Instructions

Please complete the following exercises. For any testing and troubleshooting of Vault, we recommend the use of a Unix-based operating system since that is what the vast majority of our work and our customers use. If you complete this exercise outside of a Unix-based operating system, please note as such.

We encourage you to challenge yourself and use this as a hands-on exploration of Vault. We are looking for correct answers (and believe you can get there with the resources provided below), but we are also interested in learning how you approach the problems and what you discover. If you struggle with a response, let us know your troubleshooting steps, what your next steps would be, or how you'd ask for help.

Some tools that you may find helpful include the following. You do not need to read or study all of the below information, instead, use these links as starting points, and dig in deeper if you get stuck.

- Vault Dev Server (Dev Server Mode | Vault by HashiCorp)
- Vault documentation
    - Docs: Documentation | Vault by HashiCorp
    - API: HTTP API | Vault by HashiCorp
    - Learn Guides: Vault Tutorials - HashiCorp Learn
      HashiCorp Cloud Platform Tutorials

Technical: AWS Secrets Engine

For the below, please show your work (preferably as terminal commands/output) or outline the steps you took to determine the answers. If you're not confident in your testing, show us how far you got, tell us where you got stuck and/or ask pertinent questions.

1. Please set up a basic AWS secrets engine following our documentation (AWS - Secrets Engines | Vault by HashiCorp). Note that you do not have to set up an AWS account nor use valid AWS keys to walk through the exercise - feel free to use the following

configuration command when you get to the corresponding step:

```
$ vault write aws/config/root \
      access_key=AKIA \
      secret_key=abcdefg \
      region=us-east-1
```

2. Convert the below Vault CLI command (AWS Secrets Engine) into a direct Vault API request. You can use curl or any other http client:

```
$ vault write aws/roles/my-role \
    credential_type=iam_user \
    policy_document=-<<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    }
  ]
}
EOF
```

3. Write a Vault ACL policy that would give you the permissions to run the above command.[1]

Functional: Customer Response to a known issue

1. Consider the following ticket from a customer, which is related to a known issue related to using `+` in paths and that they are treated as “less specific” than a path without `+`. You can read more about this here. The answer here is straightforward, they will need to use named paths for the apps and environments but that will understandably be a less flexible solution than they want. Given the information above, craft a response to the customer.

Hi,

I'm having some issues writing a policy that does exactly what I need it to do.

We have a top level apps tree with several apps: apple, orange, grape, etc. We have many environments: staging, nonprod, etc. Each app folder has several secrets in it such as postgres, elasticsearch, redis, etc.

[1] Hint: create a non-root token with a specified policy easily using `vault token create -policy="foo"`

It also has a key named secrets and a directory named secrets.
Thus we have a tree like:

apps
---apple
-----staging
------ postgres
-----nonprod
---orange
-----staging

I would like to write a policy that denies write access to everything on the top level of the app/env, except for secrets and gives full access to the secrets directory as well.

Here is the policy that I've been working with:
```

# Allow devs to see the apps tree
path "apps/*" {
capabilities = ["list"]
}
# Allow devs to read any apps/ secret
path "apps/data/*" {
capabilities = ["read", "list"]
}
# Allow devs to make changes to their secrets key
path "apps/data/+/+/secrets" {
capabilities = ["create", "read", "update", "delete", "list"]
}
# Allow devs to make changes to their secrets trees
path "apps/data/+/+/secrets/*" {
capabilities = ["create", "read", "update", "delete", "list"]
}
# Allow devs to delete versions of secrets in another way
path "apps/delete/*" {
capabilities = ["update"]
}```
It currently works as expected for the keys that I wish to deny for. It also works for the leaf of secrets; however, secrets as a directory unfortunately gives me permission denied, which I do not expect.

I had thought that vault use the most specific path to evaluate paths so:
`apps/data/+/+/secrets/*` should apply rather than `apps/data/*`

Can you help?

Thanks,
Joe Smith
```

HCP - Networking

1. Respond to the customer question below. After providing a response to the customer, outline how you came to your answer.

> Good Morning,
>
> I have a quick question. I created a Vault cluster on HCP using this guide. I now want to access the private address of the Vault cluster from the EC2 instances in my organization's AWS VPC. Do you have any guides or suggestions on how I can achieve this with HCP Vault?
>
> Thanks,
>
> HashiCorp User