****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Contents

[Document Revision History 3](#_Toc77534460)

[Client 3](#_Toc77534461)

[Instructions 3](#_Toc77534462)

[Developer 4](#_Toc77534463)

[1. Interpreting Client Needs 4](#_Toc77534464)

[2. Areas of Security 5](#_Toc77534465)

[3. Manual Review 5](#_Toc77534466)

[4. Static Testing 6](#_Toc77534467)

[5. Mitigation Plan 8](#_Toc77534468)

[References 9](#_Toc77534469)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/17/2021** | **Patrick Knight** | **Project One** |

## 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

Patrick Knight

## 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?

Secure communications are incredibly important to any company. The information that customers and users supply to a company must be protected by that company. This obligation is something that will keep customers coming back to use their services. If information is ever leaked because of unsecured communications, then the company can expect to lose revenue and customers.

If the company handles any foreign investments, then there will be international transactions. Also, if the company opens their services to foreign countries, that will include international transactions. However, there is nothing listed in the scenario that would imply that either of these are valid leading to a conclusion that they do not handle foreign transactions.

Transactions that the company handles will have to follow the Payment Card Industry Data Security Standard. This standard details how customer’s information, mainly in the form of their credit cards, are to be handled. It states that the information must be protected. (PCI SECURITY, n.d.) The Gramm-Leach-Bliley Act of 1999 is broader than the PCI DSS, covering more information outside of credit cards, but still requires the company to protect the privacy of the customers. This act is still oriented towards customer’s finances and history. (Kagan, 2021) Finally, if the company operates in California, it will have to follow the California Consumer Privacy Act of 2018 that covers most personal data that is obtained by the company. (California Consumer Privacy Act (CCPA), n.d.)

Some external threats would come in the form of bad actors trying to steal the stored information on the customers. This can come in many forms: either through injection where the individuals gain access to the database or from gathering current employee account information, usernames, and passwords. Another is the possibility of denial-of-service attacks where an individual overloads a server with requests that will block customers from being able to access the services. Finally, the bad actor could gain access to the system to upload malware, like ransomware, that will shut down the system.

The addition of open-source libraries can save some time when adding features to the web application. This can also end up saving the company some money. However, there will need to be a review of the open-source library to verify if there are any potential threats that will be added with their addition to the product. Addition of new technologies can also be considered and will need to be properly vetted. There will need to be an active community in case any security threats arise so that there is a team that can update the library. (Jeganathan, 2019) An example could be the inclusion of Artificial Intelligence that can give the customer advice on investment. With the AI inclusion, the threat of injections and denial-of-service attacks will still need to be taken care of.

## 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.

* Input Validation – Artemis Financial will provides services to their customers in the form of financial plans. This will mean that the customer will have an account with the client and will need to be able to log into this account. Also, the customer will be asked various questions that will require input. Input validation will be an area that will need focus because of the amount of input that the application will need to process. Input validation will protect against injections that can lead to a compromise in the integrity of the system. (Manico & Deltefsen, 2014)
* APIs – The client is already using the RESTful web API. Any vulnerabilities that RESTful has will also become vulnerabilities of the client. These vulnerabilities might not affect the customers, however there is still a risk involved. An example of a vulnerability with RESTful is that HTTP requests and HTTP response can be accessed by attackers. (Levin, 2019) Also, if the client wishes to incorporate any more libraries or technologies, then there will need to be a vetting process of the libraries and technologies.
* Cryptography – Cryptography will be an important area of security. There will be a collection of personal information obtained from the customers. This information will mainly be in the form of finances. Artemis Financial will need to protect this information from attacker by using Cryptography to encrypt the data. Cryptography can also be used to authenticate access. (What is cryptography and how do we use it in modern day computing?, 2018)
* Client/Server – Since this will be a web application, there will need to be a client/server set up for customers to be able to access their profiles. This will mean that the server will need to be protected from attackers. Denial-of-service can slow server processing speeds down leading to a loss of customers. (Nsrav, n.d.) Validation and verification will also be important because employees will need to access the servers for various reasons. Validation and verification can prevent access to the server by attackers. (Manico & Deltefsen, 2014)
* Code Error – Code errors will need to be handled before production. A code error can allow for unvalidated input to get processed on the server granting access to information. Also, coding errors in the form of error messages can give bad actors enough information to access systems. (Manico & Deltefsen, Access Control, 2014)
* Code Quality – Input given to the system will need to be validated. This can be handled by using various trusted methods for handling all untrusted data. Access to various parts of the system should only be given to the roles who need access. Additional access to the roles that do not need the rights can be viewed as potential risks. Should avoid mixing untrusted data with trusted data.

## 3. Manual Review

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

* In the CRUDController class there is a vulnerability to Cross Site Request Forgery. Cross Site Request Forgery is where a bad actor can use customer’s authentication to execute code on the web application at the expense of the customer’s identity. (KirstenS, n.d.) This is made possible by the @RequestMapping tag that does not protect the method that is underneath it. (pedrorijo91, 2017)
* In the CRUDContoller class input is requested from the user and is not validated using the @RequestParam tag.
* In the CRUD class the data that is supplied in the CRUDController is also added to the system without verification. While it is not necessary to do in this class it could be beneficial to have in either.
* In the DocData class there is an exception handling that will supply the customers and potential bad actors too much information in the form of the printStackTrace() method. This can give attackers important information on how the system is designed. (Information exposure through a stack trace, n.d.)
* In the customer class there is a vulnerability with the variable account\_balance. Account\_balance is a package private variable that will allow all class the ability to access and modify it so long as it is in the same package.
* Same as the CRUDController class there is a no input validation with the @RequestParam tag in the GreetingController class.
* In the Greeting class the unvalidated input is stored within. It is also accessed with accessor methods.

## 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:

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.

* Bouncy Castle Legion Cryptography API Dependency version 1.46
  + CVE-2016-1000338, CVE-2016-1000342
    - Vulnerability that leads to possible injections due to verification not being fully validated.
    - There are two solutions listed, first is a patch available on GitHub and the other is an update of the current version.
    - An email recording when the vulnerability was first found is attached: (Koschany, 2018) From CVE-2016-1000338 through 1000346 and 1000352 all fall under the same discovery email.
  + CVE-2016-1000339
    - The Primary engine class, AESFastEngine, has the potential leak information with its use.
    - The solutions are to either avoid the use of AESFastEngine, update the package, or use an available patch located on GitHub.
    - (Koschany, 2018)
  + CVE-2016-1000341
    - Timing attack vulnerability where with the use of timing attackers can obtain signatures for authentication.
    - A patch on GitHub is available and updating the package are available solutions mentioned.
    - (Koschany, 2018)
  + CVE-2016-1000343
    - Generation of weak private keys when using default values is another vulnerability. Data can be breached because of weak encryption.
    - Use of better values other than default, patches, and updating the package are all potential solutions.
    - (Koschany, 2018)
  + CVE-2016-1000344, CVE-2016-1000345, CVE-2016-1000352
    - Unsafe modes are offered in this version that is no longer supported. Padding attacks due to the weak nature of the timing and encryptions are potential vulnerabilities from these modes.
    - Avoid using the DHIES/ECIES CBC modes, patches, and updates are available.
    - (Koschany, 2018)
  + CVE-2016-1000346
    - The public key is not fully validated allowing attackers the ability to figure out the private key.
    - An available patch is located on GitHub and an update are the recommendations.
    - (Koschany, 2018)
* Apache Log4j version 2.12.1
  + CVE-2020-9488
    - Improper validation can lead to host mismatch resulting in man-in-the-middle attacks.
    - An upgrade to a newer version or setting the system property mail.smtp.ssl.checkserveridentity to true globally are solutions.
    - A log of the discovery can be found in the references. (Sicker, 2020)
* SnakeYAML version 1.25
  + CVE-2017-18640
    - Entity expansion which can lead to denial-of-service attacks.
    - Data validation and updating the version are solutions to the vulnerability.
    - A thread can be found on the topic in the references (Allow configuration for preventing billion laughs attack, 2017)
* FasterXML Jackson-Databind version 2.10.2
  + CVE-2020-25649
    - The Jackson-Databind has a flaw that did not properly secure entity expansion leading to data integrity vulnerabilities.
    - There is a patch available on GitHub as well as updates that will solve the vulnerability.
    - The vulnerability is written in detail in the thread referenced. (Viewing email #r2b6ddb3a4f4cd11d8f6305011e1b7438ba81351... (and replies):, 2020)
* Apache Tomcat version 9.0.3
  + CVE-2021-33037
    - Apache Tomcat would incorrectly parse HTTP requests lead to data leaks.
    - The only solution is to update the version of Apache Tomcat.
    - An email of the report can be found in the references. (Viewing email #r612a79269b0d5e5780c62dfd34286a8037232fe... (and replies):, 2021)
* Redhat Hibernate Validator version 6.0.18
  + CVE-2020-10693
    - Expression Language injection is a vulnerability present in this dependency. There is a bug that does not properly validate the input.
    - A solution in the form of properly handling the input and or updating the system are available.
    - Discussions on the vulnerability as were as the mitigations are on the thread listed in the references. (Bug 1805501 (CVE-2020-10693) - CVE-2020-10693 hibernate-validator: Improper input validation in the interpolation of constraint error messages, 2020)
* Spring Framework version 5.2.3
  + CVE-2021-22118
    - The WebFlux application can allow for privilege escalation in this vulnerability. Attackers can read, write, and overwrite files uploaded by the application.
    - The only solution is to update the package to a safer version.
    - The report on the vulnerability is in the references. (CVE-2021-22118: Local Privilege Escalation within Spring Webflux Multipart Request Handling, n.d.)
  + CVE-2020-5421
    - A vulnerability where Reflected File Download attacks are accessible to attackers depending on the browser the are on.
    - The only solution is to update the package to a newer version.
    - There is a report located in the references. (CVE-2020-5421: RFD Protection Bypass via jsessionidc, n.d.)

## 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.

Manual Review Mitigation Plan

The first step would be to handle the input that is requested in the code. The classes that involve input need to have some form of validation to protect the system. Second, there would need to be a change to the @RequestMapping tag to a @PostMapping tag. This will protect the method underneath where there looks to be information being processed. Third, the exception handling from the try catch block should include a more generic error message instead of the detailed oriented one. There can be the addition of logging to track detailed error reports. Finally, the account\_balance variable will need to be changed to a private variable so that the data is not accessible from outside the class.

Static Testing Mitigation Plan

The first step would be to update most of the dependencies that will be used in the application. These would include Bouncy Castle Legion Cryptography API, Spring Framework, Apache Tomcat, SnakeYAML, FasterXML Jackson-Databind, and Apache Log4j. These dependencies are the ones that will be used the most in the application. Second, there will be a large focus on input validation since most of the dependencies have some form of a vulnerability related to validation.

# References

*Allow configuration for preventing billion laughs attack*. (2017, May 12). Retrieved from bitbucket: https://bitbucket.org/asomov/snakeyaml/issues/377/allow-configuration-for-preventing-billion

*Bug 1805501 (CVE-2020-10693) - CVE-2020-10693 hibernate-validator: Improper input validation in the interpolation of constraint error messages*. (2020, Feb 20). Retrieved from Red Hat Bugzilla: https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693

*California Consumer Privacy Act (CCPA)*. (n.d.). Retrieved from State of California Department of Justice: https://oag.ca.gov/privacy/ccpa

*CVE-2020-5421: RFD Protection Bypass via jsessionidc*. (n.d.). Retrieved from vmware Tanzu: https://tanzu.vmware.com/security/cve-2020-5421

*CVE-2021-22118: Local Privilege Escalation within Spring Webflux Multipart Request Handling*. (n.d.). Retrieved from vmware Tanzu: https://tanzu.vmware.com/security/cve-2021-22118

*Information exposure through a stack trace*. (n.d.). Retrieved from lgtm: https://lgtm.com/rules/6780073/

Jeganathan, S. (2019). DevSecOps A Systemic Approach for Secure Software Development. *ISSA Journal*, 20-27.

Kagan, J. (2021, Jan 18). *The Gramm-Leach-Bliley Act of 1999*. Retrieved from Investopedia: https://www.investopedia.com/terms/g/glba.asp

KirstenS. (n.d.). *Cross Site Request Forgery (CSRF)*. Retrieved from OWASP: https://owasp.org/www-community/attacks/csrf

Koschany, M. (2018, Jul 07). *[SECURITY] [DLA 1418-1] bouncycastle security update*. Retrieved from Debian: https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html

Levin, G. (2019, Jan 09). *TOP 7 REST API Security Threats*. Retrieved from RestCase: https://blog.restcase.com/top-7-rest-api-security-threats/

Manico, J., & Deltefsen, A. (2014). Access Control. In *Iron Clad Java.* Oracle Press.

Manico, J., & Deltefsen, A. (2014). Web Application Security Basics. In *Iron-Clad Java.* Oracle Press.

Nsrav. (n.d.). *Denial of Service*. Retrieved from OWASP: https://owasp.org/www-community/attacks/Denial\_of\_Service

*PCI SECURITY*. (n.d.). Retrieved from PCI Security Standards Council: https://www.pcisecuritystandards.org/pci\_security/

pedrorijo91. (2017, Apr 03). *codacy-find-sec-bugs*. Retrieved from github: https://github.com/codacy/codacy-find-sec-bugs/blob/master/src/main/resources/docs/description/SPRING\_CSRF\_UNRESTRICTED\_REQUEST\_MAPPING.md

Sicker, M. (2020, Apr 13). *Add support for specifying an SSL configuration for SmtpAppender*. Retrieved from Apache: https://issues.apache.org/jira/browse/LOG4J2-2819

*Viewing email #r2b6ddb3a4f4cd11d8f6305011e1b7438ba81351... (and replies):*. (2020, Dec 08). Retrieved from Apache: https://lists.apache.org/thread.html/r2b6ddb3a4f4cd11d8f6305011e1b7438ba813511f2e3ab3180c7ffda@%3Ccommits.druid.apache.org%3E

*Viewing email #r612a79269b0d5e5780c62dfd34286a8037232fe... (and replies):*. (2021, Jul 12). Retrieved from Apache: https://lists.apache.org/thread.html/r612a79269b0d5e5780c62dfd34286a8037232fec0bc6f1a7e60c9381%40%3Cannounce.tomcat.apache.org%3E

*What is cryptography and how do we use it in modern day computing?* (2018, Aug 02). Retrieved from Beaming: https://www.beaming.co.uk/knowledge-base/cryptography-use-modern-day-networking/