****

# 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** | **11/10/2020** | **Joshua Gauthier** |  |

## Client



## Developer

Joshua Gauthier

## 1. Interpreting Client Needs

Artemis Financial deals with the private information and financial data of its clients to develop a financial plan. Insecure communications with their web application could mean stolen personal information like social security numbers and contact information. It could also cause leaks of financial data such as bank accounts, investment information, and insurance data. Stolen financial data could result in attackers stealing money from clients or Artemis Financial, attackers committing insurance fraud, and loss of Artemis Financial’s clientele.

Laws such as the Privacy Act and regulations set by the federal trade commission will need to be considered with this application’s security. In addition, laws in the banking, insurance, and investment fields will need to be considered.

Artemis Financial may develop financial plans for international patrons. Therefore, international government laws and regulations will need to be meet for numerous countries.

There are several external threats to consider for the Artemis Financial web application. Phishing attacks are used to trick people into clicking on a malicious link or trick them into giving out confidential information. The phishing attacker then gains a foothold into the financial system and then has access to money, valuable information, or service alterations. Another external threat to a web application in the financial industry would be malicious attackers using an intercepting proxy to manipulate HTTP messages. This can result in numerous security vulnerabilities being exploited by innovative malicious attackers. For example, a SQL injection attack from a proxy could give an attacker access to confidential database records if the API has a dynamic SQL query that is not parameterized. Through a combination of SQL injection and phishing attacks, malware such as ransomware could be used to make Artemis Financial pay a ransom to unencrypt their own data. Another external threat to the web app would be denial of service attacks. A sophisticated DDoS attack is impossible to prevent and can prevent legitimate users from accessing the financial services they need. Finally, new malware is constantly being produced by external threats to steal information and money in the financial industry in innovative ways. Because of this, Artemis Financial must constantly be aware of constantly changing cybersecurity solutions.

According to Manico (2014), access control is the security layer in software that limits users to accessing only authorized functionality and data. Considering the best access control scheme for the situation is important for effective security. The principle of least privilege should be used to reduce the impact of breached employee or customer access.

There are new vulnerabilities uncovered every day. Libraries and frameworks that Artemis Financial’s application depend on should be checked regularly for newly discovered vulnerabilities.

Cloud technologies are becoming increasingly popular with web applications. However, security as it relates to infrastructure-as-a-service, platform-as-a-service, and even software-as-a-service must be considered. A Cloud service provider is not legally bound to protecting the system or data it provides for (Hub Security, 2020). Because of this, care must be taken to restrict threats and avoid API vulnerabilities. A web application should not expect that its infrastructure security to protect them completely—the web app must provide its own comprehensive protection.

## 2. Areas of Security

A RESTful web API must be competent in all seven security topics to impede external security threats.

Input validation is important for any software application, including web applications. The app should deny by default—meaning it should only allow expected input and reject anything else. By doing this, the web app mitigates threats such as HTTP request modification that would otherwise allow exploitation of the app.

Secure API interactions are extremely important for a web application. The API needs to be able to authenticate outside users, and subsequently determine what authorized access the user has to the API. If authentication or authorization is not properly implemented, malicious individuals could gain access to the API data illegally.

Cryptography is important for a web application so that sensitive information is not stored in plain text. If financial information such as bank account numbers are stored without encryption, they are vulnerable to theft. Cryptography is also important so that all HTTP requests and responses are encrypted. Secure distributed computing involves private connections between the client and server, so data is harder to steal by sniffers. This makes secure distributed composing an important area of security for the web application. A strong encryption algorithm must be chosen so it is hard for attackers to decipher the encrypted text by guessing the key. In addition, the encryption process must be implemented effectively, and special care must be taken for confidentiality, data integrity, and data availability.

Web application errors that are not handled properly can create security concerns. For example, an error may cause confidential data or API structure to be displayed to the user. Another problem that could happen from improper error-handling is the error could take up excessive resources, preventing the app from performing well. This in turn could cause a DoS vulnerability.

Coding patterns and practices are solutions and preventative measures to known problems. Not utilizing them causes security vulnerabilities that should have been prevented to begin with. For example, the principle of least privilege must be used to keep applications and users from extraneous access that could cause security risks.

Improper encapsulation of data can lead to exploitation of exposed classes and functionalities of a web application. The principle of least privilege must be used so that users and applications only have the access they need to complete their requirements.

## 3. Manual Review

After reviewing this code base, several security concerns are apparent:

1. There is no sign of an authentication system for users trying to access the API services. Users should have username and passwords that allow them to access certain features of the site that should be restricted. As the site becomes more complex, more controllers will be developed and many will need a way to restrict access.
2. There is no access control or authorization for users accessing the services. Again, as the system becomes more complex, certain features need to be restricted. For example, the DocData.read\_document() may need to be restricted to users that have accounts with Artemis Financial. To do this, access control needs to be implemented.
3. There is not enough input validation for the controller. For example, in the CRUDController class, the business\_name must be validated.
4. The REST controller has improper error handling. The user will get error pages that show too much API information. This could pose a security threat.
5. There is no encryption for the application’s HTTP transmissions. Any data sniffer can intercept requests and services easily.
6. There is no encryption for a customer’s account\_number. Financial information stored this way is extremely insecure.
7. The app fails to log activity into a nonvolatile medium. This means if there is an error or attack, the application will not log important information about the threat. Fighting security threats would be next to impossible without useful system logs.
8. There is not enough encapsulation. For example, customer.account\_balance is a public attribute, and can easily be altered outside of the class. In addition, myDateTime attributes and methods are all public.

## 4. Static Testing

List of Identified Dependency Vulnerabilities:

* The Bouncy Castle Crypto package – bcprov-jdk15on.1.46.jar

1. CVE-2013-1624 - The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.
   1. Update to version 1.48 or later
   2. Retrieved from http://www.isg.rhul.ac.uk/tls/TLStiming.pdf
2. CVE-2015-6644 - Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows attackers to obtain sensitive information via a crafted application, aka internal bug 24106146.
   1. Update to build LMY49F, version 6.0 or later
   2. Retrieved from https://securitytracker.com/id/1034592
3. CVE-2015-7940 - The Bouncy Castle Java library before 1.51 does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack."
   1. Install “yum” update for Fedora, and update to Red Hat JBoss A-MQ 6.3 and Red Hat JBoss Fuse 6.3
   2. Retrieved from https://lists.fedoraproject.org/pipermail/package-announce/2016-January/174915.html and https://access.redhat.com/errata/RHSA-2016:2036
4. CVE-2016-1000338 - In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
   1. Apply Oracle’s recommended critical patch updates. Update Debian to 1.49+dfsg-3+deb8u3 or later. Update Red Hat. Update Ubuntu.
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html> and <https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html> and <https://ubuntu.com/security/notices/USN-3727-1> and https://www.oracle.com/security-alerts/cpuoct2020.html
5. CVE-2016-1000339 - In the Bouncy Castle JCE Provider version 1.55 and earlier the primary engine class used for AES was AESFastEngine. Due to the highly table driven approach used in the algorithm it turns out that if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak information on the AES key being used. There was also a leak in AESEngine although it was substantially less. AESEngine has been modified to remove any signs of leakage (testing carried out on Intel X86-64) and is now the primary AES class for the BC JCE provider from 1.56. Use of AESFastEngine is now only recommended where otherwise deemed appropriate.
   1. Apply Oracle’s recommended critical patch updates. Update Debian to 1.49+dfsg-3+deb8u3 or later. Update Red Hat. Update Ubuntu.
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html> and <https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html> and <https://ubuntu.com/security/notices/USN-3727-1> and https://www.oracle.com/security-alerts/cpuoct2020.html
6. CVE-2016-1000341 - In the Bouncy Castle JCE Provider version 1.55 and earlier DSA signature generation is vulnerable to timing attack. Where timings can be closely observed for the generation of signatures, the lack of blinding in 1.55, or earlier, may allow an attacker to gain information about the signature's k value and ultimately the private value as well.
   1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
7. CVE-2016-1000342 - In the Bouncy Castle JCE Provider version 1.55 and earlier ECDSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
   1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
8. CVE-2016-1000343 - In the Bouncy Castle JCE Provider version 1.55 and earlier the DSA key pair generator generates a weak private key if used with default values. If the JCA key pair generator is not explicitly initialised with DSA parameters, 1.55 and earlier generates a private value assuming a 1024 bit key size. In earlier releases this can be dealt with by explicitly passing parameters to the key pair generator.
   1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
9. CVE-2016-1000344 – In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
   1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
   2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
10. CVE-2016-1000345 - In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES CBC mode vulnerable to padding oracle attack. For BC 1.55 and older, in an environment where timings can be easily observed, it is possible with enough observations to identify when the decryption is failing due to padding.
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
11. CVE-2016-1000346 - In the Bouncy Castle JCE Provider version 1.55 and earlier the other party DH public key is not fully validated. This can cause issues as invalid keys can be used to reveal details about the other party's private key where static Diffie-Hellman is in use. As of release 1.56 the key parameters are checked on agreement calculation.
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
12. CVE-2016-1000352 - In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
13. CVE-2017-13098 - BouncyCastle TLS prior to version 1.0.3, when configured to use the JCE (Java Cryptography Extension) for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. This vulnerability is referred to as "ROBOT."
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
14. CVE-2018-1000613 - Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later.
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>
15. CVE-2018-5382 - Bouncy Castle BKS version 1 keystore (BKS-V1) files use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS-V1 keystore. All BKS-V1 keystores are vulnerable. Bouncy Castle release 1.47 introduces BKS version 2, which uses a 160-bit MAC.
    1. Follow Oracle’s recommended security patches on all software that depends on bouncy castle
    2. Retrieved from <https://www.oracle.com/security-alerts/cpuoct2020.html>

* The Apache Log4j API - log4j-api-2.12.1.jar

1. CVE-2020-9488 - 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.
   1. Recommended to upgrade to the Apache Log4j API version 2.13.2 which fixes the man-in-the-middle vulnerability.
   2. Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2013-1624

* YAML 1.1 parser and emitter for Java - snakeyaml-1.25.jar

1. CVE-2017-18640 - The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
   1. Do the following:
      1. If the YAML is not coming from untrusted source (it is merely a configuration file) then it is a false positive. Just ignore it. The quality of NVD database is very low and contains tons of issues which appear to be false positives.
      2. Read the YAML and check its quality before giving the document to SnakeYAML (count \* and & for instance)
      3. 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)
   2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-18640

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

1. CVE-2020-5421 - 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, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.
   1. upgrade Spring Framework to either 5.2.9, 5.1.18, 5.0.19, 4.3.29
   2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5421
   * Core Tomcat implementation - tomcat-embed-core-9.0.30.jar
2. CVE-2019-17569 - The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
   1. Upgrade Tomcat to one of these versions:
      1. 9.0.31-1~deb10u1 or later
      2. 8.5.54-0+deb9u1 or later
      3. 7.0.100 or later
   2. <https://www.debian.org/security/2020/dsa-4673> and https://lists.apache.org/thread.html/r88def002c5c78534674ca67472e035099fbe088813d50062094a1390%40%3Cannounce.tomcat.apache.org%3E
3. CVE-2020-11996 - A specially crafted sequence of HTTP/2 requests sent to 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 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.
   1. Upgrade Tomcat to these versions:
      1. 10.0.0-M6 or later
      2. 9.0.36 or later
      3. 8.5.56 or later
   2. https://lists.apache.org/thread.html/r93ca628ef3a4530dfe5ac49fddc795f0920a4b2a408b57a30926a42b@%3Ccommits.ofbiz.apache.org%3E
4. CVE-2020-13934 - An h2c direct connection to 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 did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.
   1. Upgrade Tomcat to these versions:
      1. 10.0.0-M7 or later
      2. 9.0.37 or later
      3. 8.5.57 or later
   2. https://lists.apache.org/thread.html/r61f411cf82488d6ec213063fc15feeeb88e31b0ca9c29652ee4f962e%40%3Cannounce.tomcat.apache.org%3E
5. CVE-2020-13935 - The payload length in a WebSocket frame was not correctly validated in 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. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.
   1. Upgrade Tomcat to one of these versions:
      1. 10.0.0-M7 or later
      2. 9.0.37 or later
      3. 8.5.57 or later
   2. https://lists.apache.org/thread.html/rd48c72bd3255bda87564d4da3791517c074d94f8a701f93b85752651%40%3Cannounce.tomcat.apache.org%3E
6. CVE-2020-13943 - If an HTTP/2 client connecting to 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 exceeded the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent 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 users seeing responses for unexpected resources.
   1. Upgrade Tomcat to one of these versions:
      1. 10.0.0-M8 or later
      2. 9.0.38 or later
      3. 8.5.58 or later
   2. https://lists.apache.org/thread.html/r4a390027eb27e4550142fac6c8317cc684b157ae314d31514747f307%40%3Cannounce.tomcat.apache.org%3E
7. CVE-2020-1935 - In 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 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
   1. Upgrade Tomcat to one of these versions:
      1. 9.0.31 or later
      2. 8.5.51 or later
      3. 7.0.100 or later
   2. https://lists.apache.org/thread.html/r127f76181aceffea2bd4711b03c595d0f115f63e020348fe925a916c%40%3Cannounce.tomcat.apache.org%3E
8. CVE-2020-1938 - When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
   1. Upgrade Tomcat to one of these versions:
      1. 9.0.31 or later
      2. 8.5.51 or later
      3. 7.0.100 or later
   2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1938
9. CVE-2020-9484 - When using 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 a) an attacker is able to control the contents and name of a file on the server; and b) the server is configured to use the PersistenceManager with a FileStore; and c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter="null" (the default unless a SecurityManager is used) or a sufficiently lax filter to allow the attacker provided 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, using a specifically crafted request, the attacker will be able to trigger remote code execution via deserialization of the file under their control. Note that all of conditions a) to d) must be true for the attack to succeed.
   1. Upgrade to ePO 5.10.0 Update 9 to receive fixes for the Java, Tomcat, and Xss issues
   2. https://kc.mcafee.com/corporate/index?page=content&id=SB10332
   * Core Tomcat Implementation – tomcat-embed-websocket-9.0.30.jar
10. CVE-2019-17569 – explained earlier in Tomcat Core vulnerabilities
11. CVE-2020-11996 – explained earlier in Tomcat Core vulnerabilities
12. CVE-2020-13934 – explained earlier in Tomcat Core vulnerabilities
13. CVE-2020-13935 – explained earlier in Tomcat Core vulnerabilities
14. CVE-2020-13943 – explained earlier in Tomcat Core vulnerabilities
15. CVE-2020-1935 – explained earlier in Tomcat Core vulnerabilities
16. CVE-2020-1938 – explained earlier in Tomcat Core vulnerabilities
17. CVE-2020-8022 - A Incorrect Default Permissions vulnerability in the packaging of tomcat on 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 allows local attackers to escalate from group tomcat to root. This issue affects: SUSE Enterprise Storage 5 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP4 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 12-SP5 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 15-LTSS tomcat versions prior to 9.0.35-3.57.3. SUSE Linux Enterprise Server for SAP 12-SP2 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 12-SP3 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 15 tomcat versions prior to 9.0.35-3.57.3. SUSE OpenStack Cloud 7 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud 8 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud Crowbar 8 tomcat versions prior to 8.0.53-29.32.1.
    1. this is a SUSE issue more than a Tomcat issue. To fix this problem, download openSUSE Leap 15.1 (noarch) package
    2. https://bugzilla.suse.com/show\_bug.cgi?id=1172405
18. CVE-2020-9484 – explained earlier in Tomcat Core vulnerabilities

## 5. Mitigation Plan

Local Vulnerabilities Action Plan:

1. Authentication- Authentication should be implemented with Spring Security. Usernames and passwords are recommended if any aspect of the web app should be restricted. Passwords should be stored using the Encryptors class from Spring Security Crypto. This class will use a symmetric encryption process and generate a key to decipher password encoding.
2. Access control- Access control could be implemented by using Spring Security framework or by building a custom implementation. Role-base access control can be easily implemented. However, if the web app is going to be complex, permission-based access control through Apache Shiro should be considered, or a more robust attribute-based access control should be considered via Spring Security.
3. Input Validation- The REST controllers should be accompanied by @Validated and other related annotations provided by Spring which helps with input validation. Whitelist validation should be used over blacklist validation whenever possible. Numerical string input should be parsed into formal Java numeric classes as soon as possible, and the result should be checked that it falls within the accepted length and range.
4. Exception Handling- controller methods should be accompanied with @ExceptionHandler annotation that denotes how to handle exceptions raised with the controller. Specific error pages should be created that explain to the user what happened without revealing API internals.
5. Secure Distributed Computing- SSL should be implemented with Spring Security so that client/server communications are encrypted and secure.
6. Unencrypted bank account numbers - An established third-party API should be considered for handling all financial information such as account numbers and account balances. For example, an open banking/financial API such as a Citi API can let a web app access customer bank information and allow money to be moved across accounts, among other features. Artemis Financial should rely on these reliable financial APIs to deal with confidential financial information and should never store a customer’s financial data unencrypted or without expert guidance.
7. Activity Logging- Log4j is already a dependency of this program but is not used. It should be implemented to record server requests and activity.
8. The principle of least privilege should be used when considering encapsulation. Class attributes and methods that must be restricted to that class or package should never be made public.

Dependency Vulnerabilities Action Plan:

1. Apache Log4j API must be updated to version 2.13.2 or later (or preferably the latest version)
2. Bouncy Castle package must be updated to current version
3. 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)
4. upgrade Spring Framework to either 5.2.9, 5.1.18, 5.0.19, 4.3.29 (or preferably the latest version)
5. Upgrade Tomcat to one of these versions (or preferably the latest version):
   1. 10.0.0-M8 or later
   2. 9.0.38 or later
   3. 8.5.58 or later
   4. 7.0.100 or later
6. To fix the Tomcat/SUSE vulnerability, download openSUSE Leap 15.1 (noarch) package (or preferably latest version)

**References**

Altexsoft, (November 20, 2018). Open Banking and Financial APIs: How to Integrate Your Company with the Digital Financial Ecosystem. Retrieved from https://www.altexsoft.com/blog/engineering/open-banking-and-financial-apis-how-to-integrate-your-company-into-digital-financial-ecosystem/

Hub Security. (2020). Top 5 Cyber Threats Facing Banks in 2020. Retrieved from https://hubsecurity.io/top-5-cyber-threats-facing-banks/

Manico, Jim. (2014). Iron-Clad Java. Oracle Press. Retrieved from https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/