

sonar


RULES


Products


▼


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


Tests should not execute any code after "done()" is called





"default" clauses should be last





"await" should only be used with promises





A conditionally executed single line should be denoted by indentation





Conditionals should start on new lines





Cognitive Complexity of functions should not be too high





"void" should not be used





Loop counters should not be assigned to from within the loop body





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




Functions should not be empty



Server-side requests should not be vulnerable to forging attacks



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

Server hostnames should be verified during SSL/TLS connections

Analyze your code

Vulnerability

Critical

cwe privacy owasp ssl

To establish a SSL/TLS connection not vulnerable to man-in-the-middle attacks, it's essential to make sure the server presents the right certificate.

The certificate's hostname-specific data should match the server hostname.

It's not recommended to re-invent the wheel by implementing custom hostname verification.

TLS/SSL libraries provide built-in hostname verification functions that should be used.

Noncompliant Code Example

https built-in module:

```
let options = {
  hostname: 'www.example.com',
  port: 443,
  path: '/',
  method: 'GET',
  secureProtocol: 'TLSv1_2_method',
  checkServerIdentity: function() {} // Noncompliant: hostna
};

let req = https.request(options, (res) => {
  res.on('data', (d) => {
    process.stdout.write(d);
  });
}); // Noncompliant
```

tls built-in module:

```
let options = {
  secureProtocol: 'TLSv1_2_method',
  checkServerIdentity: function() {} // Noncompliant: hos
};





let socket = tls.connect(443, "www.example.com", options, ()
  process.stdin.pipe(socket);
  process.stdin.resume();
}); // Noncompliant
```

request module:

```
let socket = request.get({
  url: 'https://www.example.com',
  secureProtocol: 'TLSv1_2_method',
  checkServerIdentity: function() {} // Noncompliant: hos
});
```

https://rules.sonarsource.com/javascript/RSPEC-5527

1/2

 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

Compliant Solution

[https](#) built-in module:

```
let options = {
  hostname: 'www.example.com',
  port: 443,
  path: '/',
  method: 'GET',
  secureProtocol: 'TLSv1_2_method'
};

let req = https.request(options, (res) => {
  res.on('data', (d) => {
    process.stdout.write(d);
  });
}); // Compliant: default checkServerIdentity function is se
```

[tls](#) built-in module:

```
let options = {
  secureProtocol: 'TLSv1_2_method',
  checkServerIdentity: (servername, peer) => {
    if (servername !== "www.example.com") {
      return new Error ('Error'); // Compliant: there s
    }
  }
};

let socket = tls.connect(443, "www.example.com", options, ()
process.stdin.pipe(socket);
process.stdin.resume();
}); // Compliant
```

[request](#) module:

```
let socket = request.get({
  url: 'https://www.example.com/',
  secureProtocol: 'TLSv1_2_method' // Compliant
}); // Compliant: default checkServerIdentity function is s
```

See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP Top 10 2021 Category A7](#) - Identification and Authentication Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [OWASP Top 10 2017 Category A6](#) - Security Misconfiguration
- [Mobile AppSec Verification Standard](#) - Network Communication Requirements
- [OWASP Mobile Top 10 2016 Category M3](#) - Insecure Communication
- [MITRE, CWE-297](#) - Improper Validation of Certificate with Host Mismatch

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 