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...

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

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

Enabling Attribute-Based Access Control for Kubernetes is security-sensitive
Security Hotspot

Creating custom roles allowing privilege escalation is security-sensitive
Security Hotspot

Creating App Engine handlers without requiring TLS is security-sensitive
Security Hotspot

Excessive granting of GCP IAM

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:

```
resource "aws_s3_bucket_policy" "mynoncompliantpolicy"
  bucket = aws_s3_bucket.mybucket.id
  policy = jsonencode({
    Id = "mynoncompliantpolicy"
    Version = "2012-10-17"
    Statement = [{
            Effect = "Allow"
            Principal = {
                AWS = "*"
            }
            Action = [
                "s3:PutObject"
            ]
            Resource: "${aws_s3_bucket.mybucket.arn}/*"
        }
    ]
  })
}
```

Compliant Solution

This policy allows only the authorized users:

```
resource "aws_s3_bucket_policy" "mycompliantpolicy" {
  bucket = aws_s3_bucket.mybucket.id
```

permissions is security-sensitive

Security Hotspot

Enabling project-wide SSH keys to access VM instances is security-sensitive

Security Hotspot

Granting public access to GCP resources is security-sensitive

Security Hotspot

Creating GCP SQL instances without requiring TLS is security-sensitive

Security Hotspot

Creating DNS zones without DNSSEC enabled is security-sensitive

```
  policy = jsonencode({
    Id = "mycompliantpolicy"
    Version = "2012-10-17"
    Statement = [{
            Effect = "Allow"
            Principal = {
                AWS = [
                    "arn:aws:iam::${data.aws_caller_ide
                ]
            }
            Action = [
                "s3:PutObject"
            ]
            Resource = "${aws_s3_bucket.mybucket.arn}/*
        }
    ]
  })
}
```

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