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

CloudFormation static code analysis

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

All rules 27 | Vulnerability 3 | Security Hotspot 20 | Code Smell 4

Tags | Search by name...

Policies authorizing public access to resources are security-sensitive
Security Hotspot

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

AWS IAM policies should not allow privilege escalation
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

Allowing public ACLs or policies on a S3 bucket is security-sensitive
Security Hotspot

Authorizing HTTP communications with S3 buckets is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

"Log Groups" should be configured with a retention policy
Code Smell

Defining a short backup retention duration is security-sensitive
Security Hotspot

Using unencrypted EFS file systems is security-sensitive
Security Hotspot

Using unencrypted SQS queues is security-sensitive
Security Hotspot

Using unencrypted SNS topics is security-sensitive
Security Hotspot

Using unencrypted SageMaker notebook instances is

Policies authorizing public access to resources are security-sensitive

Analyze your code

Security Hotspot | Blocker | aws cwe owasp

Resource-based policies granting access to all users can lead to information leakage.

Ask Yourself Whether

- The AWS resource stores or processes sensitive data.
- The AWS resource is not designed to be public.

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 principle, i.e. to grant necessary permissions only to users for their required tasks. In the context of resource-based policies, list the principals that need the access and grant to them only the required privileges.

Sensitive Code Example

This policy allows all users, including anonymous ones, to access an s3 bucket:

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3BucketPolicy:
    Type: 'AWS::S3::BucketPolicy' # Sensitive
    Properties:
      Bucket: !Ref S3Bucket
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS: "*" # all principals / anonymous access
            Action: "s3:PutObject" # can put object
            Resource: arn:aws:s3:::mybucket/*
```

Compliant Solution

This policy allows only the authorized users:

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3BucketPolicy:
    Type: 'AWS::S3::BucketPolicy' # Compliant
    Properties:
      Bucket: !Ref S3Bucket
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS:
                - !Sub 'arn:aws:iam::${AWS::AccountId}:root' # only this principal
            Action: "s3:PutObject" # can put object
            Resource: arn:aws:s3:::mybucket/*
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- AWS Documentation - Grant least privilege
- MITRE, CWE-732 - Incorrect Permission Assignment for Critical Resource
- MITRE, CWE-284 - Improper Access Control
- OWASP Top 10 2017 Category A5 - Broken Access Control

Available In:

sonarcloud | sonarqube