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...

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
Security Hotspot

Using unencrypted EBS volumes is security-sensitive
Security Hotspot

Disabling logging is security-sensitive
Security Hotspot

Administration services access should be restricted to specific IP addresses
Vulnerability

Unversioned Google Cloud Storage buckets are security-sensitive
Security Hotspot

Disabling S3 bucket MFA delete is security-sensitive

Creating DNS zones without DNSSEC enabled is security-sensitive

Analyze your code

Security Hotspot Major gcp

Domain Name Systems (DNS) are vulnerable by default to various types of attacks.

One of the biggest risks is DNS cache poisoning, which occurs when a DNS accepts spoofed DNS data, caches the malicious records, and potentially sends them later in response to legitimate DNS request lookups. This attack typically relies on the attacker's MITM ability on the network and can be used to redirect users from an intended website to a malicious website.

To prevent these vulnerabilities, Domain Name System Security Extensions (DNSSEC) ensure the integrity and authenticity of DNS data by digitally signing DNS zones.

The public key of a DNS zone used to validate signatures can be trusted as DNSSEC is based on the following chain of trust:

- The parent DNS zone adds a "fingerprint" of the public key of the child zone in a "DS record".
- The parent DNS zone signs it with its own private key.
- And this process continues until the root zone.

Ask Yourself Whether

The parent DNS zone (likely managed by the DNS registrar of the domain name) supports DNSSEC and

- The DNS zone is public (contains data such as public reachable IP addresses).

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

It's recommended to use DNSSEC when creating private and public DNS zones.

Private DNS zones cannot be queried on the Internet and provide DNS name resolution for private networks. The risk of MITM attacks might be considered low on these networks and therefore implementing DNSSEC is still recommended but not with a high priority.

Note: Choose a robust signing algorithm when setting up DNSSEC, such as `rsasha256`. The insecure `rsasha1` algorithm should no longer be used.

Sensitive Code Example

```
resource "google_dns_managed_zone" "example" { # Sensit
  name     = "foobar"
  dns_name = "foo.bar."
}
```

Security Hotspot

Disabling versioning of S3 buckets is security-sensitive

Security Hotspot

Disabling server-side encryption of S3 buckets is security-sensitive

Security Hotspot

AWS tag keys should comply with a naming convention

Code Smell

Terraform parsing failure

Code Smell

Compliant Solution

```
resource "google_dns_managed_zone" "example" {
  name     = "foobar"
  dns_name = "foo.bar."

  dnssec_config {
    default_key_specs {
      algorithm = "rsasha256"
    }
  }
}
```

See

- OWASP Top 10 2021 Category A8 - Software and Data Integrity Failures
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- GCP Documentation - Manage DNSSEC configuration
- MITRE, CWE-345 - Insufficient Verification of Data Authenticity
- MITRE, CWE-353 - Missing Support for Integrity Check

Available In:

sonarcloud sonarqube