GLOBALRAIN

CS 305 Project One
Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	[Date]	[Your Name]	

Client

ARTEMIS FINANCIAL

Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found. Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

Developer
Kevin Schroeder

1. Interpreting Client Needs

Cybersecurity is crucial for any company in the finance industry. Many are trusting Artemis Financial with keeping their money and out of the hands of hackers. Not only is Artemis Financial trusted with their customer's money, but they also have access to very important personally identifiable information. If this information got into the wrong person's hands, the customer could find themselves in financial turmoil for years, if not forever. Artemis Financial must stay on top of its cybersecurity efforts to keep the trust of its customers and gain the trust of new customers.

Artemis Financial may have clients in different countries, which means that they must disclose all the information about the exchange rates, fees & taxes payable for the transfer amount to be transferred. This information is additional PII that must be handled with care. In addition, the United States government's oversight of financial services cybersecurity reflects a complex mix of state and federal laws, regulators, regulations, and guidance. They recognize that cybersecurity is critical to protecting the vital services to the economy provided by the financial sector. In recognition of the importance of the information systems that support financial services, the government has increasingly focused on cybersecurity concerns by issuing regulations and various forms of guidance. We will need to reach out to the office of our state's senator to ensure that our API is meeting all legal requirements.

Before diving into the code and coming up with a mitigation plan, some cybersecurity attacks are becoming increasingly more common in the fin-tech space. One dangerous type of cybersecurity threat to customers is account takeover. This is when a hacker gains access to a customer's account and then changes information on it so that the real owner does not have access to it nor control of the account. This attack is often a result of a technique called credential stuffing, which is when hackers use computers to keep inserting various credentials until they break into an account. The criminals also may use the account to commit identity theft. Account takeover statistics show that the number of these attacks nearly doubled from 2017 to 2018.

Phishing attacks are another common cyberattack that we must be aware of. Phishing is when a hacker gains access to the login credentials of another user. Many of these cyberattacks occur when customers or employees click on an email or link that looks harmless. The email may state that their account has been compromised and they need to log in with new information and the hacker will intercept the login credentials without the account holder knowing. Alternatively, this strategy may be used to install malware on the computer system that can give attackers full control of a user's system.
The last common cyberattack that could affect Artemis Financial now or shortly is ransomware. Ransomware is a type of malware that may be planted as a result of a phishing attack. This malware is particularly dangerous because takes over a victim's computer system by encrypting data and making it impossible for the owner to access it unless they pay a large fee. Many of these attacks target financial companies because the criminals expect that they can afford large payouts.

Fortunately, we can use techniques to modernize the cybersecurity of the network and mitigate the chance of Artemis becoming a victim to a cybersecurity attack. The first technique is to apply the principle of least privilege. A person should have the required set of permissions to make requests to our API and each request will be verified from the very begging. the second technique is called purpose-driven access. Familiar methods, such as a “one-time password” sent to an email address, are too prone to compromise. Instead, access should be contextual and time-bound. Password-less multifactor authentication is both more secure and faster for users than multiple password reset and insecure email delivery.

Another modernization technique is to build a culture that normalizes frequent security updates and proactively identifying security vulnerabilities and risks. Most security solutions focus on compliance rather than risk and typically are reactive after an attack. While compliance is always necessary, additional technology risks and vulnerabilities are contextual. This is why Artemis Financial must regularly install security updates for their existing dependencies and thoroughly vet any open source library that is put on the system.

The last modernization technique is to encrypt personally identifiable information and any sensitive information that may be gathered by the system. The developers on the project will have access to the database, which stores all of the customer’s personal information. To protect them and the Artemis Financial employees, all personally identifiable information and password will be encrypted on the backend.

2. Areas of Security

Input Validation - Input will block improperly formed data from entering our system. Because it is difficult to detect a malicious user who is trying to attack our application, all input entering our system must be validated. Hackers commonly use input fields to send malicious queries to a system's database to extract private information. To keep our customers safe, we will enforce tight input validation for every input field on the application.

Secure API Interactions - Artemis Financial built a REST API to connect its user interface to the database. This type of communication can lead to data if it's not managed properly and kept up to date with the latest security techniques. To ensure that our data is secure, we will need to implement best practices for secure API interactions.

Code Quality - Code quality goes with the first two areas of security. Both input validation and secure API interactions require high-quality code that is tailored for security.

Cryptography - As mentioned previously, we recommend encrypting sensitive information in the database to protect Artemis Financial employees and its customers. We will ensure that the process in which this information is encrypted and decrypted is secure and prevents data leaks.

3. Manual Review

Overall

- No authorization on any of the API endpoints.
- No encryption of sensitive data.
- No limit user limits on API hits.
- No sanitization of any input values. We must remove potentially hazardous characters like: <, >, ", ', %, (,), &, +, \,)
- No validation on input data length.
- No validation on input data types.
- No validation on input data range (where necessary).

CRUD.java

- Content1 and content2 parameters need to be validated before added to the instance.

CRUDController.java

- The name parameter isn't used.

DocData.java

- Constructor serves no purpose. The compiler will automatically provide an instance with no arguments.
- Authorization must be added to the getID method.
- Database name and password are still defaults.

Greeting.java

- Constructor parameters must be validated.
- getID and getContent don't have authorization.

GreetingController.java

- name parameter is not validated.

myDateTime.java

- All class variables are not encapsulated.
- Input is not validated on either method.
- No authorization on either method.

4. Static Testing
Run a dependency check on Artemis Financial's software application to identify all security vulnerabilities in the code. Record the output from the dependency check report. Include the following:

a. The names or vulnerability codes of the known vulnerabilities
b. A brief description and recommended solutions provided by the dependency check report
c. Attribution (if any) that documents how this vulnerability has been identified or documented previously

bcprov-jdk15on-1.46.jar

- **Issue:** CVE-2013-1624 - The TLS use in the Bouncy Castle Java package before 1.48 doesn't properly consider the timing of side-channel attacks on a noncompliant MAC check operation. Side-channel attacks are when an attacker adds malicious software to an application rather than exploiting an already-established algorithm. This can also lead to the software being vulnerable to plaintext attacks, where the attacker is able to decrypt secure information.
- **Solution** - We can suppress this issue. The Bouncy-Castle code does careful sanity checking of the padding length (as indicated by the last byte of plaintext) but treats the padding as having length 1. This deviates slightly from the recommendation of the RFCs to treat the padding as having length zero.

- **Issue:** CVE-2013-1624 - Bouncy Castle JCE Provider v. 1.55 and earlier doesn't validate ASN.1 encoding of signature on verification correctly. This leaves the application vulnerable to database injections that can manipulate private data.
- **Solution** - Back up existing installation of Red Hat Fuse, including all applications, configuration files, databases and database settings, and so on. Install Fuse 7.1 security update.

- **Issue:** CVE-2016-1000339 - The AESFastEngine class in Bouncy Castle JCE Provider v 1.55 is vulnerable to data leaking on the AES key being used.
- **Solution** - Upgrade to AES v 1.56.

- **Issue:** CVE-2016-1000341 - In Bouncy Castle JCE Provider v 1.55 and earlier, the DSA signature generation is vulnerable to timing attack and could allow an attacker access to the signature's k value.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2016-1000342 - Bouncy Castle JCE Provider v. 1.55 and earlier doesn't validate ASN.1 encoding of signature on verification correctly. This leaves the application vulnerable to database injections that can manipulate private data.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61 (same as for issue CVE-2013-1624).

- **Issue:** CVE-2016-1000343 - Bouncy Castle JCE Provider v 1.55 and earlier generates a weak private key when the default values aren't changed. This can leave our application vulnerable to attack through brute force attacks.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2016-1000344 - In Bouncy Castle JCE Provider version 1.55 and earlier the DHIES allows for encryption through the electronic codebook model. This is the simplest form of encryption and has been regarded as unsafe.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2016-1000345 - In Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES allows for CBC mode, which is commonly associated and vulnerable to a padding oracle attack, where attachers can decrypt private data.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2016-1000346 - In the Bouncy Castle JCE Provider v 1.55 and earlier the other party DH public key is not validated correctly and can cause data leaking of the connector's private key.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2016-1000352 -In the Bouncy Castle JCE Provider v 1.55 and earlier the ECIES allows for encryption through the electronic codebook model. This is the simplest form of encryption and has been regarded as unsafe.
-
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2017-13098 - If BouncyCastle TLS prior to version 1.0.3 is configured with JCE (Java Cryptography Extension) for cryptographic functions, the application provides a weak Bleichenbacher oracle if a TLS cipher suite using RSA key exchange is negotiated. This leaves the application vulnerable to leaking private key data.
- **Solution** - Upgrade to Bouncy Castle JCE Provider version 1.61 and/or don't use JCE for cryptographic functions.

- **Issue:** CVE-2018-1000613 - Legion of the Bouncy Castle Java Cryptography APIs 1.58 and 1.59 contains a CWE-470, the use of Externally-Controlled Input to Select Classes or Code vulnerability in the private key deserialization that could deserialize a private key and execute unexpected code
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2018-5382 - The default BKS keystore uses an HMAC that is only 16 bits long, which is not long enough for today's secure keystore standards and leads the application vulnerable to brute-force attaches.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

- **Issue:** CVE-2020-26939 - There is a security flaw in the error handling of invalid input that leaves the application vulnerable to data leaks and database injections.
- **Solution** - Upgrade to Bouncy Castle JCE Provider v 1.61.

classmate-1.5.1.jar

- **False positive:** There were no CVE codes provided and there aren't any known issues that I found through research.

hibernate-validator-6.0.18.Final.jar

- **Issue:** CVE-2020-10693 - Hibernate Validator version 6.1.2.Final contains a known code quality error that enables invalid EL expressions to be evaluated as if they were valid.
- **Solution** - Apply the interim fix or Fix Pack containing APAR PH29942 for each named product as soon as practical.

jackson-core-2.10.2.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

jackson-databind-2.10.2.jar

- **Issue:** CVE-2020-25649 - FasterXML Jackson Databind conatins a known code quality error where it doesn't handle entity expansion properly, which could lead to attacks on data integrity.
- **Solution** - Upgrade to FasterXML Jackson Databind v 2.10.5.

jakarta.annotation-api-1.3.5.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

jakarta.validation-api-2.0.2.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

jboss-logging-3.4.1.Final.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

jul-to-slf4j-1.7.30.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

log4j-api-2.12.1.jar

- **Issue:** CVE-2020-9488 - Apache Log4j SMTP appender improperly validates certificates. This could allow an SMTPS connection to be exploited by a man-in-the-middle attack where an attacker could gain access to log messages sent through that appender.
- **Solution** - Upgrade to SmtpAppender version 2.13.2.

logback-core-1.2.3.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

slf4j-api-1.7.30.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

snakeyaml-1.25.jar

- **Issue:** CVE-2017-18640 - The Alias feature contained within SnakeYAML 1.18 allows entity expansion, which can allow attackers to define entities within our application.
- **Solution** - Upgrade to SnakeYAML v 1.26.

spring-boot-2.2.4.RELEASE.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

spring-core-5.2.3.RELEASE.jar

- **Issue:** CVE-2020-5421 - The protections against RFD attacks in Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions can be bypassed depending on the browser the attacker is using. Certain browsers will allow exploitation of the application through a jsessionid path parameter.
- **Solution** - Upgrade to Spring Framework v 5.2.15.

- **Issue:** CVE-2021-22118 - WebFlux application in Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7 is vulnerable to authorization errors and allows attackers to elevate their privileges on the application.
- **Solution** - Upgrade to Spring Framework v 5.2.15.

tomcat-embed-core-9.0.30.jar

- **Issue**: CVE-2019-17569 - Tomcat 9.0.28 - 9.0.30, 8.5.48 - 8.5.50, and 7.0.98 - 7.0.99. Invalid Transfer-Encoding headers are not processed correctly. This may lead to HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a specific manner.
- **Solution:** Upgrade TomEE plus(7.0.7) with Apache Tomcat 9.0.42

- **Issue:** CVE-2020-11996 - A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 -10.0.0-M5, 9.0.0.M1 -9.0.35 and 8.5.0 -8.5.55 could trigger high CPU usage for several seconds and could lead to the server becoming unresponsive.
- **Solution**: Upgrade to Apache Tomcat 9.0.42

- **Issue**: CVE-2020-13934 - An h2c direct connection to Apache Tomcat 10.0.0-M1 - 10.0.0-M6, 9.0.0.M5 - 9.0.36 and 8.5.1 - 8.5.56 didn't remove the HTTP/1.1 processor after the upgrade to a HTTP/2 processor. If there are a lot of hits to the server, an OutOfMemoryException could occur and the server will crash.
- **Solution**: Run this command in the console to install the security patch: openSUSE Leap 15.1: zypper in -t patch openSUSE-2020-1102=1

- **Issue**: CVE-2020-13935 - The length of a payload in a WebSocket frame is incorrectly validated in Apache Tomcat 10.0.0-M1 - 10.0.0-M6, 9.0.0.M1 - 9.0.36, 8.5.0 - 8.5.56 and 7.0.27 to 7.0.104. This coud lead to an infinite loop, which would cause a ton of hits to the server at once and possibly crashing the server as well.
- **Solution**: Run the following command in the console to install the security patch: openSUSE Leap 15.1: zypper in -t patch openSUSE-2020-1102=1

- **Issue**: CVE-2020-13943 - If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 - 10.0.0-M7, 9.0.0.M1 - 9.0.37 or 8.5.0 - 8.5.57 goes over the maximum number of concurrent streams for a connection, another request made on that connection could contain HTTP headers - including HTTP/2 pseudo headers - from a previous request rather than the intended headers. This could lead to a server response from an unexpected source
- **Solution:** Run the following command in the console to install the security patch: openSUSE Leap 15.2: zypper in -t patch openSUSE-2020-1799=1

- **Issue**: CVE-2020-17527 - Apache Tomcat 10.0.0-M1 - 10.0.0-M9, 9.0.0-M1 - 9.0.39 and 8.5.0 - 8.5.59 could re-use an HTTP request header value from the past stream received on an HTTP/2 connection for the request for the next streams. This could possibly lead to data leak between requests.
- **Solution:** Upgrade to Apache Tomcat 9.0.42

- **Issue:** CVE-2020-1935 - In Apache Tomcat 9.0.0.M1 - 9.0.30, 8.5.0 - 8.5.50 and 7.0.0 - 7.0.99 the HTTP header parsing uses a technique that could allow some invalid HTTP headers to be parsed as valid. This could lead to HTTP Request Smuggling if Tomcat was located behind a reverse proxy that does not handle the invalid Transfer-Encoding header correctly.
- **Solution:** Upgrade TomEE plus to version 7.0.7 with Apache Tomcat to verson 9.0.42

- **Issue:** CVE-2020-1938 - Tomcat regards AJP connections as more trustworthy than a similar HTTP connection. AJP connections are available to an attacker, they can be exploited. In Apache Tomcat 9.0.0.M1 - 9.0.0.30, 8.5.0 - 8.5.50, and 7.0.0 - 7.0.99, Tomcat AJP Connector is enabled by default. It was recommended that this Connector would be disabled if not required.
- **Solution**: Upgrade to Apache Tomcat 9.0.42 or disable AJP connections

- **Issue:** CVE-2020-9484 - Apache Tomcat versions 10.0.0-M1 - 10.0.0-M4, 9.0.0.M1 - 9.0.34, 8.5.0 - 8.5.54 and 7.0.0 - 7.0.103 if:

    a) an attacker can control the contents and name of a file on the server, and
    b) the server uses the PersistenceManager with a FileStore, and
    c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter="null" or a sufficiently relaxed filter to allow the attacker's object to be deserialized, and
    d) the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over;

  then, the attacker will be able to trigger remote code execution. Note that all the conditions a) to d) must be true for the attack to succeed.
- **Solution**: Upgrade to Apache Tomcat 9.0.42

- **Issue**: CVE-2021-24122 - Apache Tomcat versions 10.0.0-M1 - 10.0.0-M9, 9.0.0.M1 - 9.0.39, 8.5.0 - 8.5.59, and 7.0.0 - 7.0.106 were vulnerable to JSP source code disclosure in some configurations when they were serving resources from a network location using the NTFS file system.
- **Solution**: Upgrade to Apache Tomcat 9.0.42

- **Issue:** CVE-2021-25122 - When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 - 10.0.0, 9.0.0.M1 - 9.0.41, and 8.5.0 - 8.5.61 could duplicate request headers and a limited amount of request body between requests. This could lead to data leaking from one request to another.
- **Solution**: Upgrade to Apache Tomcat 9.0.42

- **Issue:** CVE-2021-25329 - Apache Tomcat 10.0.0-M1 - 10.0.0, 9.0.0.M1 - 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. - 7.0.107. There is a configuration edge case that is highly unlikely to be used, but it can leave Tomcat still vulnerable to CVE-2020-9494.
- **Solution**: Upgrade to Apache Tomcat 9.0.42

tomcat-embed-el-9.0.30.jar

- **False positives:** No CVEs, low confidence on the Spring report, and no known issues with this dependency.

tomcat-embed-websocket-9.0.30.jar

- **All issues addressed in tomcat-embed-core-9.0.30.jar**

5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial's software application.

- Upgrade to Apache Tomcat version 9.0.42
- Upgrade to Spring Framework version 5.2.15.
- Upgrade to SnakeYAML version 1.26.
- Install Fuse 7.1 security update.
- Apply the interim fix or Fix Pack containing APAR PH29942 for each named product as soon as practical.
- Upgrade AES to version 1.56
- Upgrade to Bouncy Castle JCE Provider version 1.61.
- Upgrade to SmtpAppender version 2.13.2. Upgrade to FasterXML Jackson Databind version 2.10.5.
- Upgrade to SnakeYAML version 1.26.
- Encrypt all sensitive and PII that is stored in the database to ensure that your employees do not have access and to prevent a hacker from obtaining that information.
- Limit the number of hits one IP address can make to our API in a specific timeframe.
- Sanitize all input values to remove hazardous characters like: <, >, ", ', %, (,), &, +, \,).
- The input for the parameters should be validated. A user could send a request with a malicious value in the name parameter.
- All string attributes should have character length restrictions that are validated before saving an instance of a class.
- All integer attributes should be limited to character values and have integer count restrictions that are validated before saving an instance of a class.
- Integer values should be whitelisted if possible. Any value that gets sent to our endpoint that doesn't contain a number on the whitelist should return an error message and not be saved into the database.
- We would want to have some sort of error handling for invalid input that prevents creating an instance of any of the classes. If we don't validate our input and we want to save it to a database, we are opening up the possibility for hackers to write malicious code that can leak data.
- We'd like to practice the principle of least privilege. A person should have the required set of permissions to make requests to our API. These permissions can be added and/or revoked.
- All requests should be sent as a POST request to prevent any sensitive data to be leaked in the headers of a GET request.
- The Greeting class should be changed to a final class to avoid malicious subclass from adding finalizers, cloning, and overriding methods.
- Remove the unnecessary accessors from all classes.
- Remove name parameter in the CRUDController.java.
- Remove constructor from the DocData class.
- Add authorization to the getID() method in the DocData class. You should create a whitelist of IP addresses that are allowed to access that method. When an IP address that is not on the whitelist tries to obtain an id, the system should throw an error message.
- Make instance variables in the myDateTime class private.

Sources

CISOMAG. (2020, October 4). The Biggest Cybersecurity Threats for Financial Services. CISO MAG | Cyber Security Magazine. https://cisomag.eccouncil.org/cyberthreats-financial-service-providers/

CompareRemit. (2020, February 10). Federal Remittance Transfer Rules. Compareremit.com; CompareRemit.com. https://www.compareremit.com/money-transfer-guide/federal-remittance-transfer-rules/

Cybersecurity Modernization & Implementation to Tackle Threats. (2020). Tcs.com. https://www.tcs.com/perspectives/articles/cybersecurity-modernization-iot-implementation

Financial Services and Cybersecurity: The Federal Role. (n.d.). Retrieved July 7, 2021, from https://crsreports.congress.gov/product/pdf/R/R44429

Secure Coding Guidelines for Java SE. (2019). Oracle.com. https://www.oracle.com/java/technologies/javase/seccodeguide.html

The Importance of Cybersecurity in Banking – Bank Business eMagazine. (2021). Bankbusiness.us. https://www.bankbusiness.us/the-importance-of-cybersecurity-in-banking/

What Is Cryptography and How Does It Work? | Synopsys. (2021). Synopsys.com. https://www.synopsys.com/glossary/what-is-cryptography.html