Code Smell (101)

Analyze your code

cwe owasp

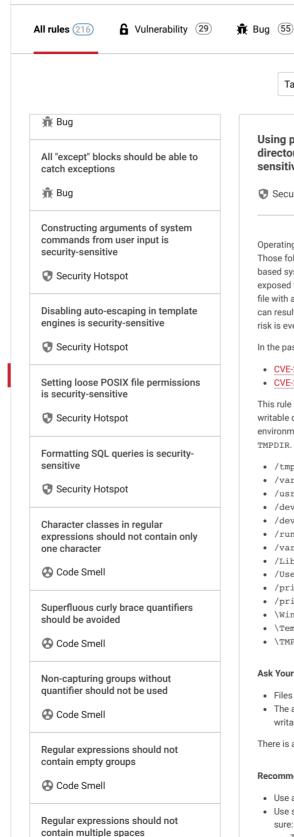






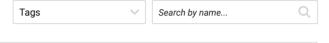
Python static code analysis

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



Code Smell

Single-character alternations in



Security Hotspot 31

Security Hotspot Oritical Operating systems have global directories where any user has write access. Those folders are mostly used as temporary storage areas like /tmp in Linux based systems. An application manipulating files from these folders is exposed to race conditions on filenames: a malicious user can try to create a file with a predictable name before the application does. A successful attack

can result in other files being accessed, modified, corrupted or deleted. This

risk is even higher if the application runs with elevated permissions.

In the past, it has led to the following vulnerabilities:

CVE-2012-2451

Using publicly writable directories is security-

sensitive

CVE-2015-1838

This rule raises an issue whenever it detects a hard-coded path to a publicly writable directory like /tmp (see examples bellow). It also detects access to environment variables that point to publicly writable directories, e.g., TMP and TMPDTR

- /tmp
- /var/tmp
- /usr/tmp
- /dev/shm
- /dev/mqueue
- /run/lock
- /var/run/lock
- /Library/Caches
- /Users/Shared • /private/tmp
- /private/var/tmp
- \Windows\Temp
- \Temp
- \TMP

Ask Yourself Whether

- · Files are read from or written into a publicly writable folder
- The application creates files with predictable names into a publicly writable folder

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

Recommended Secure Coding Practices

- Use a dedicated sub-folder with tightly controlled permissions
- Use secure-by-design APIs to create temporary files. Such API will make
 - o The generated filename is unpredictable
 - o The file is readable and writable only by the creating user ID
 - o The file descriptor is not inherited by child processes
 - o The file will be destroyed as soon as it is closed

regular expressions should be replaced with character classes

Code Smell

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

Code Smell

Values assigned to variables should match their type annotations

Code Smell

Function return types should be consistent with their type hint

Code Smell

Ob-----

Sensitive Code Example

```
file = open("/tmp/temporary_file","w+") # Sensitive
```

```
tmp_dir = os.environ.get('TMPDIR') # Sensitive
file = open(tmp_dir+"/temporary_file","w+")
```

Compliant Solution

```
import tempfile
file = tempfile.TemporaryFile(dir="/tmp/my_subdirectory
```

See

- OWASP Top 10 2021 Category A1 Broken Access Control
- OWASP Top 10 2017 Category A5 Broken Access Control
- OWASP Top 10 2017 Category A3 Sensitive Data Exposure
- MITRE, CWE-377 Insecure Temporary File
- MITRE, CWE-379 Creation of Temporary File in Directory with Incorrect Permissions
- OWASP, Insecure Temporary File
- Python tempfile module

Available In:

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