

Graph-Driven Audits for



A Deep Dive with

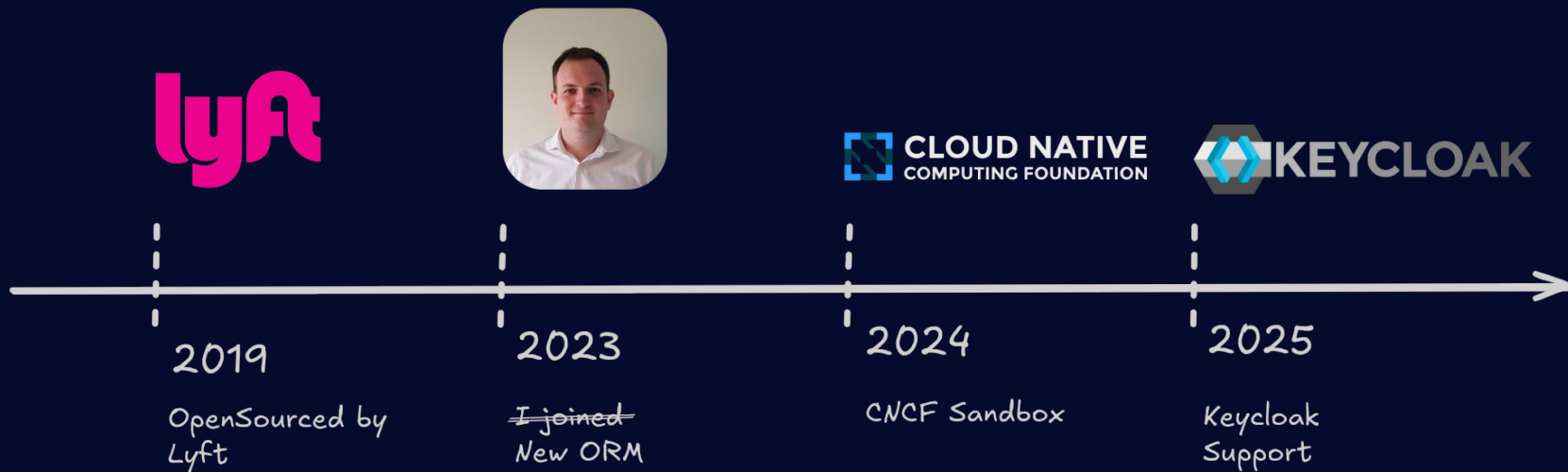


Do you really know who can access what?

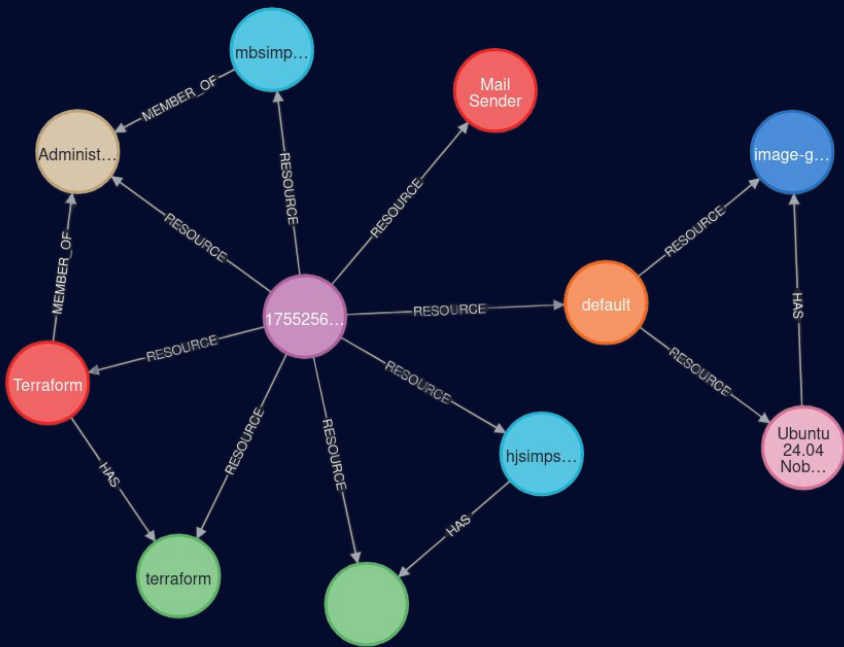
- **Hidden inheritance**
group memberships, composite roles, nested permissions
- **External identity providers**
complex trust relationships, often hard to track
- **Authentication workflows**
flexible but create blind spots in security review
- **Scaling issues**
manual audits don't scale with multi-realm, multi-client setups

Security teams lack a **holistic view** of "Who can access what?"

Cartography's Journey



Cartography in 1 slide




- **Ingestion** from provider APIs
- **Modeling** in a graph
- **Querying** the graph using the Cypher language
- Bonus: Drift detection

“BloodHound for Cloud”

Use cases

- Drift detection / compliance
- Access management
- Vulnerability management
- Asset inventory
- Incident response
- Log enrichment
- Data / secrets lineage


Thread
#drift-detection

 **intelgraphbot** APP [redacted]
DRIFT DETECTION ALERT

Query Name: RedshiftCluster-admin
New Results: 2
Removed Results: 0
Query:

```
MATCH (acc:AWSAccount)-[:RESOURCE]->
(rsc:RedshiftCluster)<-[:CAN_ADMINISTER]-
(admin:AWSPrincipal) RETURN acc.name, acc.id,
rsc.id, rsc.db_name, admin.arn ORDER BY acc.name,
rsc.id
```

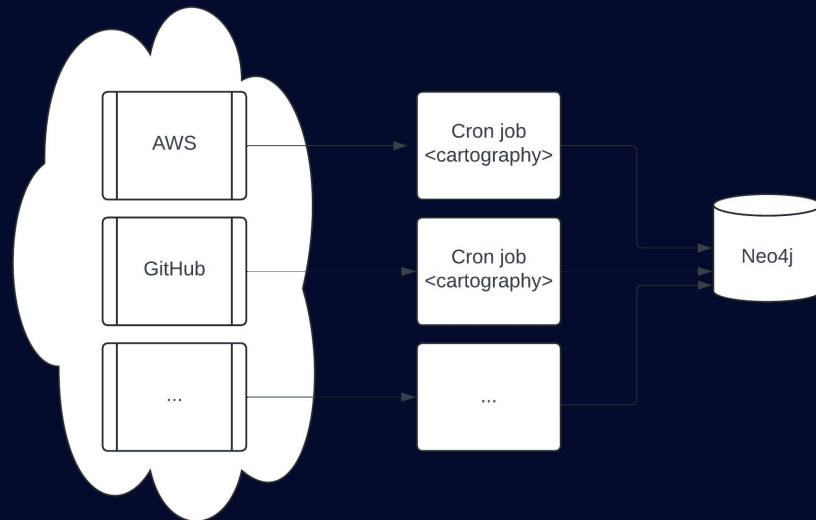
7 replies

 **intelgraphbot** APP 3 months ago
new results ▾

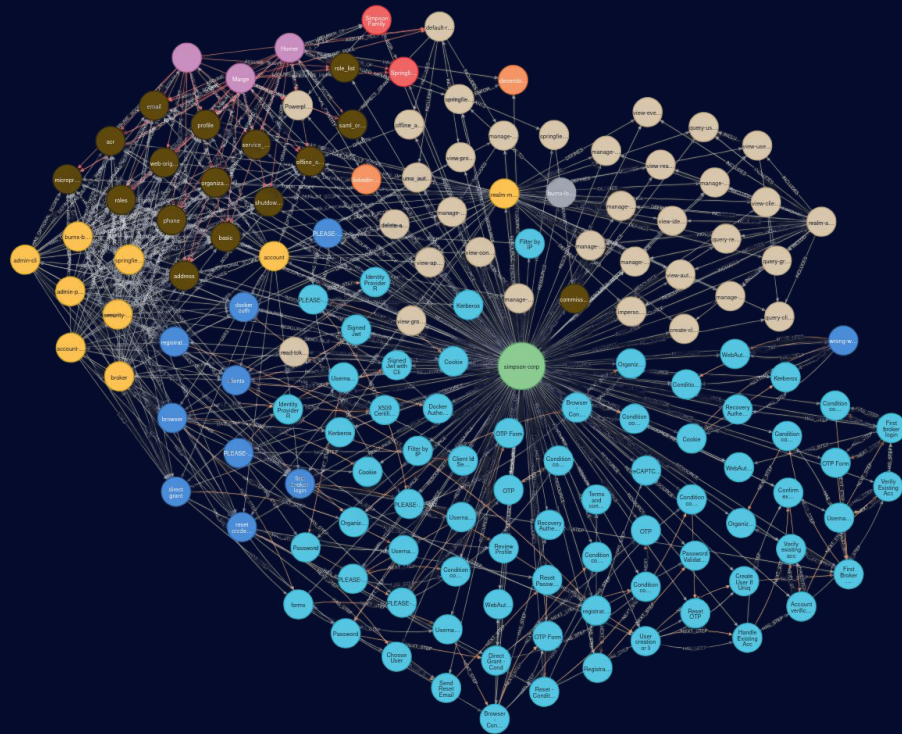
```
1 {
2   "acc.id": "[redacted]"
3   "acc.name": "[redacted]"
4   "admin.arn": "[redacted]"
5   "arn:aws:iam::[redacted]:[redacted]"
6   "rsc.db_name": "[redacted]"
7 }
```

Technical Architecture

- Modular **Python** architecture
- **Extensible** with custom modules
- Global or per-module execution
- Simple deployment with **Docker**
- Easily integrates with any orchestrator
- Visibility through **Neo4j**
- Successive runs, no continuous monitoring



How to with Keycloak



- **Clone** the repository
- **Register an Application** in Keycloak
- **Configure** secrets & URLs (env vars / parameters)
- **Run** Cartography against Keycloak
- **Explore** the graph (Neo4j browser / Cypher queries)

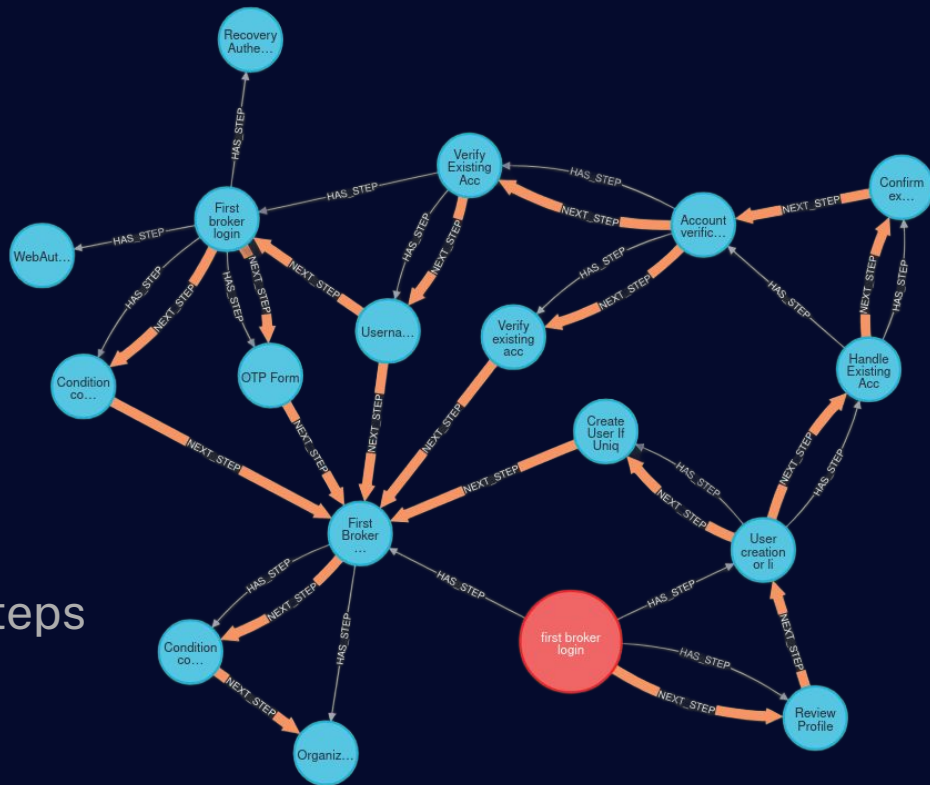
Understanding Authentication Flows in Keycloak

Two types of relations:

- **Composition** links (black)
→ native Keycloak structure.
- **Execution** links (orange)
→ computed by Cartography

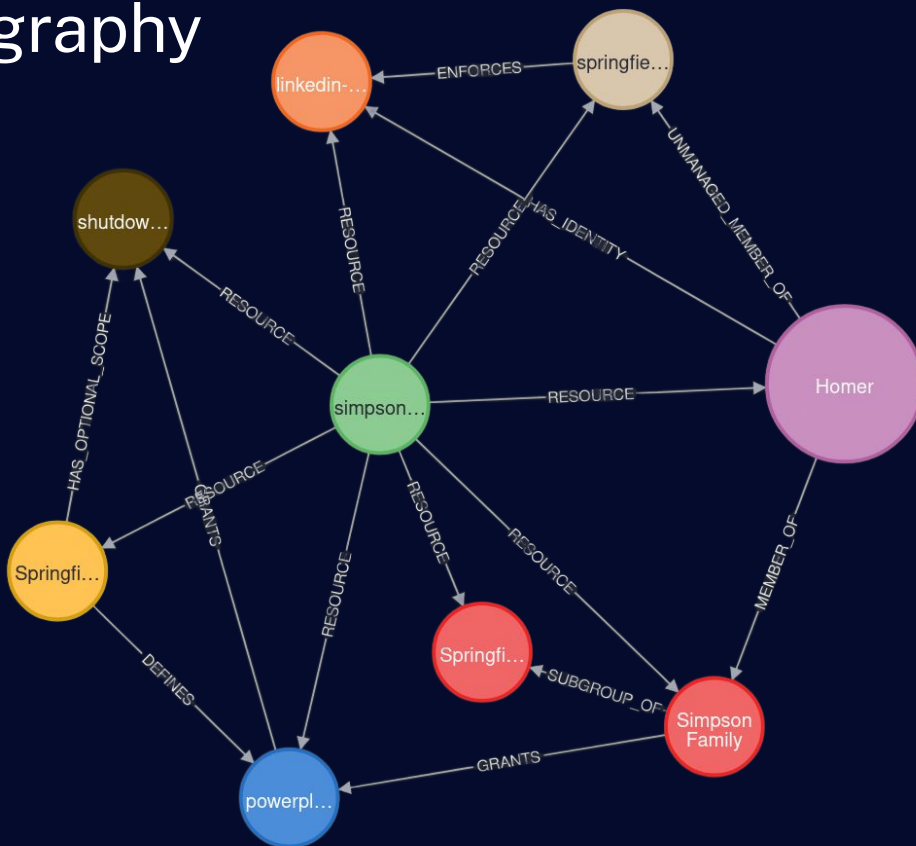
Objectives:

- Detect unreachable executions.
- Identify flows missing critical steps
- Consistency across realms



Visualizing Users in Cartography

- Users are modeled as nodes
- Relations to groups, roles, are explicit
- External IdPs and optional scopes are captured
- Graph view could reveals hidden memberships and privileges



Flattening Inheritance to Reveal Effective Access

- **Composite roles & subgroups flattened**

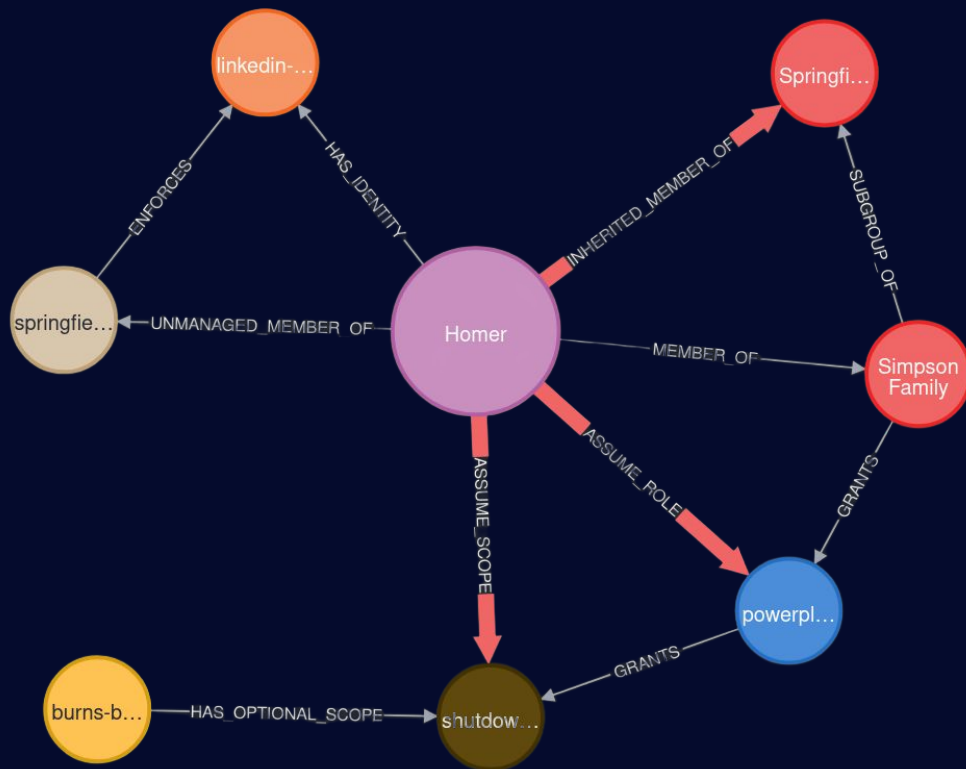
Hidden inheritance is resolved into explicit relationships.

- **Scope propagation**

Scopes assigned to roles and groups are propagated to users.

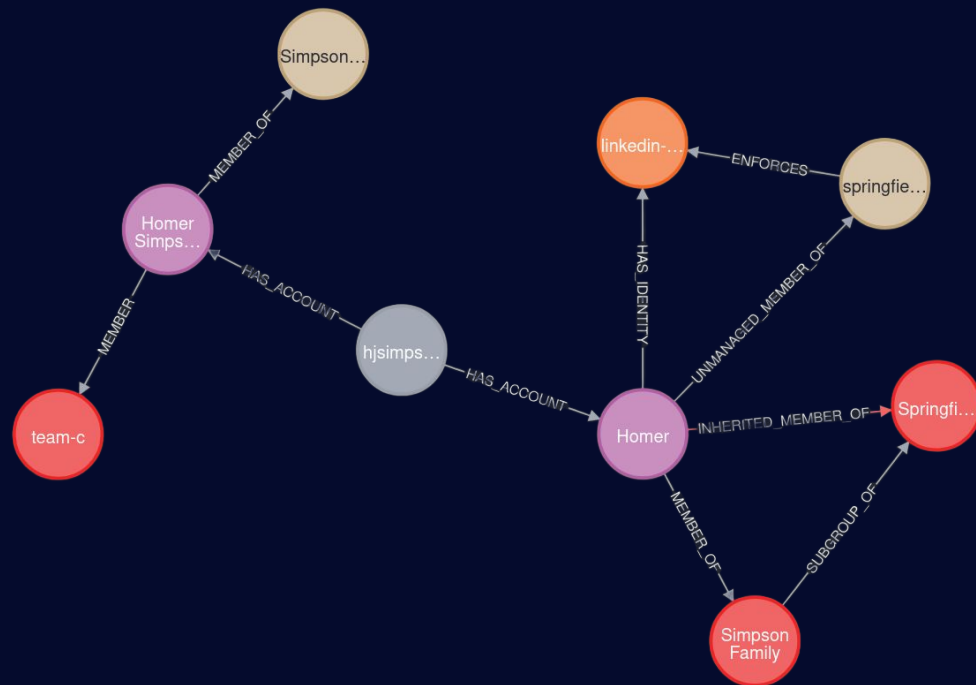
- **Effective access visibility**

We can answer precisely: “Which user can access which scope?”

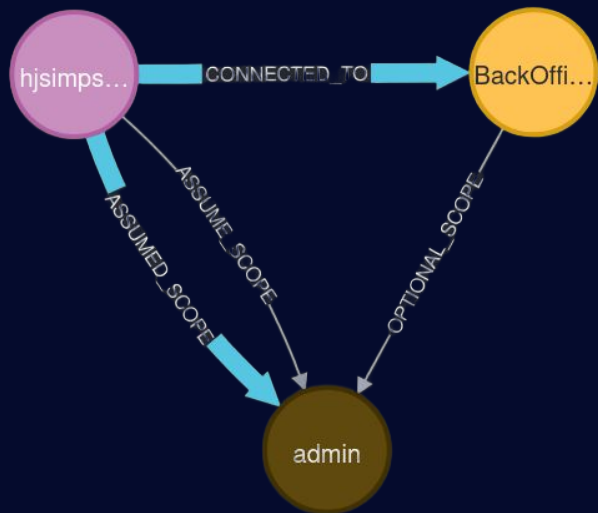


Beyond Keycloak: External Identity Integration

- Ingest identities from multiple providers (e.g., GitHub)
- Cross-check group and role consistency
- Propagate memberships
- Unified graph view of trust relationships



Log ingestion: from theory to reality



Bridging theory and reality:

- What a user can do
- What a user has done

Log and event ingestion:

- Access logs
- Network logs
- Application events

POC in progress for AWS CloudTrail

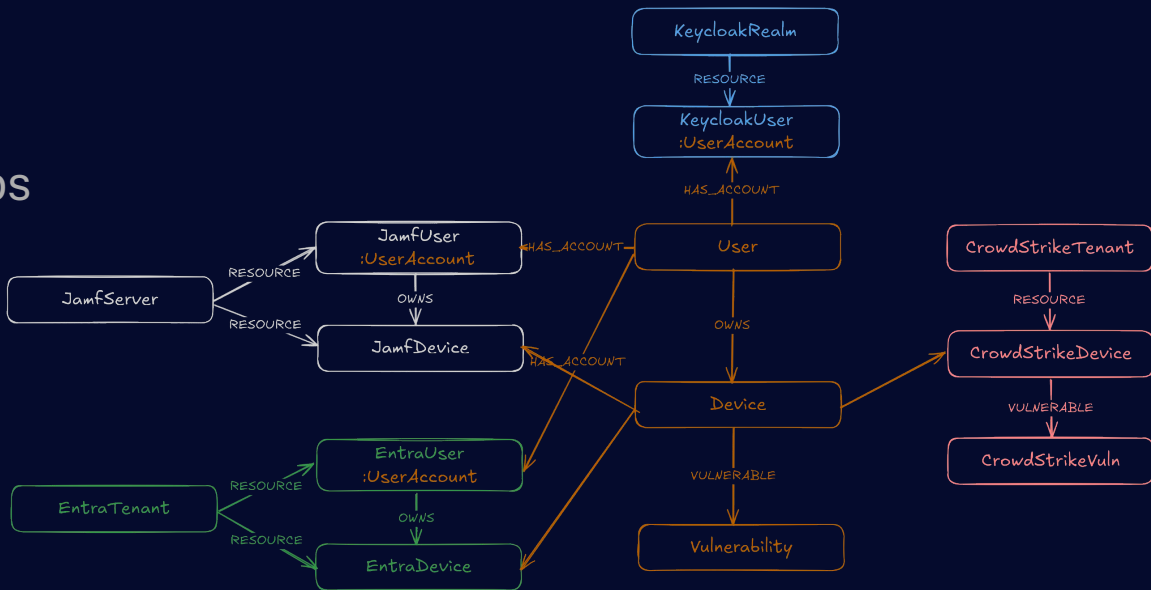
First steps toward an ontology

A lightweight **ontology**:

- Cross-domain concepts
- Standardized relationships

Objectives:

- Share knowledge across domains
- Simplify queries
- Facilitate integration



Bonus: AI fancy stuff

Get the GraphDB Schema before processing the following request:
Give me all gorups of Honer in Keycloak

> Ran `get_neo4j_schema` – cartography-neo4j (MCP Server) ✓

Now let me search for Honer (assuming the user meant "Homer") in the Keycloak data:

> Ran `read_neo4j_cypher` – cartography-neo4j (MCP Server) ✓

Now let me find all the groups that Homer (hjsimpson) belongs to:

> Ran `read_neo4j_cypher` – cartography-neo4j (MCP Server) ✓

Based on the Neo4j database query results, Homer (username: `hjsimpson`) is a member of the following groups in Keycloak:

- 1. Simpson Family**
 - Path: `/Springfield Residents/Simpson Family`
 - Description: "Toh!"
- 2. Springfield Residents**
 - Path: `/Springfield Residents`
 - Description: (empty)

Homer appears to be a member of both a parent group "Springfield Residents" and a subgroup "Simpson Family" within that hierarchy.

Bonus: more AI for buzz

Now give me all scopes Homer can assume

> Ran `read_neo4j_cypher` - cartography-neo4j (MCP Server) ✓

Based on the Neo4j database query, Homer (username: `hjsimpson`) can assume the following scopes in Keycloak:

OpenID Connect Scopes:

1. **acr** - OpenID Connect scope for add acr (authentication context class reference) to the token
2. **address** - OpenID Connect built-in scope: address
3. **basic** - OpenID Connect scope for add all basic claims to the token
4. **email** - OpenID Connect built-in scope: email
5. **microprofile-jwt** - Microprofile - JWT built-in scope
6. **offline_access** - OpenID Connect built-in scope: offline_access
7. **organization** - Additional claims about the organization a subject belongs to
8. **phone** - OpenID Connect built-in scope: phone
9. **profile** - OpenID Connect built-in scope: profile
10. **roles** - OpenID Connect scope for add user roles to the access token
11. **service_account** - Specific scope for a client enabled for service accounts
12. **shutdown-control** - No more power.
13. **web-origins** - OpenID Connect scope for add allowed web origins to the access token

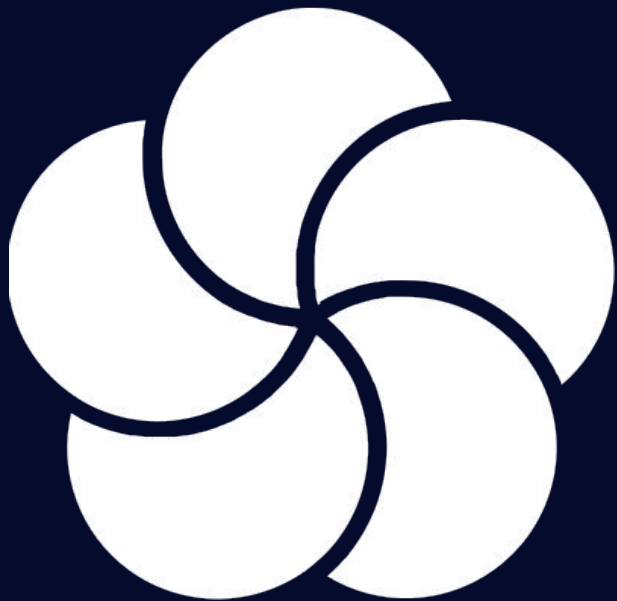
SAML Scopes:

14. **role_list** - SAML role list
15. **saml_organization** - Organization Membership

Homer has access to a comprehensive set of both standard OpenID Connect scopes (like profile, email, phone, address) and some custom scopes including a humorous "shutdown-control" scope with the description "No more power." which seems fitting for Homer Simpson's character!

🔄 ↶ 🍏 📄

Thank you for your attention



Reach us on GitHub and Slack:



www.cartography.dev

Contact me on:



<https://github.com/jychp>



https://x.com/jychp_fr



<https://jychp.medium.com>