JavaScript static code analysis

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

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

Using Sockets is security-sensitive

Analyze your code

Security Hotspot   Critical

Using sockets is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2011-178
- CVE-2017-5645
- CVE-2018-6597

Sockets are vulnerable in multiple ways:

- They enable a software to interact with the outside world. As this world is full of attackers it is necessary to check that they cannot receive sensitive information or inject dangerous input.
- The number of sockets is limited and can be exhausted. Which makes the application unresponsive to users who need additional sockets.

This rules flags code that creates sockets. It matches only the direct use of sockets, not use through frameworks or high-level APIs such as the use of http connections.

Ask Yourself Whether

- sockets are created without any limit every time a user performs an action.
- input received from sockets is used without being sanitized.
- sensitive data is sent via sockets without being encrypted.

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

Recommended Secure Coding Practices

- In many cases there is no need to open a socket yourself. Use instead libraries and existing protocols.
- Encrypt all data sent if it is sensitive. Usually it is better to encrypt it even if the data is not sensitive as it might change later.
- Sanitize any input read from the socket.
- Limit the number of sockets a given user can create. Close the sockets as soon as possible.

Sensitive Code Example

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

var socket = new net.Socket(); // Sensitive
socket.connect(80, 'google.com');

// net.createConnection creates a new net.Socket, initiates
net.createConnection({ port: port }, () => {}); // Sensitive

// net.connect is an alias to net.createConnection
net.connect({ port: port }, () => {}); // Sensitive
```

See

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure

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

- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- MITRE, CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor
- SANS Top 25 - Risky Resource Management
- SANS Top 25 - Porous Defenses

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud | sonarqube