TypeScript static code analysis

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

Disabling Vue.js built-in escaping is security-sensitive
Security Hotspot

Disabling Angular built-in sanitization is security-sensitive
Security Hotspot

Hard-coded credentials are security-sensitive
Security Hotspot

Function returns should not be invariant
Code Smell

Assertions should be complete
Code Smell

Tests should include assertions
Code Smell

Octal values should not be used
Code Smell

Switch cases should end with an unconditional "break" statement
Code Smell

"switch" statements should not contain non-case labels
Code Smell

A new session should be created during user authentication
Vulnerability

JWT should be signed and verified with strong cipher algorithms
Vulnerability

Cipher algorithms should be robust
Vulnerability

HTTP request redirections should not be open to forging attacks

Analyze your code

Vulnerability Blocker injection cwe sans-top25 owasp

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Applications performing HTTP redirects based on tainted data could enable an attacker to redirect users to a malicious site to, for example, steal login credentials.

This problem could be mitigated in any of the following ways:

- Validate the user-provided data based on an allowlist and reject input not matching.
- Redesign the application to not perform redirects based on user-provided data.

Noncompliant Code Example

```
function redirect(req, res) {
  const url = req.query.url; // user-controlled input

  res.redirect(url); // Noncompliant
}

function setLocationHeader(req, res) {
  const url = req.query.url; // user-controlled input

  res.location(url); // Noncompliant
  res.sendStatus(302);
}
```

Compliant Solution

Validate the URL with an allowlist:

```
function isValidUrl(url) {
  if(url.startsWith("https://www.safe.com/")) {
    return true;
  }

  return false;
}

function redirect(req, res) {
  const url = req.query.url; // user-controlled input

  if(isValidUrl(url)) {
    res.redirect(url); // Compliant
  }
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control

Encryption algorithms should be used with secure mode and padding scheme

🔓 Vulnerability

Server hostnames should be verified during SSL/TLS connections

🔓 Vulnerability

Server certificates should be verified during SSL/TLS connections

🔓 Vulnerability

Cryptographic keys should be robust

🔓 Vulnerability

- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-601 - URL Redirection to Untrusted Site ('Open Redirect')
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud sonarqube Developer Edition