# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **21/05/2025** | **Justin Crouch** | **Add client’s needs & areas of security** |
| **1.0** | **23/05/2025** | **Justin Crouch** | **Add manual review** |
| **1.0** | **24/05/2025** | **Justin Crouch** | **Add static testing & mitigation plan** |

## Client



## Developer

Justin Crouch

**1. Interpreting Client Needs**

The client, Artemis Financial, requests expertise regarding how to secure their RESTful application programming interface (API). The company consults with their clients and builds custom financial plans regarding savings, retirement, investments, and insurance.

This report’s domain of assessment consists of the client’s API source code, including any dependencies used within the software. Any subject outside the stated domain, such as the company’s database, firewalls, routing configurations, and physical security, are assessed only within the context of the client’s source code.

Due to the nature of the service the company provides its clients, information sent through the client’s API should be considered strongly sensitive and highly classified. All data that is sent through and saved by the API must be treated as Personally Identifiable Information (PII) and, therefore, must be properly encrypted using modern encryption practices that abide by the company’s working nation’s handling of PII regulations. This will prevent third-party entities from viewing a user’s sensitive information if the database is compromised or a network packet is captured.

At every point in the API that accepts data from the user (for example, an input field, a URL’s parameters, a GET request packet, etc), these areas must effectively handle user input validation. This will prevent multiple forms of injection attacks throughout the API.

The third-party libraries used within the software must utilize up-to-date versions. Any data sent to these third-party libraries must be formatted correctly in regard to the libraries’ methods and classes. This will prevent known security issues with the respected libraries.

**2. Areas of Security**

Given the client’s needs and potential threats stated in the previous section, the following are the areas of security that will be considered for the security assessment:

* **Input validation:** Given that the API will accept and process input generated (directly or indirectly) by the user, the software must securely validate all input vectors. Preferably, the software should parameterize any input data before it is used in third-party library method or system calls. Input data should only be accepted if it falls within a certain size range, contains only elements within the set of the minimally required characters, and is of the correct data type and encoding; else, the input data should be immediately rejected.
* **Secure API Interactions:** Some endpoints in the client’s API will, potentially, allow access to a user’s account information, thus needing a user’s login information or authorization key to access these endpoints. Therefore, the API must use the correct requests (for example, POST instead of GET) so to prevent the user’s login information or authorization key from being exposed in the URL, where it is stored to the user’s browser history or shown to a malicious entity capturing network packets.
* **Cryptography:** Because the company stores a customer’s sensitive information (i.e. financial data) and will send this information to the authorized and authenticated user of the API, the data must be properly encrypted when stored and in-transit. Specifically, the data should be encrypted in compliance with the company’s working nation’s regulations on storing PII.

**3. Manual Review**

File: GreetingController.java

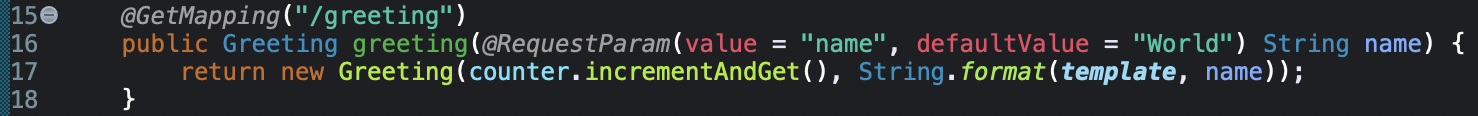
Class: GreetingController

Endpoint: /greeting

Line: 17

Summary: *See Figure 1*

* Lack of input validation. Susceptible to DoS with function String.format()

****

***Figure 1: GreetingController.java, endpoint /greeting***

File: DocData.java

Class: DocData

Function: read\_document

Line: 27

Summary: *See Figure 2*

* Database connection uses root privilege
* Database connection displays a hardcoded login account
* Database root account uses default login and password
* Database connection failure leads to a leak of the error stack

***A screen shot of a computer code

Description automatically generated***

***Figure 2: DocData.java, function read\_document***

File: CRUDController.java

Class: CRUDController

Endpoint: /read

Line: 13

Summary: *See Figure 3*

* Lack of input validation with request parameter name
* Lack of access control to DocData

***A black background with text

Description automatically generated***

***Figure 3: CRUDController.java, endpoint /read***

File: customer.java

Class: customer

Line: 5

Summary: *See Figure 4*

* A customer’s *account\_balance* is exposed to the entire package (in this case, the entire application). *Account\_balance* should be private

***A black background with blue and orange letters

Description automatically generated***

***Figure 4: customer.java***

File: myDateTime.java

Class: myDateTime

Line: 5-7

Summary: *See Figure 5*

* Attributes *mySecond, myMinute,* and *myHour* are exposed to the entire package (in this case, the entire application). These attributes should be private

***A close up of text

Description automatically generated***

***Figure 5: myDateTime.java***

**4. Static Testing**

**bcprov-jdk15on-1.46.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2016-1000343 | Weak private key generation from DSA generator with default values | Explicitly pass parameters to generator | (*CVE-2016-1000343,* 2018) |

**hibernate-validator-6.0.18.Final.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2020-10693 | Improper input validation | Upgrade to 6.0.20.Final | (*CVE-2020-10693,* 2020) |

**jackson-databind-2.10.2.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2020-25649 | |  |  | | --- | --- | |  | Improper Restriction of XML External Entity Reference | | Upgrade to 2.10.5.1 | (*CVE-2020-25649*, 2020) |

**log4j-api-2.12.1.jar**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vulnerability** | **Summary** | | **Solutions** | **Attribution** |
| CVE-2020-9488 | | Improper Certificate Validation | Upgrade to 2.12.3  Set system property mail.smtp.ssl.checkserveridentity to true | (*[LOG4J2-2819]*, 2020) |

**logback-core-1.2.3.jar**

**logback-classic-1.2.3.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2021-42550 | Deserialization of untrusted data | Set logback configuration files as read-only | (*Release of Version 1.2.9*, 2021) |
| CVE-2023-6378 | Deserialization of untrusted data | Upgrade to 1.2.13 | (*Release of Logback Versions 1.3.12 and 1.4.12*, 2023) |

**snakeyaml-1.25.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| |  |  | | --- | --- | |  | CVE-2022-41854 | | Out-of-bounds write  Stack-based buffer overflow | Validate user supplied YAML files/input before parsing | (*CVE-2022-41854*, 2022) |
| CVE-2022-1471 | Deserialization of untrusted data  Improper input validation | Update to 2.0  Use SnakeYaml's SafeConstructor | (*CVE-2022-1471*, 2022) |

**spring-boot-2.2.4.RELEASE.jar**

**spring-boot-starter-web-2.2.4.RELEASE.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2023-20883 | Uncontrolled resource consumption | Upgrade to 2.7.12 | (*CVE-2023-20883*, 2023) |
| CVE-2023-20873 | Susceptible to a security bypass | Upgrade to 2.7.12 | (*CVE-2023-20873*, 2023) |
| CVE-2022-27772 | Susceptible to temporary directory hijacking | Upgrade to 2.7.12 | (*CVE-2022-27772*, 2022) |

**spring-context-5.2.3.RELEASE.jar**

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

**spring-expression-5.2.3.RELEASE.jar**

**spring-web-5.2.3.RELEASE.jar**

**spring-webmvc-5.2.3.RELEASE.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2021-22118 | Susceptible to privilege escalation | Upgrade to 5.2.24.RELEASE | (*CVE-2021-22118*, 2021) |
| CVE-2022-22965 | Susceptible to remote code execution | Upgrade to 5.2.24.RELEASE | (*CVE-2022-22965*, 2022) |
| CVE-2023-20863 | Susceptible to ReDoS | Upgrade to 5.2.24.RELEASE | (*CVE-2023-20863*, 2023) |

**tomcat-embed-core-9.0.30.jar**

**tomcat-embed-websocket-9.0.30.jar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Summary** | **Solutions** | **Attribution** |
| CVE-2020-1938 | Susceptible to privilege escalation and remote code execution | Upgrade to 9.0.103 | (*CVE-2020-1938*, 2020) |
| CVE-2025-24813 | Path Equivalence Vulnerability | Upgrade to 9.0.103 | (*CVE-2025-24813*, 2025) |
| CVE-2025-31651 | Improper Encoding or Escaping of Output | Upgrade to 9.0.103 | (*CVE-2025-31651*, 2025) |

**5. Mitigation Plan**

The manual review revealed the main issue in the source code is a lack of input validation. Specifically, data provided through a URL parameter is used directly in functions, without any validation. This can lead in injection attacks and business logic bypasses. Other issues in the source code involve using database root credentials explicitly in code, lack of access control to the database, and exposing class attributes to the application that should be private.

The static testing report revealed most of the dependencies are out-of-date. This is resulting in numerous security vulnerabilities.

The following list of procedures will improve the security of the client’s API:

1. Update dependencies to the recommended versions stated in the Static Testing section
2. Apply recommended configurations stated in the Static Testing section
3. Validate all forms of user input
4. Insert access control before a user can access the database (this includes only reading the database)
5. Store database credentials as an environment variable on the server
6. Parameterize database queries
7. Use database credentials with the least amount of privilege required for a query to function
8. Ensure all class attributes are private
9. Ensure only the minimum needed class methods are exposed as public in the application
10. Apply recommended coding techniques and classes stated in the Static Testing section

**References**

*CVE-2016-1000343*. (2018). National Vulnerability Database. <https://nvd.nist.gov/vuln/detail/CVE-2016-1000343>

*CVE-2020-10693 Hibernate-Validator: Improper input validation in the interpolation of constraint error messages*. (2020). Red Hat Bugzilla. https://bugzilla.redhat.com/show\_bug.cgi?id=CVE-2020-10693

*CVE-2020-1938*. (2020). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2020-1938

*CVE-2020-25649 Jackson-DaTABind: FasterXML DOMDeserializer Insecure entity expansion is vulnerable to XML External Entity (XXE)*. (2020). Red Hat Bugzilla. <https://bugzilla.redhat.com/show_bug.cgi?id=1887664>

*CVE-2021-22118: Local Privilege Escalation within Spring WebFlux Multipart Request Handling*. (2021). Spring. https://spring.io/security/cve-2021-22118

*CVE-2022-1471*. (2022). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2022-1471

*CVE-2022-22965: Spring Framework RCE via Data Binding on JDK 9+*. (2022). Spring. https://spring.io/security/cve-2022-22965

*CVE-2022-27772*. (2022). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2022-27772#vulnCurrentDescriptionTitle

*CVE-2022-41854 Dev-Java/SnakeyaML: DOS via Stack Overflow*. (2022). Red Hat Bugzilla. https://bugzilla.redhat.com/show\_bug.cgi?id=2151988

*CVE-2023-20863: Spring Expression DOS vulnerability*. (2023). Spring. https://spring.io/security/cve-2023-20863

*CVE-2023-20873*. (2023). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2023-20873

*CVE-2023-20883*. (2023). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2023-20883

*CVE-2025-24813*. (2025). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2025-24813

*CVE-2025-31651*. (2025). National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2025-31651

*[LOG4J2-2819] Add support for specifying an SSL configuration for SmtpAppender - ASF JIRA*. (2020, April 13). Apache. https://issues.apache.org/jira/browse/LOG4J2-2819

*Release of Logback versions 1.3.12 and 1.4.12*. (2023). Logback News. https://logback.qos.ch/news.html#1.3.12

*Release of version 1.2.9*. (2021). Logback News. https://logback.qos.ch/news.html