

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

JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

All rules285

Vulnerability29

Bug62

Security Hotspot43

Code Smell151

Quick Fix41

Tags

Search by name...

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Object literal syntax should be used

Code Smell

"undefined" should not be assigned

Code Smell

Trailing commas should not be used

Code Smell

Array constructors should not be used

Code Smell

Quotes for string literals should be used consistently

Code Smell

Statements should end with semicolons

Code Smell

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

Code Smell

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

Code Smell

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

Code Smell

Lines should not end with trailing whitespaces

Code Smell

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

Analyze your code

Security Hotspot

Minor

cwe sans-top25 privacy owasp express.js

When a cookie is configured with the `HttpOnly` attribute set to `true`, the browser guaranties that no client-side script will be able to read it. In most cases, when a cookie is created, the default value of `HttpOnly` is `false` and it's up to the developer to decide whether or not the content of the cookie can be read by the client-side script. As a majority of Cross-Site Scripting (XSS) attacks target the theft of session-cookies, the `HttpOnly` attribute can help to reduce their impact as it won't be possible to exploit the XSS vulnerability to steal session-cookies.

Ask Yourself Whether

the cookie is sensitive, used to authenticate the user, for instance a *session-cookie*

the `HttpOnly` attribute offer an additional protection (not the case for an *XSRF-TOKEN* cookie / CSRF token for example)

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

Recommended Secure Coding Practices

By default the `HttpOnly` flag should be set to `true` for most of the cookies and it's mandatory for session / sensitive-security cookies.

Sensitive Code Example

cookie-session module:

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

express-session module:

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

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





cookies module:

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

cookies.set('LastVisit', new Date().toISOString(), {
```

https://rules.sonarsource.com/javascript/RSPEC-3330

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 |

```
    httpOnly: false // Sensitive
  }); // Sensitive
```

csrf module:

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

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

Compliant Solution

cookie-session module:

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

express-session module:

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

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

cookies module:

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

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

csrf module:

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

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

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP HttpOnly](#)
- [OWASP Top 10 2017 Category A7](#) - Cross-Site Scripting (XSS)
- [MITRE, CWE-1004](#) - Sensitive Cookie Without 'HttpOnly' Flag
- [SANS Top 25](#) - Insecure Interaction Between Components
- Derived from FindSecBugs rule [HTTPONLY_COOKIE](#)

Available In:

