




 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... 🔍


Hard-coded credentials are security-sensitive

 Security Hotspot


Function returns should not be invariant

 Code Smell


Assertions should be complete

 Code Smell


Tests should include assertions

 Code Smell


Octal values should not be used

 Code Smell


Switch cases should end with an unconditional "break" statement

 Code Smell


"switch" statements should not contain non-case labels

 Code Smell


A new session should be created during user authentication

 Vulnerability


JWT should be signed and verified with strong cipher algorithms

 Vulnerability


Cipher algorithms should be robust

 Vulnerability

Encryption algorithms should be used with secure mode and padding scheme




 Vulnerability

Server hostnames should be verified during SSL/TLS connections

 Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks

Analyze your code

 Vulnerability Blocker injection cwe sans-top25 owasp

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Furthermore, when processing an HTTP request, a web server may copy user-provided data into the body of the HTTP response that is sent back to the user. This behavior is called a "reflection". Endpoints reflecting tainted data could allow attackers to inject code that would eventually be executed in the user's browser. This could enable a wide range of serious attacks like accessing/modifying sensitive information or impersonating other users.

Typically, the solution is one of the following:

- Validate user-provided data based on a whitelist and reject input that is not allowed.
- Sanitize user-provided data from any characters that can be used for malicious purposes.
- Encode user-provided data when it is reflected back in the HTTP response. Adjust the encoding to the output context so that, for example, HTML encoding is used for HTML content, HTML attribute encoding is used for attribute values, and JavaScript encoding is used for server-generated JavaScript.

When sanitizing or encoding data, it is recommended to only use libraries specifically designed for security purposes. Also, make sure that the library you are using is being actively maintained and is kept up-to-date with the latest discovered vulnerabilities.

Noncompliant Code Example

```
function (req, res) {
  const tainted = req.query.name;

  res.send(tainted); // Noncompliant
};
```

Compliant Solution

```
import sanitizeHtml from "sanitize-html";

function (req, res) {
  const tainted = req.query.name;

  res.send(sanitizeHtml(tainted)); // Noncompliant
};
```

See

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Cheat Sheet](#) - XSS Prevention Cheat Sheet
- [OWASP Top 10 2017 Category A7](#) - Cross-Site Scripting (XSS)

https://rules.sonarsource.com/typescript/RSPEC-5131

1/2

Server certificates should be verified during SSL/TLS connections

 Vulnerability


Cryptographic keys should be robust

 Vulnerability

Weak SSL/TLS protocols should not be used

 Vulnerability

Origins should be verified during cross-origin communications

 Vulnerability

- [MITRE, CWE-79](#) - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- [SANS Top 25](#) - Insecure Interaction Between Components

Available In:



© 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](#)