Lab: Sensitive Data in Terraform State

Terraform state can contain sensitive data depending on the resources used. Sometimes it can contain initial database passwords or other secret data returned by a provider. Every time you deploy infrastructure with Terraform it stores lots of data about that infrastructure including all the parameters you passed in, within the state file.

We recommend that you protect Terraform state as you would any other sensitive piece of data.

- Task 1: View state file in raw format
- Task 2: Suppress sensitive information
- Task 3: View the Terraform state file

Task 1: View Terraform State in Raw Format

By default the local state is stored in plain text as JSON. There is no additional encryption beyond your hard disk. View the Terraform state file of your most recent deployment:

`terraform.tfstate`


```
{
  "version": 4,
  "terraform_version": "1.1.1",
  "serial": 28,
  "lineage": "3f020a98-09d2-d827-e51d-673c7ec480bf",
  "outputs": {
    "public_dns": {
      "value": "ec2-34-224-58-141.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_dns_server_subnet_1": {
      "value": "ec2-3-239-20-18.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_ip": {
      "value": "34.224.58.141",
      "type": "string"
    },
    "public_ip_server_subnet_1": {
      "value": "3.239.20.18",
      "type": "string"
    },
    "size": {
      "value": "t2.micro",
```


```
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "data",
      "type": "aws_ami",
      "name": "ubuntu",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
    ...
```

Other backends do support features like state encryption. With Amazon S3 for example you can enable encryption on the bucket, IAM access and logging, and only connect to state via TLS. Terraform Cloud also encrypts data in transit and at rest with a unique encryption key for each state version.

Task 2: Suppress sensitive information

Regardless of where state is stored it should be treated like any other sensitive piece of data. To showcase this, add the following terraform code configuration to a new file in your working directory called `contactinfo.tf`

```
variable "first_name" {
  type = string
  sensitive = true
  default = "Terraform"
}

variable "last_name" {
  type = string
  sensitive = true
  default = "Tom"
}

variable "phone_number" {
  type = string
  sensitive = true
  default = "867-5309"
}

locals {
  contact_info = {
      first_name = var.first_name
      last_name = var.last_name
      phone_number = var.phone_number
```


```
  }

  my_number = nonsensitive(var.phone_number)
}

output "first_name" {
  value = local.contact_info.first_name
}

output "last_name" {
  value = local.contact_info.last_name
}

output "phone_number" {
  value = local.contact_info.phone_number
}

output "my_number" {
  value = local.my_number
}
```

Execute a `terraform apply` with the variables in the `terraform.auto.tfvars`.

```
terraform apply
```

You will notice that several of the output blocks error as they need to have the `sensitive = true` value set.

```
| Error: Output refers to sensitive values
|
|   on variables.tf line 73:
|   73: output "phone_number" {
|
| To reduce the risk of accidentally exporting sensitive data that was intended to be on
| sensitive data be explicitly marked as sensitive, to confirm your intent.
|
| If you do intend to export this data, annotate the output value as sensitive by adding
|     sensitive = true
```

Update the outputs to set the `sensitive = true` attribute and rerun the apply.

```
output "first_name" {
  sensitive = true
  value     = local.contact_info.first_name
}

output "last_name" {
  sensitive = true
```


```
  value     = local.contact_info.last_name
}

output "phone_number" {
  sensitive = true
  value     = local.contact_info.phone_number
}
```

```
terraform apply
```

```
Outputs:

first_name = <sensitive>
last_name = <sensitive>
my_number = "867-5309"
phone_number = <sensitive>
```

Task 3: View the Terraform State File

Even though items are marked as sensitive within the Terraform configuration, they are stored within the Terraform state file. It is therefore critical to limit the access to the Terraform state file.

View the most current terraform state file:

```
{
  "version": 4,
  "terraform_version": "1.1.1",
  "serial": 29,
  "lineage": "3f020a98-09d2-d827-e51d-673c7ec480bf",
  "outputs": {
    "first_name": {
      "value": "Terraform",
      "type": "string",
      "sensitive": true
    },
    "last_name": {
      "value": "Tom",
      "type": "string",
      "sensitive": true
    },
    "my_number": {
      "value": "867-5309",
      "type": "string"
    },
    "phone_number": {
      "value": "867-5309",
```


```
      "type": "string",
      "sensitive": true
    },
    ...
```

Once complete delete the `contactinfo.tf` configuration from your working directory, and run a `terraform destroy`.

Terraform State Best Practices

Treat State as Sensitive Data

We recommend that you protect Terraform state as you would any other sensitive piece of data. Depending on the Terraform resources, state may contain sensitive data including database passwords. This behavior is resource-specific and users should assume plain-text by default.

Encrypt State Backend

Store Terraform state in a backend that supports encryption. Instead of storing your state in a local terraform.tfstate file, Terraform natively supports a variety of backends, such as S3, GCS, and Azure Blob Storage. Many of these backends support encryption, so that instead of your state files being in plain text, they will always be encrypted, both in transit (e.g., via TLS) and on disk (e.g., via AES-256).

Control Access to State File

Strictly control who can access your Terraform backend. Since Terraform state files may contain secrets, you'll want to carefully control who has access to the backend you're using to store your state files. For example, if you're using S3 as a backend, you'll want to configure an IAM policy that solely grants access to the S3 bucket for production to a small handful of trusted devs (or perhaps solely just the CI server you use to deploy to prod).