JavaScript static code analysis

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

All rules 285 | Vulnerability 29 | Bug 62 | Security Hotspot 43 | Code Smell 151 | Quick Fix 41

Tags | Search by name...

Code Smell

"for" loop increment clauses should modify the loops' counters
Code Smell

Functions should not be empty
Code Smell

Server-side requests should not be vulnerable to forging attacks
Vulnerability

Non-empty statements should change control flow or have at least one side-effect
Bug

Regular expressions with the global flag should be used with caution
Bug

Replacement strings should reference existing regular expression groups
Bug

Regular expressions should not contain control characters
Bug

Alternation in regular expressions should not contain empty alternatives
Bug

Mocha timeout should be disabled by setting it to "0".
Bug

Unicode Grapheme Clusters should be avoided inside regex character classes
Bug

Assertions should not be given twice the same argument
Bug

File uploads should be restricted

Analyze your code

Vulnerability Critical cwe sans-top25 owasp express.js

These minimum restrictions should be applied when handling file uploads:

- the file upload folder to restrict untrusted files to a specific folder.
- the file extension of the uploaded file to prevent remote code execution.

Also the size of the uploaded file should be limited to prevent denial of service attacks. This requirement is covered by the rule {rule:javascript:S5693}.

Noncompliant Code Example

formidable module:

```
const Formidable = require('formidable');

const form = new Formidable(); // Noncompliant, this form is
form.uploadDir = ""; // because upload dir is not defined (b
form.keepExtensions = true; // and file extensions are kept
```

multer (Express.js middleware) module:

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

let diskStorage = multer.diskStorage({ // Noncompliant: no d
  filename: (req, file, cb) => {
    const buf = crypto.randomBytes(20);
    cb(null, buf.toString('hex'))
  }
});

// This upload is not safe as no destination specified, /var
let diskupload = multer({
  storage: diskStorage,
});
```

Compliant Solution

formidable module:

```
const Formidable = require('formidable');

const form = new Formidable(); // Compliant
form.uploadDir = "./uploads/";
form.keepExtensions = false;
```

multer (Express.js middleware) module:

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

let diskStorage = multer.diskStorage({  // Compliant
```

Alternatives in regular expressions should be grouped when used with anchors

Bug

Promise rejections should not be caught by 'try' block

Bug

Collection elements should not be replaced unconditionally

Bug

Errors should not be created without being thrown

```
  filename: (req, file, cb) => {
    const buf = crypto.randomBytes(20);
    cb(null, buf.toString('hex'))
  },
  destination: (req, file, cb) => {
    cb(null, './uploads/')
  }
});

let diskupload = multer({
  storage: diskStorage,
});
```

See

- OWASP Top 10 2021 Category A4 - Insecure Design
- MITRE, CWE-434 - Unrestricted Upload of File with Dangerous Type
- MITRE, CWE-400 - Uncontrolled Resource Consumption
- OWASP Unrestricted File Upload - Unrestricted File Upload
- SANS Top 25 - Insecure Interaction Between Components

Available In:

sonarlint | sonarcloud | sonarqube