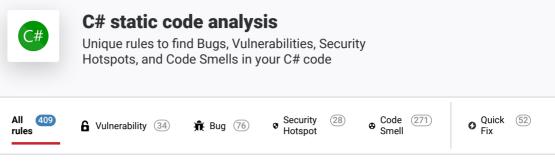
Q

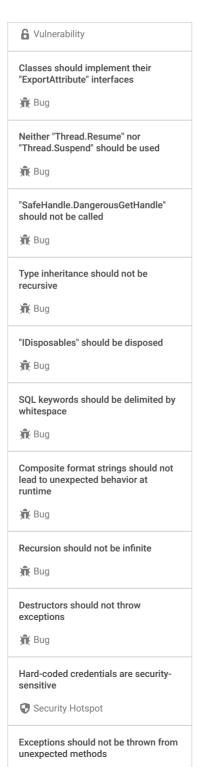






Tags

execute arbitrary operating system commands.



Code Smell



Search by name..

injection cwe owasp sans-top25

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing SQL queries directly from tainted data enables attackers to inject specially crafted values that change the initial meaning of the query itself. Successful database query injection attacks can read, modify, or delete

sensitive information from the database and sometimes even shut it down or

Typically, the solution is to use prepared statements and to bind variables to SQL query parameters with dedicated methods like setParameter, which ensures that user-provided data will be properly escaped. Another solution is to validate every parameter used to build the guery. This can be achieved by transforming string values to primitive types or by validating them against a white list of accepted

## Noncompliant Code Example

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore:
using WebApplication1.Controllers;
namespace WebApplicationDotNetCore.Controllers
{
    public class RSPEC3649SQLiNoncompliant : Controller
        private readonly UserAccountContext _context;
        public RSPEC3649SQLiNoncompliant(UserAccountContext
            context = context;
        public IActionResult Authenticate(string user)
            string query = "SELECT * FROM Users WHERE Userna
            // an attacker can bypass authentication by sett
            // user = "' or 1=1 or ''='";
            var userExists = false:
            if ( context.Database.ExecuteSqlCommand(query) >
            {
                userExists = true;
            return Content(userExists ? "success" : "fail");
   }
}
```

"operator==" should not be overloaded on reference types

Code Smell

Type should not be examined on "System.Type" instances

Code Smell

Test method signatures should be correct

Code Smell

Method overloads with default parameter values should not overlap

Code Smell

## Compliant Solution

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebApplication1.Controllers;
namespace WebApplicationDotNetCore.Controllers
    public class RSPEC3649SQLiCompliant : Controller
        private readonly UserAccountContext _context;
        public RSPEC3649SQLiCompliant(UserAccountContext con
            _context = context;
        public IActionResult Authenticate(string user)
        {
            var query = "SELECT * FROM Users WHERE Username
            var userExists = false;
            if (_context.Database.ExecuteSqlCommand(query,
                userExists = true;
            return Content(userExists ? "success" : "fail");
    }
}
```

## See

- OWASP Top 10 2021 Category A3 Injection
- OWASP Top 10 2017 Category A1 Injection
- MITRE, CWE-20 Improper Input Validation
- MITRE, CWE-89 Improper Neutralization of Special Elements used in an SQL Command
- MITRE, CWE-564 SQL Injection: Hibernate
- <u>MITRE, CWE-943</u> Improper Neutralization of Special Elements in Data Query Logic
- OWASP SQL Injection Prevention Cheat Sheet
- SANS Top 25 Insecure Interaction Between Components

Available In:

sonarcloud 🚷 | sonarqube | Develope 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