****

# CS 305 Project One

**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** | **9.17.2021** | **Gary Clark** |  |

## Client



## Developer

Gary Clark

## 1. Interpreting Client Needs

At Artemis Financial, clients place trust in the organization to handle their financial wellbeing which can have cascading effects across their entire livelihood. It is both an ethical and legal obligation to handle this important personal and financial information in a responsible way. To better serve their client base, Artemis Financial has implemented a web application utilizing the REST API which provides easy access to account management 24/7. However, despite the many benefits that this service presents, the nature of the application opens the door to a wide variety of new threats to the confidentiality, integrity, and availability of their client’s information. As such, we at Global Rain are tasked with providing a security analysis which can help to identify any vulnerabilities withing the system.

When formulating this analysis, there are certain factors which must be considered involving the method of business, government regulations, and both the short-term and long-term challenges which the system may face. The Artemis web platform will allow clients to manage their accounts at any time and by extension that will also mean at any place. Therefore, we can infer without knowing anything about their client base that this system will have to validate information from anywhere in the world. The other consideration regarding the geographic location of the users involves adhering to governmental regulations around the transfer and communication of certain types of protected data. For example, the United States passed the Gramm-Leach Bliley Act in 1999 which requires that financial institutions such as Artemis must ensure the security of information pertaining to names, phone numbers, financial accounts, income, credit histories, and social-security numbers. The European Union passed a similar piece of legislation called the General Data Protection Regulation (GDPR) which services as a comprehensive act requiring the security of user’s information. While these regulations will vary in terminology, we must utilize best practices as presented by organizations such as the NIST to maintain the three pillars of data security: Confidentiality, Integrity, and Availability.

Threats to these pillars will come from a multitude of actors including both internal and external whose goals will vary immensely. While we cannot foresee every potential scenario, we can make predictions based on current data and design the system to uphold the highest of security standards in defense of these threats. Many of these external threats will come in the form of malware, phishing, trojan, or ransomware-based attacks. According to the 2021 Cyber Security Threat Trends Report by Cisco, 86% of organizations had at lest one user try to connect to a phishing site. In this same report 48% of organizations found information-stealing malware which led Cisco to confirm a continuation of a trend towards more complex, multi-staged attacks involving multiple threat types. Many of these attacks rely on some form of social engineering where users are deceived into loading malicious software onto the system. These threats must be considered when analyzing the Artemis web application and current trends for the near future must also be accounted for.

Artemis is not alone in facing these threats as more systems move online to provide for easier access and this transition is leading to a variety of new technologies. The nature of the tech world is one of constant evolution and the ability to upgrade or modernize a system must also be considered. Open-source APIs such as the REST framework can help by allowing more users to access and thereby discover new vulnerabilities present within the code. As a result, these vulnerabilities can be made public where organizations can then move to patch these weaknesses. With evolving web application technologies, new security issues can arise at any time and the need for swift detection is paramount. Some organizations also exist to bolster support to these APIs and create new frameworks such as the Spring framework. These frameworks can provide the benefit of offloading some of the maintenance and vulnerability detection to a third-party. Regardless of the framework used, a web application should always be using the most up-to-date stable release version of any API which will provide the greatest resistance to existing threats. Our analysis will take all these elements into consideration with the goal of providing Artemis Financial with the most secure and stable platform to provide their clients with.

## 2. Areas of Security

This section aims to identify which aspects of data security best apply to the Artemis web application which give our team direction and focus when analyzing the system.

* **Input Validation**
  + Users of this application will be accessing protected information which should remain private and confidential. As such, we must ensure that any entity viewing this information is authorized to do so. This will require the application to accurately validate the user’s credentials before allowing access to the system. Compliance in this regard will help prevent data breaches and many types of phishing attacks maintaining the clients’ trust in the organization. By authenticating each user, Artemis can ensure that it is following all regulations set forth by governments across the world.
* **APIs**
  + This application has already been created using the REST API so the successful integration of this is crucial to the success of the application. Importantly, the usage of the newest release versions of the APIs will enable the best defense against known vulnerabilities and prevent attackers for exposing older versions of the API.
* **Cryptography**
  + With much of the data transmitted through the application pertaining to sensitive information under the protection of multiple governmental regulations, it is important to adhere to the best practices while sending and receiving data. Notably, personal client information should be packaged and encrypted before it is transmitted or transported to another location. Proper utilization of cryptography will help prevent hijacking or interception-based attacks and will maintain the confidentiality of users’ information.
* **Client/Server**
  + Again, as this application is web-based, there is a necessary focus on securing the interactions between the client and the server. Improper implementation of this type of framework can leave an application vulnerable to many types of attacks such as hijacking, XML injection, and cross-scripting. Therefore, we must check that the APIs handling these interactions do not suffer from any known vulnerabilities and ensure that there are fail safes in place should the connection be compromised.

These areas will be the primary focus of the analysis, but it is important to note that all aspects of the vulnerability assessment process do apply in some way. The three which were omitted from the above list include code error, code quality, and encapsulation. All of these do apply to this application, but to a lesser extent than the other four. Inaccurate entries in the login process or within the account management functions will require some form of error handling to prevent a critical system failure in the face of unexpected data. Encapsulation and code quality will apply to the data structures with the applications code, but through following best practices, issues with either of those should be avoided. Securing a system always requires consideration of all aspects of coding, but depending on the architecture of the application, different elements may play larger roles than others.

## 3. Manual Review

This section will provide the findings of our manual inspection of the code and document any areas of concerns. Each concern will have visual documentation as well as a technical explanation.

Exhibit 1:

Text

Description automatically generated

Within the “customer” class there were several instances of improper access control definitions which could allow attackers to view confidential information. The most glaring was the variable which holds the value for the clients account balance which is currently inheriting the class’s public level of access. Any personal information should be stored in private variables using getting and setting data structures to prevent external code from viewing the data. The method which returns the client’s account number is also currently set to be a public function which makes it visible to all parts of the application. Although the account number variable is only visible to this class, attackers can still infiltrate the application and run the “showInfo” method which will return the private value for the account number. This poses a risk to the confidentiality of the client’s data and corrective action should be taken.

Exhibit 2:

Text

Description automatically generated

In a similar fashion to the vulnerabilities present within the “customer” class, the “DocData” class also presents some potential risks to confidential data. The class is currently public which makes it visible to all areas of the code, but the more concerning area is the method which returns the private value for “id”. The hidden nature of the variable “id” can be defeated if an attacker can hijack the “getId” function which will allow further access to even more sensitive data. Again, corrective actions must be taken to ensure the integrity and confidentiality of the client’s data.

Exhibit 3:

Text

Description automatically generated

The final vulnerability discovered through our manual review stems from the version of the Spring Boot API. This dependency is crucial for creating a framework for the application to function and the version included with the Artemis application is 2.2.4.RELEASE. This is an older version which can be vulnerable to certain types of attacks which the newest stable release version of 2.5.4 does not suffer from.

## 4. Static Testing

**Dependency Check Report**

Graphical user interface, text, application, email

Description automatically generated

Vulnerabilities:

1. Bouncy Castle Crypto Package
   * Dependencies: bcprov-jdk15on-1.46.jar
   * CVE Count: 17
   * Severity: Unknown
   * Description: This package serves to implement cryptographic algorithms into Java which adds to the concern regarding the multiple documented vulnerabilities. Of the published vulnerabilities, CVE-2018-5382 is the most concerning as it applies to all versions prior to 1.47 of the Bouncy Castle API. The default BKS keystore uses an HMAC of only 16bits which can allow attackers to compromise the integrity of the keystore. This will allow attackers to infiltrate the application and gain access to confidential information. As this improper validation of integrity check directly affects the cryptography of the application, failing to remedy this issue can put Artemis in violation of governmental regulations regarding the security of certain protected data types.
2. Hibernate’s Bean Validation \**POTENTIAL FALSE POSITVE\**
   * Dependencies: hibernate-validator-6.0.18.Final.jar
   * CVE Count: 1
   * Severity: Medium
   * Description: This issue involves the message interpolation processor which allows invalid EL expressions to return as valid. This can allow attackers to bypass input sanitation controls. CVE-2020-10693 specifies version 6.1.2.Final which could mean this is a false positive as our version is 6.0.x.x. Regardless, newer versions may remedy the issue.
3. Jackson Data-bind
   * Dependencies: jackson-databind-2.10.2.jar
   * CVE Count: 1
   * Severity: High
   * Description: This flaw involves the FasterXML Jackson Databind where entity expansion is not secured properly. This creates a vulnerability to external entity attacks which can affect the data integrity. The severity is high, but the vulnerability has been modified and is currently undergoing reanalysis according to CVE-2020-25649 on the NIST NVD Database.
4. Apache Log4j API
   * Dependencies: log4j-api-2.12.1.jar
   * CVE Count: 1
   * Severity: Low
   * Description: This issue is caused by improper certificate validation with a host mismatch in the SMTP appender. Attackers can use this vulnerability to intercept SMTPS connections and leak data. This issue has been resolved with the latest versions 2.14.1 of the API and is awaiting reanalysis according to CVE-2020-9488.
5. YAML Parser \**POTENTIAL FALSE POSITVE\**
   * Dependencies: snakeyaml-1.25.jar
   * CVE Count: 1
   * Severity: High
   * Description: The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation which allows for attackers to cause a denial-of-service attack. This may be a false positive as CVE-2017-18640 only specifies version 1.18 and we are using 1.25. The newest versions have been confirmed to fix the issue. While a high-level vulnerability, versions 1.26 or newer resolve the issue.
6. Spring Framework
   * Dependencies: spring-aop-5.2.3.RELEASE.jar, spring-core-5.2.3.RELEASE.jar
   * CVE Count: 2
   * Severity: High
   * Description: Spring AOP and Spring Core are dependent on the Spring Framework which is vulnerable to an RFD attack (CVE-2020-5421). Certain browsers can allow attackers to use a jsession path parameter to bypass previous protections. Also, Webflux applications are vulnerable to a privilege escalation. The newest version 5.3.9 should not have this issue.
7. Core Tomcat Implementation
   * Dependencies: tomcat-embed-core-9.0.30.jar, tomcat-embed-websocket-9.0.30.jar
   * CVE Count: 14, 15
   * Severity: Critical
   * Description: Multiple vulnerabilities have been identified ranging from medium impact to critical impact. The most severe of the issues involves the usage of the AJP where Tomcat gives it a higher trust than an HTTP connection. The AJP connector should be disabled, but it was shipped with it enabled by default (CVE-2020-1938). Attackers can read and write to files and make remote code executions through this vulnerability. Version 9.0.31 has hardened the default configuration. Version 9.0.30 is quite outdated and the latest versions of 9.0.5x or higher have fixed many of these issues.

## 5. Mitigation Plan

The analysis of the Artemis application shows that in it’s current state, there are many vulnerabilities which can put the organization in direct contention with it’s obligation to both governmental entities and their client base. Most of these issues can allow attackers to compromise the application and gain access to client information such as the account number, account balance, or their personal data. The vulnerabilities identified range from deviations to best data structure practices to outdated APIs with known weaknesses. Each vulnerability will have a unique solution, but many related to the dependencies can be fixed by using newer versions of the respective APIs. The issues identified in the manual review should be easily remedied by reworking the access levels of the java classes within the application. The suggested fixes for the identified vulnerabilities as related to the respective areas of security are listed below.

Input Validation:

Hibernate’s Bean Validation (CVE-2020-10693)

Maven Dependencies -> hibernate-validator-6.0.18.Final.jar

Ensure API version is the newest stable release to prevent input sanitation bypass attacks.

Input Validation

DocData.java: lines 21 -> 31

Define wrapper around public native methods

APIs:

Multiple APIs

Maven Dependencies -> tomcat-embed-core-9.0.30.jar

* tomcat-embed-websocket-9.0.30.jar
* spring-aop-5.2.3.RELEASE.jar
* spring-core-5.2.3.RELEASE.jar
* log4j-api-2.12.1.jar
* jackson-databind-2.10.2.jar

Verify APIs are using the newest stable release versions and update dependencies.

Cryptography:

Bouncy Castle Crypto Package (CVE-2018-5382)

Maven Dependencies -> bcprov-jdk15on-1.46.jar

Update API to prevent cryptographic key exposure improperly formatted plaintext data.

Client/Server:

Spring Boot Application v2.2.4.RELEASE.

RestServiceApplication.java, spring-boot-2.2.4.RELEASE.jar

Migrate to v2.5.4 of the Spring Boot framework to ensure secure application environment for client usage.

Code Quality:

Secure Coding Practices

customer.java: lines 4 -> 9

Ensure proper access allocation of variables to hide certain data from external entities.

Secure Coding Practices

DocData.java: lines 5 -> 20

Modify data structures using hidden or guarded constructor as outlined by practices in *Secure Coding Guidelines for Java SE* (v8.0)

Resources

Secure Coding Guidelines for Java SE (September 2020). Input Validation - Mutability. (5 & 6). Retrieved from <https://www.oracle.com/java/technologies/javase/seccodeguide.html>

National Institute of Standards and Technology. (n.d.). *National Vulnerability Database*. NVD. Retrieved September 19, 2021, from <https://nvd.nist.gov/>.

Gramm-Leach-Bliley Act, 5 U.S.C. § 501-508 (November 12th, 1999). <https://www.congress.gov/bill/106th-congress/senate-bill/900/text>

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)

OJ L 119, 4.5.2016, p. 1–88 (BG, ES, CS, DA, DE, ET, EL, EN, FR, GA, HR, IT, LV, LT, HU, MT, NL, PL, PT, RO, SK, SL, FI, SV)

<https://eur-lex.europa.eu/eli/reg/2016/679/oj>

Cisco. (2021). (rep.). 2021 Cyber Security Threat Trends. Retrieved September 19, 2021, from https://umbrella.cisco.com/info/2021-cyber-security-threat-trends-phishing-crypto-top-the-list?utm\_medium=search-paid&amp;utm\_source=google&amp;utm\_campaign=UMB\_22Q1\_NA\_EN\_GS\_Nonbrand\_Threats&amp;utm\_term=pgm&amp;utm\_content=UMB-FY21-Q4-content-ebook-2021-cyber-security-threat-trends&amp;\_bt=531409955734&amp;\_bk=cybersecurity+threats&amp;\_bm=p&amp;\_bn=g&amp;\_bg=122023015112&amp;gclid=CjwKCAjw-ZCKBhBkEiwAM4qfFyAF8qSUrKztHxX5fZZS1\_fGh-vx0as3E1aeZMWjzSbN58Uu-MONXxoCQtMQAvD\_BwE.