Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

Python static code analysis

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

All rules (216) Vulnerability (29) Bug (55) Security Hotspot (31) Code Smell (101)

Tags

Search by name...

The number and name of arguments passed to a function should match its parameters

Bug

The "open" builtin function should be called with a valid mode

Bug

Only defined names should be listed in "__all__"

Bug

Calls should not be made to non-callable values

Bug

Property getter, setter and deleter methods should have the expected number of parameters

Bug

Special methods should have an expected number of parameters

Bug

Instance and class methods should have at least one positional parameter

Bug

Boolean expressions of exceptions should not be used in "except" statements

Bug

Caught Exceptions must derive from BaseException

Bug

Item operations should be done on objects supporting them

Bug

Raised Exceptions must derive from BaseException

Bug

XML parsers should not be vulnerable to XXE attacks

Analyze your code

Vulnerability Blocker cwe owasp

XML standard allows the use of entities, declared in the DOCTYPE of the document, which can be internal or external.

When parsing the XML file, the content of the external entities is retrieved from an external storage such as the file system or network, which may lead, if no restrictions are put in place, to arbitrary file disclosures or server-side request forgery (SSRF) vulnerabilities.

It's recommended to limit resolution of external entities by using one of these solutions:

- If DOCTYPE is not necessary, completely disable all DOCTYPE declarations.
- If external entities are not necessary, completely disable their declarations.
- If external entities are necessary then:
  - Use XML processor features, if available, to authorize only required protocols (eg: https).
  - And use an entity resolver (and optionally an XML Catalog) to resolve only trusted entities.

Noncompliant Code Example

lxml module:

- When parsing XML:

```
parser = etree.XMLParser() # Noncompliant: by default resolv
tree1 = etree.parse('ressources/xxe.xml', parser)
root1 = tree1.getroot()

parser = etree.XMLParser(resolve_entities=True) # Noncomplia
tree1 = etree.parse('ressources/xxe.xml', parser)
root1 = tree1.getroot()
```

- When validating XML:

```
parser = etree.XMLParser(resolve_entities=True) # Noncomplia
treexsd = etree.parse('ressources/xxe.xsd', parser)
rootxsd = treexsd.getroot()
schema = etree.XMLSchema(rootxsd)
```

- When transforming XML:

```
ac = etree.XSLTAccessControl(read_network=True, write_networ
transform = etree.XSLT(rootxsl, access_control=ac)
```

xml.sax module:

```
parser = xml.sax.make_parser()
myHandler = MyHandler()
parser.setContentHandler(myHandler)
```

Operators should be used on compatible types

Bug

Function arguments should be passed only once

Bug

Iterable unpacking, "for-in" loops and "yield from" should use an Iterable object

Bug

Variables, classes and functions should be defined before being used

```
parser.setFeature(feature_external_ges, True) # Noncompliant
parser.parse("ressources/xxe.xml")
```

Compliant Solution

lxml module:

- When parsing XML, disable `resolve_entities` and *network access*:

```
parser = etree.XMLParser(resolve_entities=False, no_network=
tree1 = etree.parse('ressources/xxe.xml', parser)
root1 = tree1.getroot()
```

- When validating XML (note that network access cannot be completely disabled when calling XMLSchema):

```
parser = etree.XMLParser(resolve_entities=False) # Compliant
treexsd = etree.parse('ressources/xxe.xsd', parser)
rootxsd = treexsd.getroot()
schema = etree.XMLSchema(rootxsd) # Compliant
```

- When transforming XML, disable access to network and file system:

```
parser = etree.XMLParser(resolve_entities=False) # Compliant
treexsl = etree.parse('ressources/xxe.xsl', parser)
rootxsl = treexsl.getroot()

ac = etree.XSLTAccessControl.DENY_ALL  # Compliant
transform = etree.XSLT(rootxsl, access_control=ac) # Complia
```

To prevent xxe attacks with xml.sax module (for other security reasons than XXE, xml.sax is not recommended):

```
parser = xml.sax.make_parser()
myHandler = MyHandler()
parser.setContentHandler(myHandler)
parser.parse("ressources/xxe.xml") # Compliant: in version 3

parser.setFeature(feature_external_ges, False) # Compliant
parser.parse("ressources/xxe.xml")
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A4 - XML External Entities (XXE)
- OWASP XXE Prevention Cheat Sheet
- MITRE, CWE-611 - Information Exposure Through XML External Entity Reference
- MITRE, CWE-827 - Improper Control of Document Type Definition

Available In:

sonarlint | sonarcloud | sonarqube