THOMAS MORE

Lab 1.3

RBAC Fix: Test Plan and Terraform Experiment

Toepasbare Cloud Security voor IT-Professionals

2025 - 2026

Contents

1. RBAC Fix: Test Plan and Terraform Experiment . . . 3
   1.1. Context . . . 3
   1.2. Learning goals . . . 3
   1.3. Estimated time . . . 3
   1.4. Scenario . . . 4
2. Starting point . . . 5
   2.1. Question 1 . . . 5
   2.2. Question 2 . . . 5
3. Test plan . . . 6
   3.1. Baseline: before changing Terraform . . . 6
   3.2. Change: alter Terraform . . . 6
   3.3. Question 3 . . . 7
   3.4. Question 4 . . . 7
4. Verification after your change . . . 8
   4.1. Question 5 . . . 8
   4.2. Question 6 . . . 8
   4.3. Question 7 . . . 8
5. Result comparison . . . 9
6. Reflection . . . 10
   6.1. Question 8 . . . 10
   6.2. Question 9 . . . 10
   6.3. Question 10 . . . 10
   6.4. Question 11 . . . 10
7. Final conclusion . . . 11

1. RBAC Fix: Test Plan and Terraform Experiment

1.1. Context

In the previous labs you discovered and tested an intentionally imperfect access model.

The backend application identity was given a broad role at resource group scope. That showed two important things:

- The backend identity could do too much on the Azure management plane.
- The backend identity still did not necessarily have the correct data-plane permission to read secrets from Key Vault.

In this lab you will not follow a fully guided step-by-step fix. Instead, you will use Terraform, Azure CLI, and the application endpoints to investigate and improve the access model yourself.

The goal is to make the application work with the smallest reasonable set of permissions.

1.2. Learning goals

By the end of this lab, you should be able to explain:

- Which Terraform role assignment created the risky access.
- Which role assignment the backend actually needs.
- Why scope matters in Azure RBAC.
- Why management-plane access and data-plane access are not the same thing.
- How you tested that the fix changed the behavior of the environment.

1.3. Estimated time

60 minutes

Suggested timing:

Time	Activity
10 minutes	Read the test plan and inspect the current Terraform
20 minutes	Change Terraform and apply your fix
20 minutes	Run the verification tests
10 minutes	Compare results and discuss

1.4. Scenario

The team now understands the problem:

“The backend application identity has broad access, but not necessarily the correct access.”

Your task is to alter the Terraform configuration so the backend identity no longer has broad Contributor access at resource group scope, but does have the specific permission needed for its intended application behavior.

You decide how to make the change. Work in pairs if possible.

2. Starting point

Before you change anything, confirm that you are working in the correct environment.

```
az account show --output table
```

Then go to the Terraform folder for Day 1.

```
cd <DAY_1_TERRAFORM_FOLDER>
```

Inspect the current Terraform files.

```
ls
```

Look for the role assignments related to:

- the backend managed identity
- the resource group
- the Key Vault
- the current Terraform-running user

> **Minimal instruction**
> Do not copy commands blindly. First read the Terraform file and decide which access model you want to end up with.

2.1. Question 1

Which role assignment currently gives the backend identity broad access?

Your answer:

2.2. Question 2

Which role assignment should the backend identity probably have instead?

Your answer:

3. Test plan

Use this plan to guide your experiment.

3.1. Baseline: before changing Terraform

First test the current behavior.

Test	Expected before fix	Observed result
`POST /api/security/impact-demo/tag-self`	Works	
`GET /api/security/secret-demo`	Fails or returns authorization error	
Backend identity role assignment	Contributor at resource group scope	

Useful commands or actions:

```
terraform output
```

```
az role assignment list `
  --assignee <BACKEND_PRINCIPAL_ID> `
  --all `
  --output table
```

Call the application endpoints from a browser, REST client, or terminal, depending on what is easiest in your environment.

```
curl -X POST https://<BACKEND_APP_HOSTNAME>/api/security/impact-demo/tag-self
```

```
curl https://<BACKEND_APP_HOSTNAME>/api/security/secret-demo
```

> **Command examples**
> The examples in this lab use PowerShell line continuation with a backtick (`). If you use Bash, replace the backtick with a backslash (\).

3.2. Change: alter Terraform

Change the Terraform so that the backend identity is no longer a Contributor on the full resource group.

Then give the backend identity the narrowly scoped access it actually needs for reading the demo secret.

You decide exactly how to edit the Terraform, but your final model should express this idea:

```
Remove broad management-plane access.
Add specific Key Vault secret access at Key Vault scope.
```

After editing the Terraform, format and validate your configuration.

```
terraform fmt
terraform validate
```

Inspect the planned change before applying it.

```
terraform plan --out deployment.plan
```

Only apply when the plan matches what you intended.

```
terraform apply deployment.plan
```

3.3. Question 3

What did Terraform plan to remove?

Your answer:

3.4. Question 4

What did Terraform plan to add?

Your answer:

4. Verification after your change

Run the same tests again.

Test	Expected after fix	Observed result
`POST /api/security/impact-demo/tag-self`	Fails	
`GET /api/security/secret-demo`	Works	
Backend identity role assignment	No Contributor at resource group scope	
Backend Key Vault access	Key Vault Secrets User at Key Vault scope	

Check the backend identity role assignments again.

```
az role assignment list `
  --assignee <BACKEND_PRINCIPAL_ID> `
  --all `
  --output table
```

4.1. Question 5

Did your fix remove the broad access without breaking the intended application behavior?

Your answer:

4.2. Question 6

Which test proves that the previous misuse path is blocked?

Your answer:

4.3. Question 7

Which test proves that the application now has the access it actually needs?

Your answer:

5. Result comparison

Complete the table below.

Check	Before	After	What changed?
Backend can modify its App Service tags			
Backend can read the demo secret			
Backend has Contributor at resource group scope			
Backend has Key Vault secret read access			

6. Reflection

6.1. Question 8

Why is the fix not simply “give the application more access”?

Your answer:

6.2. Question 9

Why is it better to assign the Key Vault role at Key Vault scope instead of resource group scope?

Your answer:

6.3. Question 10

What could go wrong if you removed the Contributor role but forgot to add the Key Vault role?

Your answer:

6.4. Question 11

What could go wrong if you added the Key Vault role but forgot to remove Contributor?

Your answer:

7. Final conclusion

At the end of this lab, your conclusion should look similar to this:

```
The backend managed identity no longer has Contributor access at resource group
scope.
The backend identity now has a narrower role that matches the application need.
The impact endpoint no longer works because broad management-plane access was
removed.
The secret endpoint works because the correct Key Vault data-plane access was added.
```

The important lesson is:

“Least privilege is not only about less access. It is about the right access, assigned to the right identity, at the right scope.”