Security

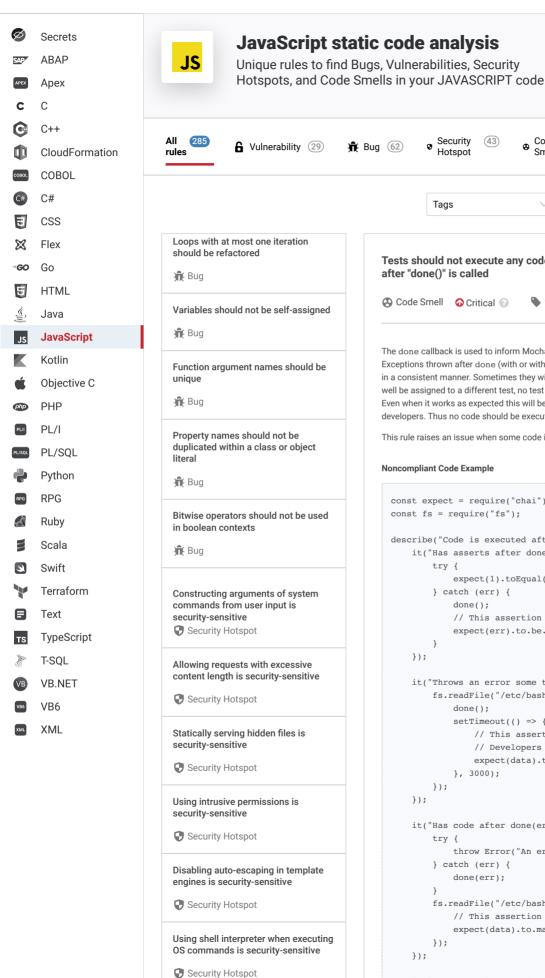
Hotspot

(43)



Products >

O Quick 41 Fix



```
Q
                                       Search by name.
          Tags
Tests should not execute any code
                                            Analyze your code
after "done()" is called
tests unpredictable mocha
The done callback is used to inform Mocha when an asynchronous test ends.
Exceptions thrown after done (with or without parameters) is called are not handled
in a consistent manner. Sometimes they will be correctly handled, but they might as
well be assigned to a different test, no test at all, or even be completely ignored.
Even when it works as expected this will be a source of confusion for other
developers. Thus no code should be executed after done is called
This rule raises an issue when some code is executed after a call to done.
Noncompliant Code Example
 const expect = require("chai").expect;
 const fs = require("fs");
 describe("Code is executed after Done", function() {
      it("Has asserts after done()", function(done) {
          try {
               expect(1).toEqual(2);
          } catch (err) {
              done();
               // This assertion will be ignored and the test w
               expect(err).to.be.an.instanceof(RangeError); //
      });
      it("Throws an error some time after done()", function(do
          fs.readFile("/etc/bashrc", 'utf8', function(err, dat
               done();
               setTimeout(() => { // Noncompliant
                   // This assertion error will not be assigned
                   // Developers will have to guess which test
                   expect(data).to.match(/some expected string/
               }, 3000);
          });
      });
      it("Has code after done(err)", function(done) {
          try {
               throw Error("An error");
          } catch (err) {
               done(err);
          fs.readFile("/etc/bashrc", 'utf8', function(err, dat
               // This assertion error will be assigned to "Oth
               expect(data).to.match(/some expected string/);
          });
      });
      it("Other test", function(done)
```

(151)

Setting loose POSIX file permissions

is security-sensitive Security Hotspot Formatting SQL queries is securitysensitive Security Hotspot

Code Smell

Comma operator should not be used

Regular expressions should not contain empty groups

Code Smell

Regular expressions should not contain multiple spaces

```
});
```

Compliant Solution

```
const expect = require("chai").expect;
const fs = require("fs");
describe("Code is executed after Done", function() {
   it("Has asserts after done()", function(done) {
            expect(1).toEqual(2);
        } catch (err) {
            expect(err).to.be.an.instanceof(RangeError);
    });
    it("Throws an error some time after done()", function(do
        fs.readFile("/etc/bashrc", 'utf8', function(err, dat
            setTimeout(() => {
                expect(data).to.match(/some expected string/
            }, 3000);
        });
   });
    it("Has code after done(err)", function(done) {
            throw Error("An error"):
        } catch (err) {
            return done(err);
        fs.readFile("/etc/bashrc", 'utf8', function(err, dat
           // This assertion error will be assigned to "Oth
            expect(data).to.match(/some expected string/);
            done();
        });
    });
    it("Other test", function(done) {
        done()
    });
});
```

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube

© 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