

PL/SQL

Python **RPG**

Ruby

Scala

Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

XML

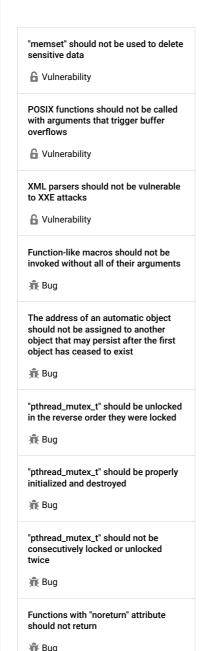


C static code analysis

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

⊗ Code (206) O Quick 14 ΑII 311 Security 18 6 Vulnerability (13) ₩ Bug (74) rules Hotspot Smell

Tags



"memcmp" should only be called with pointers to trivially copyable types

with no padding

🖷 Bug



Search by name.

Setting capabilities can lead to privilege escalation.

Linux capabilities allow you to assign narrow slices of root's permissions to files or processes. A thread with capabilities bypasses the normal kernel security checks to execute high-privilege actions such as mounting a device to a directory, without requiring (additional) root privileges.

Ask Yourself Whether

Capabilities are granted:

- To a process that does not require all capabilities to do its job.
- . To a not trusted process

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

Recommended Secure Coding Practices

Capabilities are high privileges, traditionally associated with superuser (root), thus make sure that the most restrictive and necessary capabilities are assigned to files and processes.

Sensitive Code Example

When setting capabilities:

```
cap_t caps = cap_init();
cap_value_t cap_list[2];
cap_list[0] = CAP_FOWNER;
cap_list[1] = CAP_CHOWN;
cap set flag(caps, CAP PERMITTED, 2, cap list, CAP SET);
cap_set_file("file", caps); // Sensitive
cap_set_fd(fd, caps); // Sensitive
cap_set_proc(caps); // Sensitive
capsetp(pid, caps); // Sensitive
capset(hdrp, datap); // Sensitive: is discouraged to be used
```

When setting SUID/SGID attributes:

```
chmod("file", S_ISUID|S_ISGID); // Sensitive
fchmod(fd, S_ISUID|S_ISGID); // Sensitive
```

- OWASP Top 10 2021 Category A1 Broken Access Control
- OWASP Top 10 2017 Category A5 Broken Access Control
- MITRE, CWE-250 Execution with Unnecessary Privileges
- MITRE, CWE-266 Incorrect Privilege Assignment
- False Boundaries and Arbitrary Code Execution
- Linux manual page capabilities(7)

Stack allocated memory and nonowned memory should not be freed

Bug

Closed resources should not be
accessed

Bug

Dynamically allocated memory should
be released

Bug

Freed memory should not be used

Available In:

sonarcloud sonarqube Developer Edition

© 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