****

# 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** | **May 05th, 2022** | **Jacob Silberstein** |  |

## Client



## 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

Jacob Silberstein

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following regarding how companies protect against external threats based on the scenario information:

* **What is the value of secure communications to the company?**
  + The value of secure communications is incalculable with this being the primary avenue of business. Any lapse in secure communications could mean terrible repercussions for Artemis financial ranging from, but not limited to, lost or stolen user information, credentials, or assets, loss of trust from the patrons and an insecurity about the status of their ability to maintain secure communications and transactions, and many more. The value of secure communications is of the utmost importance.
* **Are there any international transactions that the company produces?**
  + There is no current information to advise us that the company for sure engages in international transactions, but seeing that they are a financial institution, it is not out of the realm of thought that they may encounter some international transactions. Whether it be patrons who travel abroad and require the use of local ATM machines or a patron who utilizes the services of Artemis financial, remotely from a different country, international transactions are riddled throughout traditional financial institution interactions.
* **Are there governmental restrictions about secure communications to consider?**
  + Regarding common regulations that all financial institutions face, there are several in regard to the transfer and usage of communication and user data. Some of the required regulations are the SOX or Sarbanes Oxley Act which established requirements for audits regarding security, access controls and data backups. Additionally, there is the Gramm-Leach Bliley Act or GLBA which regulates the safekeeping, collection and use of private financial information. The last major one to discuss here is the PCI DSS compliance which governs all companies and organizations that utilize store, or process cardholder data. *(A simplified regulatory checklist for Financial Institutions, 2022)*
* **What external threats might be present now and in the immediate future?**
  + In terms of all financial institutions, many face common threats, which is largely the reason for so many kinds of compliances that they must meet. In terms of cyber threats there are some that are more intense than others, but those threats range from everything of Malware, in which malicious software is introduced into the institutions systems only to later compromise the security of the system, to Phishing and business fraud, where the institution will be sent emails or phone calls or some type of interaction that is largely avoidable through proper employee training (Hock, 2020). Overall, these threats are present now, and more new and nefarious threats can become ever present at any new day. The important thing to remember when talking about potential for attack is to remember that there needs to be s strong defense in order to repel that attack.
* **What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies?**
  + Some major “Modernization” requirements that must be considered in addition to the role of open-source libraries and evolving web application technologies, is the migration of server-based infrastructure to cloud based infrastructure. With this migration the company also needs to identify whether they want hosted services or managed cloud as a service or whether they want that to be something in house due to the confidentiality of the information financial institutions utilize. Additionally, Artemis financial should consider the necessity for device or lifecycle modernization. With device modernization, this refers to being on new devices that are either managed as a bring your own device or a Device as a service, which places more emphasis on the IT department, but increases the security by having non-recreational devices utilized for work. *(Insight, 2021)*

Based on the information provided by the customer, there are several specific items we would like to outline on the grounds of three major aspects: needs, potential threats, and security requirements. To begin, it is important to outline exactly what the company does based on our examination. Artemis Financial is a financial group that is responsible for developing and maintaining financial plans for savings accounts, retirement accounts, investment accounts, and insurance for their customers/patrons. It is also important to illustrate that Artemis financial is looking to make their systems and operations more modern by implementing and applying the most up to date security software. They (Artemis Financial) currently have a RESTful web application programming interface that is being examined by Global Rain for the purposes of security and vulnerabilities.

Artemis Financials needs are that of any major financial institution with the added task of ensuring security within and throughout online applications. As all financial institutions must, Artemis Financial is responsible for meeting several regulatory actions, be it SOX, PCI DSS, and GLBA. In terms of additional needs, the company has, they are required to maintain the highest level of confidentiality and security with their data, as well as upgrade to the latest information security and standards. The company also needs to have updated software to ensure security is at the forefront of the topics.

To discuss the potential threats the company faces, we need to examine the current structure. As it stands now, without the interaction of Global Rain, Artemis Financial is running on RESTful API’s that are currently using outdated software and non-updated patches leaving them susceptible to any potential incursion that has been identified as viable to the specific dependencies. Additionally, Artemis Financial is susceptible to a wave of other major attacks the likes of which all major financial institutions face such as, malware, phishing, Ransomware, DOS attacks, Watering Hole attacks, and many more. *(Hock, 2020)*

Finally, we get to the security requirements. In terms of what is required of the institution for their requirements, as previously outlined, they are required to adhere to several regulations in terms of how they get, use, and maintain their data (A simplified regulatory checklist for Financial Institutions, 2022). That being said, it is important for the institution to keep up with the RESTful API practices that they currently have going on as well as the necessity and routine to continually check for updates and patches within shared libraries. Additionally, it is paramount for Artemis financial to routinely audit the security of their online applications. There are additional steps that we would recommend such as ensuring encryption of data, privacy policies and training, and multifactor authentication, but some of these go beyond the scope of the online program. *(ABA)*

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financials’ software application. Justify your reasoning for why each area is relevant to the software application.

After reviewing the information provided and referencing the vulnerability assessment flow diagram, we have determined that there are several areas of this project that pose security concerns off the bat. The major points we would like to hit for areas in which security is going to be the most pivotal, is going to be in the input validation, API, Client/server interactions, code quality, Cryptography, and code errors. Below we will give our justification of each one and why we deemed them as the most important in this case.

Let’s begin with Input validation. Input validation is the process of validating that input from any access point is sent to the correct area it needs to go. I the input is invalid, it lets the person inputting the incorrect input know it is incorrect and won’t let them proceed any further. On the other side of that spectrum, if an input is valid, it authorizes and validates it to continue doing what it needs to do. This process is paramount for the security of Artemis Financials as all information with regards to the internal workings of the institution is of the highest confidentiality. Not checking for valid input would put all of the information inside the program at a high risk for attack. These attacks come in several different forms but often as SQL injections. *(NTT)*

The next major vulnerability to discuss is to ensure that there are secure API interactions. AS Artemis Financial already utilizes a RESTful API, which utilizes some great practices for the safety of its users, but as it gets changed, it is important to make sure those interactions remain the same or improve. This part of the vulnerabilities is one of the most important when considering how to approach it since API’s interact directly with company applications as well as have public endpoints. *(Akana)*

Client/Server interactions are also very important to check given what we will be going through, with updating and changing some security features. This section of potential vulnerability is ver5y pivotal in ensuring that the customers do not experience a lapse of excellence in service as well as ensuring that due to changes additional vulnerabilities do not become exposed. This could lead to a leak of data or other major issues if not properly identified and taken care of on both sides of the program. (*Client-side enforcement of server-side security*)

Code Quality and code error are the next major vulnerabilities we would like to discuss. Since we will be building off of prior architecture, it is important to ensure what we have already built follows good standard ethical building practices, as well as making sure the foundation for the updated program does not have any major holes. In general, this helps to ensure the solidity of the program going forward, as well as reducing the likelihood of code from before the update becomes an issue causing harm to the company. (*Why is code quality important?* 2020) Code error is the other item we should talk about here, as we are already talking architecture. Correct code error handling will provide a response out to the system out log. That being said, due to the upcoming changes to the program, verifying that when the code gets an error it sends it to the correct place, is of vital importance. If information is not sent to the correct place, this can lead to many other types of attacks and remote executions based on incorrect output of input from the user. (*Why error handling is so important - profocus technology - open it positions and technology job postings,* 2017)

Cryptography is perhaps going to be the most utilized service you have, and code pose a tremendous vulnerability if not implemented correctly. Cryptography as it pertains to this is the encoding of sensitive data to ensure that it is only readable and authenticated by a decoder on the other end of where it is being transmitted. In the world of financial institutions, money is coming and going in millions of different places all at the same time, each with their own unique signatures due to cryptography. In terms of vulnerabilities, If the cryptography of interactions with the bank isn’t secure, that could mean a loss of data, assets, reputation, and many other negative unintended effects. (*Why is Cryptography Important?*)

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

As we continue to review the code, we would point the attention to several areas of analysis to ensure the code is functioning to the extent that ensures safety. Initially, we should take a look at the controllers. The package contains two major controllers being the CRUD controller and the Greeting controller. As far as I can tell, there are no major issues with either of these controllers given that both maintain independence and show good encapsulation of program from server side. The only major thing I would consider revising is the direction of information going directly to a new document without filtration. This occurs in lines 12-16 of the CRUD controller. Additionally, the change should have a filtration method that ensures the correct data types are being implemented to the new doc.

The next issue I would like to outline is that of data access and generation. Overall, data access is always a big concern as it is telling of whom has access, how they have access, and what they have access to. From a client-side perspective, the client need only be able to access information pertinent to their own confidential information. Based on that assumption, we see that in the customer.java application which is the accessor for the account number, account balance, and deposit void, we have a well encapsulated piece of program. This differs a bit from the DocData.java program where information is generated and stored. In this class, we see on lines 23 and 24, unimplemented methods that denote what should be read in terms of the documents, as well as an unimplemented auto-generated catch block on line 29. While these are marked as “To do”, the issue occurs in that this program was live prior to this analysis, which leads to the potential for attack from SQL attacks and remote executions through misinterpreted data that may have never been caught. Additionally, the test information for the database name has the root as username and password (line 32, DocData.java), which while not easily interpretable, is definitely at high risk for attack and potential information leaking.

## 4. Static Testing

Run a dependency check on Artemis Financials’ software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Code | Description and Solution | Attribution | Risk |
| bcprov-jdk15on-1.46.jar | CVE-2016-1000352 | * In versions 1.55 and earlier of the Bouncy Castle JCE, ECB mode was enabled with ECIES, which is unsafe for use, being removed by the provider. | MITRE | V3.0(High)  V2.0(Medium) |
| CVE-2016-1000346 | * Not fully validated party DH public key in Bouncy Castle JCE 1.55 or earlier. * Makes it so invalid keys could be used to reveal information about private keys while static Diffie-Hellman is in use. | MITRE | V3.0(Low)  V2.0(Medium) |
| CVE-2016-1000345 | * DHIES/ECIES CBC mode was vulnerable to a padding attack in Bouncy Castle JCE 1.55 or earlier. * Possible to identify when encryption is failing due to padding. | MITRE | V3.0(Medium)  V2.0(Medium) |
| CVE-2016-1000344 | * In versions 1.55 and earlier of the Bouncy Castle JCE, ECB mode was enabled with ECIES, which is unsafe for use, being removed by the provider. | MITRE | V3.0(High)  V2.0(Medium) |
| CVE-2016-1000343 | * Weak private key generation based on default values in Bouncy Castle 1.55 and earlier. * 1.55 and earlier assumes a 1024 bit key if JCA key pair generator isn’t initialize with DSA parameters. | MITRE | V3.0(High)  V2.0(Medium) |
| CVE-2016-1000342 | * On verification, ASN.1 encoding isn’t fully validating the signature in Bouncy Castle JCE 1.55 or earlier ECDSA. * Injection of extra elements in sequenced data is possible with no detection, leading to shrouded or invisible data into a structure. | MITRE | V3.0(High)  V2.0(Medium) |
| CVE-2016-1000341 | * vulnerable to timing attack in Bouncy Castle 1.55 and earlier DSA signatures. * Based on the ability to observe timings for signature generation, attackers can gain information on a signatures K value and private value. | MITRE | V3.0(Medium)  V2.0(Medium) |
| CVE-2016-1000339 | * If table driven approach is used in the algorithm of AESFastEngine in Bouncy Castle JCE 1.55 and earlier, CPU monitoring is available, leading to a leak of the used AES key. * AESEngine has been modified to remove leakage. * AESFastEngine is recommended only where deemed appropriate. | MITRE | V3.0(Medium)  V2.0(Medium) |
| CVE-2016-1000338 | * On verification, ASN.1 encoding isn’t fully validating the signature in Bouncy Castle JCE 1.55 or earlier ECDSA. * Injection of extra elements in sequenced data is possible with no detection, leading to shrouded or invisible data into a structure. | CERT/CC | V3.0(High)  V2.0(Medium) |
| CVE-2018-5382 | * Legacy files on BC versions 1.49 or earlier in need of format conversion are subject to the use of a BKS-V1 key store which still operates on the low security 16-bit checksum. * Only use BKS-V1 key store when absolutely necessary. | CERT/CC | V3.0(Medium)  V2.0(Low) |
| CVE-2017-13098 | * Whilst configured to JCE for cryptographic functions in Bouncy Castle versions 1.0.3, a weak Bleichenbacher oracle is given when TLS Cipher suite is used when the RSA key exchange is undertaken. * Key is recoverable by attacker through vulnerable application referred to as a “ROBOT.” | CERT/CC | V3.0(Medium)  V2.0(Medium) |
| CVE-2013-1624 | * During the processing for malformed CBC padding in response to a noncompliant MAC check, Bouncy Castle Java Library before 1.48 and C# do not properly identify and consider timing side-channel attacks. * Allows for attackers to conduct distinguishing attacks and plaintext-recovery attacks through the use of statistical analysis of crafted packets timing data. * Related to CVE-2013-0169. | MITRE | V3.0(NA)  V2.0(Medium) |
| hibernate-validator-6.0.18.Final.jar | CVE-2020-10693 | * Flaw in hibernate validator enabling invalid EL expressions to be evaluated as valid, in version 6.1.2.Final of Hibernate Validator. * Attackers are able to bypass input sanitation controls aimed at controlling user-controlled data in error messages. | Red Hat, Inc. | V3.1(Medium)  V2.0(Medium) |
| jackson-databind-2.10.2.jar | CVE-2020-36518 | * Allowance for Java StackOverflow exception DOS through use of large depth nested objects in Jackson-databind 2.13.0 and earlier. | MITRE | V3.1(High)  V2.0(Medium) |
| CVE-2020-25649 | * Flaw found where there was improperly secured entity expansion in FasterXML Jackson Databind. * High threat to data integrity resulting from vulnerability to XXE attacks. | Red Hat, Inc. | V3.1(High)  V2.0(Medium) |
| log4j-api-2.12.1.jar | CVE-2021-44832 | * Vulnerable to RCE attacks when config uses a JDBC Appender utilizing a JNDI LDAP data source URI while attacker is in control of LDAP server in Apache Log4j2 versions 2.0-beta7 through 2.17.0. * Limited JNDI data source names to java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2, fixes this issue. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-45105 | * Apache Log4j2 Versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3 and 2.3.1) * Didn’t protect from uncontrolled recursion of self-referential lookups. * Attackers with control of the thread context map data can cause a DOS on interpretation of crafted string. * Fixed in Log4j 2.17.0, 2.12.3, and 2.3.1. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-45046 | * Incomplete fix of CVE-2021-44228 in Apache Log4j 2.15.0 in some non-default configurations. * Attackers with control of Thread Context map input data were able to craft malicious input data, which resulted in a leak of information and RCE in some environments. * Issue is fixed by removing support for message lookup patterns and disabling JNDI functions by default. | Apache Software Foundation | V3.1(Critical)  V2.0(Medium) |
| CVE-2021-44228 | * Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) * JNDI features, log messages, and parameters don’t protect against controlled LDAP and JNDI endpoints controlled by attackers. * Attacker has control over log messages and parameters and can execute arbitrary code from LDAP servers, when message lookup substitution is enabled. * Disabled in log4j 2.15.0 by default. * Functionality was removed in version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1). * specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects. | Apache Software Foundation | V3.1(Critical)  V2.0(High) |
| CVE-2020-9488 | * improper certificate validation with mismatch host in Apache Log4j SMTP appender. * Allows for SMTPS connections to be intercepted through man-in-the-middle attacks, leaking any log messages transmitted through the appender. * Fixed in Apache Log4j 2.12.3 and 2.13.1 | Apache Software Foundation | V3.1(Low)  V2.0(Medium) |
| logback-core-1.2.3.jar | CVE-2021-42550 | * Malicious configurations could be crafted by attackers with required privileges to edit configurations in logback version 1.2.7 and earlier. * Attackers able to execute arbitrary code from LDAP servers. | Switzerland Government Common Vulnerability Program | V3.1(Medium)  V2.0(High) |
| snakeyaml-1.25.jar | CVE-2017-18640 | * Allowance for entity expansion during load operation in SnakeYAML 1.18. * Related to CVE-2003-1564. | MITRE | V3.1(High)  V2.0(Medium) |
| spring-aop-5.2.3.RELEASE.jar | CVE-2022-22950 | * Potential for user crafted SpEL expression resulting in DOS condition in Spring framework 5.3.0-5.3.16 or older unsupported versions. | VMware | V3.1(Medium)  V2.0(Medium) |
| CVE-2022-22971 | * spring framework versions prior to 5.3.20+, 5.2.22+ and old unsupported versions * Applications with a STOMP WebSocket endpoint were vulnerable to a DOS by authenticated users. | VMware | V3.1(Medium)  V2.0(Medium) |
| CVE-2022-22970 | * spring framework versions prior to 5.3.20+, 5.2.22+ and old unsupported versions. * File uploads are subject to DOS attacks if they utilize data binding to set a MultipartFile or javax.servlet.Part to a field in a model object. | VMware | V3.1(High)  V2.0(Medium) |
| CVE-2022-22968 | * Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions. * Ineffective patterns for disallowed fields on DataBinders with case sensitive fields ineffectively protecting property path. | VMware | V3.1(Medium)  V2.0(Medium) |
| CVE-2022-22965 | * Spring MVC or Spring WebFlux application running on JDK 9+. * Vulnerability to RCE attack through data binding. * WAR deployment is required to utilize this exploit. * If deployed as a Spring Boot executable jar (default), application isn’t vulnerable. | VMware | V3.1(Critical)  V2.0(High) |
| CVE-2021-22060 | * Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions. * User can supply malicious input resulting in insertion of additional log entries. * follow-up to CVE-2021-22096 that protects against additional types of input. | VMware | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-22096 | * Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions. * User can supply malicious input resulting in insertion of additional log entries. | VMware | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-22118 | * Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7. * Vulnerability to privilege escalation in WebFlux application. * Recreated temporary storage directory, allowing locally authenticated malicious users to read and modify uploaded files in the WebFlux application. * Ability to also overwrite arbitrary files with multipart request data. | VMware | V3.1(High)  V2.0(Medium) |
| CVE-2020-5421 | * 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. * Bypassing of RFD protections outlined from CVE-2015-5211, due to dependency on browser used during jsessionid path parameter. | Pivotal Software, Inc. | V3.1(Medium)  V2.0(Low) |
| CVE-2016-1000027 | * Spring Framework up through 5.3.16. * Potential for RCE attack if utilized for Java deserialization of untrusted data. * Depending on the implementation of the library, authentication may be required, and issue may not occur. | MITRE | V3.1(Critical)  V2.0(High) |
| spring-boot-2.2.4.RELEASE.jar | CVE-2022-27772 | * spring-boot versions prior to version v2.2.11. * Release vulnerable to temporary directory hijacking. * Major impact to org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. * Vulnerability only affects unsupported products or versions from the maintainer. | MITRE | V3.1(High)  V2.0(Medium) |
| spring-core-5.2.3.RELEASE.jar | Same as spring-aop-5.2.3.RELEASE.jar | * Same as spring-aop-5.2.3.RELEASE.jar | Same as spring-aop-5.2.3.RELEASE.jar | Same as spring-aop-5.2.3.RELEASE.jar |
| tomcat-embed-core-9.0.30.jar | CVE-2022-29885 | * Incorrect documentation for the EncryptInterceptor in several versions of Apache Tomcat that stated it enabled Tomcat clustering when in fact it does not. * EncryptInterceptor provides confidentiality and integrity protection, but not against all risks on untrusted networks, and specifically not DOS risks. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2021-41079 | * Apache Tomcat 8.5.0 to 8.5.63, 9.0.0-M1 to 9.0.43 and 10.0.0-M1 to 10.0.2. * Improper Validation of incoming TLS packets. * Specially crafted packets could be created and utilized to trigger infinite loops leading to DOS. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2021-33037 | * Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66. * Incorrect parsing of HTTP transfer-encoding request header. * Potential for request smuggling when utilized with a reverse proxy. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-30640 | * Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65. * Attackers are abvle to authenticate using variations of valid usernames, and in some cases are able to bypass protection by the LockOut Realm within the JNDI realm of Apache Tomcat. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2021-25329 | * Incomplete fix on vulnerability CVE-2021-25329 resulting in vulnerability to code CVE-2020-9494. * Previous published prerequisites of CVE-3030-9484 and mitigations of CVE-2020-9484 apply to this issue. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2021-25122 | * Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 * Requests headers could be duplicated and visible to two separate users when responding to new h2c connection requests in the aforementioned versions. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2021-24122 | * Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106. * Susceptible to JSP source code disclosures in certain configurations of the aforementioned versions, when serving resources from network locations utilizing an NTFS file system. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2020-17527 | * Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59. * The above-mentioned versions were able to reuse HTTP request header values from prior streams received on HTTP/2 connections for request corresponding to the subsequent stream. * Likely closure of the HTTP/2 connection, but potential for leaked data between requests. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2020-13943 | * Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57. * Subsequent requests made to a connection could contain HTTP/2 headers while connecting from the same stream due to an overage on maximum agreed number of concurrent streams. * Potential for visibility of alternate users’ responses. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2020-13935 | * Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. * Incorrect validation on payload length h in WebSocket frame. * Potential for triggering infinite loop and Dos. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2020-13934 | * Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56. * Potential for DOS based on an h2c connection not releasing the HTTP/1.1 processor, post upgrade to HTTP/2. * Could lead to an OutOfMemoryException. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2020-8022 | * SUSE Enterprise Storage 5, SUSE Linux Enterprise Server 12-SP2-BCL, SUSE Linux Enterprise Server 12-SP2-LTSS, SUSE Linux Enterprise Server 12-SP3-BCL, SUSE Linux Enterprise Server 12-SP3-LTSS, SUSE Linux Enterprise Server 12-SP4, SUSE Linux Enterprise Server 12-SP5, SUSE Linux Enterprise Server 15-LTSS, SUSE Linux Enterprise Server for SAP 12-SP2, SUSE Linux Enterprise Server for SAP 12-SP3, SUSE Linux Enterprise Server for SAP 15, SUSE OpenStack Cloud 7, SUSE OpenStack Cloud 8, SUSE OpenStack Cloud Crowbar 8 * Vulnerability stemming from incorrect default permissions in packaging of Tomcat on the aforementioned systems which allowed local attackers to escalate from group to Root in Tomcat. | SUSE | V3.1(High)  V2.0(High) |
| CVE-2020-11996 | * Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55. * Potential for an unresponsive server due to multiple concurrent specially crafted requests of the HTTP/2 connections, that trigger high CPU usage for a few seconds. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2020-9484 | * Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103. * If an attacker has a very specific set of circumstances the attacker is able to execute an RCE attack. | Apache Software Foundation | V3.1(High)  V2.0(Medium) |
| CVE-2020-1938 | * Ensure adequate care whilst utilizing Apache JServ Protocol for trusting incoming connections to Tomcat. * Tomcat treats AJP requests as higher trust than other connections. | Apache Software Foundation | V3.1(Critical)  V2.0(High) |
| CVE-2020-1935 | * Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99. * Allowance for invalid HTTP headers to be parsed as valid based on code using an approach to end-of-line parsing. * Potential for HTTP request smuggling through reverse proxies. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| CVE-2019-17569 | * Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99. * Regression in the above listed versions of Tomcat through refactoring which led to invalid transfer-encoding headers being validated and enabling request smuggling over reverse proxies. | Apache Software Foundation | V3.1(Medium)  V2.0(Medium) |
| tomcat-embed-websocket-9.0.30.jar | Same as tomcat-embed-core-9.0.30.jar | * Same as tomcat-embed-core-9.0.30.jar | Same as tomcat-embed-core-9.0.30.jar | Same as 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 Financials’ software application.

Overall, from what we have glinted of the application there are several major security concerns we should address and ensure are being managed. Based on what was seen in the vulnerabilities, many of the potential issue that were uncovered were version specific to many older versions of our plugins. That being said, this does not mean that there could not be vulnerabilities that aren’t yet exposed, it simply means that much of what we saw was based on older versions and unsupported structures. That being the case, I would of course recommend having a consistent system of updating software’s to their newest versions. As I said before, this does not entirely prevent vulnerabilities, but the longer a patch is out and visible, the more time attackers have to try to manipulate it in ways that best fit their own purposes.

Additionally, another issue we would like to discuss is the input validation and generation, as well as the sanitation protocols of the system. A few of the vulnerabilities that were returned were of attackers having access to improperly sanitized material, leaving the user and the program open to attack from malicious users. We would recommend ensuring that the program is correctly sanitizing the users material after each session and in some cases during sessions on periods of absence. Overall, this would be a major boon to both the users and the company as it leaves the possibility of leaked data from users, in the past. We also wanted to discuss the input validation. From some of the vulnerabilities in Apache, although often referring to older or unsupported versions, issues of incorrect input validation were prominent and could have some very dire consequences on the company as a financial institution. This also goes for information generation, although we did not notice any major issues with data generation, we believe these fall into a similar category as input validation and sanitation protocols in that they should be reworked to ensure that proper input is being taken only, and user pages are being sanitized before anyone other than the users have time to utilize them.

The final major issue we would like to see worked through is the system for keys. We noticed that in several of the vulnerabilities, the generation of weak keys are keys that were easily bypassed was a very real concern. The Keysets are what ensure that information in transit and static information are safe and only readable by the intended user. For this, we would recommend reworking the keyset system as well as ensuring that the available version of Bouncy Castle is up to date and functioning correctly in all cases. As you deal in financials, this is not the style of situation anyone can afford to get wrong even once.

References

ABA. (n.d.). *Protecting customers*. ABA. Retrieved May 22, 2022, from https://www.aba.com/banking-topics/technology/cybersecurity/protecting-customers#:~:text=To%20help%20ensure%20users%E2%80%99%20safety%20while%20offering%20the,privacy%20policies%20and%20training%2C%20anomaly%20detection%2C%20and%20more.

Akana. (n.d.). *API security best practices*. Akana by Perforce. Retrieved May 22, 2022, from https://www.akana.com/resources/api-security-best-practices#:~:text=Because%20APIs%20interact%20with%20your%20company%27s%20applications%2C%20securing,their%20vulnerabilities%20are%20often%20the%20target%20of%20hackers.

*Client-side enforcement of server-side security*. Martello Security. (n.d.). Retrieved May 22, 2022, from https://www.martellosecurity.com/kb/mitre/cwe/602/

Editor, I. (2021, June 15). *Your guide to it modernization*. Insight. Retrieved May 22, 2022, from https://www.insight.com/en\_US/content-and-resources/2021/your-guide-to-it-modernization.html

Hock, J. (2020, October 8). *Financial institutions are one of the largest targets for cyber attacks.* Doeren Mayhew. Retrieved May 22, 2022, from https://doeren.com/5-common-cyber-threats-to-the-financial-institution-industry/#:~:text=Here%20are%20five%20of%20the%20most%20common%20cyber,flood%20the%20institution%E2%80%99s%20servers%20with%20...%20More%20items

*International Financial Institutions: United Nations Development Programme*. UNDP. (n.d.). Retrieved May 22, 2022, from https://www.undp.org/partners/international-financial-institutions

NTT. (n.d.). *Input validation*. WhiteHat Security Glossary. Retrieved May 22, 2022, from https://www.whitehatsec.com/glossary/content/input-validation#:~:text=Input%20validation%20prevents%20improperly%20formed%20data%20from%20entering,and%20validate%20all%20input%20entered%20into%20a%20system.

Ravan, Admin, Jay, Person, Young, D., Packer, J., Choudhary, C., Luis, Tiamo, Alexander, Anto, 应琪瑜, Sunny, Gaurav, Talada, R., Metkar, P., Iris, & Masne, S. (2022, April 7). *What is rest*. REST API Tutorial. Retrieved May 22, 2022, from https://restfulapi.net/

*Secure Coding Guidelines for Java SE*. Secure coding guidelines for java SE. (n.d.). Retrieved May 22, 2022, from https://www.oracle.com/java/technologies/javase/seccodeguide.html

*Secure coding practices - quick reference guide - owasp*. (n.d.). Retrieved May 22, 2022, from https://owasp.org/www-pdf-archive/OWASP\_SCP\_Quick\_Reference\_Guide\_v2.pdf

*A simplified regulatory checklist for Financial Institutions*. Arctic Wolf. (2022, February 1). Retrieved May 22, 2022, from https://arcticwolf.com/resources/blog/a-simplified-regulatory-checklist-for-financial-institutions

U.S. Chamber of Commerce. (n.d.). *Finance*. U.S. Chamber of Commerce. Retrieved May 22, 2022, from https://www.uschamber.com/finance

*Why error handling is so important - profocus technology - open it positions and technology job postings*. ProFocus Technology. (2017, December 1). Retrieved May 22, 2022, from https://www.profocustechnology.com/general/error-handling-important/

*Why is code quality important?* Codegrip. (2020, December 10). Retrieved May 22, 2022, from https://www.codegrip.tech/productivity/why-is-code-quality-important/

*Why is Cryptography Important?* RSA Labs FAQ - why is cryptography important? (n.d.). Retrieved May 22, 2022, from http://security.nknu.edu.tw/netsec/faq/html/1-7.html