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...

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 security-sensitive
Security Hotspot

Using unencrypted Elasticsearch domains is security-sensitive
Security Hotspot

Using unencrypted RDS databases is security-sensitive

AWS IAM policies should not allow privilege escalation

Analyze your code

Vulnerability | Critical | cwe owasp aws

AWS Identity and Access Management (IAM) is the service that defines access to AWS resources. One of the core components of IAM is the policy which, when attached to an identity or a resource, defines its permissions. Policies granting permission to an Identity (a User, a Group or Role) are called identity-based policies. They add the ability to an identity to perform a predefined set of actions on a list of resources.

Here is an example of a policy document defining a limited set of permission that grants a user the ability to manage his own access keys.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey"
            ],
            "Resource": "arn:aws:iam::245500951992:user/${aws:username}",
            "Effect": "Allow",
            "Sid": "AllowManageOwnAccessKeys"
        }
    ]
}
```

Privilege escalation generally happens when an identity policy gives to an identity the ability to grant more privileges than the ones it already has. Here is another example of a policy document that hides a privilege escalation. It allows an identity to generate a new access key for any user from the account, including users with high privileges.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "AllowManageOwnAccessKeys"
        }
    ]
}
```

Although it looks like it grants a limited set of permissions, this policy would, in practice, give the highest privileges to the identity it's attached to.

Privilege escalation is a serious issue as it allows a malicious user to easily escalate to a high privilege identity from a low privilege identity it took control of.

The example above is just one of many permission combinations that the rule can detect. There are other variants based on `iam:PassRole`, `lambda:UpdateFunctionCode`, and other IAM permissions that allow to update, create, and attach IAM policies.

The general recommendation to protect against privilege escalation is to restrict the resources that are granted sensitive permissions to. The first example above is a good demonstration of sensitive permissions being used with a narrow scope of resources and where no privilege escalation is possible.

Noncompliant Code Example

This policy allows to update the code of any lambda function. Updating the code of a lambda executing with high privileges will lead to privilege escalation.

```
AWSTemplateFormatVersion: 2010-09-09

Resources:
  # Update Lambda code
  lambdaUpdatePolicy:
    # Noncompliant
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: lambdaUpdatePolicy
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - lambda:UpdateFunctionCode
            Resource: "*"
```

Compliant Solution

Narrow the policy to only allow to update the code of certain lambda functions.

```
AWSTemplateFormatVersion: 2010-09-09

Resources:
  # Update Lambda code
  lambdaUpdatePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: lambdaUpdatePolicy
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - lambda:UpdateFunctionCode
            Resource: "arn:aws:lambda:us-east-2:123456789012:function:my-function:1"
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- Rhino Security Labs - AWS IAM Privilege Escalation – Methods and Mitigation
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-269 - Improper Privilege Management

Available In:

sonarcloud | sonarqube