****

# 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** | **3/23/2024** | **Jason Kremhelmer** | Updates to: Interpreting Client NeedsAreas of SecurityManual ReviewStatic TestingMitigation Plan |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Jason Kremhelmer

## Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Document your findings in your vulnerability assessment report. Consider the scenario information and the following questions regarding how companies protect against external threats:

* 1. What is the value of secure communications to the company?
  2. Does the company make any international transactions?
  3. Are there governmental restrictions about secure communications to consider?
  4. What external threats might be present now and in the immediate future?
  5. What are the modernization requirements that you must consider? For example:
     1. The role of open-source libraries
     2. Evolving web application technologies

*Secure Communication*

The use of secure communication is vital when personal information is being delt with. It is also needed when personal information is not involved, as it helps to prevent attacks from those who would cause harm. With Artemis Financial dealing with financial plans and insurance, among other items, for their clients, it is imperative that secure communication is at the top of the priority list.

*International Transactions*

Based on the documents provided and an initial code review, there does not appear to be any internal transactions. However, this does not mean there will not be in the future. Principals of security would dictate that we treat it as a possibility, as there is no outright statement excluding the possibility. With that in mind, Artemis Financial needs to have a robust method for secure communications both domestic and foreign.

*Governmental Restrictions*

The United States government in May 2021 issues an Executive Order which states: Section 1.  Policy.  The United States faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public sector, the private sector, and ultimately the American people’s security and privacy.  The Federal Government must improve its efforts to identify, deter, protect against, detect, and respond to these actions and actors.  The Federal Government must also carefully examine what occurred during any major cyber incident and apply lessons learned.  But cybersecurity requires more than government action.  Protecting our Nation from malicious cyber actors requires the Federal Government to partner with the private sector.  The private sector must adapt to the continuously changing threat environment, ensure its products are built and operate securely, and partner with the Federal Government to foster a more secure cyberspace.  In the end, the trust we place in our digital infrastructure should be proportional to how trustworthy and transparent that infrastructure is, and to the consequences we will incur if that trust is misplaced.  
  
Incremental improvements will not give us the security we need; instead, the Federal Government needs to make bold changes and significant investments in order to defend the vital institutions that underpin the American way of life.  The Federal Government must bring to bear the full scope of its authorities and resources to protect and secure its computer systems, whether they are cloud-based, on-premises, or hybrid.  The scope of protection and security must include systems that process data (information technology (IT)) and those that run the vital machinery that ensures our safety (operational technology (OT)).   
  
It is the policy of my Administration that the prevention, detection, assessment, and remediation of cyber incidents is a top priority and essential to national and economic security.  The Federal Government must lead by example.  All Federal Information Systems should meet or exceed the standards and requirements for cybersecurity set forth in and issued pursuant to this order. (Jr., 2021)

It is imperative that Artemis Financial abide by the laws set forth in this country and others around the globe.

*External Threats*

Artemis Financial works with sensitive financial and personal information. This alone makes them a target for external threats. The largest threat for this company that I can see would be an individual or small group attack. Based on the size of the firm, I do not see any government or high-level actor being interested in taking on this company. There are many ways attacks can come, a few are, SQL injection, invalid inputs, data leaking into browser history (personal data, such as username and password). Another threat is having authorized users accessing data they should not have access to. This could happen with accessing administrator files by gaining elevated privileges. The use of a robust API would help to deter these attacks.

*Modernization Requirements*

The use of open-source libraries can present unforeseen threats to the system. Artemis Financial deals in financial planning, and as such, they may have the need to use complied stock/bond information obtained by other companies. If this information was integrated into the system, it could pose a threat to the system as there may have malware installed in the files. With securely built API as well as other security structures, would help protect them from these issues. Another item to ponder with modernization is web applications. While web applications are great for some instances, they do often have a lot of vulnerabilities that can be exploited.

## 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.

The primary areas of security include the following:

* ***Input Validation****:* Input validation is crucial any time you are dealing with input. This program does allow input, or at least leave itself open to input. For example, line 12 of CRUDController.java allows input to be passed in the form of an expected string. Validation of that string, to avoid failures or SQL injection, will make this program secure. Input validation is essential because we are asking the user to enter something into our system. If they have malicious intent, this could do serious damage or result in unauthorized access of private data.
* ***APIs****:* Because this application is designed to be run in an outside environment, such as a web browser, a good API will be essential. This defines how third-party software interacts with our program. The API will determine what methods are acceptable or unacceptable as well as what data is acceptable or unacceptable. Furthermore, this project may involve using third-party software which this software may depend on. To mitigate the risks posed by using someone else’s software, a secure API is vital.
* ***Cryptography****:* Cryptography is included in this list because of the uncertainty of international transfers. As discussed above, cryptography is regulated when it is an export item, so if secure, encrypted communications are used and Artemis Financial does do business overseas, then steps must be taken to secure the data but also do so in a manner that complies with the laws of both the United States and the destination country.
* ***Code Error****:* I included code error because the description of that is secure error handling. I see this as going hand-in-hand with input validation because errors caused by an input need to be handled properly. Any errors, especially those relating to input validation, need to be handled properly. Errors also need to be handled properly to avoid privilege escalation and mitigate other vulnerabilities.
* ***Code Quality****:* Code quality is included because it is an area of concern whenever an API and input validation are present. Quality code will ensure that no data is unintentionally exposed and it will also ensure that methods that are restricted from end users will not be made available to them. In short, only authorized users will get the data and methods that match their tier of authorization if quality code is used.

## Manual Review

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

My findings after the manual review of the code base are as follows:

* The first part I looked over was the input validation. First, I looked over the pom.xml file included in the codebase; this was do the ability to write in some Apache type validators in this file. After which, I looked in places where user inputs are coming in. The greeting controller is the primary place this occurs. I was able to input information here, however I was unable to see if any of it was kept. This being due to no output available.
* The API would be the next place to look, however there isn’t any. This program accepts data in a completely unsecure fashion. It accepts via URL instead of the POST method, as such, will leak data into the browser history. This has potential for exploitation. Another glaring issue, is this program has no way to interact with it unless you are using the source code. This program as it sits is not publishable to the general public.
* I followed this up by looking for any sort of cryptography in case there are international communications. This was a quick review because I did not find any sort of encryption at all. If international communications are needed by Artemis Financial, they will have to develop some sort of encryption system that complies with United States and international regulations and laws.
* Fourth, I looked for code error particularly in error handling. The one that stood out to me was in the DocData.java class. This class had a method for reading a document when given a key and a value. This does include a try/catch block but does not have any actual error handling. No other classes have any sort of error handling to speak of, so there was nothing else to evaluate there.
* Lastly, I looked at code quality. The code that exists is okay, though not entirely functional. It needs to be expanded a lot in order to get this program fully ready. The API, as mentioned, makes nothing available to the end user informing them what is expected. Input validation needs to be implemented. Input must be moved from the URL to the POST method to secure the data and prevent it from leaking into browser history. Code quality is a superset of the API and input validation. The issue with this aspect is to actually code, and code securely, the subsets.

## 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

All of the dependencies and vulnerability ID’s found below are from the same reference, footnoted here.[[1]](#footnote-2)

* bcprov-jdk15on-1.46.jar – Bouncy Castle Crypto package works on cryptographic algorithms for JDK 1.5 through JDK 1.7 (vulnerability ID: cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-cryptography-api:1.46.\*.\*.\*.\*.\*.\*.\*)
  + This potentially has several issues. One of the major ones, involving cryptography algorithms, can have a big impact on secure communications as well as exposing sensitive data. This version of Bouncy Castle (1.46) has a vulnerability where the second party DH public key is not fully validated and can lead to invalid keys being used to reveal details about the other party’s private key. If the private key of another user is found, it can make encryption useless. Even exposure of the algorithm used can lead to finding the key and render those “secure” communications essentially public.
* jackson-databind-2.10.2.jar – General data-binding functionality for Jackson that works on core streaming API (vulnerability ID: cpe:2.3:a:fasterxml:jackson:2.10.2.\*.\*.\*.\*.\*.\*.\* - AND – cpe:2.3:a:fasterxml:jackson-databind:2.10.2.\*.\*.\*.\*.\*.\*.\*)
  + This vulnerability can allow an attacker to submit an XML file that defines something outside of the intended scope of the application and have that data sent back as an error message. This can be used to allow an attacker to view sensitive data if they submit something that accesses files in the right way. The error message can be informative, but sometimes can be overly-informative in this instance.
* log4j-api-2.12.1.jar – Apache Log4j API (vulnerability ID: cpe:2.3:a:apache:log4j:2.12.1.\*.\*.\*.\*.\*.\*.\*)
  + This vulnerability is capable of impacting the SSL. The certificate checks performed during a session could either be not validated at all or validated incorrectly. This can allow software to connect to a malicious host while believing it is a trusted host. The lack of validation of an SSL certificate essentially renders that level of protection useless and gives users a false sense of security by seeing the little lock icon or the green address bar.
* snakeyaml-1.25.jar – YAML 1.1 parser and emitter for Java (vulnerability ID: cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25.\*.\*.\*.\*.\*.\*.\*)
  + This vulnerability can lead to denial of service (DoS). The reason is because the document type definition does not control the number of recursive definitions inside it. This can cause exponential growth when the data is parsed. DoS happens when a website or server has more coming in than it can handle and shuts down. An infinite, or even a very large but not infinite, recursion can be seriously taxing on any computer. It is critical to defend against this since DoS and DDoS attacks are among the most common.
* spring-core-5.2.3.RELEASE.jar – Spring Core (vulnerability ID: cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release.\*.\*.\*.\*.\*.\*.\* - AND – cpe:2.3:a:springsource:spring\_framework:5.2.3:release.\*.\*.\*.\*.\*.\*.\* - AND – cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release.\*.\*.\*.\*.\*.\*.\*)
  + This vulnerability impacts versions of Spring all the way up to 5.2.8 and the vulnerability is that protections against RFD (reflected file download) attacks can be bypassed using the jssessionid path. RFD is quite an interesting and scary exploit. It can be used to completely control the targeted system. This exploit gets around the more traditional methods of blocking RFD.
* tomcat-embed-core-9.0.30.jar – Core Tomcat implementation (vulnerability ID: cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* - AND - cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* - AND - cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*)
  + This dependency involves many vulnerabilities that are known and documented. One major vulnerability is that it can be exploited to give unauthorized people access to sensitive information. This can work when the code inserts information directly, indirectly, or that the code manages resources that intentionally contain sensitive information and the resources become available to unauthorized users. If a user can access a method that involves sensitive information, they could accidentally stumble upon it. They could also deliberately mess with the program method to try and further exploit it to gain that data that is not protected.
* tomcat-embed-websocket-9.0.30.jar – Core Tomcat implementation (vulnerability ID: cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* - AND - cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* - AND - cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*)
  + See previous – details are the same and the CPEs are the same.

## 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.

The general mitigation plan involves transforming this DevOps pipeline into a DevSecOps pipeline. Since this project is still in the early stages, it would help to get everyone in the development team in a security mindset now. That way things will be built with security in mind and we can go forward with this project the right way. If we do this, then things like input validation, which has been cause for concern for the NIST many times, can be mitigated before this project goes into operation.

Specifically, the threats identified above have mitigation measures to be taken as well. Again, the early stage of development works to our advantage here. Those measures will be outlined in the list below.

* bcprov-jdk15on-1.46.jar – mitigating this threat will involve using a version numbered 1.56 or later because the later versions check the key parameters on agreement calculation. Keeping keys secure is what makes cryptography worth it in the first place. It is of critical importance to have those keys calculated properly and stored. Another part of this is making sure that they are in fact validated properly before any data is exchanged.
* jackson-databind-2.10.2.jar – mitigating this threat will involve configuring the XML parser and validator to disable eternal entity expansion. This means that attackers will no longer be able to submit altered DTD files within the XML documents because the parser will catch it and ignore the malicious code, if done properly.
* log4j-api-2.12.1.jar – mitigating this threat will involve not using certificate pinning or to make sure that all properties of the certificate are validated before pinning with a special focus on the hostname. Browsers trust the SSL certificate and so do the end users of those browsers. It would not do us much good to give people a false sense of security when using our platform to have an SSL certificate that is not enforced properly.
* snakeyaml-1.25.jar – mitigating this threat will involve prohibiting the use of DTDs or to use a parser that limits the expansion of those recursive definitions. Recursive calls can generate huge loops, either by mistake of by design. I use recursion often in code and it always needs to be checked. If a parser limits the expansion, that could stop this sort of attack in its tracks. Some recursion may take place, but it would be prevented from expanding to such an extent as to actually cause a DoS.
* spring-core-5.2.3.RELEASE.jar – There are no mentioned mitigation measures listed associated with this vulnerability, however, the current version is not listed in this set of vulnerable Spring frameworks. Therefore, a mitigating measure could be upgrading to the latest version, 5.3.1. Always using updated software helps to ensure the latest patches to known vulnerabilities are applied. That being said, it is also risky to use the newest thing because it doesn’t always have the real-world exposure to find, and therefore identify publicly, a vulnerability yet.
* tomcat-embed-core-9.0.30.jar & tomcat-embed-websocket-9.0.30.jar – mitigating these threats will involve using the principle of least privilege, along with safe zones in the system, will help to mitigate this threat. This allows sensitive information to remain within the safe zone. If sensitive data is places within the bounds of a secure structure, and that structure has authorization enforcement mechanisms in place, the leaking of that data becomes dramatically less likely.

# References

Jr., J. R. (2021, May 12). *Executive Order on Improving the Nation’s Cybersecurity*. Retrieved from The White House: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

1. Dependency-Check-Report (included in submission documents as an HTML file) [↑](#footnote-ref-2)