SECTION 06

Security in Fabric

Define Once. Enforce Everywhere.

Who Can See Our Margins?

> "Alex, these margin reports are great. But who else can see this? The supplier costs are confidential. Can you lock this down?"

Confidential Rates

Supplier costs are negotiated rates

Regional Isolation

Managers shouldn't see each other's data

Access Control

The intern just got Fabric access...

The question every organization asks. Eventually.

Demo: Securing Data in Fabric

Act	What Happens
1	The VP's security concern
2	Semantic model RLS (what you know)
3	The "multiple doors" problem
4	OneLake Security—the solution
5	Row and column level security

~40 minutes, live demo

DEMO PLACEHOLDER

DEMO

Live demonstration happens here

Semantic Model RLS & OLS

Security Type	What It Does
Row-Level Security (RLS)	Filter rows by user identity
Object-Level Security (OLS)	Hide columns or tables

Example RLS:

```
[Region] = USERPRINCIPALNAME()
```

You've done this before. It works great... for reports.

One Lock. Four Doors.

Report
✓ Secured

SQL Endpoint
✗ OPEN

Notebook
✗ OPEN

OneLake
✗ OPEN

> **Semantic model RLS only locks ONE door.**

How Users Bypass RLS

Access Method	**Semantic Model RLS**
Power BI Report	✓ Enforced
SQL Analytics Endpoint	✗ **Full access**
Spark Notebook	✗ **Full access**
OneLake APIs/Explorer	✗ **Full access**
Excel via XMLA	✗ **Full access**

> *"I'll just query the SQL endpoint directly..."*

If they know the door exists, they can walk through it.

OneLake Security

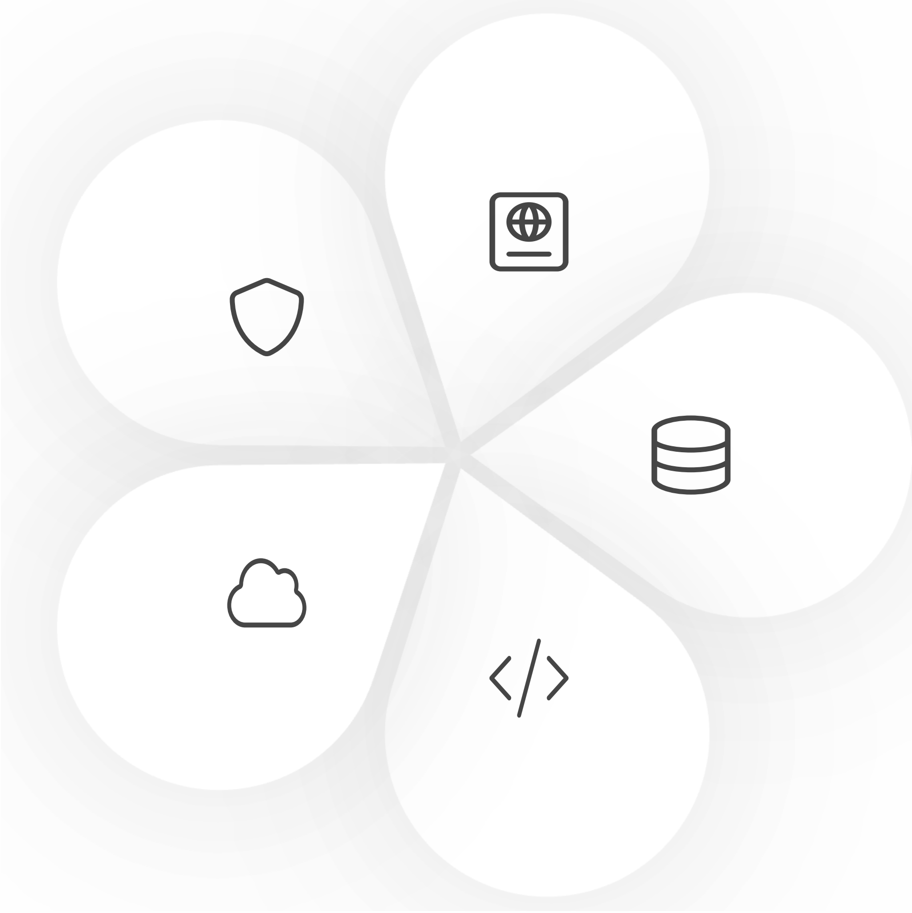
Security at the DATA layer, not the compute layer.

Aspect	How It Works
Where defined	Lakehouse (data item)
Where enforced	ALL access points
What it secures	Tables, rows, columns
Who it applies to	Everyone accessing the data

Define once. Enforced everywhere.

One Definition. Every Door.

OneLake Security Roles



Report

✓ Secured

SQL

✓ Secured

OneLake

✓ Secured

Notebook

✓ Secured

Same security definition. Every access point.

Under the Hood

Aspect	How It Works
Architecture	Parquet file-level filtering at query time
Engine	Evaluated by Fabric compute, not client tools
Scope	Delta tables in Lakehouses
Ownership	Data item owner enables and manages

Key insight: Security is applied *before* data leaves OneLake.

The query engine reads the security role, applies filters, and only returns permitted data. Tools never see what they can't access.

What It Can (and Can't) Do

✓ Works

- SQL Analytics Endpoint
- Spark Notebooks
- OneLake APIs & Explorer
- Power BI via Direct Lake
- Dataflows Gen2
- External tools via XMLA

⚠️ Limitations

- Shortcuts inherit *source* security
- Mirrored databases have own security
- Complex predicates = performance overhead
- Can't disable once enabled (preview)
- 5-minute propagation delay
- 250 roles maximum per Lakehouse

Bottom line: Covers native Lakehouse access comprehensively. External sources follow their own rules.

When to Use What

Approach	Scope	Best For
OneLake Security	All access to Lakehouse	Production data, multi-tool access
Semantic Model RLS	Reports only	Quick setup, report-specific logic
Workspace Roles	Entire workspace	Coarse access control (dev/test/prod)
Source System Security	External sources	Data that stays external (shortcuts)

01

Workspace roles = who can **administer**

02

OneLake Security = who can **read what data**

03

Semantic Model RLS = additional **report-specific** filtering

Use OneLake Security as your foundation. Add semantic model RLS for refinement.

Table → Row → Column

Level	What It Controls	Example
Table	Which tables users can see	Sales ✓, SupplierCosts ✗
Row (RLS)	Which rows within a table	`Region = 'Pacific Northwest'`
Column (CLS)	Which columns within a table	Hide `SupplierCost` column

Combine them for precise control.

Creating a Data Access Role

Step by Step

01

Open Lakehouse → **Manage OneLake security**

02

Click **New** role

03

Name it (e.g., `Regional Manager - Pacific NW`)

04

Choose **Read** permission

05

Select specific tables/folders

06

(Optional) Add RLS predicates

07

(Optional) Hide columns (CLS)

08

Add members

09

Save

Takes 2 minutes. Secures everything.

Filter Rows by User

Scenario	Predicate
Region filter	`Region = 'Pacific Northwest'`
User's own data	`OwnerEmail = @User`
Manager sees team	`ManagerID = @UserID`
Date range	`OrderDate >= '2025-01-01'`

Same SQL WHERE clause syntax you know.

Decision Matrix

Scenario	**Recommended Approach**
Report-only consumers	Semantic model RLS (simple)
Multiple access points	**OneLake Security** (comprehensive)
Hiding sensitive columns	OneLake CLS + Semantic model OLS
Quick prototyping	Semantic model RLS (faster)
Production security	**OneLake Security** (no bypass)
IT-managed data	**OneLake Security** (centralized)

Watch Out For

Gotcha	What to Know
5-minute delay	Role changes take time to propagate
Can't disable once enabled	Preview limitation—commit before enabling
DefaultReader role	Gives everyone full access initially—modify or delete it
250 roles max	Plan your role strategy
Preview status	Production-ready, but check latest docs

Best Practices

Practice	Why
Start with OneLake Security	Comprehensive coverage
Use groups, not individuals	Easier to manage
Test with real scenarios	Verify before production
Document your roles	Future you will thank you
Review regularly	Access needs change

Security is a process, not a one-time setup.

Getting Started

1. Enable OneLake Security on your Lakehouse
2. Review the DefaultReader role
3. Create your first custom role
4. Test access via SQL endpoint (verify it works!)
5. Plan your production role strategy

Key insight: Define security at the data layer. Enforce everywhere.

Resources:

- MS Learn: "OneLake security overview"
- MS Learn: "Data access control model"

The VP's Question, Answered

Before	**After**
RLS in semantic model only	Security at data layer
Multiple places to manage	Single definition
Gaps in coverage	All access points secured
Easy to bypass	No bypass possible
Worried VP	Confident VP

> *"Who can see our margins? Only the people you explicitly grant access to. Period."*

Key Takeaways

Data Layer Security

OneLake Security protects at the source, not the tool

Comprehensive Coverage

One definition enforced across all access points

No Bypass

SQL, notebooks, APIs—all secured automatically

SECTION COMPLETE

End of Section 06

Security in Fabric: Define Once. Enforce Everywhere.