THOMAS MORE

Lab 2.1

Mapping Public and Internal Exposure

Toepasbare Cloud Security voor IT-Professionals

2025 - 2026

Contents

1. Mapping Public and Internal Exposure

1.1. Context

In Day 2 you are working with a four-layer application:

```
User / Browser
→ Frontend
→ Public Backend API
→ Internal Backend Service
→ Data Layer (Table Storage)
```

The environment was deployed quickly. No network boundaries have been added yet.

In this lab you will identify which components are publicly reachable and classify them by whether they should be public.

1.2. Learning goals

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

- Which components in the Day 2 environment are reachable from the public internet.
- Which components should not be publicly reachable and why.
- What the difference is between a service being reachable and being authorized to receive requests.
- How to map exposure systematically across an environment.

1.3. Estimated time

60 minutes

Suggested timing:

Time	Activity
5 minutes	Read context and collect outputs
15 minutes	List and test all components
15 minutes	Complete the exposure worksheet
15 minutes	Reflection questions
10 minutes	Group discussion

1.4. Before you start

Make sure you have:

- [ ] Access to the Azure Portal
- [ ] Access to the correct Azure tenant and subscription

> **Note**
>
> In this lab, do not change anything. Your goal is to observe, document and explain.

2. Collect environment information

Collect the URLs and resource names for your environment from the Azure Portal.

Navigate to your resource group in the Azure Portal. You can find it under **Resource groups** in the left menu.

Write down the values you will use throughout this lab: You can use the table below and the UI or terraform outputs to fill in the table below.

Resource	**Where to find it**
Frontend URL	Storage account → **Data management** → **Static website** → Primary endpoint
Public backend URL	App Service `app-...-api` → **Overview** → Default domain
Internal backend URL	App Service `app-...-internal` → **Overview** → Default domain
DB Storage Account name	Storage account starting with `stdb`
Key Vault name	Key Vault
Log Analytics name	Log Analytics workspace → **Overview** → Workspace Name
Resource group name	Resource groups → your resource group → **Overview**

Write down the collected values:

Value	**Collected**
Frontend URL	
Public backend URL	
Internal backend URL	
DB Storage Account name	
Key Vault name	
Log Analytics workspace name	
Resource group name	

3. Test reachability of each component

Test each endpoint from your local machine or Azure Cloud Shell.

3.1. Frontend

```
curl -I <FRONTEND_URL>
```

HTTP response code:

Your answer:

3.2. Public Backend API

```
curl https://<PUBLIC_BACKEND_URL>/api/health
```

Does it respond? Write down the response:

Your answer:

3.3. Internal Backend Service

```
curl https://<INTERNAL_BACKEND_URL>/internal/health
```

Does it respond?

Your answer:

3.4. DB Storage Account endpoint

```
curl -I https://<DB_STORAGE_ACCOUNT_NAME>.table.core.windows.net
```

You will need to authenticate to be sure, you can do so by generating a SAS token in the Azure Portal:

Use the Access Key to generate a Shared Access Signature (SAS):

1. Open your storage account in the Azure Portal.
2. Navigate to Shared access signature under the security settings menu.
3. Check the Table service box, set your allowed permissions (e.g., Read, Write), and click Generate SAS and connection string.Copy the generated Table service SAS URL.
4. Paste that URL directly into your browser as a GET request.
5. Append your specific table name between the main URL endpoint (/) and the ? query string.
6. Navigate to the URL.

Example URL format:

```
https://stdbmmommersdevfdhikm.table.core.windows.net/audit?sv=2026-02-06&ss=t&srt=o&
sp=rwdlacu&se=2026-05-28T03:39:17Z&st=2026-05-27T19:24:17Z&spr=https&sig=MCmYE<snip>
```

If it shows or downloads a file (open it with notepad) you should see something like this if the endpoint is reachable:

```
<link rel="self" title="audit" href="audit" />
<author>
  <name />
</author>
```

HTTP response code:

Your answer:

4. Exposure worksheet

Fill in the worksheet based on your test results.

Component	Reachable from internet?	Should be public?	Reason
Frontend		Yes	Users access the UI directly
Public Backend API		Yes	Users call it via the frontend
Internal Backend Service		No	
DB Storage Account		No	
Key Vault		No	

> **Note**
>
> Key Vault is included in the Terraform outputs and the exposure worksheet because it is part of the environment. It is not included in the testing section because it does not have a public endpoint to test against.

5. Reflection questions

5.1. Question 1

What is the difference between a service being reachable and being authorized to receive requests?

Your answer:

5.2. Question 2

If someone calls the internal backend directly, what controls on the public backend can they bypass?

Your answer:

5.3. Question 3

The DB storage account endpoint is also publicly reachable. What would an attacker need to use it?

Your answer:

6. Final conclusion

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

```
The internal backend service and the DB storage account are publicly reachable.

This is a problem because these components are only intended for
internal use by other services, not for direct access from the internet.

Without explicit network controls, all Azure App Services and Storage
Accounts are accessible from the internet by default. Infrastructure
configuration is required to create a real boundary.
```