sonar RULES

Products

- 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

Terraform static code analysis

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

All rules 50 | Vulnerability 5 | Security Hotspot 43 | Code Smell 2

Tags | Search by name...

Creating public APIs is security-sensitive
Security Hotspot

Allowing public network access to cloud resources is security-sensitive
Security Hotspot

Having AWS policies that grant access to all resources of an account is security-sensitive
Security Hotspot

Having policies that grant all privileges is security-sensitive
Security Hotspot

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

Having policies that grant all privileges is security-sensitive

Analyze your code

Security Hotspot Blocker cwe owasp aws gcp

A policy that grants all permissions may indicate an improper access control, which violates the principle of least privilege. Suppose an identity is granted full permissions to a resource even though it only requires read permission to work as expected. In this case, an unintentional overwriting of resources may occur and therefore result in loss of information.

Ask Yourself Whether

Identities obtaining all the permissions:

- only require a subset of these permissions to perform the intended function.
- have monitored activity showing that only a subset of these permissions is actually used.

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

Recommended Secure Coding Practices

It's recommended to apply the least privilege principle, i.e. by only granting the necessary permissions to identities. A good practice is to start with the very minimum set of permissions and to refine the policy over time. In order to fix overly permissive policies already deployed in production, a strategy could be to review the monitored activity in order to reduce the set of permissions to those most used.

Noncompliant Code Example

A customer-managed policy for AWS that grants all permissions by using the wildcard (*) in the `Action` property:

```
resource "aws_iam_policy" "example" {
  name = "noncompliantpolicy"

  policy = jsonencode({
    Version   = "2012-10-17"
    Statement = [
      {
        Action   = [
          "*" # Sensitive
        ]
        Effect   = "Allow"
        Resource = [
          aws_s3_bucket.mybucket.arn
        ]
      }
    ]
  })
}
```

Security Hotspot

Google Cloud load balancers SSL policies should not offer weak cipher suites

Vulnerability

Azure custom roles should not grant subscription Owner capabilities

Vulnerability

Excluding users or groups activities from audit logs is security-sensitive

Security Hotspot

Defining a short log retention duration is security-sensitive

Security Hotspot

A customer-managed policy for GCP that grants all permissions by using the actions admin role `role` property:

```
resource "google_project_iam_binding" "example" {
  project = "example"
  role    = "roles/owner" # Sensitive

  members = [
    "user:jane@example.com",
  ]
}
```

Compliant Solution

A customer-managed policy for AWS that grants only the required permissions:

```
resource "aws_iam_policy" "example" {
  name = "compliantpolicy"

  policy = jsonencode({
    Version   = "2012-10-17"
    Statement = [
      {
        Action   = [
          "s3:GetObject"
        ]
        Effect   = "Allow"
        Resource = [
          aws_s3_bucket.mybucket.arn
        ]
      }
    ]
  })
}
```

A customer-managed policy for GCP that grants restricted permissions by using the actions admin role `role` property:

```
resource "google_project_iam_binding" "example" {
  project = "example"
  role    = "roles/actions.Viewer"

  members = [
    "user:jane@example.com",
  ]
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- AWS Documentation - Grant least privilege
- Google Cloud Documentation - Understanding roles
- 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