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...

Item operations should be done on objects supporting them
Bug

Raised Exceptions must derive from BaseException
Bug

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
Bug

Identity operators should not be used with dissimilar types
Bug

Only strings should be listed in "__all__"
Bug

"__init__" should not return a value
Bug

"yield" and "return" should not be used outside functions
Bug

String formatting should not lead to runtime errors
Bug

Recursion should not be infinite

LDAP queries should not be vulnerable to injection attacks

Analyze your code

Vulnerability | Blocker | injection cwe owasp

User-provided data such as URL parameters should always be considered as untrusted and tainted. Constructing LDAP names or search filters directly from tainted data enables attackers to inject specially crafted values that changes the initial meaning of the name or filter itself. Successful LDAP injections attacks can read, modify or delete sensitive information from the directory service.

Within LDAP names, the special characters `' '`, `'#'`, `'"'`, `'+'`, `','`, `';'`, `'<'`, `'>'`, `'\'` and `null` must be escaped according to RFC 4514, for example by replacing them with the backslash character `'\'` followed by the two hex digits corresponding to the ASCII code of the character to be escaped. Similarly, LDAP search filters must escape a different set of special characters (including but not limited to `'*'`, `'('`, `')'`, `'\'` and `null`) according to RFC 4515.

Noncompliant Code Example

```
from flask import request
import ldap

@app.route("/user")
def user():
    dn =  request.args['dn']
    username =  request.args['username']

    search_filter = "(&(objectClass=*)(uid="+username+"))"
    ldap_connection = ldap.initialize("ldap://127.0.0.1:389"
    user = ldap_connection.search_s(dn, ldap.SCOPE_SUBTREE,
    return user[0]
```

Compliant Solution

```
from flask import request
import ldap
import ldap.filter
import ldap.dn

@app.route("/user")
def user():
    dn = "dc=%s" % ldap.dn.escape_dn_chars(request.args['dc'
    username = ldap.filter.escape_filter_chars(request.args[

    search_filter = "(&(objectClass=*)(uid="+username+"))"
    ldap_connection = ldap.initialize("ldap://127.0.0.1:389"
    user = ldap_connection.search_s(dn, ldap.SCOPE_SUBTREE,
    return user[0]
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection

Bug

Silly equality checks should not be made

Bug

Granting access to S3 buckets to all or authenticated users is security-sensitive

Security Hotspot

Hard-coded credentials are security-sensitive

Security Hotspot

- RFC 4514 - LDAP: String Representation of Distinguished Names
- RFC 4515 - LDAP: String Representation of Search Filters
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-90 - Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')

Available In:

sonarcloud | sonarqube Developer Edition