Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

TypeScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your TYPESCRIPT code

All rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags | Search by name...

expression that can't match the empty string
Code Smell

Tests should check which exception is thrown
Code Smell

Character classes in regular expressions should not contain the same character twice
Code Smell

Names of regular expressions named groups should be used
Code Smell

Regular expressions should not be too complicated
Code Smell

Optional property declarations should not use both '?' and 'undefined' syntax
Code Smell

Shorthand promises should be used
Code Smell

Template literals should not be nested
Code Smell

"undefined" should not be passed as the value of optional parameters
Code Smell

"in" should not be used on arrays
Code Smell

Assignments should not be redundant
Code Smell

Functions should not have identical implementations
Code Smell

Server-side requests should not be vulnerable to forging attacks

Analyze your code

Vulnerability | Major | injection cwe sans-top25 owasp

User-supplied data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Performing requests from user-controlled data could allow attackers to make arbitrary requests on the internal network or to change their original meaning and thus to retrieve or delete sensitive information.

The problem could be mitigated in any of the following ways:

- Validate the user-provided data, such as the URL and headers, used to construct the request.
- Redesign the application to not send requests based on user-provided data.

Noncompliant Code Example

```
const request = require('request');

function ssrf(req, res) {
  const url = req.query.url;

  request(url, callback); // Noncompliant
}
```

Compliant Solution

Validate the url with an allowlist:

```
const request = require('request');

function ssrf(req, res) {
  const white_list = [ "www.example.com", "example.com" ]

  const url             = (new URL(req.query.url));
  const remote_hostname = url.hostname;

  if (white_list.includes(remote_hostname)) {
    request(url, callback);
  }
}
```

See

- OWASP Top 10 2021 Category A10 - Server-Side Request Forgery (SSRF)
- OWASP Attack Category - Server Side Request Forgery
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-641 - Improper Restriction of Names for Files and Other Resources
- MITRE, CWE-918 - Server-Side Request Forgery (SSRF)
- SANS Top 25 - Risky Resource Management

Sparse arrays should not be declared

Code Smell

Array-mutating methods should not be used misleadingly

Code Smell

Collection and array contents should be used

Code Smell

Literals should not be thrown

Code Smell

Array indexes should be numeric

Available In:

sonarcloud sonarqube Developer Edition