****

# CS 305 Project One

**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 5](#_Toc32574613)

[4. Static Testing 5](#_Toc32574614)

[5. Mitigation Plan 8](#_Toc32574615)

[6. Citations 9](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **November 14, 2021** | **Matt Oliphant** |  |

## 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

Matt Oliphant

## 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 communication is very valuable to Artemis Financial since they deal with financial information and other sensitive user data. It could be detrimental to the company if any patron were able to see or access any other patron’s data. The scenario provided doesn’t state that Artemis Financial produces international transactions. Global Rain works with agencies around the world, but it needs to be made clear whether Artemis Financial does or not. Since Artemis Financial is a financial consulting company, their industry is bound to the Graham–Leach–Bliley Act, which requires “companies that offer consumer-centric financial services have to publicly disclose their information-sharing practices and take steps to protect consumer data (Manico, n.d.)”.

External threats present now and in the future are more than likely going to be attackers who are trying to gain access to Artemis Financials’ patrons. To prevent some of these attacks, the web application needs to be designed with secure patterns and structures. Since Artemis Financial has created a RESTful API, they modernized their applications capabilities since RESTful APIs are very flexible allowing for future modification. Open-source libraries will need to be sanitized and evaluated for defects before they are used.

## 2. 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*.

Due to the sensitive nature of the data that Artemis Financial is tasked with keeping secure, I feel as if most of the areas of security are applicable to their software application. Utilizing a RESTful API will do a lot of work toward preventing invalid input, but all input needs to be verified. It’s required for Artemis Financial patrons to enter sensitive personal data, and an attacker could enter an attack string in an attempt to gain unauthorized access. Secure API interactions are required:

“RESTful applications rely on the underlying security of the API ecosystem rather than including security within the REST architecture style. In addition to securing RESTful API calls with the HTTPS protocol, session-based authentication should be utilized. Currently, most RESTful applications leverage OAuth 2.0 and JWT is the newcomer that is gaining more and more popularity with API developers. OAuth, JWT, and Basic Auth all use headers for transmitting credentials, and API providers should be doing the same with all API keys. While easy to do as parameters, they are more secure as headers (Levin, n.d.)”.

Cryptography will be utilized to protect data at rest and in transmission. The client/ server may be the one vulnerability that Artemis Financial might not worry about since they made their API RESTful. A representational state transfer application programming interface is loosely coupled and stateless so there isn’t any need for the client or server to transfer data directly. Code error needs to be accounted for and systems need to be in place in case an error occurs. Secure coding practices is a must to protect Artemis Financials’ patrons. Creating logs and including pertinent information can provide insight to previous attacks so that vulnerabilities can be identified. The application also needs to prepared for DoS and other attacks. Finally, a RESTful API will do a great job at encapsulating data from the user, but secure data structures must still be used to further promote encapsulation.

## 3. Manual Review

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

Within the customer.java file, there is a getter that will return a bank account number. If an attacker were to gain access to the source code, this would allow them to be able to alter patron account information. Inside the myDateTime.java file an mutator is used to set the date and time. The class also uses int variables and an array to store date and time data. It would be safer to use a Java library (java.time) to construct a date object. It’s easy to get the date and time if set up properly using encapsulated methods; this is necessary because it’s also easy to change the value of an int attribute. In the same java file, a method to retrieve the date and time returns a new object which is a vulnerability. The new keyword used to construct objects should be used as little as possible. I’m still new to SQL and the Spring framework, so I’m not sure of vulnerabilities in those files.

## 4. 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*

The dependency check report located 9 .jar files whose dependencies are vulnerable which equates to 42 unique CVE codes:

**bcprov-jdk15on-1.46.jar**: *Highest Confidence – Unknown Severity*

cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46  
cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46  
cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46  
cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46

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, CVE-2020-15522,

CVE-2020-26939

**hibernate-validator-6.0.18.Final.jar**: *Highest Confidence – Medium Severity*

cpe:2.3:a:redhat:hibernate\_validator:6.0.18

CVE-2020-10693

**jackson-databind-2.10.2.jar**: *Highest Confidence – High Severity*

cpe:2.3:a:fasterxml:jackson-databind:2.10.2  
cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2

CVE-2020-25649

**log4j-api-2.12.1.jar**: *Highest Confidence – Low Severity*

cpe:2.3:a:apache:log4j:2.12.1

CVE-2020-9488

**snakeyaml-1.25.jar**: *Highest Confidence – High Severity*

cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25

CVE-2017-18640

**spring-aop-5.2.3.RELEASE.jar**: *Highest Confidence – High Severity*

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

CVE-2020-5421, CVE-2021-22096, CVE-2021-22118

**spring-core-5.2.3.RELEASE.jar**: *Highest Confidence – High Severity*

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

CVE-2020-5421, CVE-2021-22096, CVE-2021-22118

**tomcat-embed-core-9.0.30.jar**: *Highest Confidence – Critical Severity*

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

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-2020-9484, CVE-2021-24122, CVE-2021-25122, CVE-2021-25329, CVE-2021-30640, CVE-2021-33037, CVE-2021-41079, CVE-2021-42340

**tomcat-embed-websocket-9.0.30.jar**: *Highest Confidence – Critical Severity*

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

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, CVE-2021-25122, CVE-2021-25329, CVE-2021-30640, CVE-2021-33037, CVE-2021-41079, CVE-2021-42340

\***Note: Duplicate CVE codes have been combined.**

CVE-2013-1624: Cryptographic issue: not considering timing side-channel attacks

CVE-2015-6644: Information disclosure vulnerability

CVE-2015-7940: Invalid curve attack: does not validate if a point is within the ecliptic curve

CVE-2016-1000338: Does not fully validate encoding of signature on verification

CVE-2016-1000339: Data leak: data channel on CPU can be monitored

CVE-2016-1000341: Timing attacks on DSA signature generation

CVE-2016-1000342: Does not fully validate encoding of signature on verification

CVE-2016-1000343: Default values use weak private key

CVE-2016-1000344: Allows ECB mode which is considered unsafe

CVE-2016-1000345: Vulnerable to padding oracle attack

CVE-2016-1000346: Public key is not full validated which leads to invalid keys

CVE-2016-1000352: Allows ECB mode which is considered unsafe

CVE-2017-13098: Provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated

CVE-2017-18640: Allows entity expansion during a load operation

CVE-2018-5382: BKS keystore use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore

CVE-2018-1000613: Vulnerable to private key theft that can result in desterilization

CVE-2019-17569: Apache Tomcat refactoring leads to a regression

CVE-2020-1935: Parser allows invalid HTTP headers to be parsed as valid

CVE-2020-1938: Apache JServ Protocol can trust the user more than they need

CVE-2020-5421: Old Spring framework versions security can be bypassed

CVE-2020-8022: Incorrect default permission vulnerability

CVE-2020-9484: An attacker can control the contents and name of a file

CVE-2020-9488: Improper validation with host mismatch

CVE-2020-10693: A bug in message interpolation processor enable invalid expressions to be evaluated

CVE-2020-11996: Can trigger high CPU usage for several seconds: server can become unresponsive

CVE-2020-13934: Out of memory exception can occur leading to denial of service

CVE-2020-13935: Payload length in a web socket frame was not correctly validated

CVE-2020-13943: If too many concurrent streams, headers can be displayed that aren’t supposed to be

CVE-2020-15522: Timing issue with EC math library that can expose information about a private key

CVE-2020-17527: Info can leak since Tomcat allows reuse of an HTTP request header

CVE-2020-25649: Entity expansion is not secured properly

CVE-2020-26939: Attackers can use behavior to error inputs to expose information about a private key

CVE-2021-22096: Malicious input to insert additional log entries in Spring Framework

CVE-2021-22118: Vulnerable to privilege escalation in Spring Framework

CVE-2021-24122: Susceptible to JSP source code disclosure

CVE-2021-25122: Duplicate request headers could copy one user’s data to another user

CVE-2021-25329: Configuration edge case is unlikely to be used and is vulnerable

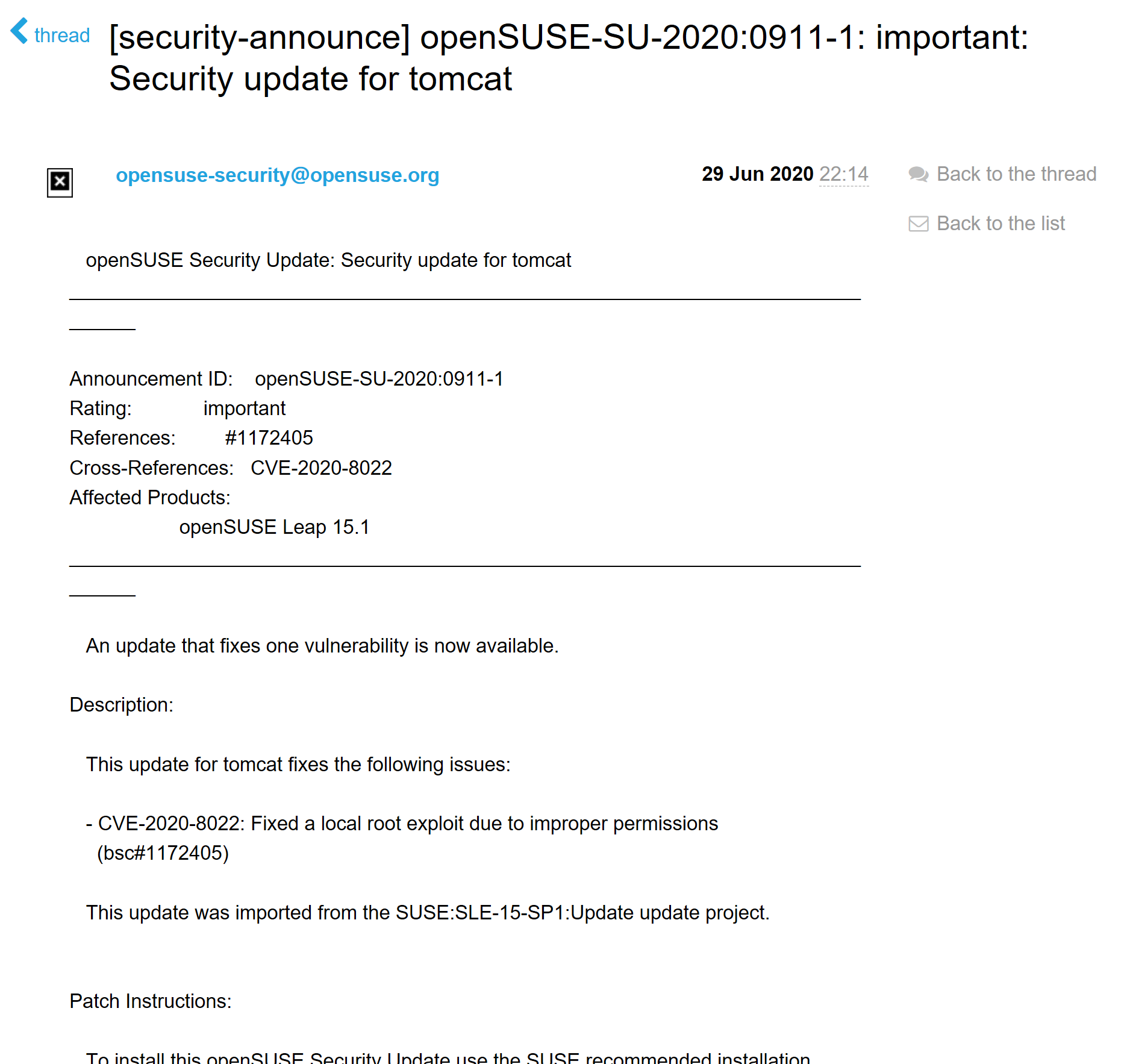
CVE-2021-30640: Attacker can authenticate using variations of valid username/ bypass protections

CVE-2021-33037: Incorrectly parses HTTP transfer encoding request header

CVE-2021-41079: Did not properly validate incoming TLS packets

CVE-2021-42340: Memory leak introduced by bug fix

Solutions to most of the vulnerabilities above are to update the version of dependencies that the application utilizes, and avoid the known problem configurations. Each CVE code above has a clickable link that describes the problem, severity, and references to advisories, solutions, and tools. Each of the CVE codes has several hyperlinks associated with the fix and would be too much detailed to go into here. For example, the following is a screenshot of one of the links that can offer remedies to vulnerabilities:



## 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 Financials’ software application.

The number of vulnerabilities Artemis Financials’ app currently has is very high and more than 30 of them are critical. It’s necessary to remove these vulnerabilities; step one is to update the current Spring Framework from version 2.2.4 to the Spring Expression Language, or another updated version of the Spring Framework. As software ages and technology advances, new vulnerabilities arise that requires maintenance or possibly restructurion an application altogether. However, a newer version of the framework attempts to solve the vulnerabilities, so it’s critical to constantly update when new versions are available. The same is true for the Apache Tomcat server that Artemis Financial is utilizing. The newest version needs to be used in order to avoid older security vulnerabilities. Since technology is rapidly advancing, it’s critical to design with security in mind and to maintain source code after it has been deployed. The principle of least privilege should be applied, input should be validated, queries should be parametrized, procedures for escapement and solutions to security breaches should be planned.

The only issue I have found with the updated Spring Framework is CVE-2011-2730, but it’s easy to mitigate this vulnerability. Disabling the double resolution functionality will remove the possibility for an expression language injection and can be done by adding the following code to the app’s web.xml file:

“<context-param>

<description>Spring Expression Language Support</description>

<param-name>springJspExpressionSupport</param-name>

<param-value>false</param-value>

</context-param> (OWASP, n.d.).”

Another strategy to avoid injection is to validate and/or encode the data to ensure it’s not expression language. Regular Expressions are a good way to ensure only valid characters are allowed to be passed. Sensitive data should not be transmitted in a header, but should instead be included in the body of an HTTP request/ response.

## 6. Citation

Manico, J. (n.d.). *Iron-Clad Java*. O’Reilly Online Learning. Retrieved November 14, 2021, from <https://www.oreilly.com/library/view/iron-clad-java/9780071835886/>

Levin, G., & Core, B. (n.d.). REST API Security - DZone Refcardz. Dzone.Com. Retrieved November 14, 2021, from <https://dzone.com/refcardz/rest-api-security-1>

Open Web Application Security Project. (n.d.). Expression Language Injection | OWASP. OWASP. Retrieved November 7, 2021, from <https://owasp.org/www-community/vulnerabilities/Expression_Language_Injection#>