

- 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



## C# static code analysis

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

All rules **409**

Vulnerability **34**

Bug **76**

Security Hotspot **28**

Code Smell **271**

Quick Fix **52**

Tags ▾

Search by name... 🔍

Encryption algorithms should be used with secure mode and padding scheme

Vulnerability

Insecure temporary file creation methods should not be used

Vulnerability

Server certificates should be verified during SSL/TLS connections

Vulnerability

LDAP connections should be authenticated

Vulnerability

Cryptographic keys should be robust

Vulnerability

Weak SSL/TLS protocols should not be used

Vulnerability

Cipher Block Chaining IVs should be unpredictable

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

Hashes should include an unpredictable salt

Vulnerability

Non-async "Task/Task<T>" methods should not return null

Bug

Calls to delegate's method "BeginInvoke" should be paired with calls to "EndInvoke"

Bug

"Shared" parts should not be created with "new"

### Destructors should not throw exceptions

Analyze your code

Bug Blocker ?

If Finalize or an override of Finalize throws an exception, and the runtime is not hosted by an application that overrides the default policy, the runtime terminates the process immediately without graceful cleanup (finally blocks and finalizers are not executed). This behavior ensures process integrity if the finalizer cannot free or destroy resources.

The rule reports on throw statements used in finalizers.

#### Noncompliant Code Example





```
class MyClass
{
    ~MyClass()
    {
        throw new NotImplementedException(); // Noncompliant
    }
}
```

#### Compliant Solution

```
class MyClass
{
    ~MyClass()
    {
        // no throw
    }
}
```

Available In:

**sonarlint** | **sonarcloud** | **sonarqube**

<b>with new</b>
 Bug
<b>Getters and setters should access the expected fields</b>  Bug
<b>Right operands of shift operators should be integers</b>  Bug
<b>Shared resources should not be used for locking</b>  Bug
<b>Locks should be released</b>