THOMAS MORE

Lab 2.5

# Protecting Key Vault with a Private Endpoint

**Toepasbare Cloud Security voor IT-Professionals**

2025 - 2026

# Contents

# 1. Protecting Key Vault with a Private Endpoint

## 1.1. Context

In Lab 2.4 you moved the DB storage account behind a private endpoint and disabled public network access.

Key Vault holds secrets. It is currently reachable from the public internet.

In this lab you will apply the same pattern to Key Vault: create a private endpoint, verify the DNS zone, confirm the application still works, and then disable public network access.

## 1.2. Learning goals

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

- Why Key Vault needs a private endpoint in addition to RBAC.
- What the private DNS zone name for Key Vault is and why it differs from Table Storage.
- What the `AzureServices` bypass does and why it is not a substitute for a private endpoint.
- How to verify that a private endpoint is working before disabling public access.
- What happens to the application when public access on Key Vault is disabled.

## 1.3. Estimated time

**45 minutes**

Suggested timing:

| Time | Activity |
|---|---|
| 5 minutes | Verify starting state and collect values from the Portal |
| 5 minutes | Test that the application retrieves the secret (baseline) |
| 15 minutes | Create Key Vault private endpoint in the Portal |
| 10 minutes | Verify private endpoint and DNS, then disable public access |
| 10 minutes | Reflection questions |

## 1.4. Before you start

Make sure you have:

- [ ] The Key Vault name and URI from Lab 2.1 (or Portal: Key Vault → Overview)
- [ ] The `public_backend_url` from Lab 2.1
- [ ] Azure Portal access
- [ ] Lab 2.4 completed (private endpoint for the DB storage account already in place)

# 2. Micro-theory: Key Vault and network access

Key Vault supports the same private endpoint pattern as Table Storage. The private DNS zone is different:

| Service | Private DNS zone |
|---|---|
| Table Storage | `privatelink.table.core.windows.net` |
| Cosmos DB | `privatelink.documents.azure.com` |
| Key Vault | `privatelink.vaultcore.azure.net` |

Key Vault also has a network ACL bypass for trusted Azure services (`AzureServices`). This allows App Services using managed identity to reach Key Vault through the Azure backbone even when public access is denied — but only as a safety net. A private endpoint gives a stronger and more explicit boundary.

> **Note**
>
> The AzureServices bypass is not a substitute for a private endpoint. It does not prevent access from non-Azure networks. A private endpoint with public access disabled is the correct production configuration.

# 3. Part 1 — Verify the starting state

## 3.1. Collect Key Vault values from the Portal

Navigate to your Key Vault in the Azure Portal.

Go to **Overview** and write down:

| Field | Value |
|---|---|
| Vault name | |
| Vault URI | |

## 3.2. Check Key Vault public access in the Portal

Navigate to your Key Vault in the Azure Portal.

Go to **Settings → Networking**.

Write down the current public network access setting:

**Your answer:**

## 3.3. Test that the vault URI is reachable

```
curl -I <KEY_VAULT_URI>
```

Write down the HTTP response code:

**Your answer:**

# 4. Part 2 — Confirm the baseline application path

Call the secret demo endpoint on the public backend. This endpoint retrieves a secret from Key Vault using the managed identity.

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

Does the endpoint return the secret value?

| Item | Value |
|---|---|
| HTTP status | |
| Secret returned? | |

# 5. Part 3 — Create the Key Vault private endpoint

In the Azure Portal, navigate to your Key Vault.

Go to **Settings → Networking → Private endpoint connections** and click **+ Create**.

Fill in the wizard:

**Basics tab:**

| Setting | Value |
|---|---|
| Resource group | Your lab resource group |
| Name | `pe-kv-<your-prefix>` |
| Region | Same region as your resources |

**Resource tab:**

| Setting | Value |
|---|---|
| Connection method | Connect to an Azure resource in my directory |
| Resource type | `Microsoft.KeyVault/vaults` |
| Resource | Your Key Vault |
| Target sub-resource | `vault` |

**Virtual Network tab:**

| Setting | Value |
|---|---|
| Virtual network | Your lab VNet |
| Subnet | `snet-private-endpoints` |
| Private IP configuration | Dynamically allocate IP address |

**DNS tab:**

| Setting | Value |
|---|---|
| Integrate with private DNS zone | Yes |
| Private DNS zone | `privatelink.vaultcore.azure.net` (created automatically) |

Click **Next, Next**, then **Create**.

The portal creates:

| Resource | Purpose |
|---|---|
| Private endpoint | Private NIC for Key Vault inside the VNet |
| Private DNS zone | `privatelink.vaultcore.azure.net` |
| DNS zone VNet link | Allows the VNet to resolve private DNS names for Key Vault |

Write down any issues encountered during creation:

**Your answer:**

# 6. Part 4 — Verify the private endpoint

## 6.1. In the Azure Portal

1. Navigate to your Key Vault → **Networking → Private endpoint connections**.
2. Confirm the endpoint status shows **Approved**.

Write down the connection state:

**Your answer:**

## 6.2. In the private DNS zone

1. Navigate to **Private DNS zones → privatelink.vaultcore.azure.net**.
2. Confirm a DNS A record exists for your Key Vault name.

Write down the private IP address from the DNS record:

| Item | Value |
|---|---|
| Key Vault name | |
| Private IP address | |

## 6.3. Confirm the application still works

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

Does the endpoint still return the secret?

**Your answer:**

# 7. Part 5 — Disable public network access

In the Azure Portal, navigate to your Key Vault.

Go to **Settings → Networking**.

Under **Allow access from**, select **Disable public access**. Keep everything else as is.

Click **Apply** and wait for the change to take effect.

# 8. Part 6 — Test after disabling public access

## 8.1. Test the application path from your laptop (should be blocked)

The most reliable way to test if Key Vault public access is blocked is to try accessing it from your laptop using the Azure CLI or PowerShell.

Try to list secrets using Azure CLI:

```
az keyvault secret list --vault-name <KEY_VAULT_NAME>
```

Or using PowerShell:

```
Get-AzKeyVaultSecret -VaultName <KEY_VAULT_NAME>
```

Write down the result:

| Item | Value |
|---|---|
| Error message | |
| Public access blocked? | |

> **Note**
>
> You should see an error like 'Public network access is disabled' or 'ForbiddenByFirewall'. Key Vault's root URI always returns 404, so curl tests are not reliable for verifying public access blocks.

## 8.2. Test the application path (should still work)

```
curl https://<PUBLIC_BACKEND_URL>/api/secret-demo
```

Write down the result:

| Item | Value |
|---|---|
| HTTP status | |
| Secret returned? | |

**Why does it still work? Your answer:**

# 9. Reflection questions

## 9.1. Question 1

What is the difference between revoking the managed identity's role assignment and disabling public network access on Key Vault?

**Your answer:**

## 9.2. Question 2

Key Vault, the DB storage account, and the internal backend are now all protected by network-level controls. Which component is still fully public, and why is that acceptable?

**Your answer:**

## 9.3. Question 3

The private DNS zones for Table Storage and Key Vault are separate. What would happen if you linked only one of them to the VNet?

**Your answer:**

# 10. Final conclusion

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

```
Key Vault now has a private endpoint inside the VNet.
The private DNS zone resolves the Key Vault hostname to a private IP
for resources inside the VNet.

Public network access is disabled. A direct connection from the internet
to the Key Vault endpoint is now refused.

The public backend can still retrieve secrets through the private
endpoint using its managed identity. The application continues to work.

Both the DB storage account and Key Vault are now accessible only from within the
VNet.
The attack surface has been reduced by two layers.
```

> **Note**
>
> The same pattern applies to any Azure PaaS service that supports private endpoints: Storage accounts (blob, table, queue, file), Cosmos DB, Service Bus, Azure Container Registry, and others. Once you have done it for Table Storage and Key Vault, the pattern is the same.