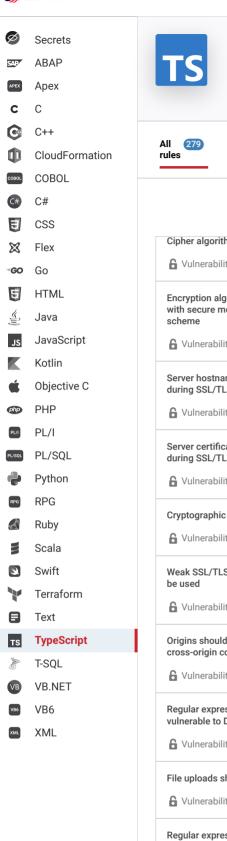
TypeScript static code analysis

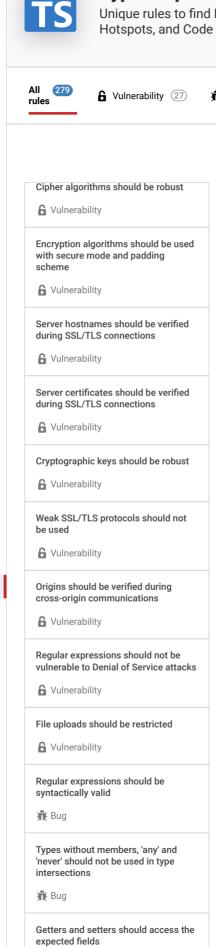


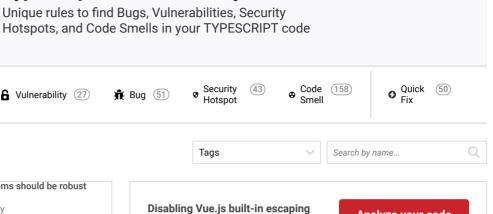
Products ✓

Analyze your code

cwe owasp







Vue.js framework prevents XSS vulnerabilities by automatically escaping HTML contents with the use of native API browsers like innerText, instead of

It's still possible to explicity use innerHtml and similar APIs to render HTML. Accidentally rendering malicious HTML data will introduce an XSS vulnerability in the application and enable a wide range of serious attacks like accessing/modifying sensitive information or impersonating other users.

Ask Yourself Whether

innerHtml.

is security-sensitive

The application needs to render HTML content which:

- could be user-controlled and not previously sanitized.
- is difficult to understand how it was constructed.

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

Recommended Secure Coding Practices

- Avoid injecting HTML content with v-html directive unless the content can be considered 100% safe, instead try to rely as much as possible on built-in autoescaping Vue is features.
- Take care when using the v-bind:href directive to set URLs which can contain malicious Javascript (javascript:onClick(...)).
- Event directives like: onmouseover are also prone to Javascript injection and should not be used with unsafe values.

Sensitive Code Example

When using Vue.js templates, the v-html directive enables HTML rendering without any sanitization:

```
<div v-html="htmlContent"></div> <!-- Noncompliant -->
```

When using a rendering function, the innerHTML attribute enables HTML rendering without any sanitization:

```
Vue.component('element', {
  render: function (createElement) {
    return createElement(
        'div',
        {
            domProps: {
               innerHTML: this.htmlContent, // Noncompliant
        }
        }
    );
},
});
```

Bug

"super()" should be invoked appropriately

📆 Bug

Results of "in" and "instanceof" should be negated rather than operands

👬 Bug

A compare function should be provided when using "Array.prototype.sort()"

👬 Bug

Jump statements should not occur in "finally" blocks

When using JSX, the domPropsInnerHTML attribute enables HTML rendering without any sanitization:

<div domPropsInnerHTML={this.htmlContent}></div> <!-- Noncom</pre>

Compliant Solution

When using Vue.js templates, putting the content as a child node of the element is safe:

```
<div>{{ htmlContent }}</div>
```

When using a rendering function, using the innerText attribute or putting the content as a child node of the element is safe:

```
Vue.component('element', {
  render: function (createElement) {
    return createElement(
        'div',
        {
            domProps: {
               innerText: this.htmlContent,
            }
        },
        this.htmlContent // Child node
    );
    },
});
```

When using JSX, putting the content as a child node of the element is safe:

```
<div>{this.htmlContent}</div>
```

See

- OWASP Top 10 2021 Category A3 Injection
- OWASP Top 10 2017 Category A7 Cross-Site Scripting (XSS)
- <u>MITRE, CWE-79</u> Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- Vue.js Security Injecting HTML

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