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...

HTTP responses should not be vulnerable to session fixation

Analyze your code

Vulnerability   Blocker   injection cwe owasp

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing cookies directly from tainted data enables attackers to set the session identifier to a known value, allowing the attacker to share the session with the victim. Successful attacks might result in unauthorized access to sensitive information, for example if the session identifier is not regenerated when the victim authenticates.

Typically, the solution to prevent this type of attack is to restrict the cookies that can be influenced with an allow-list.

Noncompliant Code Example

```
using System.Web;
using System.Web.Mvc;

[HttpGet]
public ActionResult index(string val)
{
    Response.AddHeader("Set-Cookie", val);  // Noncompliant
    HttpCookie cookie = new HttpCookie("ASP.NET_SessionId",
    Response.AppendCookie(cookie);
    return View("");
}
```

Compliant Solution

```
using System.Web;
using System.Web.Mvc;

[HttpGet]
public ActionResult index(string val)
{
    Response.AddHeader("X-Data", val);
    HttpCookie cookie = new HttpCookie("data", val);
    Response.AppendCookie(cookie);
    return View("");
}
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-384 - Session Fixation

Available In:
sonarcloud   sonarqube Developer Edition

I/O function calls should not be vulnerable to path injection attacks

Vulnerability

LDAP queries should not be vulnerable to injection attacks

Vulnerability

OS commands should not be vulnerable to command injection attacks

Vulnerability

Classes should implement their "ExportAttribute" interfaces

Bug