sonar RULES Products

- 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 rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags | Search by name...

Disabling CSRF protections is security-sensitive
Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive
Security Hotspot

Dynamically executing code is security-sensitive
Security Hotspot

Equality operators should not be used in "for" loop termination conditions
Code Smell

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

Union and intersection types should not be defined with duplicated elements
Code Smell

"default" clauses should be last
Code Smell

"await" should only be used with promises
Code Smell

A conditionally executed single line should be denoted by indentation
Code Smell

Conditionals should start on new lines
Code Smell

Cognitive Complexity of functions should not be too high
Code Smell

Server hostnames should be verified during SSL/TLS connections

Analyze your code

Vulnerability | Critical | cwe privacy owasp ssl

To establish a SSL/TLS connection not vulnerable to man-in-the-middle attacks, it's essential to make sure the server presents the right certificate.

The certificate's hostname-specific data should match the server hostname.

It's not recommended to re-invent the wheel by implementing custom hostname verification.

TLS/SSL libraries provide built-in hostname verification functions that should be used.

Noncompliant Code Example

https built-in module:

```
let options = {
  hostname: 'www.example.com',
  port: 443,
  path: '/',
  method: 'GET',
  secureProtocol: 'TLSv1_2_method',
  checkServerIdentity: function() {} // Noncompliant: hostna
};

let req = https.request(options, (res) => {
  res.on('data', (d) => {
    process.stdout.write(d);
  });
}); // Noncompliant
```

tls built-in module:

```
let options = {
    secureProtocol: 'TLSv1_2_method',
    checkServerIdentity: function() {}  // Noncompliant: hos
};

let socket = tls.connect(443, "www.example.com", options, ()
  process.stdin.pipe(socket);
  process.stdin.resume();
});  // Noncompliant
```

request module:

```
let socket = request.get({
    url: 'https://www.example.com',
    secureProtocol: 'TLSv1_2_method',
    checkServerIdentity: function() {}  // Noncompliant: hos
});
```

"void" should not be used

Code Smell

Loop counters should not be assigned to from within the loop body

Code Smell

"for" loop increment clauses should modify the loops' counters

Code Smell

Functions should not be empty

Code Smell

Server-side requests should not be vulnerable to forging attacks

Compliant Solution

https built-in module:

```
let options = {
  hostname: 'www.example.com',
  port: 443,
  path: '/',
  method: 'GET',
  secureProtocol: 'TLSv1_2_method'
};

let req = https.request(options, (res) => {
  res.on('data', (d) => {
    process.stdout.write(d);
  });
}); // Compliant: default checkServerIdentity function is se
```

tls built-in module:

```
let options = {
    secureProtocol: 'TLSv1_2_method',
    checkServerIdentity: (servername, peer) => {
        if (servername !== "www.example.com") {
            return new Error ('Error');  // Compliant: there
        }
    }
};

let socket = tls.connect(443, "www.example.com", options, ()
  process.stdin.pipe(socket);
  process.stdin.resume();
}); // Compliant
```

request module:

```
let socket = request.get({
    url: 'https://www.example.com/',
    secureProtocol: 'TLSv1_2_method' // Compliant
}); // Compliant:  default checkServerIdentity function is s
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Network Communication Requirements
- OWASP Mobile Top 10 2016 Category M3 - Insecure Communication
- MITRE, CWE-297 - Improper Validation of Certificate with Host Mismatch

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