GLOBALRAIN

CS 305 Project One
Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
2.0	07/19/2021	Connor Brereton	made requested edits

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
Connor Brereton

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?
- Are there any international transactions that the company produces?
- Are there governmental restrictions about secure communications to consider?
- What external threats might be present now and in the immediate future?
- What are the "modernization" requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Artemis Financial is in a position where security is really important to their enterprise. This is the case because they need to hold PII that is tied directly to financial data that is then tied to the financial wellbeing of the customer. According to EMSISoft, "In most cases, data theft is financially driven. After stealing your information, bad actors can use a variety of shady channels to monetize your data, including taking out loans and making purchases under your name, holding your data to ransom and selling your data on dark web marketplaces to the highest bidder". "What is vital to the customer and the company's communications is that it's all done in a secure manner for regulatory reasons amongst others. Perhaps the most important financial regulation is Gramm-Leach-Bliley-Act aka GLBA which states: "The purpose of the GLB Act is to ensure that financial institutions and their affiliates safeguard the confidentiality of personally identifiable information (PII) gathered from customer records in paper, electronic or other forms. The law requires affected companies to comply with strict guidelines that govern data security". This act also gives consumers the ability to decide what information they want to give the financial institution. However, there are looparounds and caveats for this. When you as the consumer agree to their terms and conditions it changes the rules for what is allowed to be shared and what is not able to be shared.Also for regulatory reasons they have to ensure that their data is secured in a way that allows regulators to audit from time to time. Here is how audits work in regards to this system: State and federal banking agencies have varying degrees of authority to enforce GLBA provisions. The FTC can take action in federal district courts against organizations that fail to comply with the Privacy Rule. Section 5 of GLBA grants the FTC the authority to audit privacy policies to ensure they are developed and applied fairly. They are working internationally so they have to make sure that they are not just following local regulations but also global regulations. Most if not all of the threats to their business come from bad actors wanting to get confidential information for financial gains. The piece of infrastructure that poses the greatest risk for the security of the company is their API. Their API needs to be extremely secure. If any information gets leaked out it could be the end of them as a company depending on the fines, reputational damages, etc. What used to be able to be solved with rate limiting can now be solved using 2FA (two factor authentication). 2FA is where the user has to verify that they are who they say that they are. The goal of 2FA is to sit at the application layer and prevent anyone who isn't registered with the application from getting into the application.

Another layer of security that is vital is making sure that all internet traffic is served over HTTPS. This ensures that all sensitive data that is in the headers is encrypted and not exposed to bad actors.

2. Areas of Security
Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial's software application. Justify your reasoning for why each area is relevant to the software application.

The most relevant of the VAPFD is secure coding. Secure coding is important because it's needed to prevent any outside attacks from taking place by bad actors. Secure, structured code creates a level of security that can't be matched by other banks and the more secure the bank is the more high end clients they will attract. The two most important programming principles that need to be followed are logging code errors along with having secure APIs. Logging is really important because it makes sure that you have full visibility into your application by way of the API. According to Security Metrics, "From a security point of view, the purpose of a log is to act as a red flag when something bad is happening. Reviewing logs regularly could help identify malicious attacks on your system. Given the large of amount of log data generated by systems, it is impractical to review all of these logs manually each day". Logging all API events is really important because you need to be able to see every transaction, login, refund, etc. These events all create a profile to be analyzed for the overall security of the program. There are tools that will be used for making sure there are security alerts for anything suspicious that happens at the logging level. The reason why the APIs are so important is because our web services use RESTful APIs they need to ensure that the super sensitive data that's being sent over the header packet is done in a way that someone can't get access to the information. Input validation is another important topic because when working with a database the information that the user is transferring needs to be free of any characters that can act maliciously against the database engine. An example of this is the common SQL injection.

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

- The service does not use HTTPS which is vital for securely passing PII.
- There is no verification process in place to check identity. This is vital for financial systems.
- The requests are not validated which is vital for making sure that untrusted applications are not utilizing the infrastructure.
- In the CRUDController.java class there are business names that are sent which is really stupid. This can lead to many security issues for customers that want to remain anonymous.

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 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

Project: rest-service

com.twk:rest-service:0.0.1-SNAPSHOT

Scan Information (show less):

- *dependency-check version*: 5.3.2
- *Report Generated On*: Thu, 15 Jul 2021 16:16:54 -0700
- *Dependencies Scanned*: 38 (19 unique)
- *Vulnerable Dependencies*: 7
- *Vulnerabilities Found*: 36
- *Vulnerabilities Suppressed*: 0
- *NVD CVE Checked*: 2021-07-15T16:16:44
- *NVD CVE Modified*: 2021-07-15T15:00:02
- *VersionCheckOn*: 2021-07-15T16:16:44

Display: Showing All Dependencies (click to show less)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bcprov-jdk15on-1.46.jar	cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:*:*:*:*:*:*:*	pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46	Unknown	16	Highest	37
classmate-1.5.1.jar		pkg:maven/com.fasterxml/classmate@1.5.1		0		47
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:hibernate:hibernate-validator:6.0.18:*:*:*:*:*:*:* cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	1	Highest	36
jackson-annotations-2.10.2.jar		pkg:maven/com.fasterxml.jackson.core/jackson-annotations@2.10.2		0		38
jackson-core-2.10.2.jar		pkg:maven/com.fasterxml.jackson.core/jackson-core@2.10.2		0		45
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	1	Highest	39
jackson-datatype-jdk8-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.datatype/jackson-datatype-jdk8@2.10.2		0	Low	39
jakarta.annotation-api-1.3.5.jar		pkg:maven/jakarta.annotation/jakarta.annotation-api@1.3.5		0		32
jakarta.validation-api-2.0.2.jar		pkg:maven/jakarta.validation/jakarta.validation-api@2.0.2		0		29
jboss-logging-3.4.1.Final.jar		pkg:maven/org.jboss.logging/jboss-logging@3.4.1.Final		0		45
jul-to-slf4j-1.7.30.jar		pkg:maven/org.slf4j/jul-to-slf4j@1.7.30		0		28
log4j-api-2.12.1.jar	cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:*	pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1	LOW	1	Highest	46
logback-core-1.2.3.jar	cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.3		0	Highest	32
slf4j-api-1.7.30.jar		pkg:maven/org.slf4j/slf4j-api@1.7.30		0		29
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	HIGH	1	Highest	28
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_boot:2.2.4:release:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:2.2.4:release:*:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE		0	Highest	32
spring-core-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@5.2.3.RELEASE	HIGH	2	Highest	30
tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_software_foundation:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30	CRITICAL	14	Highest	39
tomcat-embed-el-9.0.30.jar		pkg:maven/org.apache.tomcat.embed/tomcat-embed-el@9.0.30		0		36

bcprov-jdk15on-1.46.jar – several vulnerabilities on 1.46, update to latest version.

- CVE-2013-1624
- CVE-2015-6644
- CVE-2015-7940
- CVE-2016-1000338
- CVE-2016-1000339
- CVE-2016-1000341
- CVE-2016-1000342
- CVE-2016-1000343
- CVE-2016-1000344
- CVE-2016-1000345
- CVE-2016-1000346
- CVE-2016-1000352
- CVE-2017-13098
- CVE-2018-1000613
- CVE-2018-5382

Log4j-api-2.12.1.jar – one vulnerability, update to latest version.

- CVE-2020-9488

Snakeyalm-1.25.jar – one vulnerability, update to latest version.

- CVE-2017-18640

Jackson-databind-2.10.2.jar – one vulnerability, update to latest version.

- CVE-2020-25649

Tomcat-embed-core-9.0.30.jar – several vulnerabilities, update to latest tomcat version.

- CVE-2019-17569
- CVE-2020-11996
- CVE-2020-13934
- CVE-2020-13935
- CVE-2020-13943
- CVE-2020-17527
- CVE-2020-1935
- CVE-2020-1938
- CVE-2020-8022
- CVE-2020-9484
- CVE-2021-24122

Spring-core-5.2.3.RELEASE.jar – one vulnerability, update to latest version.

- CVE-2020-5421

Hibernate-validator-6.0.18.Final.jar – one vulnerability, update to latest version.

- CVE-202-10693

Links To CVE's for bcprov-jdk15on-1.46.jar

https://nvd.nist.gov/vuln/detail/CVE-2013-1624
https://nvd.nist.gov/vuln/detail/CVE-2016-1000338
https://nvd.nist.gov/vuln/detail/CVE-2016-1000339
https://nvd.nist.gov/vuln/detail/CVE-2016-1000341
https://nvd.nist.gov/vuln/detail/CVE-2016-1000342
https://nvd.nist.gov/vuln/detail/CVE-2016-1000343
https://nvd.nist.gov/vuln/detail/CVE-2016-1000344
https://nvd.nist.gov/vuln/detail/CVE-2016-1000345
https://nvd.nist.gov/vuln/detail/CVE-2016-1000346
https://nvd.nist.gov/vuln/detail/CVE-2016-1000352
https://nvd.nist.gov/vuln/detail/CVE-2017-13098
https://nvd.nist.gov/vuln/detail/CVE-2018-1000613
https://nvd.nist.gov/vuln/detail/CVE-2018-5382

Link to CVE's for tomcat-embed-core-9.0.30.jar

https://nvd.nist.gov/vuln/detail/CVE-2019-17569
https://nvd.nist.gov/vuln/detail/CVE-2020-11996
https://nvd.nist.gov/vuln/detail/CVE-2020-13934
https://nvd.nist.gov/vuln/detail/CVE-2020-13935
https://nvd.nist.gov/vuln/detail/CVE-2020-13943
https://nvd.nist.gov/vuln/detail/CVE-2020-17527
https://nvd.nist.gov/vuln/detail/CVE-2020-1935
https://nvd.nist.gov/vuln/detail/CVE-2020-1938

https://nvd.nist.gov/vuln/detail/CVE-2020-8022
https://nvd.nist.gov/vuln/detail/CVE-2020-9484
https://nvd.nist.gov/vuln/detail/CVE-2021-24122
https://nvd.nist.gov/vuln/detail/CVE-2021-25122
https://nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329
https://nvd.nist.gov/vuln/detail/CVE-2021-33037

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.

To remedy these issues the client needs to enable HTTPS on their communication channels to prevent MITM attacks amongst others. They need to move request parameters to headers in order to make sure it's passed securely. They need to remove any hard coded business or PII data so that these cannot be snooped upon and contracts breached. Finally, they should get an IAM/2FA system in place to protect the identity of the company's customers, employees, and ensure that only privileged access is allowed. They also need to update all of the dependencies from the dependency checker aka Maven.

Sources

What is the Gramm-Leach-Bliley Act?. (2021). Retrieved 19 July 2021, from https://searchcio.techtarget.com/definition/Gramm-Leach-Bliley-Act

What is Dodd-Frank Act? A definition from WhatIs.com. (2021). Retrieved 19 July 2021, from https://searchcompliance.techtarget.com/definition/Dodd-Frank-Act?_gl=1*1kxhefh*_ga*ODcxMDIxNDM0LjE2MjY3MDkzMjU.*_ga_TQKE4GS5P9*MTYyNjcwOTMyMy4xLjAuMTYyNjcwOTMyMy4w&_ga=2.25247037.1695461442.1626709325-871021434.1626709325

How do hackers make money from your stolen data? | Emsisoft | Security Blog. (2020). Retrieved 19 July 2021, from https://blog.emsisoft.com/en/35541/how-do-hackers-make-money-from-your-stolen-data/

The Importance of Log Management. (2021). Retrieved 19 July 2021, from https://www.securitymetrics.com/blog/importance-log-management

Is HTTPS the Answer to Man in the Middle Attacks?. (2021). Retrieved 19 July 2021, from https://www.catchpoint.com/blog/https-man-in-the-middle