-36518**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518)**,** [**CVE-2022-42003**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42003)**,** [**CVE-2022-42004**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42004)

**Solution:** upgrade your jackson-databind packages newer than 2.14 version.

**log4j-api-2.12.1.jar**

**Description:**

The Apache Log4j API. Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.

[**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488)

**Solution**: update to version Apache Log4j 2.12.3 and 2.13.1

**logback-core-1.2.3.jar**

**Description:**

logback-core module. In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

[**CVE-2021-42550**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550)

**Solution**: update to version later than 1.2.7 such as Logback 1.3.x or 1.4.x

**snakeyaml-1.25.jar**

**Description:**

YAML 1.1 parser and emitter for Java

**CVE-2022-1471** (OSSINDEX) , [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) , [**CVE-2022-25857**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-25857)**,** [**CVE-2022-38749**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38749)**,** [**CVE-2022-38751**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38751)**,** [**CVE-2022-38752**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38752)**,** [**CVE-2022-41854**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-41854)**,** [**CVE-2022-38750**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38750)

**Solution:** Migrate to SnakeYAML Engine. It has a configuration option to restrict aliases for collections (the aliases for scalars cannot grow and they are not restricted)

**spring-boot-2.2.4.RELEASE.jar and**

**spring-boot-starter-web-2.2.4.RELEASE.jar**

**Description:**

Spring Boot. \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

[**CVE-2022-27772**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772)

**Solution**: update and use latest version of SpringBoot.

**spring-core-5.2.3.RELEASE.jar**

**and spring-webmvc-5.2.3.RELEASE.jar**

**Description:**

Spring Core. A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

[**CVE-2022-22965**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965)**,** [**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)**,** [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)**,** [**CVE-2022-22950**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950)**,** [**CVE-2022-22971**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971)**,** [**CVE-2022-22968**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968)**,** [**CVE-2022-22970**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970)**,** [**CVE-2021-22060**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)**,** [**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)

**Solution:** Update latest version of Spring Core.

**tomcat-embed-core-9.0.30.jar**

**and tomcat-embed-websocket-9.0.30.jar**

**Description:**

Core Tomcat implementation.

[**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938)**,** [**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996) , [**CVE-2020-1393**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934)**4,** [**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935)**,** [**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527), [**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122)**,** [**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)**,** [**CVE-2022-29885**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885)**,** [**CVE-2022-42252**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252)**,** [**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484)**,** [**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329), [**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)**,** [**CVE-2022-34305**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305)**,** [**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122)**,** [**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037)**,** [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)**,** [**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935)**,** [**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943) , [**CVE-2021-43980**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980)

**Solution**: Upgrade to Apache Tomcat to 10.0.6 or later

## 5 Mitigation Plan

**Action Plan for local vulnerabilities:**

* Update pom.xml spring boot version to the most up to date 3.0 version instead of 2.2.4.
* Implement an access control method such as role base with username and password or random two factor authentication is very good.
* Input Validation- limit characters and range for input to avoid overload the system. The use of special characters such as accent should be avoided to prevent complications with system read.
* Fix API protocol to ensure working properly with GET, POST, etc. to ensure good interaction between the website application and customers.
* Implement encryption such as Secure Distributed Computing- SSL, encrypting the XML and JSON data for private information such as customers’ account numbers, financial information and other personal information. Exporting of data to meet the international protocols and regulations.
* Remove hard-coded database connection on DocData or implement change password method and encrypt this information to protect username and password.

**Action Plan for dependency vulnerabilities:**

* With dependency vulnerabilities, the easiest fix is to update the most current version of library and dependency that affected since these problems are already fixed in the newer version. Refer to solutions from part 4 for specific version update and dependency problems.

References

*What is an Open Source Library? Definition and FAQs | HEAVY.AI*. (n.d.).

Retrieved March 17, 2023, from Www.heavy.ai.

<https://www.heavy.ai/technical-glossary/open-source-library#:~:text=Benefits%20of%20Open%20Source%20Libraries>

*How to Test a REST API | AppSpider Documentation*. (n.d.). Docs.rapid7.com. Retrieved March 18, 2023 from https://docs.rapid7.com/appspider/how-to-test-a-rest-api/#:~:text=REST%20APIs%20are%20vulnerable%20to

‌

*Top 12 Cybersecurity Regulations for Financial Services | UpGuard*. (n.d.).

Retrieved March 18, 2023 from Www.upguard.com. https://www.upguard.com/blog/cybersecurity-regulations-financial-industry

‌

M, S. (2022, November 11). *10 Types of Cyber Attacks You Should Be Aware in [2021]*.

Retrieved March 18, 2023 from Simplilearn.com. https://www.simplilearn.com/tutorials/cyber-security-tutorial/types-of-cyber-attacks