




 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... 🔍

 Code Smell


Wrapper objects should not be used for primitive types




Multiline string literals should not be used




Local variables should not be declared and then immediately returned or thrown




Unused local variables and functions should be removed




Function call arguments should not start on new lines




"switch" statements should have at least 3 "case" clauses




A "while" loop should be used instead of a "for" loop




Unnecessary imports should be removed




Return of boolean expressions should not be wrapped into an "if-then-else" statement



Boolean literals should not be used in comparisons



Extra semicolons should be removed



Disabling auto-escaping in template engines is security-sensitive

Security Hotspot

Major

cwe owasp

To reduce the risk of cross-site scripting attacks, templating systems, such as Twig, Django, Smarty, Groovy's template engine, allow configuration of automatic variable escaping before rendering templates. When escape occurs, characters that make sense to the browser (eg: <a>) will be transformed/replaced with escaped/sanitized values (eg: <a>).

Auto-escaping is not a magic feature to annihilate all cross-site scripting attacks, it depends on **the strategy applied** and the context, for example a "html auto-escaping" strategy (which only transforms html characters into **html entities**) will not be relevant when variables are used in a **html attribute** because ':' character is not escaped and thus an attack as below is possible:

```
<a href="{ { myLink } }">link</a> // myLink = javascript:alert
<a href="javascript:alert(document.cookie)">link</a> // JS i
```

Ask Yourself Whether

- Templates are used to render web content and
 - dynamic variables in templates come from untrusted locations or are user-controlled inputs
 - there is no local mechanism in place to sanitize or validate the inputs.

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

Recommended Secure Coding Practices

Enable auto-escaping by default and continue to review the use of inputs in order to be sure that the chosen auto-escaping strategy is the right one.

Sensitive Code Example

mustache.js template engine:

```
let Mustache = require("mustache");

Mustache.escape = function(text) {return text;}; // Sensitive

let rendered = Mustache.render(template, { name: inputName })
```

handlebars.js template engine:

```
const Handlebars = require('handlebars');

let source = "<p>attack {{name}}</p>";

let template = Handlebars.compile(source, { noEscape: true })
```

markdown-it markup language parser:

https://rules.sonarsource.com/javascript/RSPEC-5247

1/3

Class names should comply with a naming convention

 Code Smell

Track uses of "TODO" tags

 Code Smell

Web SQL databases should not be used

 Vulnerability

Variables should be defined before being used

 Bug

```
const markdownIt = require('markdown-it');
let md = markdownIt({
  html: true // Sensitive
});

let result = md.render('# <b>attack</b>');
```

marked markup language parser:

```
const marked = require('marked');

marked.setOptions({
  renderer: new marked.Renderer(),
  sanitize: false // Sensitive
});

console.log(marked("# test <b>attack</b>"));
```

kramed markup language parser:

```
let kramed = require('kramed');

var options = {
  renderer: new kramed.Renderer({
    sanitize: false // Sensitive
  })
};
```

Compliant Solution

mustache.js template engine:

```
let Mustache = require("mustache");

let rendered = Mustache.render(template, { name: inputName })
```

handlebars.js template engine:

```
const Handlebars = require('handlebars');

let source = "<p>attack {{name}}</p>";
let data = { "name": "<b>Alan</b>" };

let template = Handlebars.compile(source); // Compliant by default
```

markdown-it markup language parser:

```
let md = require('markdown-it')(); // Compliant by default

let result = md.render('# <b>attack</b>');
```

marked markup language parser:

```
const marked = require('marked');

marked.setOptions({
  renderer: new marked.Renderer()
}); // Compliant by default sanitize is set to true

console.log(marked("# test <b>attack</b>"));
```

kramed markup language parser:

```
let kramed = require('kramed');

let options = {
  renderer: new kramed.Renderer({
    sanitize: true // Compliant
  })
};

console.log(kramed('Attack [xss?](javascript:alert("xss")).')
```

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)
- [MITRE, CWE-79](#) - Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)

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](#)