




 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... 🔍

 Code Smell


Primitive types should be omitted from initialized or defaulted declarations




Non-null assertions should not be used




"undefined" should not be assigned




Trailing commas should not be used




Array constructors should not be used




Quotes for string literals should be used consistently




Statements should end with semicolons




Comments should not be located at the end of lines of code




Loops should not contain more than a single "break" or "continue" statement






Variable, property and parameter names should comply with a naming convention



Lines should not end with trailing whitespaces



Creating cookies without the "secure" flag is security-sensitive

 Security Hotspot Minor ⓘ

When a cookie is protected with the secure attribute set to `true` it will not be send by the browser over an unencrypted HTTP request and thus cannot be observed by an unauthorized person during a man-in-the-middle attack.

Ask Yourself Whether

- the cookie is for instance a `session-cookie` not designed to be sent over non-HTTPS communication.
- it's not sure that the website contains `mixed content` or not (ie HTTPS everywhere or not)

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

Recommended Secure Coding Practices

- It is recommended to use HTTPs everywhere so setting the secure flag to `true` should be the default behaviour when creating cookies.
- Set the secure flag to `true` for session-cookies.

Sensitive Code Example

`cookie-session` module:

```
let session = cookieSession({
  secure: false, // Sensitive
}); // Sensitive
```

`express-session` module:

```
const express = require('express');
const session = require('express-session');

let app = express();
app.use(session({
  cookie: {
    secure: false // Sensitive
  }
}));
```





`cookies` module:

```
let cookies = new Cookies(req, res, { keys: keys });

cookies.set('LastVisit', new Date().toISOString(), {
  secure: false // Sensitive
}); // Sensitive
```

https://rules.sonarsource.com/typescript/RSPEC-2092

1/2

Files should contain an empty newline at the end
 Code Smell
An open curly brace should be located at the end of a line
 Code Smell
Tabulation characters should not be used
 Code Smell
Function and method names should comply with a naming convention
 Code Smell

csrf module:

```
const cookieParser = require('cookie-parser');
const csrf = require('csrf');
const express = require('express');

let csrfProtection = csrf({ cookie: { secure: false }}); //
```

Compliant Solution

cookie-session module:

```
let session = cookieSession({
  secure: true, // Compliant
}); // Compliant
```

express-session module:

```
const express = require('express');
const session = require('express-session');

let app = express();
app.use(session({
  cookie: {
    {
      secure: true // Compliant
    }
  }
}));
```

cookies module:

```
let cookies = new Cookies(req, res, { keys: keys });

cookies.set('LastVisit', new Date().toISOString(), {
  secure: true // Compliant
}); // Compliant
```

csrf module:

```
const cookieParser = require('cookie-parser');
const csrf = require('csrf');
const express = require('express');

let csrfProtection = csrf({ cookie: { secure: true }}); // C
```

See

- [OWASP Top 10 2021 Category A4](#) - Insecure Design
- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-311](#) - Missing Encryption of Sensitive Data
- [MITRE, CWE-315](#) - Cleartext Storage of Sensitive Information in a Cookie
- [MITRE, CWE-614](#) - Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
- [SANS Top 25](#) - Porous Defenses

Available In:

sonarcloud  | **sonarqube** 