Security

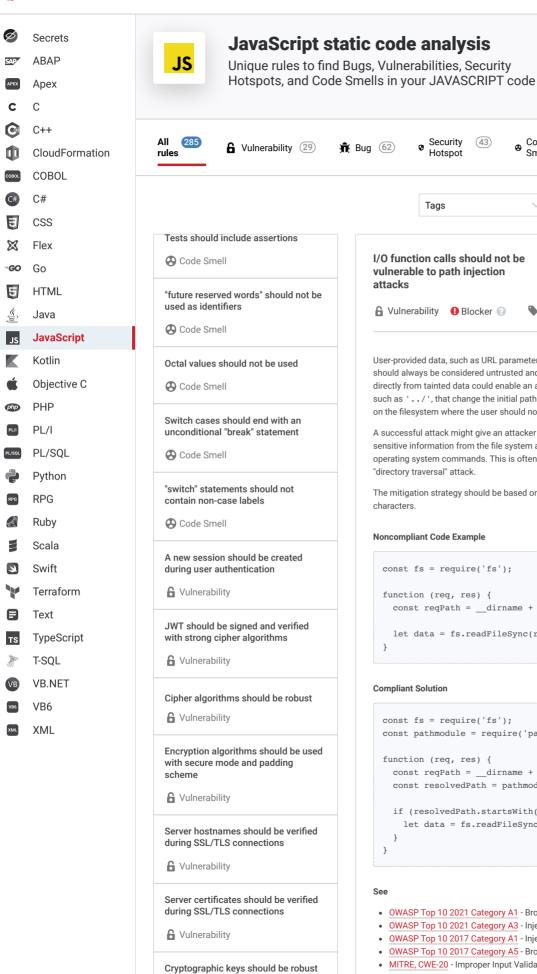
Hotspot

(43)



Products ✓

O Quick 41 Fix



```
Search by name.
            Tags
I/O function calls should not be
vulnerable to path injection
                                                Analyze your code
attacks
Vulnerability
                 Blocker 
                                    injection cwe owasp sans-top25
User-provided data, such as URL parameters, POST data payloads, or cookies,
should always be considered untrusted and tainted. Constructing file system paths
directly from tainted data could enable an attacker to inject specially crafted values,
such as '../', that change the initial path and, when accessed, resolve to a path
on the filesystem where the user should normally not have access.
A successful attack might give an attacker the ability to read, modify, or delete
sensitive information from the file system and sometimes even execute arbitrary
operating system commands. This is often referred to as a "path traversal" or
"directory traversal" attack.
The mitigation strategy should be based on the whitelisting of allowed paths or
Noncompliant Code Example
  const fs = require('fs');
  function (req, res) {
    const reqPath = dirname + req.query.filename; // user-co
    let data = fs.readFileSync(reqPath, { encoding: 'utf8', fl
Compliant Solution
  const fs = require('fs');
  const pathmodule = require('path');
  function (req, res) {
    const reqPath = __dirname + req.query.filename; // user-co
    const resolvedPath = pathmodule.resolve(reqPath); // resol
```

if (resolvedPath.startsWith(\_\_dirname + '/uploads')) { // let data = fs.readFileSync(resolvedPath, { encoding: 'ut

MITRE, CWE-22 - Improper Limitation of a Pathname to a Restricted Directory

MITRE, CWE-99 - Improper Control of Resource Identifiers ('Resource Injection')

OWASP Top 10 2021 Category A1 - Broken Access Control

• OWASP Top 10 2017 Category A5 - Broken Access Control

• OWASP Top 10 2021 Category A3 - Injection

OWASP Top 10 2017 Category A1 - Injection

• MITRE, CWE-20 - Improper Input Validation

('Path Traversal')

⊗ Code

Smell

(151)

Vulnerability

JavaScript static code analysis:
Weak SSL/TLS protocols should not be used
<b>6</b> Vulnerability
Origins should be verified during cross-origin communications
<b>₲</b> Vulnerability
Regular expressions should not be vulnerable to Denial of Service attacks
★ Vulnerability
File uploads should be restricted
<b>6</b> Vulnerability
Function calls should not pass extra

MITRE, CWE-641 - Improper Restriction of Names for Files and Other Resources
 SANS Top 25 - Risky Resource Management
 Available In:
 sonarcloud sonarqube

Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy