

# Semgrep Relatório de Vulnerabilidades

## html.security.audit.missing-integrity.missing-integrity

**Arquivo:** docs/index.html  
**Severidade:** WARNING  
**Linha do Código:** 7  
**Descrição da Vulnerabilidade:** This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you’re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** robot/goat.robot  
**Severidade:** ERROR  
**Linha do Código:** 116  
**Descrição da Vulnerabilidade:** JWT token detected

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/controller/StartLesson.java  
**Severidade:** WARNING  
**Linha do Código:** 71  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java  
**Severidade:** WARNING  
**Linha do Código:** 64  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LabelDebugService.java  
**Severidade:** WARNING  
**Linha do Código:** 77  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LessonInfoService.java  
**Severidade:** WARNING  
**Linha do Código:** 28  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LessonMenuService.java  
**Severidade:** WARNING  
**Linha do Código:** 76  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LessonProgressService.java  
**Severidade:** WARNING  
**Linha do Código:** 32  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/LessonTitleService.java  
**Severidade:** WARNING  
**Linha do Código:** 29  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/RestartLessonService.java  
**Severidade:** WARNING  
**Linha do Código:** 52  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/container/service/SessionService.java  
**Severidade:** WARNING  
**Linha do Código:** 24  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/challenges/challenge1/ImageServlet.java  
**Severidade:** WARNING  
**Linha do Código:** 17  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.spring.security.injection.tainted-sql-string.tainted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java  
**Severidade:** ERROR  
**Linha do Código:** 60  
**Descrição da Vulnerabilidade:** User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/challenges/challenge5/Assignment5.java  
**Severidade:** ERROR  
**Linha do Código:** 65  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/EncodingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 52  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.lang.security.audit.tainted-session-from-http-request.tainted-session-from-http-request

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/EncodingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 54  
**Descrição da Vulnerabilidade:** Detected input from a HTTPServletRequest going into a session command, like `setAttribute`. User input into such a command could lead to an attacker inputting malicious code into your session parameters, blurring the line between what's trusted and untrusted, and therefore leading to a trust boundary violation. This could lead to programmers trusting unvalidated data. Instead, thoroughly sanitize user input before passing it into such function calls.

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 46  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 53  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.lang.security.audit.crypto.use-of-md5.use-of-md5

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 55  
**Descrição da Vulnerabilidade:** Detected MD5 hash algorithm which is considered insecure. MD5 is not collision resistant and is therefore not suitable as a cryptographic signature. Use HMAC instead.

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 65  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/HashingAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 71  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/cryptography/SigningAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 52  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.lang.security.audit.object-deserialization.object-deserialization

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/deserialization/InsecureDeserializationTask.java  
**Severidade:** WARNING  
**Linha do Código:** 57  
**Descrição da Vulnerabilidade:** Found object deserialization using ObjectInputStream. Deserializing entire Java objects is dangerous because malicious actors can create Java object streams with unintended consequences. Ensure that the objects being deserialized are not user-controlled. If this must be done, consider using HMACs to sign the data stream to make sure it is not tampered with, or consider only transmitting object fields and populating a new object.

## java.lang.security.audit.object-deserialization.object-deserialization

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/deserialization/SerializationHelper.java  
**Severidade:** WARNING  
**Linha do Código:** 18  
**Descrição da Vulnerabilidade:** Found object deserialization using ObjectInputStream. Deserializing entire Java objects is dangerous because malicious actors can create Java object streams with unintended consequences. Ensure that the objects being deserialized are not user-controlled. If this must be done, consider using HMACs to sign the data stream to make sure it is not tampered with, or consider only transmitting object fields and populating a new object.

## java.lang.security.audit.cookie-missing-httponly.cookie-missing-httponly

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/hijacksession/HijackSessionAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 89  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/hijacksession/cas/HijackSessionAuthenticationProvider.java  
**Severidade:** WARNING  
**Linha do Código:** 48  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.lang.security.audit.crypto.weak-random.weak-random

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 53  
**Descrição da Vulnerabilidade:** Detected use of the functions `Math.random()` or `java.util.Random()`. These are both not cryptographically strong random number generators (RNGs). If you are using these RNGs to create passwords or secret tokens, use `java.security.SecureRandom` instead.

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTSecretKeyEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 58  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.servlets.security.cookie-issecure-false.cookie-issecure-false

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 130  
**Descrição da Vulnerabilidade:** Default session middleware settings: `setSecure` not set to true. This ensures that the cookie is sent only over HTTPS to prevent cross-site scripting attacks.

## java.lang.security.audit.cookie-missing-httponly.cookie-missing-httponly

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 131  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'

## java.lang.security.audit.cookie-missing-secure-flag.cookie-missing-secure-flag

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 131  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'secure' flag by calling 'cookie.setSecure(true);'

## java.servlets.security.cookie-issecure-false.cookie-issecure-false

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 135  
**Descrição da Vulnerabilidade:** Default session middleware settings: `setSecure` not set to true. This ensures that the cookie is sent only over HTTPS to prevent cross-site scripting attacks.

## java.lang.security.audit.cookie-missing-httponly.cookie-missing-httponly

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 136  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'

## java.lang.security.audit.cookie-missing-secure-flag.cookie-missing-secure-flag

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/JWTVotesEndpoint.java  
**Severidade:** WARNING  
**Linha do Código:** 136  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'secure' flag by calling 'cookie.setSecure(true);'

## java.spring.security.injection.tainted-url-host.tainted-url-host

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/jwt/claimmisuse/JWTHeaderJKUEndpoint.java  
**Severidade:** ERROR  
**Linha do Código:** 51  
**Descrição da Vulnerabilidade:** User data flows into the host portion of this manually-constructed URL. This could allow an attacker to send data to their own server, potentially exposing sensitive data such as cookies or authorization information sent with this request. They could also probe internal servers or other resources that the server running this code can access. (This is called server-side request forgery, or SSRF.) Do not allow arbitrary hosts. Instead, create an allowlist for approved hosts, hardcode the correct host, or ensure that the user data can only affect the path or parameters.

## java.lang.security.httpservlet-path-traversal.httpservlet-path-traversal

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/pathtraversal/ProfileUploadRetrieval.java  
**Severidade:** ERROR  
**Linha do Código:** 91  
**Descrição da Vulnerabilidade:** Detected a potential path traversal. A malicious actor could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.

## java.servlets.security.cookie-issecure-false.cookie-issecure-false

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 79  
**Descrição da Vulnerabilidade:** Default session middleware settings: `setSecure` not set to true. This ensures that the cookie is sent only over HTTPS to prevent cross-site scripting attacks.

## java.lang.security.audit.cookie-missing-httponly.cookie-missing-httponly

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 81  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'

## java.lang.security.audit.cookie-missing-secure-flag.cookie-missing-secure-flag

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 81  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'secure' flag. The 'secure' flag for cookies prevents the client from transmitting the cookie over insecure channels such as HTTP. Set the 'secure' flag by calling 'cookie.setSecure(true);'

## java.lang.security.audit.cookie-missing-httponly.cookie-missing-httponly

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/spoofcookie/SpoofCookieAssignment.java  
**Severidade:** WARNING  
**Linha do Código:** 98  
**Descrição da Vulnerabilidade:** A cookie was detected without setting the 'HttpOnly' flag. The 'HttpOnly' flag for cookies instructs the browser to forbid client-side scripts from reading the cookie. Set the 'HttpOnly' flag by calling 'cookie.setHttpOnly(true);'

## java.spring.security.injection.tainted-sql-string.tainted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java  
**Severidade:** ERROR  
**Linha do Código:** 67  
**Descrição da Vulnerabilidade:** User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java  
**Severidade:** ERROR  
**Linha do Código:** 69  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionChallenge.java  
**Severidade:** WARNING  
**Linha do Código:** 69  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/advanced/SqlInjectionLesson6a.java  
**Severidade:** ERROR  
**Linha do Código:** 74  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java  
**Severidade:** ERROR  
**Linha do Código:** 71  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson10.java  
**Severidade:** WARNING  
**Linha do Código:** 71  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson2.java  
**Severidade:** WARNING  
**Linha do Código:** 65  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5a.java  
**Severidade:** ERROR  
**Linha do Código:** 67  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson5b.java  
**Severidade:** ERROR  
**Linha do Código:** 86  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java  
**Severidade:** ERROR  
**Linha do Código:** 78  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java  
**Severidade:** WARNING  
**Linha do Código:** 78  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java  
**Severidade:** ERROR  
**Linha do Código:** 158  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson8.java  
**Severidade:** WARNING  
**Linha do Código:** 158  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.formatted-sql-string.formatted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java  
**Severidade:** ERROR  
**Linha do Código:** 81  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java  
**Severidade:** WARNING  
**Linha do Código:** 81  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.lang.security.audit.sqli.jdbc-sqli.jdbc-sqli

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/introduction/SqlInjectionLesson9.java  
**Severidade:** WARNING  
**Linha do Código:** 111  
**Descrição da Vulnerabilidade:** Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.

## java.spring.security.injection.tainted-sql-string.tainted-sql-string

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/sqlinjection/mitigation/Servers.java  
**Severidade:** ERROR  
**Linha do Código:** 73  
**Descrição da Vulnerabilidade:** User data flows into this manually-constructed SQL string. User data can be safely inserted into SQL strings using prepared statements or an object-relational mapper (ORM). Manually-constructed SQL strings is a possible indicator of SQL injection, which could let an attacker steal or manipulate data from the database. Instead, use prepared statements (`connection.PreparedStatement`) or a safe library.

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/lessons/xxe/SimpleXXE.java  
**Severidade:** WARNING  
**Linha do Código:** 99  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.unrestricted-request-mapping.unrestricted-request-mapping

**Arquivo:** src/main/java/org/owasp/webgoat/webwolf/FileServer.java  
**Severidade:** WARNING  
**Linha do Código:** 74  
**Descrição da Vulnerabilidade:** Detected a method annotated with 'RequestMapping' that does not specify the HTTP method. CSRF protections are not enabled for GET, HEAD, TRACE, or OPTIONS, and by default all HTTP methods are allowed when the HTTP method is not explicitly specified. This means that a method that performs state changes could be vulnerable to CSRF attacks. To mitigate, add the 'method' field and specify the HTTP method (such as 'RequestMethod.POST').

## java.spring.security.injection.tainted-file-path.tainted-file-path

**Arquivo:** src/main/java/org/owasp/webgoat/webwolf/FileServer.java  
**Severidade:** ERROR  
**Linha do Código:** 97  
**Descrição da Vulnerabilidade:** Detected user input controlling a file path. An attacker could control the location of this file, to include going backwards in the directory with '../'. To address this, ensure that user-controlled variables in file paths are sanitized. You may also consider using a utility method such as org.apache.commons.io.FilenameUtils.getName(...) to only retrieve the file name from the path.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/authbypass/html/AuthBypass.html  
**Severidade:** WARNING  
**Linha do Código:** 23  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/authbypass/html/AuthBypass.html  
**Severidade:** WARNING  
**Linha do Código:** 43  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge1.html  
**Severidade:** WARNING  
**Linha do Código:** 18  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge1.html  
**Severidade:** WARNING  
**Linha do Código:** 40  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge5.html  
**Severidade:** WARNING  
**Linha do Código:** 69  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge6.html  
**Severidade:** WARNING  
**Linha do Código:** 102  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge7.html  
**Severidade:** WARNING  
**Linha do Código:** 60  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/challenges/html/Challenge8.html  
**Severidade:** WARNING  
**Linha do Código:** 234  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html  
**Severidade:** WARNING  
**Linha do Código:** 25  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html  
**Severidade:** WARNING  
**Linha do Código:** 46  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/chromedevtools/html/ChromeDevTools.html  
**Severidade:** WARNING  
**Linha do Código:** 67  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cia/html/CIA.html  
**Severidade:** WARNING  
**Linha do Código:** 30  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## html.security.plaintext-http-link.plaintext-http-link

**Arquivo:** src/main/resources/lessons/clientsidefiltering/html/ClientSideFiltering.html  
**Severidade:** WARNING  
**Linha do Código:** 96  
**Descrição da Vulnerabilidade:** This link points to a plaintext HTTP URL. Prefer an encrypted HTTPS URL if possible.

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js  
**Severidade:** ERROR  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/lessons/clientsidefiltering/js/clientSideFiltering.js  
**Severidade:** ERROR  
**Linha do Código:** 38  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cryptography/html/Cryptography.html  
**Severidade:** WARNING  
**Linha do Código:** 31  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cryptography/html/Cryptography.html  
**Severidade:** WARNING  
**Linha do Código:** 48  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cryptography/html/Cryptography.html  
**Severidade:** WARNING  
**Linha do Código:** 65  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cryptography/html/Cryptography.html  
**Severidade:** WARNING  
**Linha do Código:** 90  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/cryptography/html/Cryptography.html  
**Severidade:** WARNING  
**Linha do Código:** 113  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/csrf/html/CSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 16  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/csrf/html/CSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 35  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/csrf/html/CSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 93  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/csrf/html/CSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 213  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/csrf/html/CSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 237  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/deserialization/html/InsecureDeserialization.html  
**Severidade:** WARNING  
**Linha do Código:** 26  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## html.security.audit.missing-integrity.missing-integrity

**Arquivo:** src/main/resources/lessons/hijacksession/html/HijackSession.html  
**Severidade:** WARNING  
**Linha do Código:** 5  
**Descrição da Vulnerabilidade:** This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you’re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/hijacksession/templates/hijackform.html  
**Severidade:** WARNING  
**Linha do Código:** 3  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/httpbasics/html/HttpBasics.html  
**Severidade:** WARNING  
**Linha do Código:** 26  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/httpproxies/html/HttpProxies.html  
**Severidade:** WARNING  
**Linha do Código:** 25  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/idor/html/IDOR.html  
**Severidade:** WARNING  
**Linha do Código:** 23  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/idor/html/IDOR.html  
**Severidade:** WARNING  
**Linha do Código:** 81  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/idor/html/IDOR.html  
**Severidade:** WARNING  
**Linha do Código:** 108  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/insecurelogin/html/InsecureLogin.html  
**Severidade:** WARNING  
**Linha do Código:** 18  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/insecurelogin/html/InsecureLogin.html  
**Severidade:** WARNING  
**Linha do Código:** 26  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_decode.adoc  
**Severidade:** ERROR  
**Linha do Código:** 8  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries.adoc  
**Severidade:** ERROR  
**Linha do Código:** 28  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries.adoc  
**Severidade:** ERROR  
**Linha do Código:** 40  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries.adoc  
**Severidade:** ERROR  
**Linha do Código:** 56  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries\_assignment.adoc  
**Severidade:** ERROR  
**Linha do Código:** 7  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries\_assignment2.adoc  
**Severidade:** ERROR  
**Linha do Código:** 7  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_libraries\_solution.adoc  
**Severidade:** ERROR  
**Linha do Código:** 7  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_signing\_solution.adoc  
**Severidade:** ERROR  
**Linha do Código:** 35  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/documentation/JWT\_signing\_solution.adoc  
**Severidade:** ERROR  
**Linha do Código:** 77  
**Descrição da Vulnerabilidade:** JWT token detected

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/jwt/html/JWT.html  
**Severidade:** WARNING  
**Linha do Código:** 20  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/jwt/html/JWT.html  
**Severidade:** WARNING  
**Linha do Código:** 125  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/jwt/html/JWT.html  
**Severidade:** WARNING  
**Linha do Código:** 158  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/html/JWT.html  
**Severidade:** ERROR  
**Linha do Código:** 322  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/html/JWT.html  
**Severidade:** ERROR  
**Linha do Código:** 388  
**Descrição da Vulnerabilidade:** JWT token detected

## generic.secrets.security.detected-jwt-token.detected-jwt-token

**Arquivo:** src/main/resources/lessons/jwt/images/logs.txt  
**Severidade:** ERROR  
**Linha do Código:** 2  
**Descrição da Vulnerabilidade:** JWT token detected

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/lessontemplate/html/LessonTemplate.html  
**Severidade:** WARNING  
**Linha do Código:** 48  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/logging/html/LogSpoofing.html  
**Severidade:** WARNING  
**Linha do Código:** 17  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/logging/html/LogSpoofing.html  
**Severidade:** WARNING  
**Linha do Código:** 39  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/missingac/html/MissingFunctionAC.html  
**Severidade:** WARNING  
**Linha do Código:** 53  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/missingac/html/MissingFunctionAC.html  
**Severidade:** WARNING  
**Linha do Código:** 76  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/missingac/html/MissingFunctionAC.html  
**Severidade:** WARNING  
**Linha do Código:** 98  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/passwordreset/html/PasswordReset.html  
**Severidade:** WARNING  
**Linha do Código:** 144  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/passwordreset/templates/password\_reset.html  
**Severidade:** WARNING  
**Linha do Código:** 12  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/pathtraversal/html/PathTraversal.html  
**Severidade:** WARNING  
**Linha do Código:** 192  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/securepasswords/html/SecurePasswords.html  
**Severidade:** WARNING  
**Linha do Código:** 21  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## html.security.audit.missing-integrity.missing-integrity

**Arquivo:** src/main/resources/lessons/spoofcookie/html/SpoofCookie.html  
**Severidade:** WARNING  
**Linha do Código:** 5  
**Descrição da Vulnerabilidade:** This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you’re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 16  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 40  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 64  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 88  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 189  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 217  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 245  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjection.html  
**Severidade:** WARNING  
**Linha do Código:** 274  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionAdvanced.html  
**Severidade:** WARNING  
**Linha do Código:** 21  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionAdvanced.html  
**Severidade:** WARNING  
**Linha do Código:** 34  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionAdvanced.html  
**Severidade:** WARNING  
**Linha do Código:** 169  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionMitigations.html  
**Severidade:** WARNING  
**Linha do Código:** 26  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionMitigations.html  
**Severidade:** WARNING  
**Linha do Código:** 45  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionMitigations.html  
**Severidade:** WARNING  
**Linha do Código:** 73  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionMitigations.html  
**Severidade:** WARNING  
**Linha do Código:** 96  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/sqlinjection/html/SqlInjectionMitigations.html  
**Severidade:** WARNING  
**Linha do Código:** 176  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/ssrf/html/SSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 13  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/ssrf/html/SSRF.html  
**Severidade:** WARNING  
**Linha do Código:** 35  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## html.security.audit.missing-integrity.missing-integrity

**Arquivo:** src/main/resources/lessons/vulnerablecomponents/html/VulnerableComponents.html  
**Severidade:** WARNING  
**Linha do Código:** 5  
**Descrição da Vulnerabilidade:** This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you’re telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/vulnerablecomponents/html/VulnerableComponents.html  
**Severidade:** WARNING  
**Linha do Código:** 104  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/webwolfintroduction/html/WebWolfIntroduction.html  
**Severidade:** WARNING  
**Linha do Código:** 19  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/webwolfintroduction/html/WebWolfIntroduction.html  
**Severidade:** WARNING  
**Linha do Código:** 40  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/webwolfintroduction/html/WebWolfIntroduction.html  
**Severidade:** WARNING  
**Linha do Código:** 77  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScripting.html  
**Severidade:** WARNING  
**Linha do Código:** 13  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScripting.html  
**Severidade:** WARNING  
**Linha do Código:** 134  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScripting.html  
**Severidade:** WARNING  
**Linha do Código:** 149  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScripting.html  
**Severidade:** WARNING  
**Linha do Código:** 169  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScriptingMitigation.html  
**Severidade:** WARNING  
**Linha do Código:** 24  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScriptingMitigation.html  
**Severidade:** WARNING  
**Linha do Código:** 44  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/lessons/xss/html/CrossSiteScriptingStored.html  
**Severidade:** WARNING  
**Linha do Código:** 68  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## javascript.express.security.audit.xss.ejs.var-in-href.var-in-href

**Arquivo:** src/main/resources/webgoat/static/js/goatApp/templates/paging\_controls.html  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. If using a relative URL, start with a literal forward slash and concatenate the URL, like this: href='/<%= link %>'. You may also consider setting the Content Security Policy (CSP) header.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/backbone-min.js  
**Severidade:** WARNING  
**Linha do Código:** 1  
**Descrição da Vulnerabilidade:** RegExp() called with a `t` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/mode-java.js  
**Severidade:** WARNING  
**Linha do Código:** 573  
**Descrição da Vulnerabilidade:** RegExp() called with a `commentRegex` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/mode-java.js  
**Severidade:** WARNING  
**Linha do Código:** 576  
**Descrição da Vulnerabilidade:** RegExp() called with a `commentRegex` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.prototype-pollution.prototype-pollution-loop.prototype-pollution-loop

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null) ), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** RegExp() called with a `r` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** RegExp() called with a `r` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.prototype-pollution.prototype-pollution-loop.prototype-pollution-loop

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null) ), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** RegExp() called with a `n` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/js/libs/underscore-min.js  
**Severidade:** WARNING  
**Linha do Código:** 6  
**Descrição da Vulnerabilidade:** RegExp() called with a `r` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/js/quiz.js  
**Severidade:** ERROR  
**Linha do Código:** 34  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** WARNING  
**Linha do Código:** 843  
**Descrição da Vulnerabilidade:** RegExp() called with a `nodeTypes` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 1189  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 1206  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 4011  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** WARNING  
**Linha do Código:** 4079  
**Descrição da Vulnerabilidade:** RegExp() called with a `className` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.lang.security.audit.detect-non-literal-regexp.detect-non-literal-regexp

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** WARNING  
**Linha do Código:** 4088  
**Descrição da Vulnerabilidade:** RegExp() called with a `className` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 4354  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 4391  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 5999  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 6202  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 6695  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 7940  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 7962  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## javascript.browser.security.insecure-document-method.insecure-document-method

**Arquivo:** src/main/resources/webgoat/static/plugins/bootstrap-wysihtml5/js/wysihtml5-0.3.0.js  
**Severidade:** ERROR  
**Linha do Código:** 8056  
**Descrição da Vulnerabilidade:** User controlled data in methods like `innerHTML`, `outerHTML` or `document.write` is an anti-pattern that can lead to XSS vulnerabilities

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/webgoat/templates/login.html  
**Severidade:** WARNING  
**Linha do Código:** 31  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.

## python.django.security.django-no-csrf-token.django-no-csrf-token

**Arquivo:** src/main/resources/webwolf/templates/files.html  
**Severidade:** WARNING  
**Linha do Código:** 32  
**Descrição da Vulnerabilidade:** Manually-created forms in django templates should specify a csrf\_token to prevent CSRF attacks.