****

# 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** | **November 9th, 2022** | **Lawrence Arundel** | **Vulnerability Assessment** |

## Client



## Developer

Lawrence Arundel

## Interpreting Client Needs

**Interpretations**

The company has expressed deeply on topics including secure communications, international transactions, governmental restrictions, present/future external threats, and modernization requirements. The following interprets the client’s needs in a coherent and formulated pattern:

**Secure Communications** -The value of secure communications is of the utmost importance to the client. The client will be handling through RESTful web application programming interfaces financial plans for their customers which include savings, retirement, investment, and insurance. Secure communications will help protect the organization from potential external threats, increase the integrity of its customers, and further enhance the lives of the surrounding communities and humanity (SNHU, 2022, p. 1).

**International Transactions** -The client may or may not be engaging in international transactions mainly due to the nature of their system. The handling of financial plans may be primarily used wherever the company is operating. If the client wanted to handle international transactions, other means of implementation within the system and proper adherence would require for cross-platform transactions. Gramm-Leach-Bliley and various international privacy laws are in effect when engaging with international transactions or any transaction that is relevant to the service or system (Manico J. & Detlefsen A., 2015, p. 253).

**Government Restrictions** -When developing the software, we must take into account the various forms of restrictions when handling sensitive information across the network. These restrictions may include laws preventing organizations that handle sensitive information to encrypt private data by utilizing in-transit encryption and at-rest encryption and employ reasonable security practices to protect nonpublic data within the infrastructure (Crane, 2021, p. 1). The system must also adhere to proper cryptographic-based security systems to protect against sensitive information (Manico J. & Detlefsen A., 2015, p. 8). Lastly, the nature of the system and dealing with financial services must adhere to the Gramm-Leach-Bliley Act which states that companies who do not protect the integrity and security of consumers’ data are subject to criminal and civil penalties (Crane, 2021, p. 1).

**External Threats –** External threats play a critical role in developing the system for the client. We must understand what the external threats are now, and how they will evolve as best we can. Some of the external threats to the developed system may include SQL injections, XML/JSON injections, and turnkey automated attacks, (Manico J. & Detlefsen A., 2015, p. 8, 236). Many other attacks may include denial-of-service attacks, exposed authentication, sensitive data exposure, non-functional access control, parameter tampering, and MITM (Man-In-The-Middle-Attack) (Levin, 2019, p. 1).

**Modernization Requirements (Open-Source / Web Application)** – We can adopt a monolithic application architecture based on IBM’s example of how to implement effective and efficient systems. The monolithic architecture includes various layers which include customer information, security, database, and additional integration testing components (IBM, 2021, p. 1). By modernizing the application, we can further enhance flexibility for the application, better control over scaling and resource allocation and ease bandwidth and computing resources throughout the infrastructure (IBM,

2021, p. 1). The implementation of the architecture can facilitate the containerization of the application that is built on open-source software, therefore, taking advantage of operational benefits including cloud-native technologies, portability, security, and scalability (IBM, 2021, p. 1). Updating the application runtimes with various Java modular enterprises including the Open Liberty project or WebSphere Liberty to reciprocate and enhance cloud deployment operations. Once the framework has been properly implemented, we can further enhance the system by refactoring the monolith to microservices supported by a cloud-native architecture. By doing so, we allow the system to break free from the need to update the entire application when replacing a component or resolving issues found in service and provide flexibility in scaling, security, and creating a more responsive application (IBM, 2021, p. 1).

## Areas of Security

**Security Features**

The following security topics are important to the web application's design:

* Cryptography (Encryption use and Vulnerabilities) (SNHU, 2022, p. 1)
* Client / Server (Secure Distributed Composing) (SNHU, 2022, p. 1)
* Code Quality (Secure Coding Practices / Patterns) (SNHU, 2022, p. 1)
* APIs (Secure API Interactions) (SNHU, 2022, p. 1)
* Input Validation (Secure Input and Representations) (SNHU, 2022, p. 1)

**Justifications**

**Cryptography** -Cryptography helps with the storage of various passwords within the software, and adheres to productive and efficient systems. Cryptography is used to secure transactions and communications, safeguard personally identifiable information and other confidential data, authenticate identity, prevent document tampering, and establishes trust between servers (Gruhn, 2022, p. 1). By utilizing cryptography for the software developed for the client, we can implement libraries such as Google Keyczar in conjunction with open-source Bouncy Castle for a more robust and nuanced environment (Manico J. & Detlefsen A., 2015, p. 256). We must utilize cryptography in the design aspect to further enhance the security of the framework, increase performance, and adhere to policies implemented by governing officials.

**Client / Server –** Client/Server considerations for the software must be considered due to the security risk that is present. These security risks include insecure direct object references, replay attacks, spoofing, cross-site scripts also known as XSS attacks, and cross-site request forgery (Subramanian, 2019, p. 1). These attacks are why we should consider the client/server area of security to be prioritized within the software development process to create adaptable and secure environments.

**Code Quality –** The use of Restful web apps is required to implement the numerous properties of code quality. One of the primary justifications for using code quality is a characteristic known as input validation. The notion of quality code includes input validation, which is a key defense strategy for web applications when used correctly (Manico, 2015, p. 14). The query parameterization method is another essential factor for ensuring quality in these systems. Because the query execution plan is cached inside the database, query parameterization both defend against SQL injections and can improve the performance of repeated requests (Manico, 2015, p. 186). To prevent injectable assaults on the

infrastructure, a comparable defense against XML and JSON injections will need to be provided, which further explains the requirement. Denial-of-service assaults, which are common during the software development process, can be prevented by the necessity of high-quality code (Manico, 2015, p. 195).

**APIs -** To guarantee that data complies with the project's business objectives, APIs (Application Programming Interfaces) might use minimal input validation about DAO (Data Access Object) APIs. APIs can add identity and access control for the different layers of the infrastructure. If DAO APIs are established, this will guarantee that even a beginner programmer may use them without jeopardizing the security and integrity of the system. APIs give the project flexibility (Manico, 2015, p. 191). The usage of APIs in a Restful web development project is justified by the fact that they can offer supplementary protection against injections within the framework and bring uniformity to the system so that different users are only granted specific privilege rights and are permitted to use. Every program should function with the least degree of privilege required to finish the job, and the same goes for privileged users of the system (Manico, 2015, p. 56). The importance of APIs is that they will assist provide secure systems and adaptability, as well as providing more flexibility overall, for the software that is being produced (MuleSoft, 2022, p. 1).

**Input Validation -** One of the main protection strategies for online applications, or the "first line of defense," is input validation (Manico, 2015, p. 14). Input validation implemented correctly can thwart adversarial attacks on valid input fields, cookies, and other components of the HTTP request. Input validation establishes boundaries for the input that a web application may legally accept and aids with internal infrastructure security (Manico, 2015, p. 16). Additionally, input validation using highly complicated regular expressions helps maintain the web application reliable, effective, and secure. We may further improve the user experience within the system by adopting the Restful framework in conjunction with input validation. The framework is built on numerous REST services, including appropriate actions, caching, redirecting, forwarding, and security (Tanzu, 2022, p. 1). To comply with OWASP secure coding principles, maintain system integrity, and improve manageability throughout the entire system's infrastructure, we consider input validation for the program design (GitGuardian, 2022, p. 5).

## Manual Review

**Manual Analysis**

**Figure 1.0**

Graphical user interface, text

Description automatically generated



***Figure 1.0 is found in the CRUD.java file, and can be seen as a vulnerability to the system. The constructors within this file should be utilizing Object Construction or Mutable Objects to protect sensitive information and classes (Oracle, 2022, p. 1).***

**Figure 1.1**

Text

Description automatically generated



***Figure 1.1 is found in the CRUDController.java file, and can be seen as a vulnerability to the system. The parameter for value is hard-coded and if the @RequestParam were to become injected by an invalid data type, the system would be at risk. Performing proper checks before storing the value would be sufficient and ultimately lead to a more secure system.***

**Figure 1.2**

Graphical user interface, text, application

Description automatically generated



***Figure 1.2 is found in the customer.java file, and can be seen as a vulnerability to the system. The account balance variable with data type int is exposed to potential attacks. The data type is recommended to have the private access modifier to create a more robust and efficient system.***

**Figure 1.3**

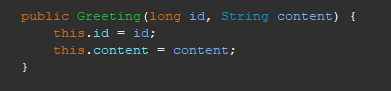
Text

Description automatically generated



***Figure 1.3 is found in the DocData.java file, and can be seen as a vulnerability to the system. The Connection object can be injectable and re-routed to a different local host or unwanted access could be permissible. Serialization and deserialization are also key concepts when performing the necessary checks for the local host and various information flowing between the client and server (JDBC, 2021, p. 1).***

**Figure 1.4**





***Figure 1.4 is found in the Greeting.java file, and can be seen as a vulnerability to the system. The constructors within this file should be utilizing Object Construction or Mutable Objects to protect sensitive information and classes (Oracle, 2022, p. 1).***

**Figure 1.5**

Text

Description automatically generated



***Figure 1.5 is found in the GreetingController.java file, and can be seen as a vulnerability to the system. The parameter for value and the default value are hard-coded and if the @RequestParam were to become injected by an invalid data type, the system would be at risk. Performing proper checks before storing the value would be sufficient and ultimately lead to a more secure system.***

**Figure 1.6**

Text

Description automatically generated



***Figure 1.6 is found in the myDateTime.java file, and can be seen as a vulnerability to the system. The mySecond, myMinute, and myHour variables with data type int are exposed to potential attacks. The***

***data types are recommended to have the private access modifier to create a more robust and efficient system.***

## Static Testing

**Dependency Report**

Text, letter

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

**Dependency List**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dependency** | **Vulnerability ID** | **CVSS Severity Level** | **Description** |
| Bcprov-jdk15on-1.46.jar | CVE-2016-1000352 | HIGH | **Description**  In Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB (Electronic Code Book) mode. Mode is regarded as unsafe and support for it has been removed from the provider (Awati, 2021, p. 1);(NIST, 2022, p. 1).  **Solution**  The model has been removed from the provider if the version is later than 1.55 (NIST, 2022, p. 1). |
| Spring-boot-2.2.4.RELEASE.jar | CVE-2022-27772 | HIGH | **Description**  Spring-boot versions before version v2.2.11.RELEASE was vulnerable to temporary directory hijacking.  **Solution**  The way to avoid this vulnerability is to make sure the version used is beyond v2.2.11.RELEASE (NIST, 2022, p. 1). |
| Logback-core-1.2.3.jar | CVE-2021-42550 | MEDIUM | **Description**  In logback version 1.2.7 and prior versions, an attack with the required privileges to edit configuration files could craft a malicious configuration allowing the execution of arbitrary code loaded from LDAP servers (NIST, 2022, p. 1).  **Solution**  Make sure the version being utilized is beyond 1.2.7 for logback (NIST, 2022, p. 1). |
| Log4j-API-2.12.1.jar | CVE-2021-44832 | MEDIUM | **Description**  Apache Log4j2 version is vulnerable to remote code execution (RCE) attack when a configuration uses a JDBC (Java database connectivity) Appender with a JNDI (Java Naming and Directory Interface API) LDAP (Lightweight Directory Access Protocol) data source URI (Uniform Resource Identifier) when an attacker has control of the target LDAP (Awati, 2021, p. 1);(IBM, 2021, p. 1); (NIST, 2022, p. 1) (Oracle, 2022, p. 1);(Tyagi, 2022, p. 1).  **Solution**  The issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2 (NIST, 2022, p. 1). |
| Snakeyaml-1.25.jar | CVE-2022-38752 | MEDIUM | **Description**  Using snakeYAML to parse untrusted YAML files may be vulnerable to denial-of-service attacks (DoS). If the parser is running on user-supplied input, an attack may supply content that causes the parser to crash by stack overflow (NIST, 2022, p. 1).  **Solution**  Make sure to store user inputs within a structure before parsing the data, running validators, or checking that the metadata is valid and noncorrupt (NIST, 2022, p. 1). |
| Jackson-databind-2.10.2.jar | CVE-2022-42004 | HIGH | **Description**  In FasterXML Jackson-databind before 2.13.4, resource exhaustion occurs because of a lack of a check-in BeanDeserializer.\_deserializerFromArry to prevent the use of deeply nested arrays (NIST, 2022, p. 1).  **Solution**  The vulnerability is undergoing analysis and will have a solution soon. I would recommend avoiding deeply nested arrays when utilizing deserialization (NIST, 2022, p. 1). |
| Tomcat-embed-core-9.0.30.jar | CVE-2021-43980 | LOW | **Description**  The simplified implementation of blocking reads and writes introduced in Tomcat 10 and back-ported to Tomcat 9.0.47 onwards exposed a long-standing concurrency bug in Apache Tomcat 10.1.0 to 10.1.0-M12, 10.0.0-M1 to 10.0.18, 9.0.0-M1 to 9.0.60 and 8.5.0 to 8.5.77 that could cause client connections to share an Http11Processor instance resulting in responses, or part responses, to be received by the wrong client (NIST, 2022, p. 1).  **Solution**  Make sure to run validators when reading and writing utilizing Tomcat 10 to resolve the issue (NIST, 2022, p. 1). |
| Spring-boot-starter-validation-2.2.4.RELEASE.jar | CVE-2022-27772 | HIGH | **Description**  Spring-boot versions before version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. Versions are no longer supported by the maintainer (NIST, 2022, p. 1).  **Solution**  Make sure that spring-boot versions are up to date to avoid such risk to the infrastructure (NIST, 2022, p. 1). |
| Hibernate-validator-6.0.18.Final.jar | CVE-2020-10693 | MEDIUM | **Description**  Flaw found in Hibernate Validator version 6.1.2. Final. A bug in the message interpolation processor enables invalid EL (Expression Language) expressions to be evaluated as if they were valid. The flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages (JSP, 2022, p. 1);(NIST, 2022, p. 1).  **Solution**  Integrating proper validations while allowing user input can help deter unknown attacks on the system (NIST, 2022, p. 1). |
| Spring-core-5.2.3.RELEASE.jar | CVE-2022-22971 | MEDIUM | **Description**  In spring framework versions before 5.3.20+, 5.2.22+, and unsupported versions, applications with a STOMP (Streaming Text Oriented Messaging Protocol) over WebSocket endpoint are vulnerable to a denial-of-service attack by an authenticated user (NIST, 2022, p. 1);(STOMP, 2022, p. 1).  **Solution**  Make sure when utilizing the spring framework, the system needs to use the supported versions to deter vulnerabilities such as these (NIST, 2022, p. 1). |



## Mitigation Plan

The following characteristics should be incorporated into the design to minimize and/or stop potential future attacks and lessen the vulnerabilities described:

**Mitigate Vulnerabilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identified Risk** | **Mitigation Plan** | **Category** | **Priority** |
| Privilege/Access Risk | **Action**  To stop SQL injections and other undesirable attacks on the system, implement the POLA or POLP concept (Manico, 2015, p. 193). | Client Management/Access | HIGH |
| Input Validation Requirements | **Action**  Utilize a variety of techniques and authentication throughout the system, ensuring that user inputs and untrusted objects are validated (Oracle, 2022, p. 1). | Client/Server | MEDIUM |
| Constructor Protection | **Action**  Implement the formation of mutable objects and make use of object building to shield sensitive class constructor constructors from exposure to undesired entities (Oracle, 2022, p. 1). | Client/Server | HIGH |
| Serialization/Deserialization Procedures | **Action**  To build a more stable and effective environment for maintaining secure coding standards, provide appropriate interfaces and encryption techniques (Oracle, 2022, p. 1). | Technical | MEDIUM |
| API Control | **Action**  Access Control is used to give the system's API functionality extra security. To create a more traditional and secure API, Access Control ensures that the system checks permissions to prevent callback methods (Oracle, 2022, p. 1). | Interface | LOW |
| Query Parameterization | **Action**  The technique expressed helps to build a dynamic SQL statement to safely bind untrusted data into placeholders within a query. Proper use of query parameterization to build dynamic database queries will protect the system from SQL injection attacks (Manico, 2015, p. 185). | Technical | HIGH |



**Assigned Task**

The actionable tasks we can carry out for the figures (1.0–1.6) are as follows evaluating the components to use to help mitigate/prevent future attacks (Note: The action item ID corresponds to the figure number where the identifiable code was analyzed in section 3 of the document):

**Actionable Task List**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Action Item ID | Rank | Priority | Owner | Assigned | Due | Done | Status | Notes |
| 1.0 | 1 | HIGH | Tom | 11/09/2022 | 12/05/2022 |  | Pending |  |
| 1.1 | 7 | LOW | Rachel | 11/08/2022 | 12/04/2022 |  | Pending |  |
| 1.2 | 3 | MEDIUM | Lawrence | 11/07/2022 | 12/03/2022 |  | Pending |  |
| 1.3 | 4 | MEDIUM | Jason | 11/06/2022 | 12/01/2022 |  | Pending |  |
| 1.4 | 5 | LOW | Melissa | 11/11/2022 | 12/10/2022 |  | Pending |  |
| 1.5 | 6 | LOW | John | 11/09/2022 | 12/01/2022 |  | Pending |  |
| 1.6 | 2 | HIGH | Chris | 11/12/2022 | 12/03/2022 |  | Pending |  |

**Utilizing Tools**

When handling the action item/task, bear in mind the following characteristics:

1. **Input Validation -** To avoid editable data or compromised data, we can provide native methods or wrappers, employ suitable input validation for the parsed variable, and validate user input. To avoid such a breach, I advise that we store the variable somewhere else (Figure 1.0). (Oracle, 2022, p. 1).
2. **Object Construction -** By adding keywords like "protected" or "copymutableinput" when implementing the object, we may use correct object creation and secure and improve the integrity of the system. To make it more difficult for outside entities to understand what the variable is used for within the system, I would also advise renaming the "id" and "content" variables (Figure 1.1). (Oracle, 2022, p. 1).
3. **Access Control -** We can apply appropriate access control to limit the kinds of users who have access to the different identification number-related data stored in the system. Using hashing techniques to store hashed information in a secure data structure will stop identification numbers from being leaked or compromised. To assist in creating a more secure system, I would also advise altering the "id" variable name (Figure 1.2). (Oracle, 2022, p. 1).
4. **Security Controls –** We can utilize multi-factor authentication, antivirus software, and firewalls to help secure the system. We must also adhere to compliance controls which include privacy laws and cybersecurity frameworks and standards to create an efficient and robust system (Reciprocity, 2022, p. 1).
5. **Automation** – Implement the correct practices to automate as much of the security within the framework as possible. By automating these processes, we can lessen the burden on the entire system and team and can facilitate remedial action when assessing vulnerabilities within the infrastructure (Reciprocity, 2022, p. 1).

## References

Awati, R. (2021, October 15). *What is a uniform resource identifier (URI)?* WhatIs.com. from <https://www.techtarget.com/whatis/definition/URI-Uniform-Resource-Identifier#:~:text=A%20Uniform%20Resource%20Identifier%20(URI)%20is%20a%20character%20sequence%20that,between%20and%20among%20these%20resources.>

Awati, R. (2021, August 27). *What is electronic code book (ECB) encryption and how does it work?* SearchSecurity., from [https://www.techtarget.com/searchsecurity/definition/Electronic-Code-Book#:~:text=Electronic%20Code%20Book%20(ECB)%20is,is%20broken%20into%20numerous%20blocks](https://www.techtarget.com/searchsecurity/definition/Electronic-Code-Book%23:~:text=Electronic%20Code%20Book%20(ECB)%20is,is%20broken%20into%20numerous%20blocks)

Crane, C. (2021, March 11). 10 data privacy and encryption laws every business needs to know., from [https://www.thesslstore.com/blog/10-data-privacy-and-encryption-laws-every-business-needs-to-know/](https://www.thesslstore.com/blog/10-data-privacy-and-encryption-laws-every-business-needs-to-know/%20)

GitGuardian, G. G. (2022, January 1). *Owasp secure coding practices-quick reference guide*. OWASP Secure Coding Practices-Quick Reference Guide | OWASP Foundation., from [https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated\_content](https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content%20)

Gruhn, D. (2022, October 19). *What is cryptography and why is it important?* Entrust Blog., from [https://www.entrust.com/blog/2021/06/why-is-cryptography-so-important-heres-what-you-need-to-know/](https://www.entrust.com/blog/2021/06/why-is-cryptography-so-important-heres-what-you-need-to-know/%20)

IBM, I. B. M. (2021, August 10). *Strategies for modernizing your applications*. IBM developer. from [https://developer.ibm.com/articles/strategies-for-modernizing-your-applications/](https://developer.ibm.com/articles/strategies-for-modernizing-your-applications/%20)

JDBC, J. D. B. C. (2021, September 2). JDBC Connection URL Attack., [from https://su18.org/post/jdbc-connection-url-attack/](file:///C:\Users\L69de\Downloads\from%20https:\su18.org\post\jdbc-connection-url-attack\)

JSP, J. S. P. (2022, July 1). *JSP Expression Language: El - javatpoint*. www.javatpoint.com., from [https://www.javatpoint.com/EL-expression-in-jsp#:~:text=The%20Expression%20Language%20(EL)%20simplifies,in%20JSP%20technology%20version%202.0.](https://www.javatpoint.com/EL-expression-in-jsp%23:~:text=The%20Expression%20Language%20(EL)%20simplifies,in%20JSP%20technology%20version%202.0.%20)

Levin, G. (2019, January 10). *Top 7 rest API security threats*. Top 7 REST API Security Threats., from [https://dzone.com/articles/top-7-rest-api-security-threats](https://dzone.com/articles/top-7-rest-api-security-threats%20)

Manico J. & Detlefsen A. (2015). *Iron-clad java: building secure web applications*. Mc Graw Hill Education. from <http://www.books24x7.com/marc.asp?bookid=72520>

MuleSoft, M. S. (2022, February 1). *What is a restful API?* MuleSoft., from [https://www.mulesoft.com/resources/api/restful-api#:~:text=One%20of%20the%20key%20advantages,the%20correct%20implementation%20of%20hypermedia.](https://www.mulesoft.com/resources/api/restful-api%23:~:text=One%20of%20the%20key%20advantages,the%20correct%20implementation%20of%20hypermedia.%20)

NIST, N. I. S. T. (2022, October 20). NATIONAL VULNERABILITY DATABASE., from <https://nvd.nist.gov/vuln/search>

Oracle, O. (2022, October 1). *Secure Coding Guidelines for Java SE*. Secure coding guidelines for Java SE. from [https://www.oracle.com/java/technologies/javase/seccodeguide.html](https://www.oracle.com/java/technologies/javase/seccodeguide.html%20)

Reciprocity, R. (2022, April 18). *Top 7 vulnerability mitigation strategies — reciprocity*. Top 7 Vulnerability Mitigation Strategies., from [https://reciprocity.com/blog/top-7-vulnerability-mitigation-strategies/](https://reciprocity.com/blog/top-7-vulnerability-mitigation-strategies/%20)

SlideModel, S. M. (2021, January 20). *Mitigation plan PowerPoint template*. SlideModel., from [https://slidemodel.com/templates/mitigation-plan-powerpoint-template/](https://slidemodel.com/templates/mitigation-plan-powerpoint-template/%20)

SNHU, S. N. H. U. (2022, October 22). Project One Guidelines and Rubric., from [https://learn.snhu.edu/d2l/le/content/1196941/viewContent/20849009/View](https://learn.snhu.edu/d2l/le/content/1196941/viewContent/20849009/View%20)

SNHU, S. N. H. U. (2022, October 22). CS 305 FlowChart, from [https://learn.snhu.edu/content/enforced/1196941-CS-305-H7328-OL-TRAD-UG.22EW2/course\_documents/CS%20305%20Vulnerability%20Assessment%20Process%20Flow%20Diagram.pdf?\_&d2lSessionVal=hKyr7Fty4usRpZFRlwuEtnZnV&ou=1196941](https://learn.snhu.edu/content/enforced/1196941-CS-305-H7328-OL-TRAD-UG.22EW2/course_documents/CS%20305%20Vulnerability%20Assessment%20Process%20Flow%20Diagram.pdf?_&d2lSessionVal=hKyr7Fty4usRpZFRlwuEtnZnV&ou=1196941%20)

STOMP, S. T. O. M. P. (2022, September 1). *What is it?* STOMP., [from https://stomp.github.io/](from%20https:/stomp.github.io/%20)

Subramanian, H. (2019, June 6). *Rest API security vulnerabilities - dzone security*. dzone.com., from [https://dzone.com/articles/rest-api-security-vulnerabilities](https://dzone.com/articles/rest-api-security-vulnerabilities%20)

Tanzu, V. M. (2022, January 1). Spring Quickstart Guide, from [https://spring.io/quickstart](https://spring.io/quickstart%20)

Tanzu, V. M. (2022, January 1). Spring Quickstart Guide, from <https://spring.io/guides/tutorials/rest/>

Tyagi, S. (2000, March 24). *LDAP and JNDI: Together forever*. InfoWorld., from [https://www.infoworld.com/article/2076073/ldap-and-jndi--together-forever.html#:~:text=LDAP%20is%20a%20standard%20way,access%20directory%20services%20like%20LDAP.](https://www.infoworld.com/article/2076073/ldap-and-jndi--together-forever.html%23:~:text=LDAP%20is%20a%20standard%20way,access%20directory%20services%20like%20LDAP.%20)

US Department Homeland Security, U. S. D. H. S. (1999, January 1). *Search CVE list*. Search CVE List., from [https://cve.mitre.org/cve/search\_cve\_list.html](https://cve.mitre.org/cve/search_cve_list.html%20)

Wittwer, J. (2021, November 15). *Action items template*. Action Item Templates., 2022, from [https://www.vertex42.com/ExcelTemplates/action-items-template.html](https://www.vertex42.com/ExcelTemplates/action-items-template.html%20)