THOMAS MORE

Lab 2.2

Demonstrating Internal Backend Exposure

Toepasbare Cloud Security voor IT-Professionals

2025 - 2026

Contents

1. Demonstrating Internal Backend Exposure

1.1. Context

In Lab 2.1 you found that the internal backend service is publicly reachable.

In this lab you will prove the problem. You will call the internal backend directly and observe what happens in the logs compared to a normal request.

You will not fix anything in this lab. The goal is to generate evidence and understand what the logs reveal.

1.2. Learning goals

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

- How to make a normal request through the expected path.
- How to make a direct request to the internal backend.
- What the difference is in the log events produced by each path.
- Why logging a suspicious event is not the same as blocking it.
- What the `expectedPath` field in the internal backend logs means.

1.3. Estimated time

60 minutes

Suggested timing:

Time	Activity
5 minutes	Review context
10 minutes	Verify data exists in Table Storage
10 minutes	Generate normal request and record trace
10 minutes	Generate direct request and record trace
15 minutes	Compare log events
10 minutes	Reflection questions

1.4. Before you start

Make sure you have:

- [ ] The `public_backend_url` from Lab 2.1
- [ ] The `internal_backend_url` from Lab 2.1
- [ ] The DB storage account name from Lab 2.1
- [ ] Access to Application Insights or the App Service log stream
- [ ] Azure Storage Explorer installed on your local machine

Note

> In this lab, do not change anything. Your goal is to demonstrate the problem and record the evidence.

2. Micro-theory: expected caller header

When the public backend calls the internal backend, it adds a header to the request:

`X-Internal-Caller: backend-api`

The internal backend checks for this header.

If the header is present with the correct value, the internal backend logs an expected call event.

If the header is missing or incorrect, the internal backend logs a direct access event.

> **Note**
>
> The expected caller header is a teaching tool, not a security control. Any caller can add this header. Real protection requires network-level segmentation, which is the goal of Lab 2.3.

3. Confirm your Azure context

```
az account show --output json
```

Item	Value
Subscription name	
Subscription ID	
Signed-in user	

4. Part 1 — Verify data exists in the Table Storage

Before testing the endpoints, confirm that the DB storage account has actual data.

1. Open Azure Storage Explorer on your local machine.
2. Sign in with your Azure credentials.
3. Navigate to your subscription → Storage Accounts → DB storage account (`stdb...`) → Tables
4. Open the `messages` table.
5. Look for a row with:
   - PartitionKey: `message`
   - RowKey: `welcome`

Does the welcome message exist?

Your answer:

If the row does not exist, add it manually using Azure Storage Explorer:

Field	Value
PartitionKey	`message`
RowKey	`welcome`
text	`[Something you like to insert here]`
classification	`internal`

> **Note**
>
> The application tries to read this item when you call the `/internal/data/welcome` endpoint. If it doesn't exist, the app returns a fallback message. Adding it ensures you see real data being retrieved from Table Storage.

5. Part 2 — Generate a normal request

Call the trace demo endpoint on the public backend.

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

Record the response:

Field	Value
`correlationId`	
`message`	
`path`	

6. Part 3 — Generate a direct request to the internal backend

Call the internal backend trace demo endpoint directly, without going through the public backend.

```
curl https://<INTERNAL_BACKEND_URL>/internal/trace-demo
```

Record the response:

Field	Value
`correlationId`	
`expectedPath`	
`message`	

7. Part 4 — Call the data endpoint directly

Also call the data welcome endpoint on the internal backend.

```
curl https://<INTERNAL_BACKEND_URL>/internal/data/welcome
```

Does it return data?

Your answer:

What `classification` field is shown in the returned item?

Your answer:

> **Note**
> If you see the message you added in Part 1 and the classification field shows `internal`, the application successfully retrieved the data from Table Storage. If you see a fallback message, verify that you added the row with the correct PartitionKey and RowKey.

8. Part 5 — Compare log events

Open Application Insights in the Azure Portal or use the App Service log stream.

Search for events from both calls using the correlation IDs you recorded.

Complete the comparison table:

Event name	Normal request	Direct request
`internal-service-request-received`		
`internal-service-expected-call`		
`internal-service-direct-access-detected`		
`expectedPath` value		

9. Part 6 — Test with the expected caller header

Try adding the expected caller header manually to the direct call.

```
curl `
  -H "X-Internal-Caller: backend-api" `
  https://<INTERNAL_BACKEND_URL>/internal/trace-demo
```

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

What is the `expectedPath` value in the response now?

Your answer:

9.1. Question 5

You just spoofed the `X-Internal-Caller` header with a single curl flag. If this header were the only authentication mechanism protecting a production API, how long do you think it would take an attacker to figure out how to bypass it?

Your answer:

10. Reflection questions

10.1. Question 6

A security team says: “We have full audit logging enabled — if someone tries to attack us, we’ll see it in the logs.” What critical assumption are they making about the time between detection and response?

Your answer:

11. Final conclusion

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

```
The internal backend is directly reachable from the public internet.

A direct call bypasses any checks that the public backend API applies,
such as authentication, rate limiting, and input validation.

The internal backend logs a direct-access-detected event when called
without the expected caller header. This logging is valuable evidence,
but it does not stop the call.

Network segmentation is needed to block the unexpected path.
```

> **Note**
>
> Write down both correlation IDs before you continue. After Lab 2.3 blocks the direct path, you will not be able to re-generate the direct-access trace.