




 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... 🔍


 Bug


Non-existent operators '+=', '--' and '!=' should not be used







"NaN" should not be used in comparisons







A "for" loop update clause should move the counter in the right direction







Return values from functions without side effects should not be ignored







Special identifiers should not be bound or assigned





Values should not be uselessly incremented





Related "if/else if" statements should not have the same condition


Objects should not be created to be dropped immediately without being used


Identical expressions should not be used on both sides of a binary operator


All code should be reachable

Loops with at most one iteration should be refactored

Disabling CSRF protections is security-sensitive

 Security Hotspot

 Critical

 cwe sans-top25 owasp express.js

A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn't intend, such as updating his profile or sending a message, more generally anything that can change the state of the application.

The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive.

Ask Yourself Whether

- The web application uses cookies to authenticate users.
- There exist sensitive operations in the web application that can be performed when the user is authenticated.
- The state / resources of the web application can be modified by doing HTTP POST or HTTP DELETE requests for example.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Protection against CSRF attacks is strongly recommended:
 - to be activated by default for all **unsafe HTTP methods**.
 - implemented, for example, with an unguessable CSRF token
- Of course all sensitive operations should not be performed with **safe HTTP** methods like GET which are designed to be used only for information retrieval.

Sensitive Code Example

Express.js CSURF middleware protection is not found on an unsafe HTTP method like POST method:

```
let csrf = require('csrf');
let express = require('express');

let csrfProtection = csrf({ cookie: true });

let app = express();

// Sensitive: this operation doesn't look like protected by
app.post('/money_transfer', parseForm, function (req, res) {
  res.send('Money transferred');
});
```


Protection provided by **Express.js CSURF middleware** is globally disabled on unsafe methods:

```
let csrf = require('csrf');
let express = require('express');
```


https://rules.sonarsource.com/typescript/RSPEC-4502

1/2


Variables should not be self-assigned

 Bug


Bitwise operators should not be used in boolean contexts

 Bug

Constructing arguments of system commands from user input is security-sensitive

 Security Hotspot

Allowing requests with excessive content length is security-sensitive

 Security Hotspot

```
app.use(csrf({ cookie: true, ignoreMethods: ["POST", "GET"]
```

Compliant Solution

[Express.js CSURF middleware](#) protection is used on unsafe methods:

```
let csrf = require('csrf');
let express = require('express');

let csrfProtection = csrf({ cookie: true });

let app = express();

app.post('/money_transfer', parseForm, csrfProtection, function (req, res) {
  res.send('Money transferred')
});
```

Protection provided by [Express.js CSURF middleware](#) is enabled on unsafe methods:



```
let csrf = require('csrf');
let express = require('express');

app.use(csrf({ cookie: true, ignoreMethods: ["GET"] })); //
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [MITRE, CWE-352](#) - Cross-Site Request Forgery (CSRF)
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [OWASP: Cross-Site Request Forgery](#)
- [SANS Top 25](#) - Insecure Interaction Between Components

Available In:

https://rules.sonarsource.com/typescript/RSPEC-4502

2/2

© 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](#)