Security

Hotspot

Tags

Server certificates should be

process.stdin.resume();

}); // Noncompliant

request module:

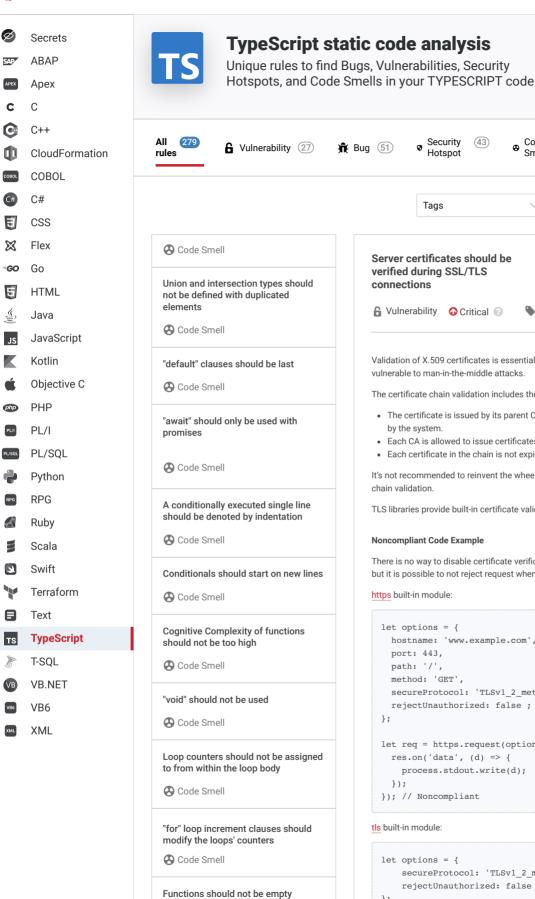
(43)

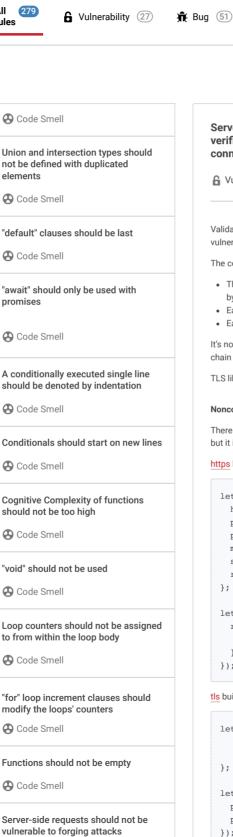


Products >

O Quick 50 Fix

Q





```
verified during SSL/TLS
                                               Analyze your code
connections
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
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: 'TLSv1_2_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: 'TLSv1_2_method',
      rejectUnauthorized: false ; // Noncompliant
 let socket = tls.connect(443, "www.example.com", options, ()
    process.stdin.pipe(socket):
```

(158)

Search by name.

Code Smell

Vulnerability

Mon-amntu etatamente chould change

TypeScript static code analysis: Server certificates should be verified during SSL/TLS connections

INOIL-EITIPLY STATELLIS SHOULD CHAINGE control flow or have at least one sideeffect



Regular expressions with the global flag should be used with caution



Replacement strings should reference existing regular expression groups



Regular expressions should not contain control characters



```
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
```

- 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



© 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