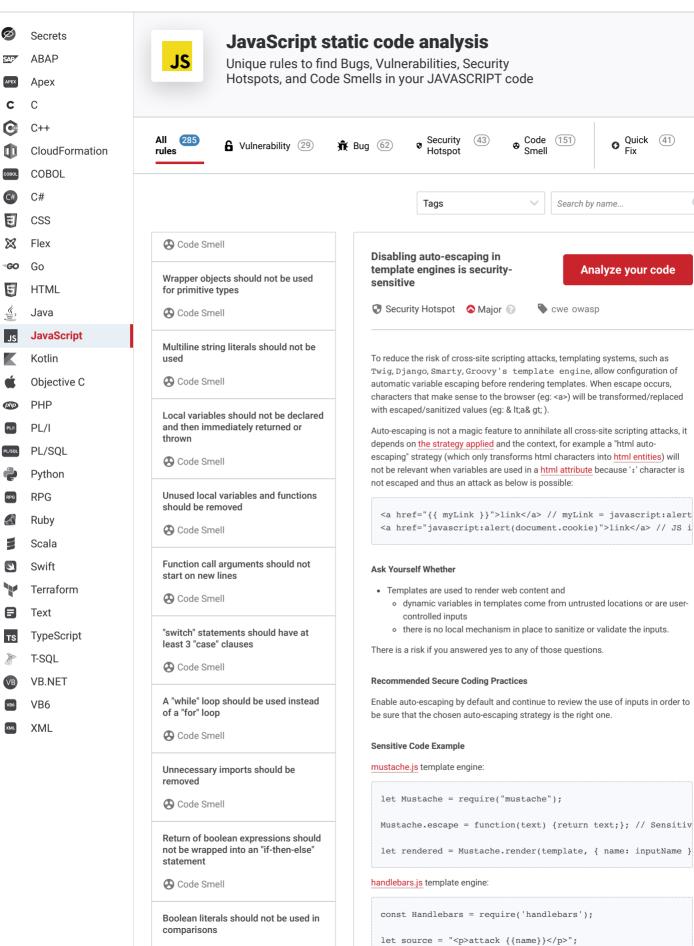


Products ✓



Code Smell

Code Smell

Extra semicolons should be removed

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

markdown-it markup language parser:

Class names should comply with a naming convention

Code Smell

Track uses of "TODO" tags

Code Smell

Web SQL databases should not be

6 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 = "attack {{name}}";
let data = { "name": "<b>Alan</b>" };
let template = Handlebars.compile(source); // Compliant by d
```

 $\underline{\text{markdown-it}} \text{ markup language parser:}$

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

sonarcloud 🙆 | sonarqube

© 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