




 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 rules279

Vulnerability27

Bug51


Security Hotspot43

Code Smell158


Quick Fix50


Tags

Search by name...


 Bug


Identical expressions should not be used on both sides of a binary operator




 Bug


All code should be reachable




 Bug


Loops with at most one iteration should be refactored




 Bug


Variables should not be self-assigned




 Bug

Bitwise operators should not be used in boolean contexts



 Security Hotspot

Constructing arguments of system commands from user input is security-sensitive



 Security Hotspot

Allowing requests with excessive content length is security-sensitive



 Security Hotspot

Statically serving hidden files is security-sensitive



 Security Hotspot

Using intrusive permissions is security-sensitive



 Security Hotspot

Disabling auto-escaping in template engines is security-sensitive




 Security Hotspot


Using shell interpreter when executing OS commands is security-sensitive




### Tests should not execute any code after "done()" is called

Analyze your code

 Code Smell

 Critical

 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 will pass
      expect(err).toBe.an.instanceof(RangeError); //
    }
  });





  it("Throws an error some time after done()", function(done) {
    fs.readFile("/etc/bashrc", "utf8", function(err, data) {
      done();
      setTimeout(() => { // Noncompliant
        // This assertion error will not be assigned
        // Developers will have to guess which test
        expect(data).toMatch(/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, data) {
      // This assertion error will be assigned to "Other test"
      expect(data).toMatch(/some expected string/);
    });
  });

  it("Other test", function(done) {
    done();
  });
});
```

https://rules.sonarsource.com/typescript/RSPEC-6079

1/2

<b>Setting loose POSIX file permissions is security-sensitive</b>
 Security Hotspot
<b>Formatting SQL queries is security-sensitive</b>
 Security Hotspot
<b>Comma operator should not be used</b>
 Code Smell
<b>Regular expressions should not contain empty groups</b>
 Code Smell

```
});
});
```

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) {
    try {
      expect(1).toEqual(2);
    } catch (err) {
      expect(err).toBe.an.instanceof(RangeError);
      done();
    }
  });

  it("Throws an error some time after done()", function(done) {
    fs.readFile("/etc/bashrc", 'utf8', function(err, data) {
      setTimeout(() => {
        expect(data).toMatch(/some expected string/);
        done();
      }, 3000);
    });
  });

  it("Has code after done(err)", function(done) {
    try {
      throw Error("An error");
    } catch (err) {
      return done(err);
    }
    fs.readFile("/etc/bashrc", 'utf8', function(err, data) {
      // This assertion error will be assigned to "Other test"
      expect(data).toMatch(/some expected string/);
      done();
    });
  });

  it("Other test", function(done) {
    done()
  });
});
```

Available In:

 |  | 