




 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



## TypeScript static code analysis

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

All rules279

Vulnerability27

Bug51


Security Hotspot43

Code Smell158


Quick Fix50


Tags


Search by name...


 Code Smell


Union and intersection types should not be defined with duplicated elements





 "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


### Server certificates should be verified during SSL/TLS connections

Analyze your code

 Vulnerability

 Critical



 cwe

privacy

owasp

ssl

Validation of X.509 certificates is essential to create secure SSL/TLS sessions not vulnerable to man-in-the-middle attacks.

The certificate chain validation includes these steps:

- The certificate is issued by its parent Certificate Authority or the root CA trusted by the system.
- Each CA is allowed to issue certificates.
- Each certificate in the chain is not expired.

It's not recommended to reinvent the wheel by implementing custom certificate chain validation.

TLS libraries provide built-in certificate validation functions that should be used.

#### Noncompliant Code Example

There is no way to disable certificate verification in tls, https and request modules but it is possible to not reject request when verification fails.

[https](#) built-in module:

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

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

[tls](#) built-in module:





```
let options = {
  secureProtocol: 'TLStls_method',
  rejectUnauthorized: false ; // Noncompliant
};

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

[request](#) module:

https://rules.sonarsource.com/typescript/RSPEC-4830

1/2

|   |
|---|
| 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 |

```
let socket = request.get({
  url: 'www.example.com',
  secureProtocol: 'TLSv1_2_method',
  rejectUnauthorized: false ; // Noncompliant
});
```

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: by default rejectUnauthorized is set to t
```

[tls](#) built-in module:

```
let options = {
  secureProtocol: 'TLSv1_2_method'
};

let socket = tls.connect(443, "www.example.com", options, ()
  process.stdin.pipe(socket);
  process.stdin.resume();
}); // Compliant: by default rejectUnauthorized is set to t
```

[request](#) module:

```
let socket = request.get({
  url: 'https://www.example.com/',
  secureProtocol: 'TLSv1_2_method' // Compliant
}); // Compliant: by default rejectUnauthorized is set to t
```

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-295](#) - Improper Certificate Validation

Available In:

sonarlint

|

sonarcloud

|

sonarqube