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...

sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive

Security Hotspot

Using non-standard cryptographic algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Constants should not be used as conditions

Code Smell

"SystemExit" should be re-raised

Code Smell

Bare "raise" statements should only be used in "except" blocks

Code Smell

Comparison to None should not be constant

Code Smell

"self" should be the first argument to instance methods

Code Smell

Function parameters' default values should not be modified or assigned

Code Smell

Some special methods should return

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

Analyze your code

Security Hotspot | Blocker | aws cwe owasp

Predefined permissions, also known as canned ACLs, are an easy way to grant large privileges to predefined groups or users.

The following canned ACLs are security-sensitive:

- `PUBLIC_READ`, `PUBLIC_READ_WRITE` grant respectively "read" and "read and write" privileges to everyone in the world (`AllUsers` group).
- `AUTHENTICATED_READ` grants "read" privilege to all authenticated users (`AuthenticatedUsers` group).

Ask Yourself Whether

- The S3 bucket stores sensitive data.
- The S3 bucket is not used to store static resources of websites (images, css …).

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

Recommended Secure Coding Practices

It's recommended to implement the least privilege policy, i.e., to grant necessary permissions only to users for their required tasks. In the context of canned ACL, set it to `PRIVATE` (the default one), and if needed more granularity then use an appropriate S3 policy.

Sensitive Code Example

All users (ie: anyone in the world authenticated or not) have read and write permissions with the `PUBLIC_READ_WRITE` access control:

```
bucket = s3.Bucket(self, "bucket",
    access_control=s3.BucketAccessControl.PUBLIC_READ_W
)

s3deploy.BucketDeployment(self, "DeployWebsite",
    access_control=s3.BucketAccessControl.PUBLIC_READ_W
)
```

Compliant Solution

With the `PRIVATE` access control (default), only the bucket owner has the read/write permissions on the buckets and its ACL.

```
bucket = s3.Bucket(self, "bucket",
    access_control=s3.BucketAccessControl.PRIVATE
)
```

"NotImplemented" instead of raising "NotImplementedError"

Code Smell

Custom Exception classes should inherit from "Exception" or one of its subclasses

Code Smell

Bare "raise" statements should not be used in "finally" blocks

Code Smell

Arguments given to functions should be of an expected type

Code Smell

`str.replace` should be preferred to

```
# Another example
s3deploy.BucketDeployment(self, "DeployWebsite",
    access_control=s3.BucketAccessControl.PRIVATE
)
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- AWS Documentation - Access control list (ACL) overview (canned ACLs)
- AWS Documentation - Controlling access to a bucket with user policies
- MITRE, CWE-732 - Incorrect Permission Assignment for Critical Resource
- MITRE, CWE-284 - Improper Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control
- AWS CDK version 2 - Class Bucket (construct)

Available In:

sonarcloud sonarqube