Python static code analysis

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

Unread "private" attributes should be removed

Analyze your code

Code Smell Critical cwe unused

Private attributes which are written but never read are a clear case of dead store. Changing their value is useless and most probably indicates a serious error in the code.

Python has no real private attribute. Every attribute is accessible. There are however two conventions indicating that an attribute is not meant to be "public":

- attributes with a name starting with a single underscore (ex: `_myattribute`) should be seen as non-public and might change without prior notice. They should not be used by third-party libraries or software. It is ok to use those methods inside the library defining them but it should be done with caution.
- "class-private" attributes have a name which starts with at least two underscores and ends with at most one underscore. These attribute's names will be automatically mangled to avoid collision with subclasses' attributes. For example `__myattribute` will be renamed as `_classname__myattribute`, where `classname` is the attribute's class name without its leading underscore(s). They shouldn't be used outside of the class defining the attribute.

This rule raises an issue when a class-private attribute (two leading underscores, max one underscore at the end) is never read inside the class. It optionally raises an issue on unread attributes prefixed with a single underscore. Both class attribute and instance attributes will raise an issue.

Noncompliant Code Example

```
class Noncompliant:
    _class_attr = 0  # Noncompliant if enable_single_undersc
    __mangled_class_attr = 1  # Noncompliant

    def __init__(self, value):
        self._attr = 0  # Noncompliant if enable_single_unde
        self.__mangled_attr = 1  # Noncompliant

    def compute(self, x):
        return x * x
```

Compliant Solution

```
class Compliant:
    _class_attr = 0
    __mangled_class_attr = 1

    def __init__(self, value):
        self._attr = 0
        self.__mangled_attr = 1

    def compute(self, x):
        return x * Compliant._class_attr * Compliant.__mangl
```

Collapsible "if" statements should be merged

Code Smell

Logging should not be vulnerable to injection attacks

Vulnerability

Repeated patterns in regular expressions should not match the empty string

Bug

Function parameters initial values should not be ignored

See

- Python documentation – Private Variables
- PEP 8 – Style Guide for Python Code

Available In:

sonarlint | sonarcloud | sonarqube