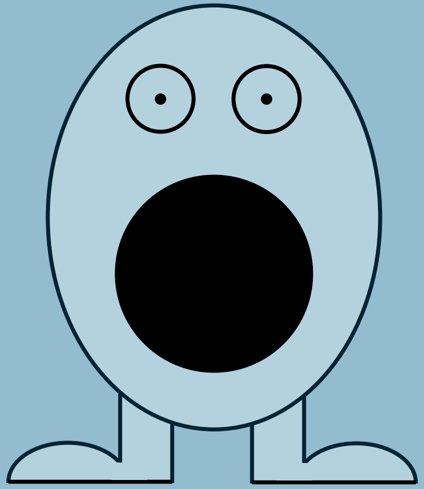
BSides Chicago

Pruning Garden Paths in AWS



Speaker

Evan Perotti
@2xxeformyshirt
2xxe.com

Lead Scientist
Security Risk Advisors
sra.io

Agenda

Background & Prior Art

Design and Goals

Graphs

Neph

Attack Paths

Background

Assessing Cloud Security

- Static checks, “CSPM”
- Graph-based
- Both necessary

Static Checks

PROWLER

CIS Center for Internet Security®

Static Checks

```
EC2 {
    Architecture: x86_64
    ImageId: ami-abc
    MetadataOptions {
        HttpTokens: optional  <-- bad
    }
}
```

Graphs

Cartography

BLOODHOUND
COMMUNITY EDITION

BLOODHOUND
ENTERPRISE

Graphs

```
(User) -- [AdminTo] -> (Device)
             ^
             |
            bad
```

Prior Art

Related Works

▪ Cartography	https://cartography-cncf.github.io/cartography
▪ awspx	https://github.com/WithSecureLabs/awspx/
▪ Pmapper	https://github.com/nccgroup/PMapper/
▪ Apeman	https://github.com/hotnops/apeman/
▪ IAMhounddog	https://github.com/VirtueSecurity/IAMhounddog/

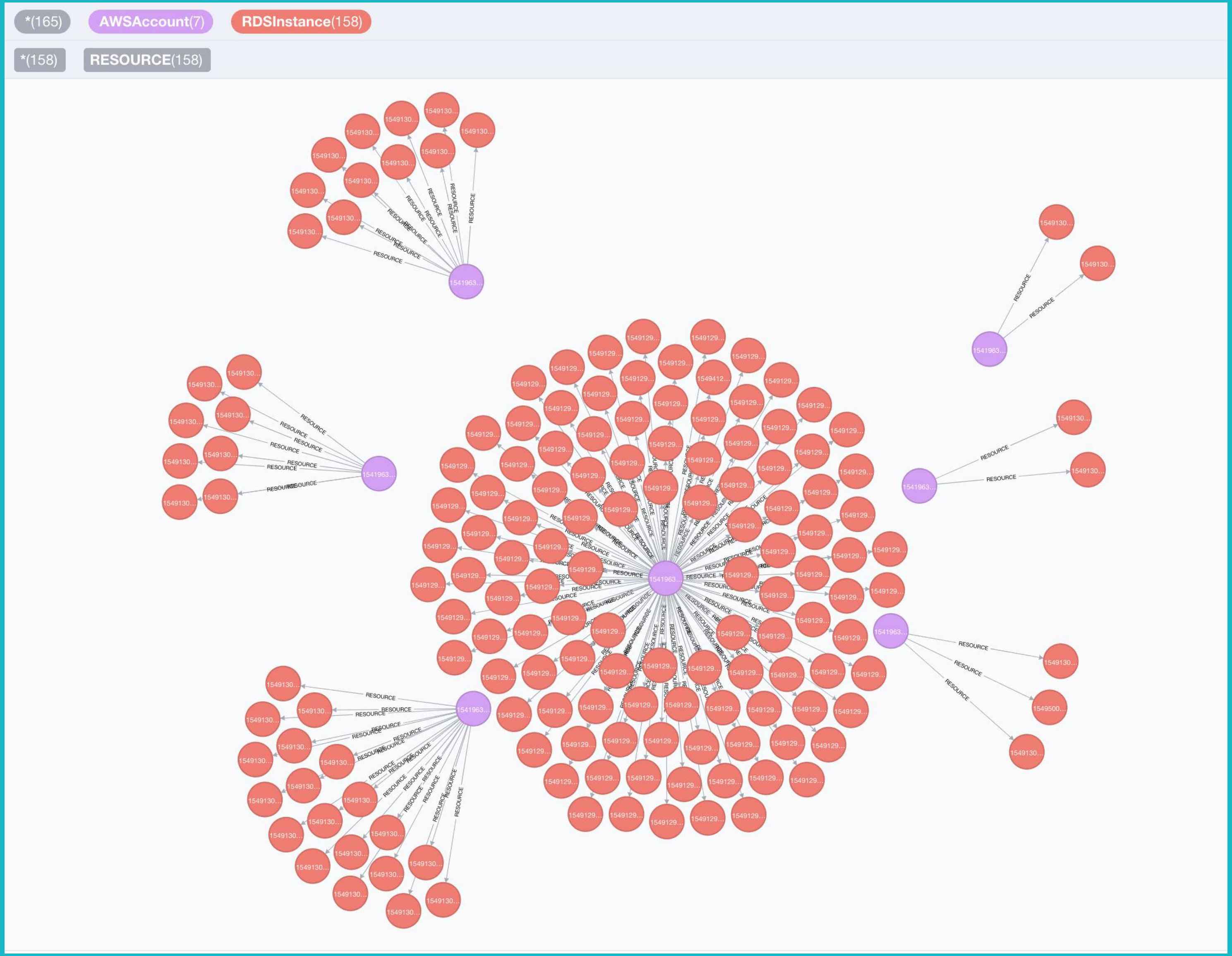
Cartography

Most mature Cloud graphing tool

Supports 30+ providers (SaaS, IaaS, etc)

Extensible (ish)

https://cartography-cncf.github.io/cartography/



https://cartography-cncf.github.io/cartography/usage/tutorial.html

Motivations & Goals

Motivations

- Reviewed existing tools and approaches
- Nothing fit well into an ideal **workflow**
- Identified several common limitations

Limitation: IAM

- Relationships based on **naïve** IAM analysis
- Do not consider **full** scope of permissions context

Limitation: Resources

- Only **small** subset of resources supported
- Adding new resources requires forking

Limitation: Analysis

- Analysis often **one and done**
- Adding new resources requires forking

Goals

- Map AWS attack **paths**
- Evaluate **complex** permission contexts
- Avoid **polluting** the graph
- **Extensible** to support new resources, analysis
- **Human**-in-the-loop analysis

- Ultimately **support** workflow not replace it; **rapid** iterations

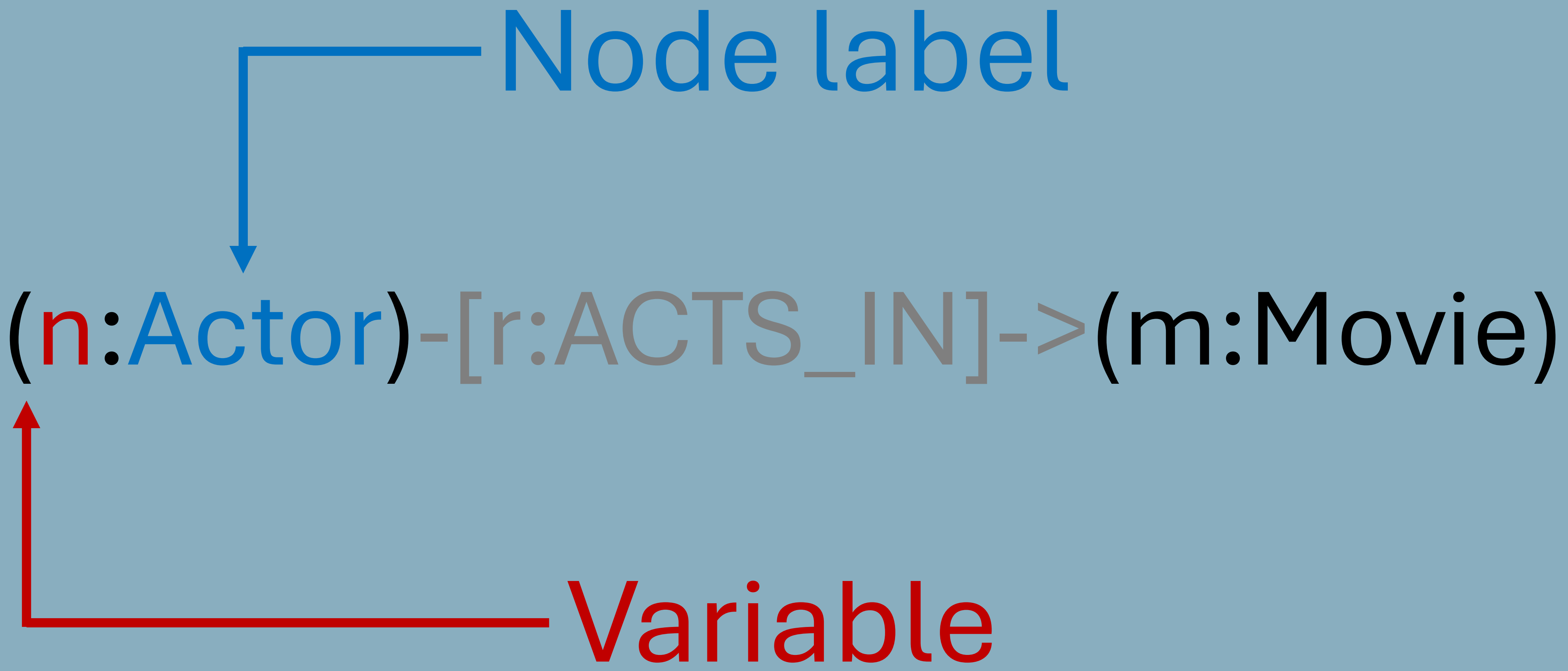
Design

Detour: Neo4j + Cypher

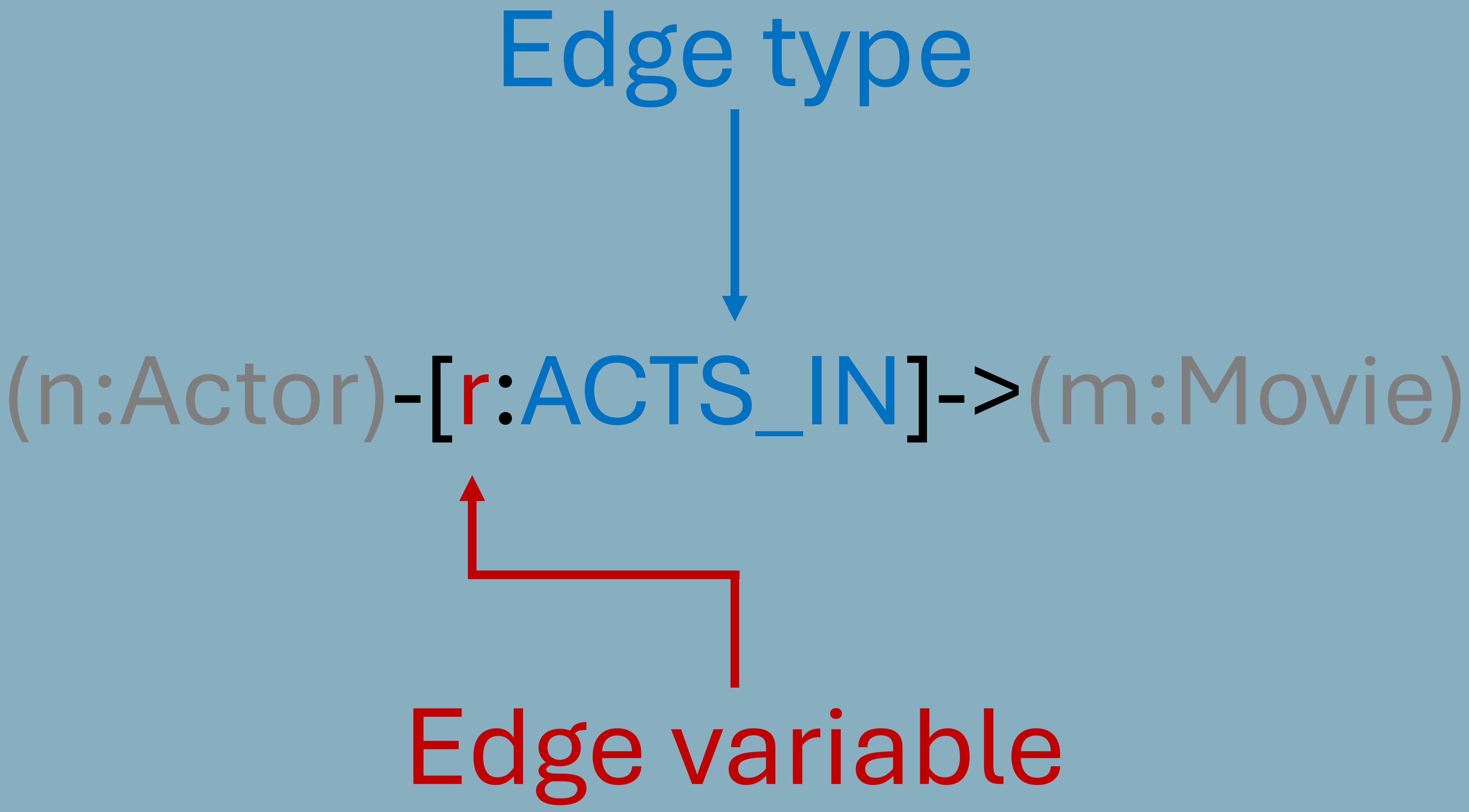
- Neo4j: popular graph database
- Data stored as nodes
- Nodes related via edges

- Cypher: graph query language

```
(n:Actor)-[r:ACTS_IN]->(m:Movie)
```



```
(n:Actor)-[r:ACTS_IN]->(m:Movie)
```



```
(n:Actor)-[r:ACTS_IN]->(m:Movie)
```

```
(n:Actor)-[r:ACTS_IN]->(m:Movie)
```

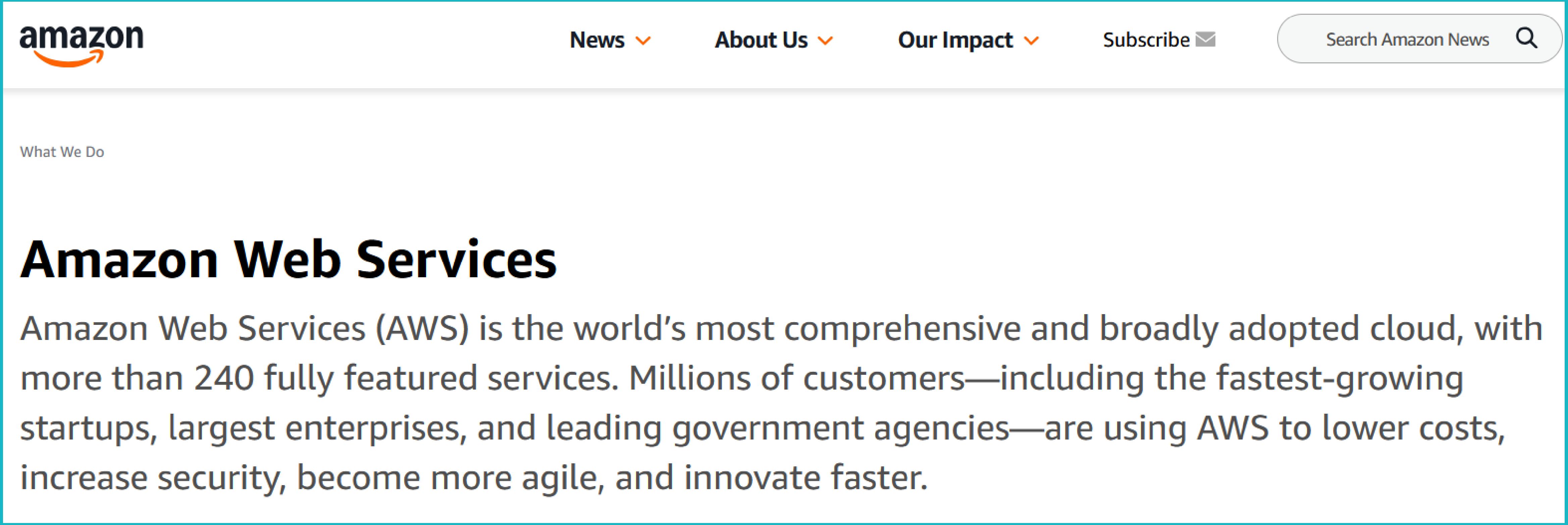
Edge direction

AWS Attack Paths

- Traditional privilege escalation
- Action on objectives
- Cross-account permissions
- Service abuse

IAM Permissions

AWS IAM underpins everything

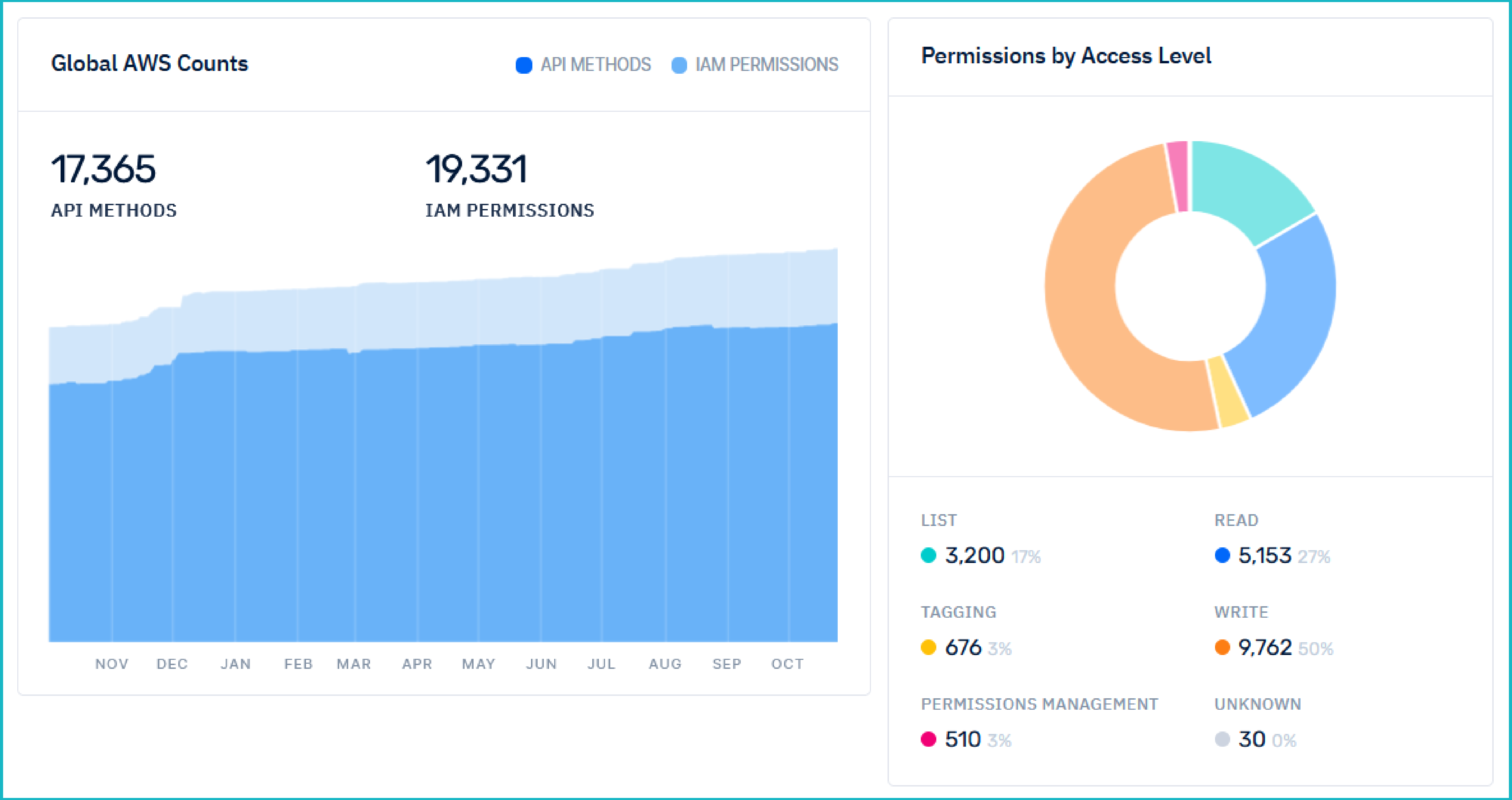


What We Do

Amazon Web Services

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud, with more than 240 fully featured services. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, increase security, become more agile, and innovate faster.

https://www.aboutamazon.com/what-we-do/amazon-web-services



Global AWS Counts

● API METHODS ● IAM PERMISSIONS

17,365
API METHODS

19,331
IAM PERMISSIONS

Permissions by Access Level

LIST	READ
● 3,200 17%	● 5,153 27%
TAGGING	WRITE
● 676 3%	● 9,762 50%
PERMISSIONS MANAGEMENT	UNKNOWN
● 510 3%	● 30 0%

https://aws.permissions.cloud/

IAM Permissions

Driving question:

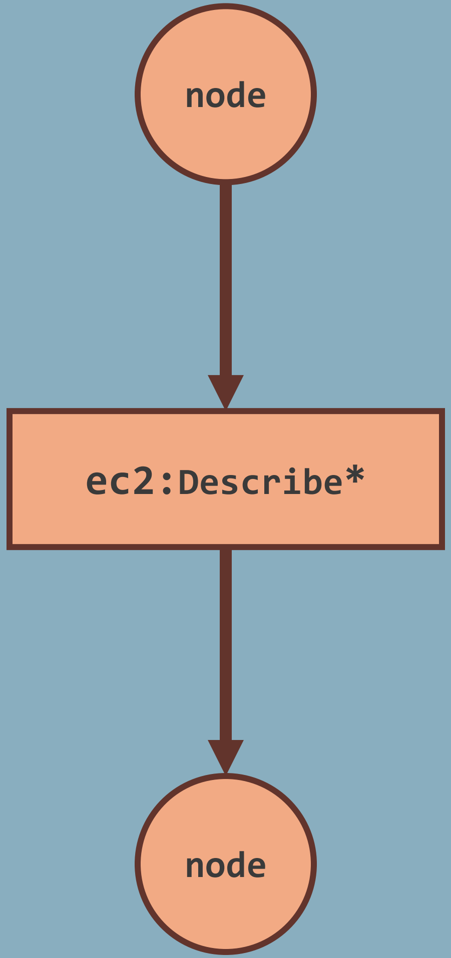
How to do you represent permissions?

An IAM policy

```
{
     "Version": "2012-10-17",
     "Statement": [{
          "Effect": "Allow",
          "Action": [
               “ec2:Describe*",
          ],
          "Resource": "*"
     }]
}
```

1 statement = 1 path

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            “ec2:Describe*",
        ],
        "Resource": "*"
    }]
}
```

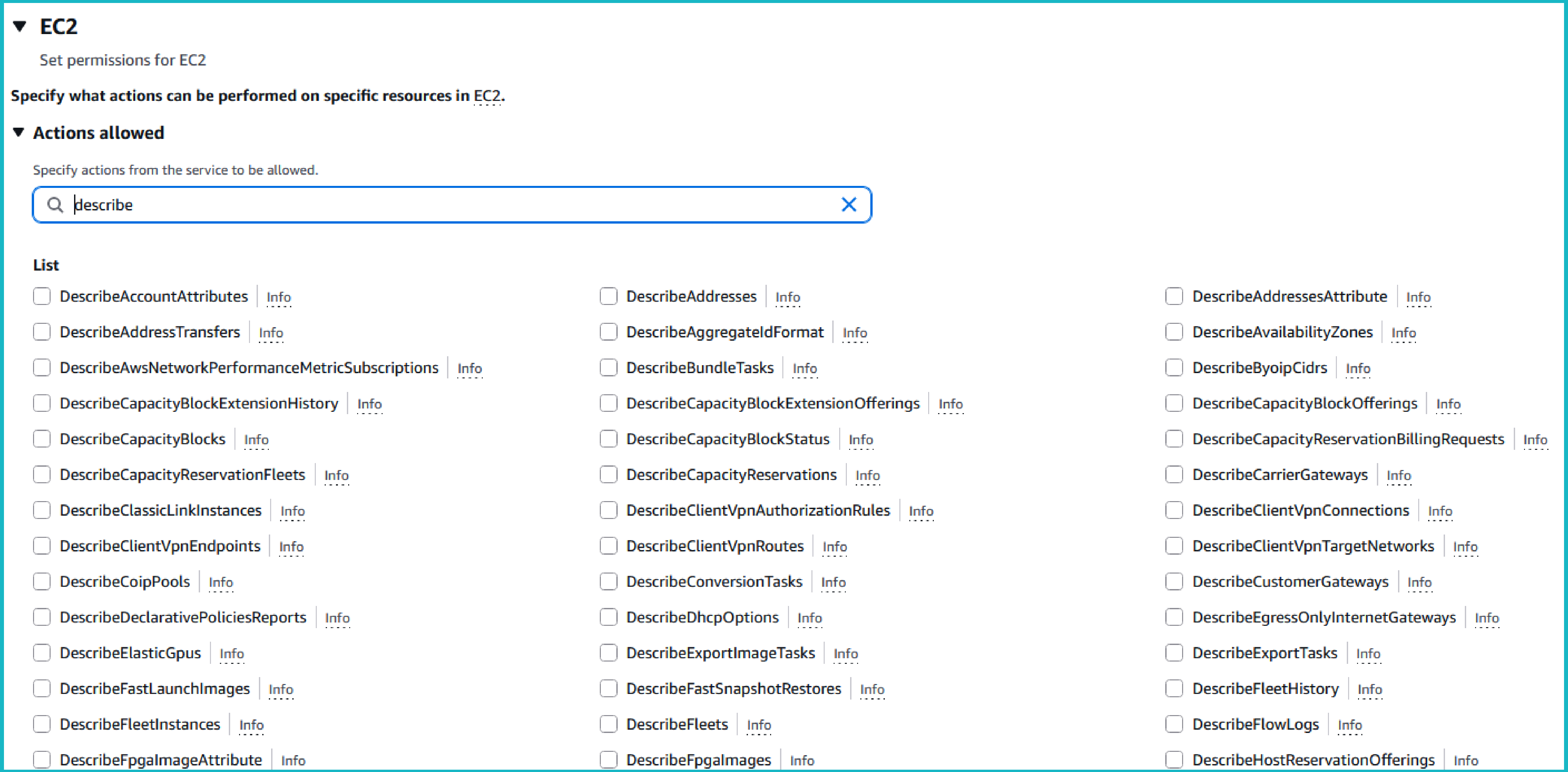


1 statement = 1 path

Wildcard permissions + resources

```
  Action = ec2:Describe*
  Resource = "*"
```

Expand the wildcards?



▼ EC2

Set permissions for EC2

Specify what actions can be performed on specific resources in EC2.

▼ **Actions allowed**

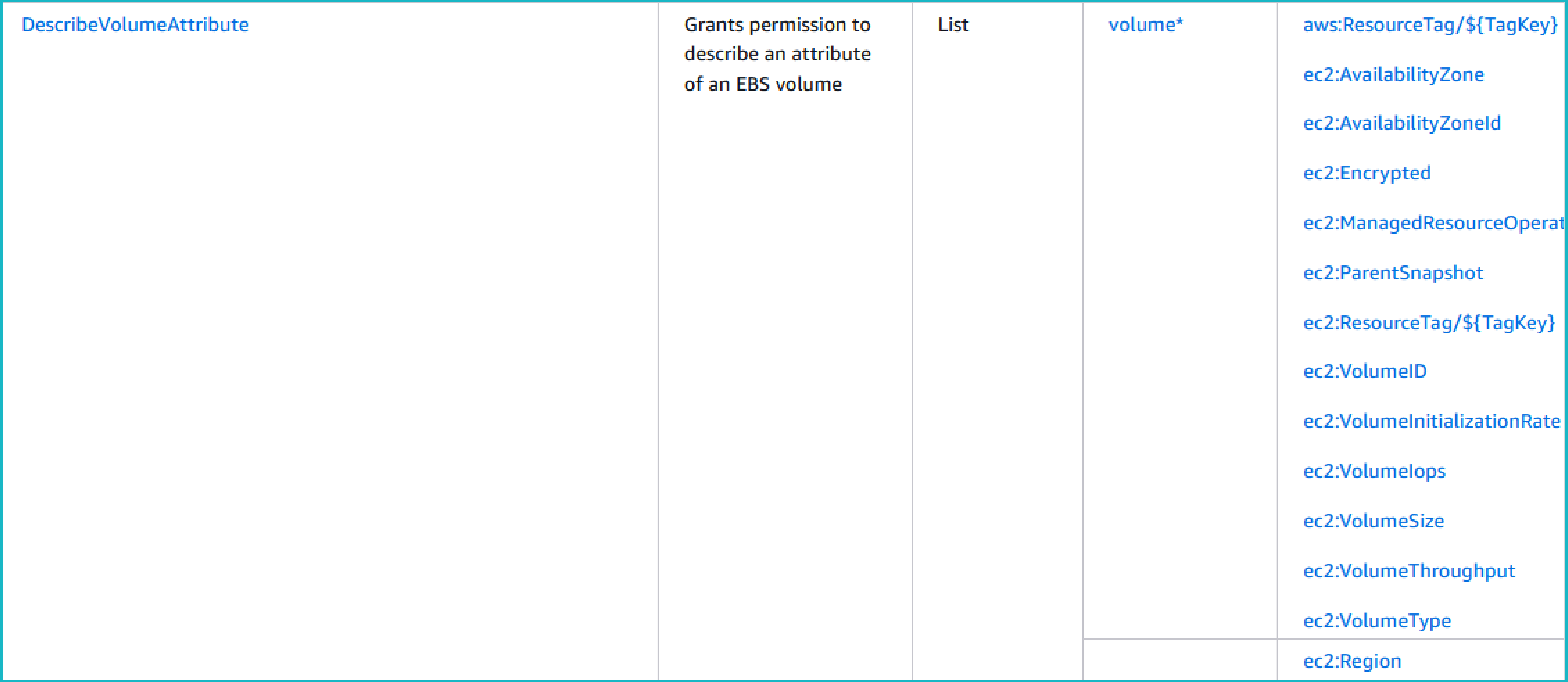
Specify actions from the service to be allowed.

describe

List

☐ DescribeAccountAttributes | Info	☐ DescribeAddresses | Info	☐ DescribeAddressesAttribute | Info
☐ DescribeAddressTransfers | Info	☐ DescribeAggregateIdFormat | Info	☐ DescribeAvailabilityZones | Info
☐ DescribeAwsNetworkPerformanceMetricSubscriptions | Info	☐ DescribeBundleTasks | Info	☐ DescribeByoipCidrs | Info
☐ DescribeCapacityBlockExtensionHistory | Info	☐ DescribeCapacityBlockExtensionOfferings | Info	☐ DescribeCapacityBlockOfferings | Info
☐ DescribeCapacityBlocks | Info	☐ DescribeCapacityBlockStatus | Info	☐ DescribeCapacityReservationBillingRequests | Info
☐ DescribeCapacityReservationFleets | Info	☐ DescribeCapacityReservations | Info	☐ DescribeCarrierGateways | Info
☐ DescribeClassicLinkInstances | Info	☐ DescribeClientVpnAuthorizationRules | Info	☐ DescribeClientVpnConnections | Info
☐ DescribeClientVpnEndpoints | Info	☐ DescribeClientVpnRoutes | Info	☐ DescribeClientVpnTargetNetworks | Info
☐ DescribeCoipPools | Info	☐ DescribeConversionTasks | Info	☐ DescribeCustomerGateways | Info
☐ DescribeDeclarativePoliciesReports | Info	☐ DescribeDhcpOptions | Info	☐ DescribeEgressOnlyInternetGateways | Info
☐ DescribeElasticGpus | Info	☐ DescribeExportImageTasks | Info	☐ DescribeExportTasks | Info
☐ DescribeFastLaunchImages | Info	☐ DescribeFastSnapshotRestores | Info	☐ DescribeFleetHistory | Info
☐ DescribeFleetInstances | Info	☐ DescribeFleets | Info	☐ DescribeFlowLogs | Info
☐ DescribeFpgaImageAttribute | Info	☐ DescribeFpgaImages | Info	☐ DescribeHostReservationOfferings | Info

EC2 Describe* actions (almost 200 total)



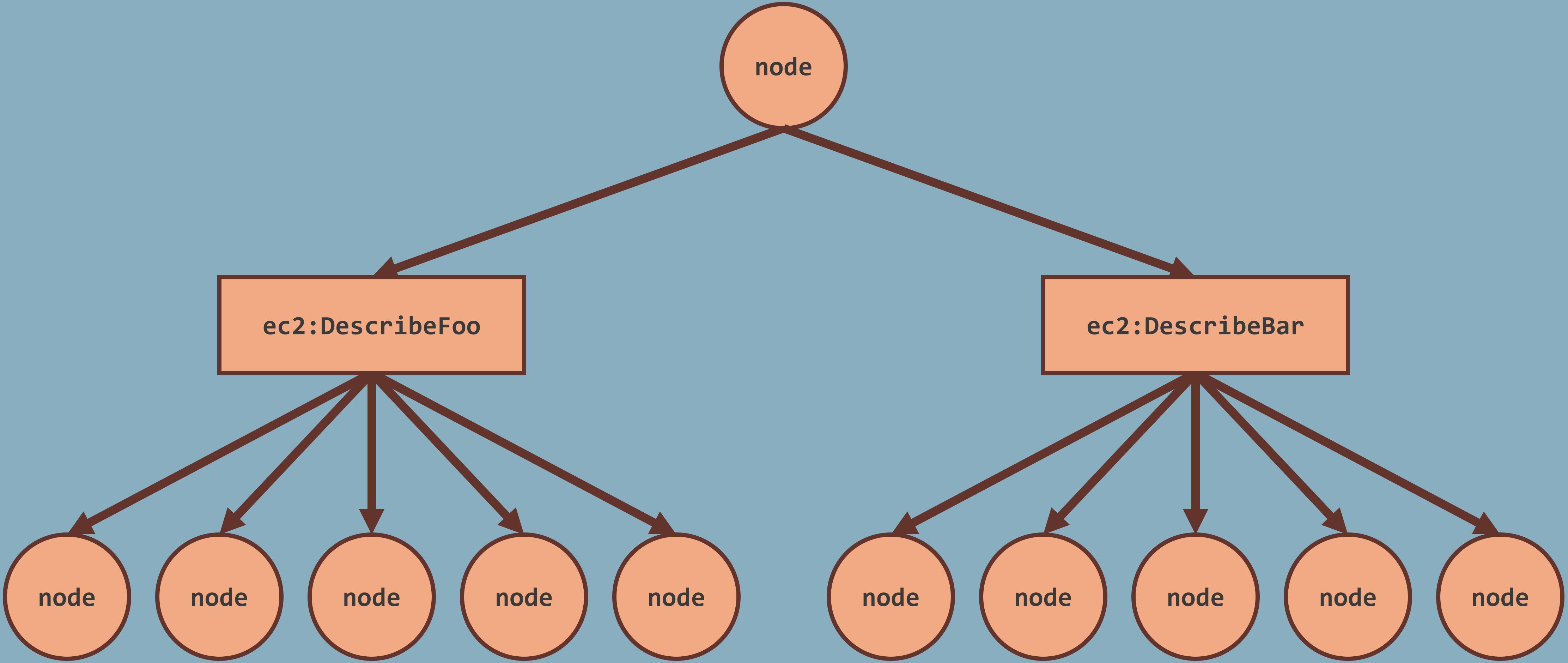
DescribeVolumeAttribute	Grants permission to describe an attribute of an EBS volume	List	volume*	aws:ResourceTag/${TagKey} ec2:AvailabilityZone ec2:AvailabilityZoneId ec2:Encrypted ec2:ManagedResourceOperat ec2:ParentSnapshot ec2:ResourceTag/${TagKey} ec2:VolumeID ec2:VolumeInitializationRate ec2:VolumeIops ec2:VolumeSize ec2:VolumeThroughput ec2:VolumeType
				ec2:Region

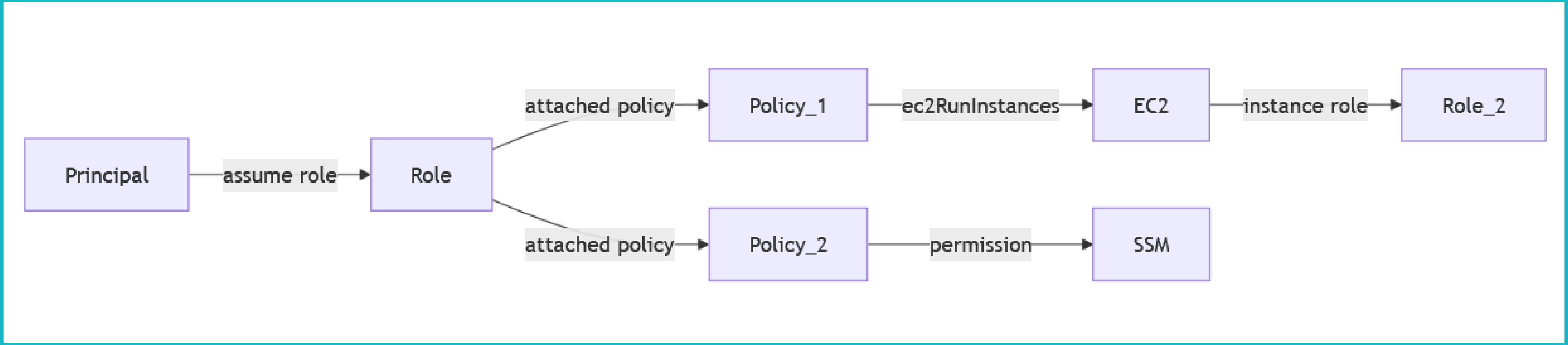
EC2 DescibeVolumeAttribute resources and conditions

Expanding Wildcards

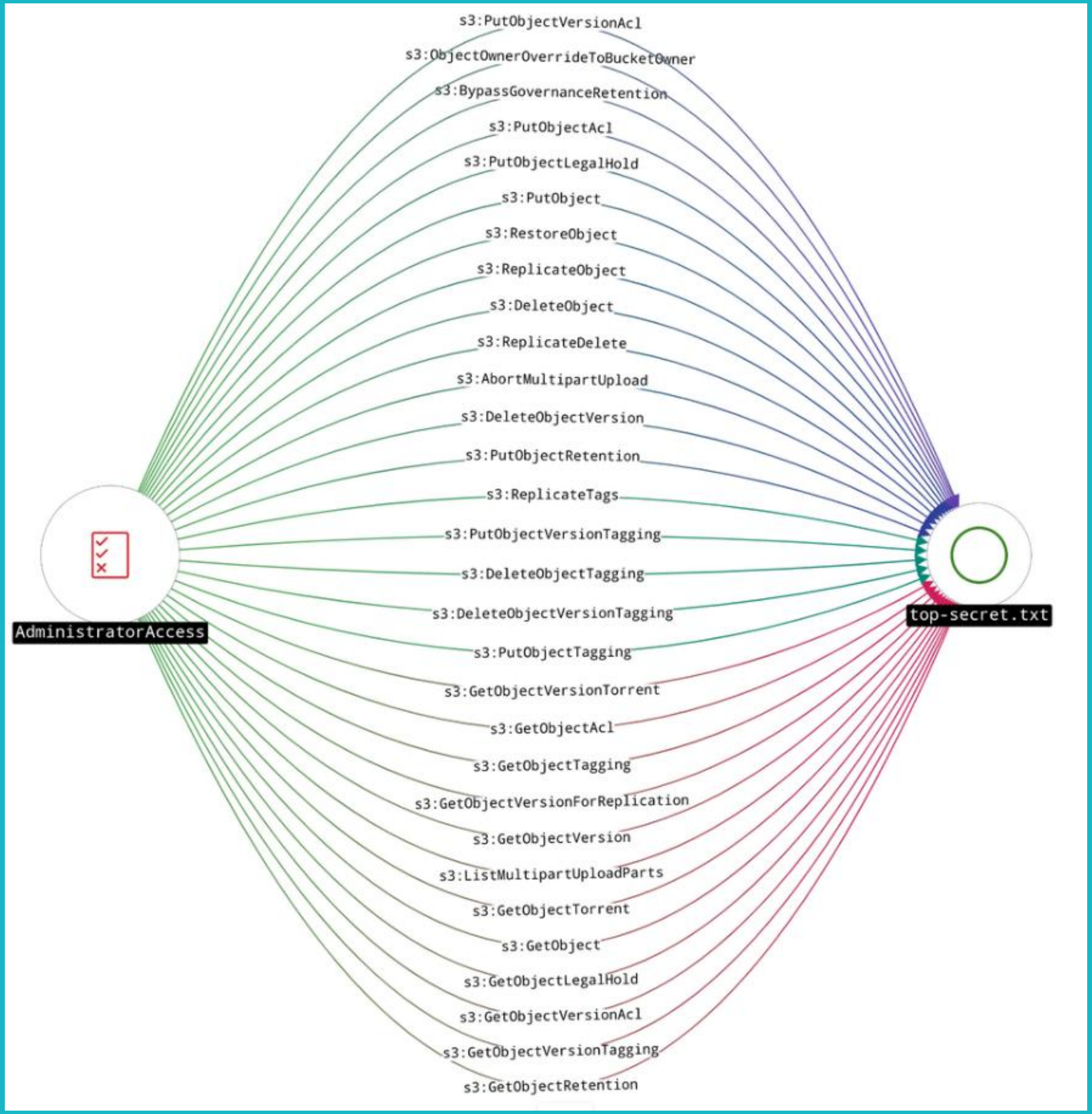
- New AWS services, actions
- New customer resources
- Very large environments
- Multiple accounts

Expanded Graph





IAMhounddog example paths



awspx example paths from policy to S3 object

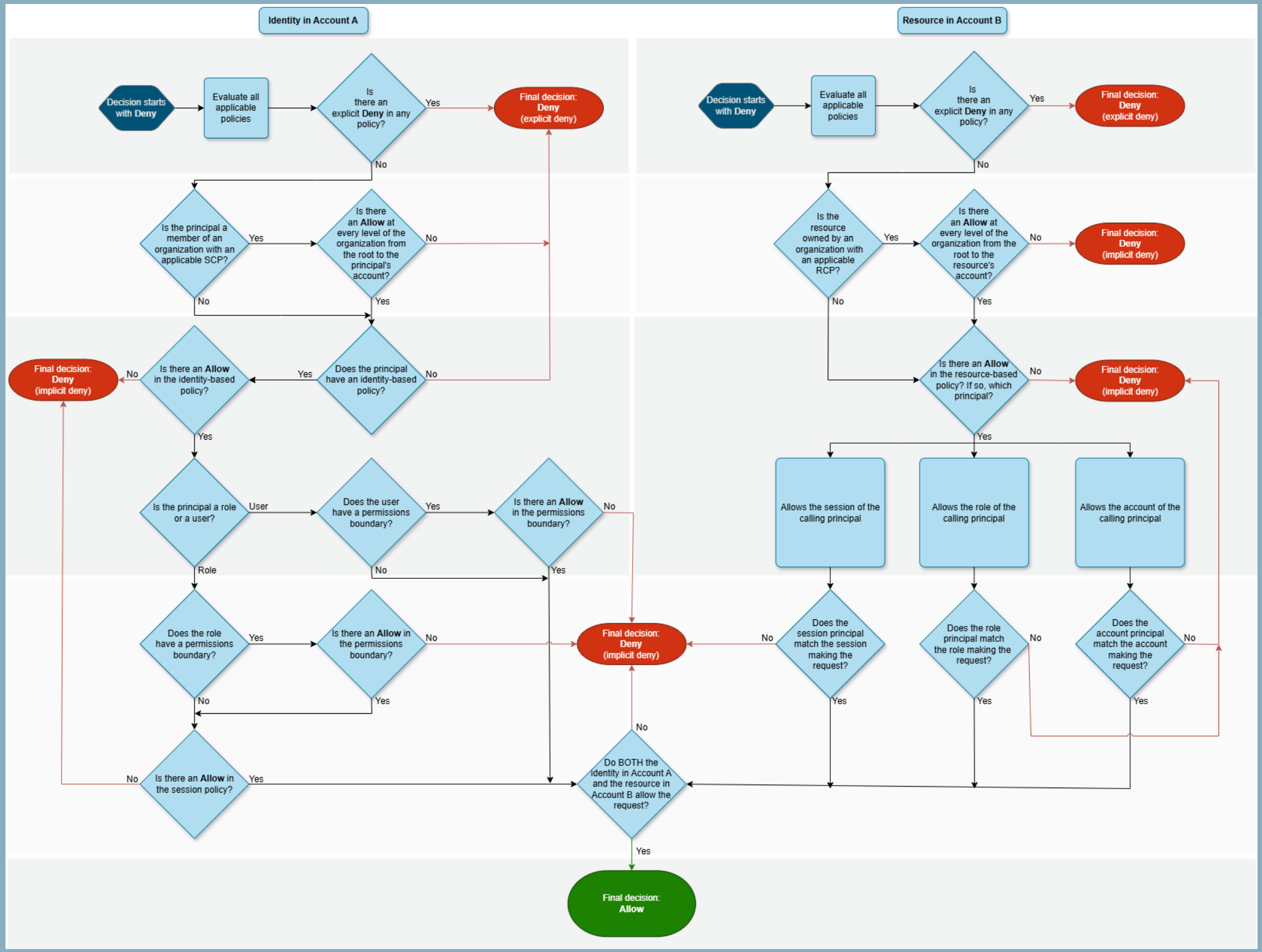
An IAM policy 2

```
{
    "Version": "2012-10-17",
    "Statement": [{
        …

        "Effect": “Deny",
         "Action": [
              “ec2:DescribeInstances",
         ],
         "Resource": "*"
    }]
}
```

Elephants in the room

- Explicit policy denies
- Deny via SCP/RCP
- Permission boundaries
- Resource policies
- ...

Elephants in the room



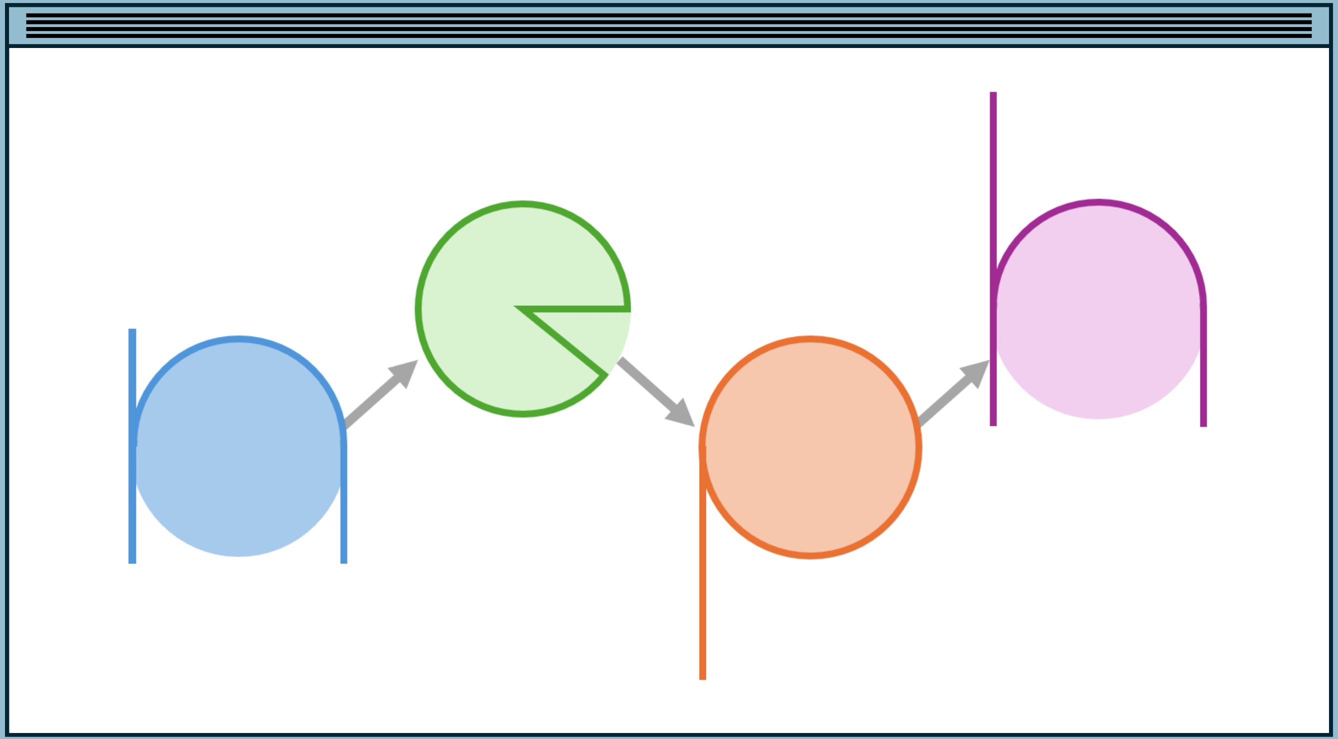
Identity in Account A
Resource in Account B
Decision starts with Deny
Evaluate all applicable policies
Is there an explicit Deny in any policy?
Yes
No
Final decision: Deny (explicit deny)
Is the principal a member of an organization with an applicable SCP?
Is there an Allow at every level of the organization from the root to the principal's account?
Final decision: Deny (implicit deny)
Is there an Allow in the identity-based policy?
Does the principal have an identity-based policy?
Is the principal a role or a user?
User
Role
Does the user have a permissions boundary?
Is there an Allow in the permissions boundary?
Does the role have a permissions boundary?
Is there an Allow in the session policy?
Do BOTH the identity in Account A and the resource in Account B allow the request?
Final decision: Allow
Is the resource owned by an organization with an applicable RCP?
Is there an Allow at every level of the organization from the root to the resource's account?
Is there an Allow in the resource-based policy? If so, which principal?
Allows the session of the calling principal
Allows the role of the calling principal
Allows the account of the calling principal
Does the session principal match the session making the request?
Does the role principal match the role making the request?
Does the account principal match the account making the request?

IAM Permissions

How to do you represent permissions and account for:

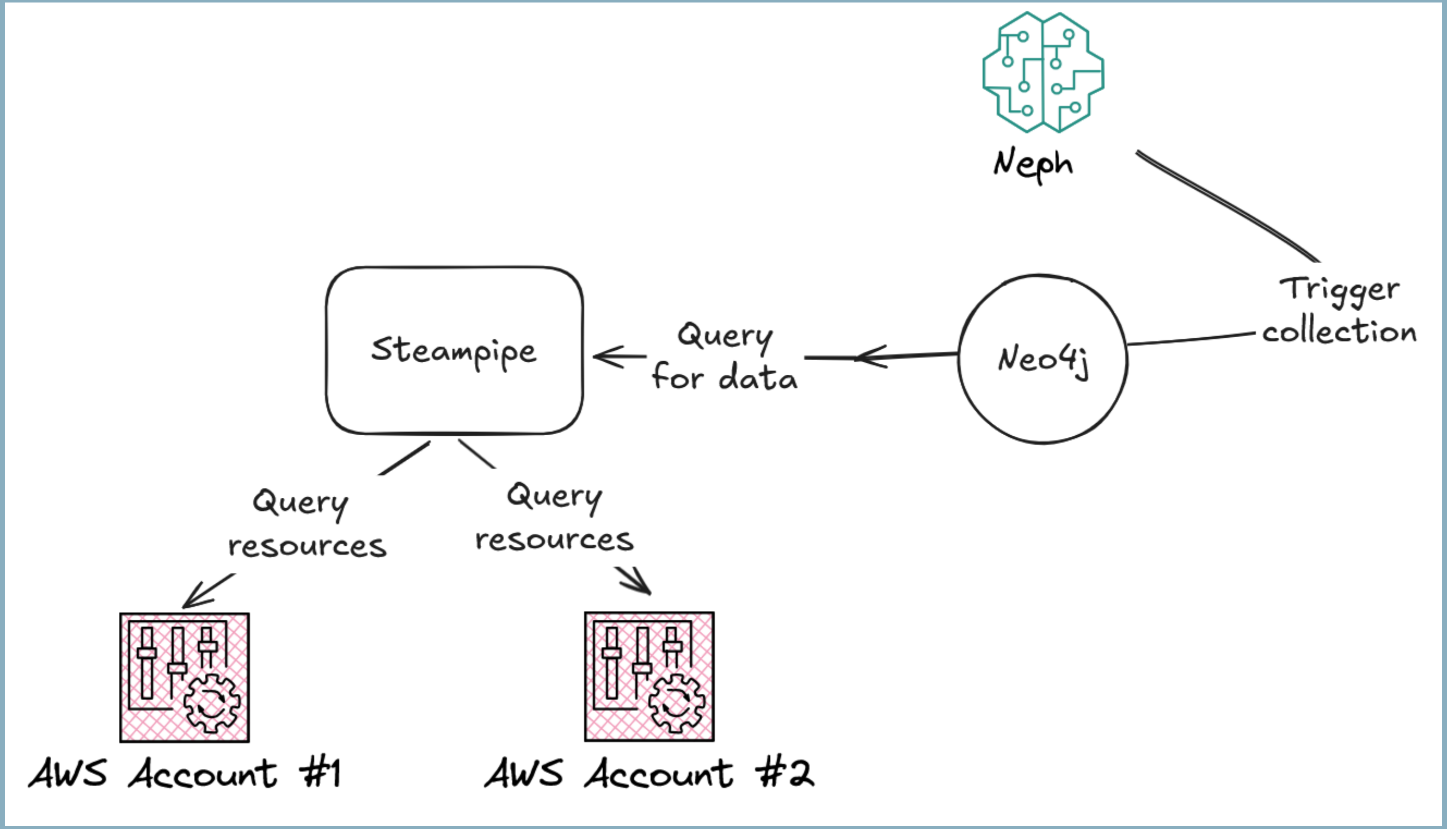
- wildcards
- other policies
- explicit denies
- conditions
- mixed resources

¯_(ツ)_/¯



Architecture pt 1

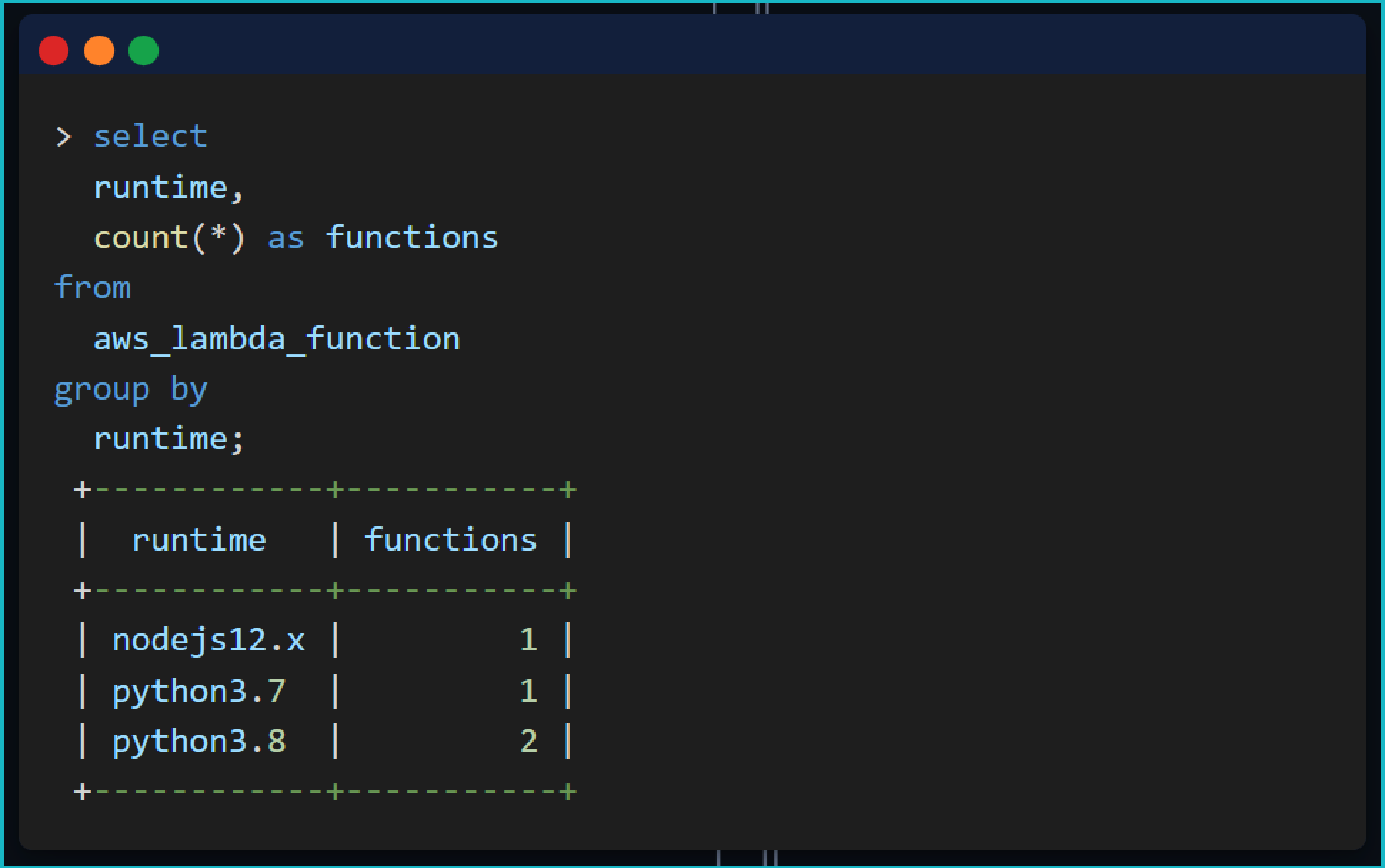
Step 1: get data



Steampipe

- SQL for your cloud resources
  > select * from aws_ec2_instances;

- Handles all data collection, multi-account, regions, errors

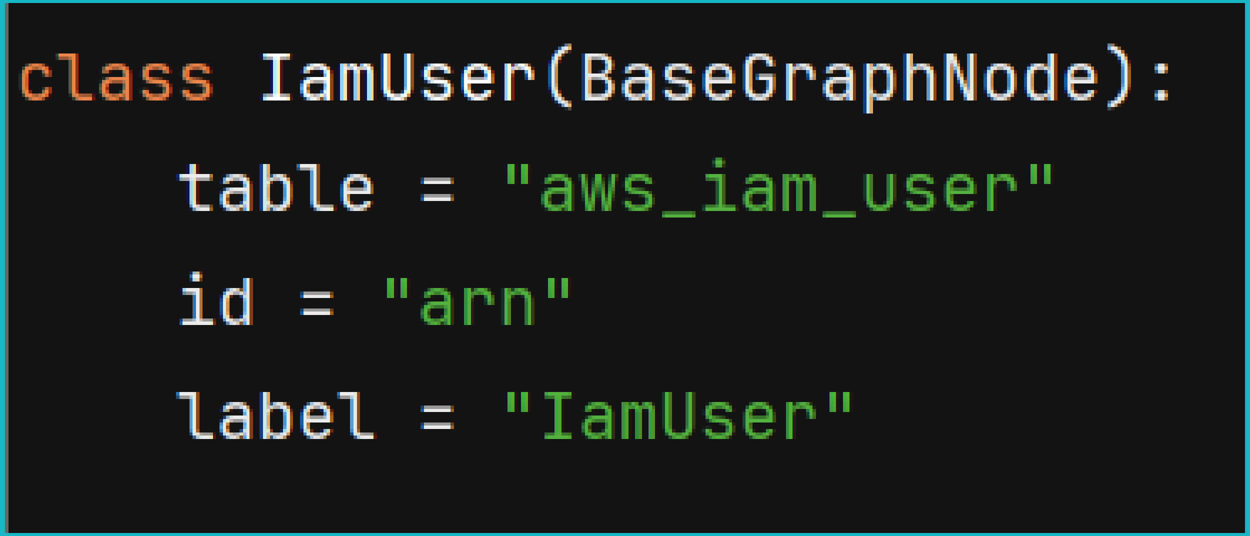


```
> select
  runtime,
  count(*) as functions
from
  aws_lambda_function
group by
  runtime;
 +------------+-----------+
 |  runtime   | functions |
 +------------+-----------+
 | nodejs12.x |         1 |
 | python3.7  |         1 |
 | python3.8  |         2 |
 +------------+-----------+
```

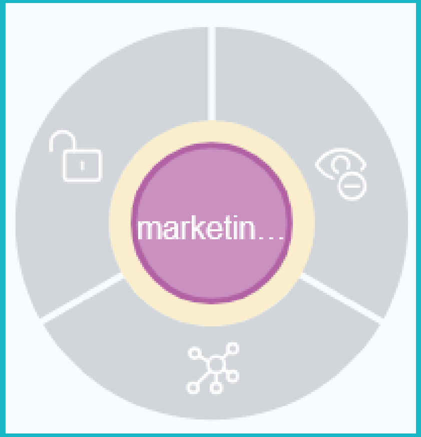
https://steampipe.io/

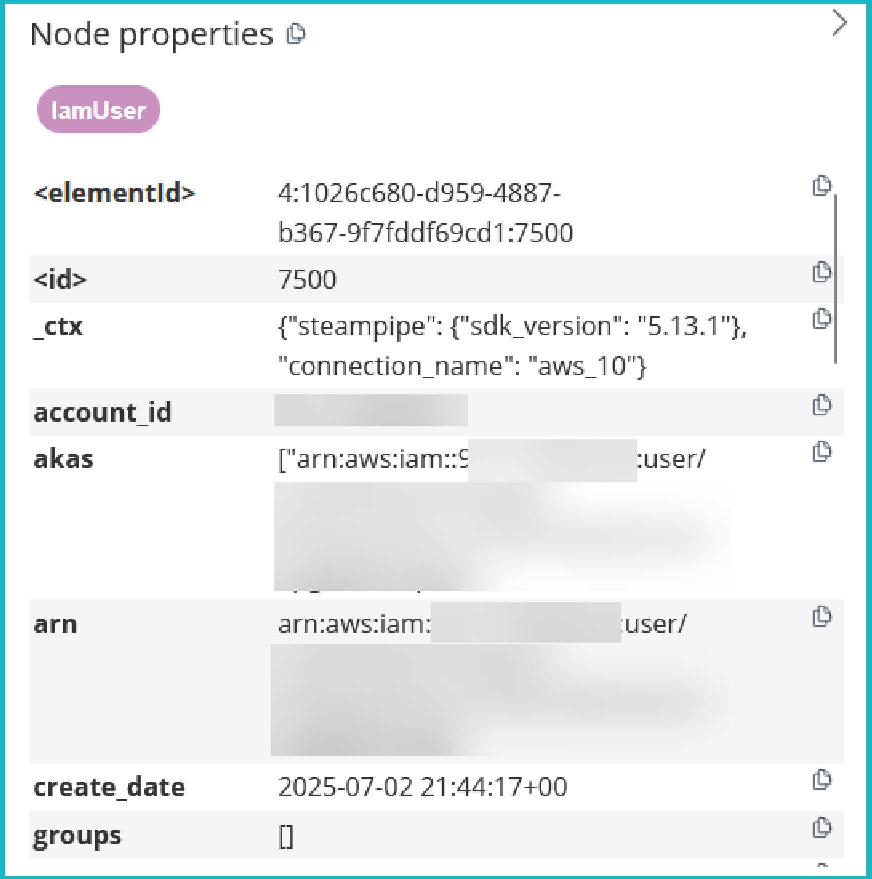
Steampipe

- Can expose PostgreSQL listener
- Neo4j supports JDBC
- Direct ingestion into Neo4j via JDBC



```python
class IamUser(BaseGraphNode):
    table = "aws_iam_user"
    id = "arn"
    label = "IamUser"
```




Node properties

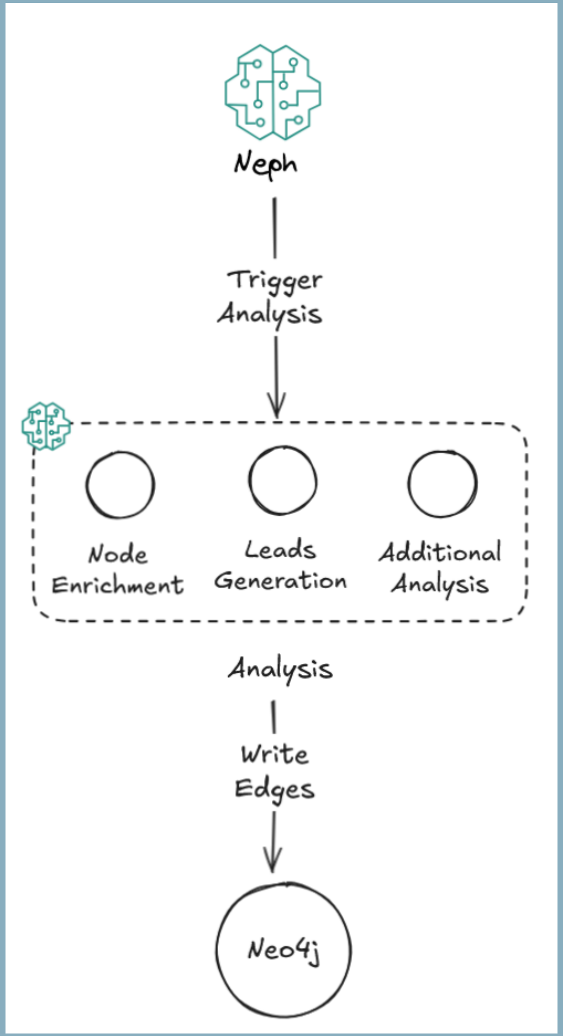
IamUser

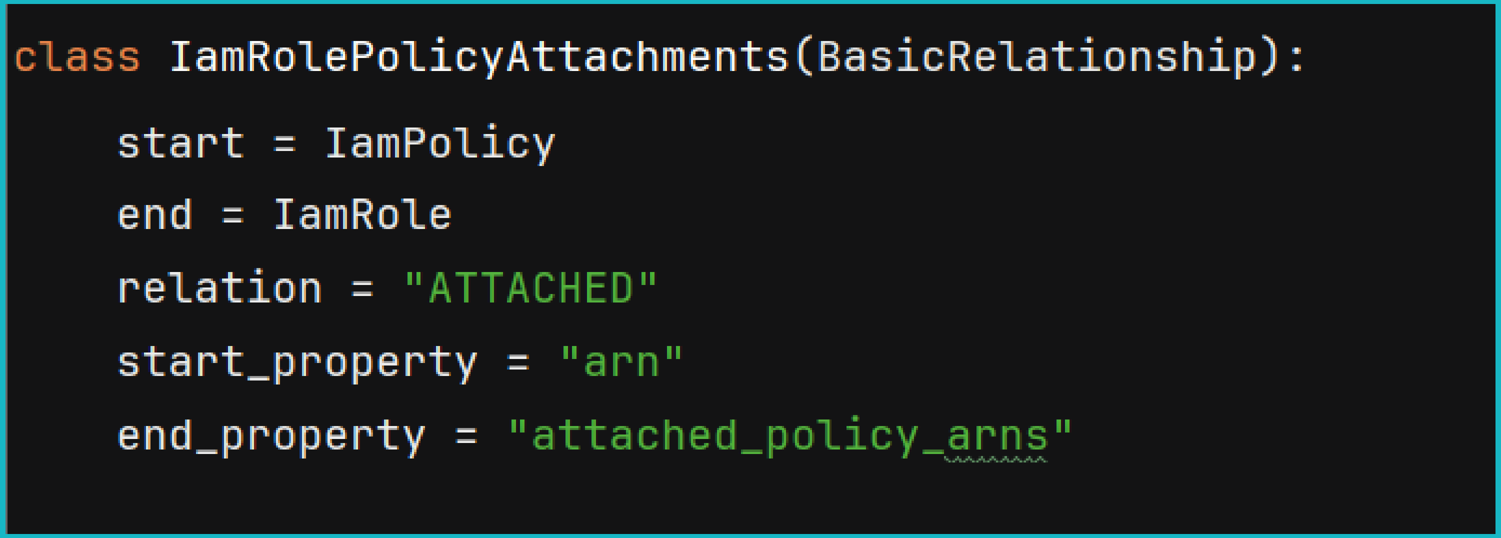
<elementId>	4:1026c680-d959-4887-b367-9f7fddf69cd1:7500
<id>	7500
_ctx	{"steampipe": {"sdk_version": "5.13.1"}, "connection_name": "aws_10"}
account_id	[redacted]
akas	["arn:aws:iam::9[redacted]:user/ [redacted]
arn	arn:aws:iam:[redacted]:user/ [redacted]
create_date	2025-07-02 21:44:17+00
groups	[]

Limitation: Resources ✓

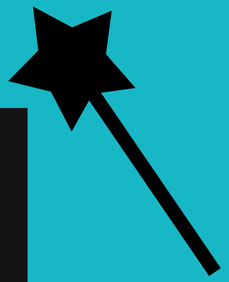
Architecture pt 2

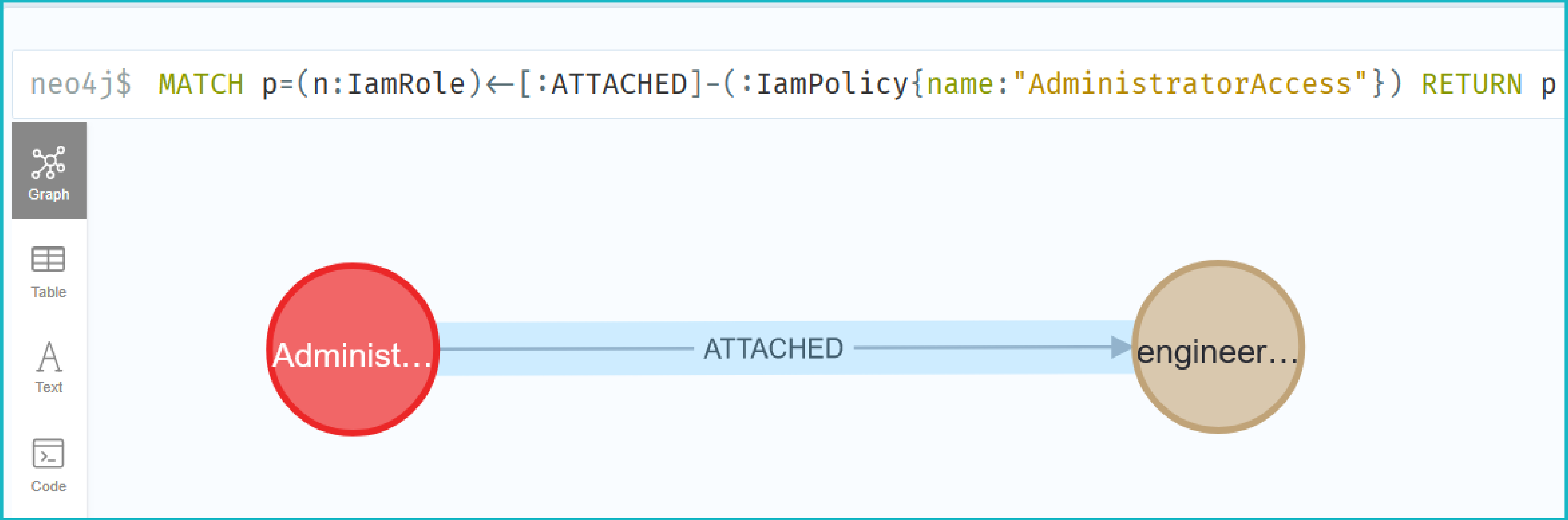
- Step 2: Analyze data
- Add custom fields to nodes
- Create edges, “leads”



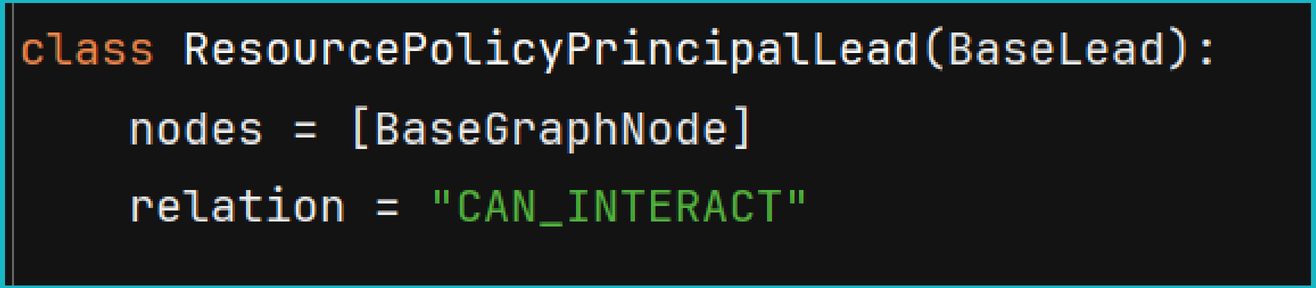


```python
class IamRolePolicyAttachments(BasicRelationship):
    start = IamPolicy
    end = IamRole
    relation = "ATTACHED"
    start_property = "arn"
    end_property = "attached_policy_arns"
```

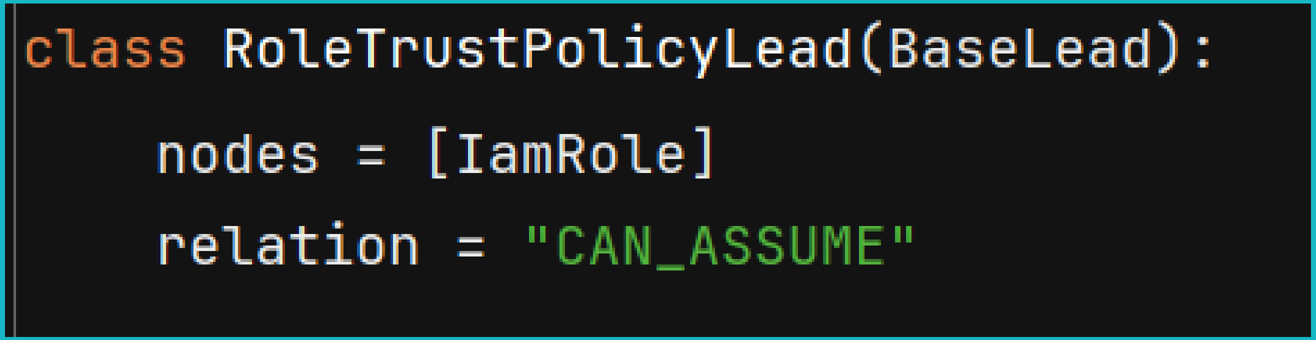




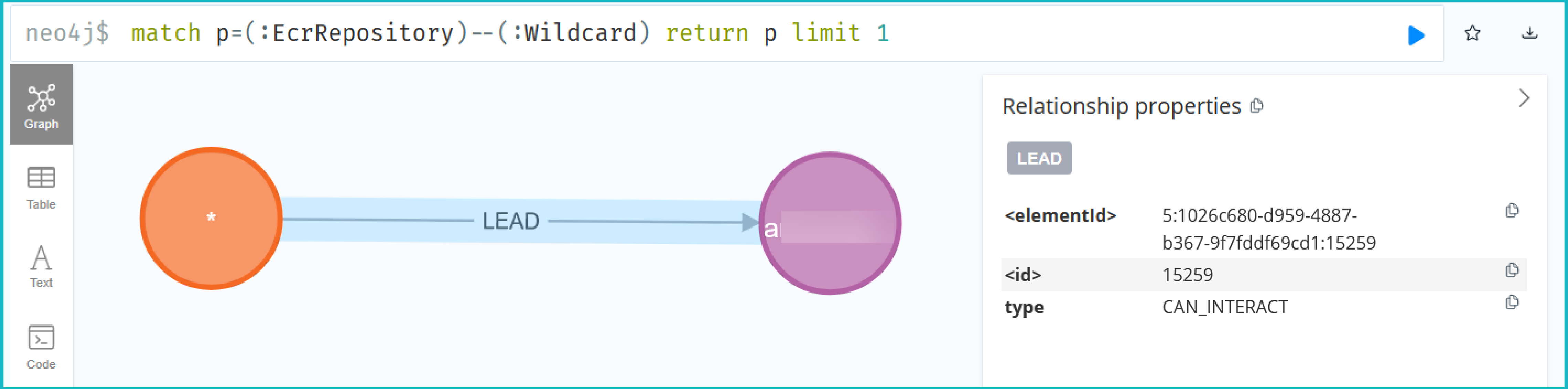
```
neo4j$ MATCH p=(n:IamRole)←[:ATTACHED]-(:IamPolicy{name:"AdministratorAccess"}) RETURN p
```



```python
class ResourcePolicyPrincipalLead(BaseLead):
    nodes = [BaseGraphNode]
    relation = "CAN_INTERACT"
```



```python
class RoleTrustPolicyLead(BaseLead):
    nodes = [IamRole]
    relation = "CAN_ASSUME"
```

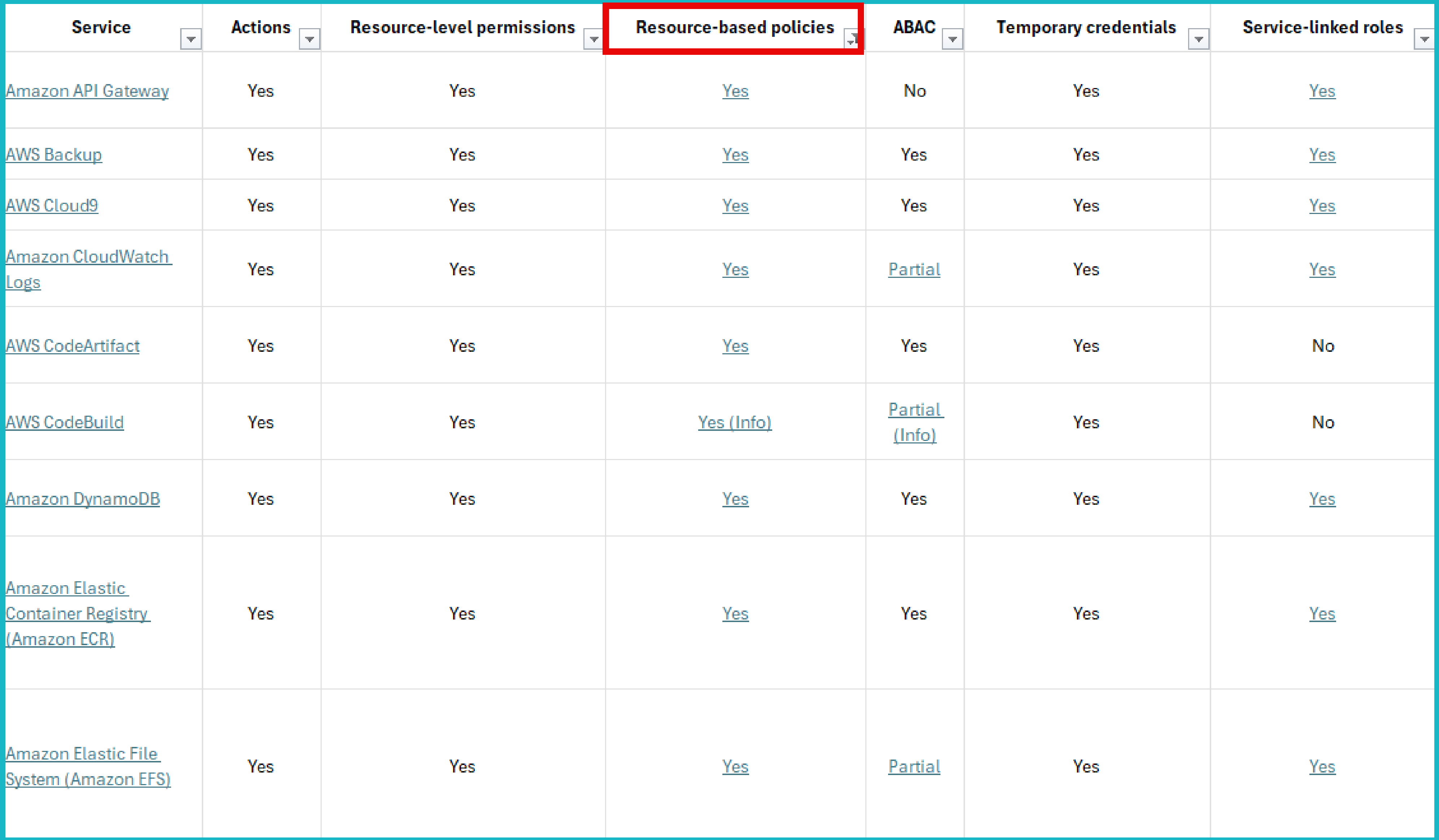


```
neo4j$ match p=(:EcrRepository)--(:Wildcard) return p limit 1
```

Relationship properties

LEAD

<elementId>	5:1026c680-d959-4887-b367-9f7fddf69cd1:15259
<id>	15259
type	CAN_INTERACT



Service	Actions	Resource-level permissions	Resource-based policies	ABAC	Temporary credentials	Service-linked roles
Amazon API Gateway	Yes	Yes	Yes	No	Yes	Yes
AWS Backup	Yes	Yes	Yes	Yes	Yes	Yes
AWS Cloud9	Yes	Yes	Yes	Yes	Yes	Yes
Amazon CloudWatch Logs	Yes	Yes	Yes	Partial	Yes	Yes
AWS CodeArtifact	Yes	Yes	Yes	Yes	Yes	No
AWS CodeBuild	Yes	Yes	Yes (Info)	Partial (Info)	Yes	No
Amazon DynamoDB	Yes	Yes	Yes	Yes	Yes	Yes
Amazon Elastic Container Registry (Amazon ECR)	Yes	Yes	Yes	Yes	Yes	Yes
Amazon Elastic File System (Amazon EFS)	Yes	Yes	Yes	Partial	Yes	Yes

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html

Limitation: Analysis ✓

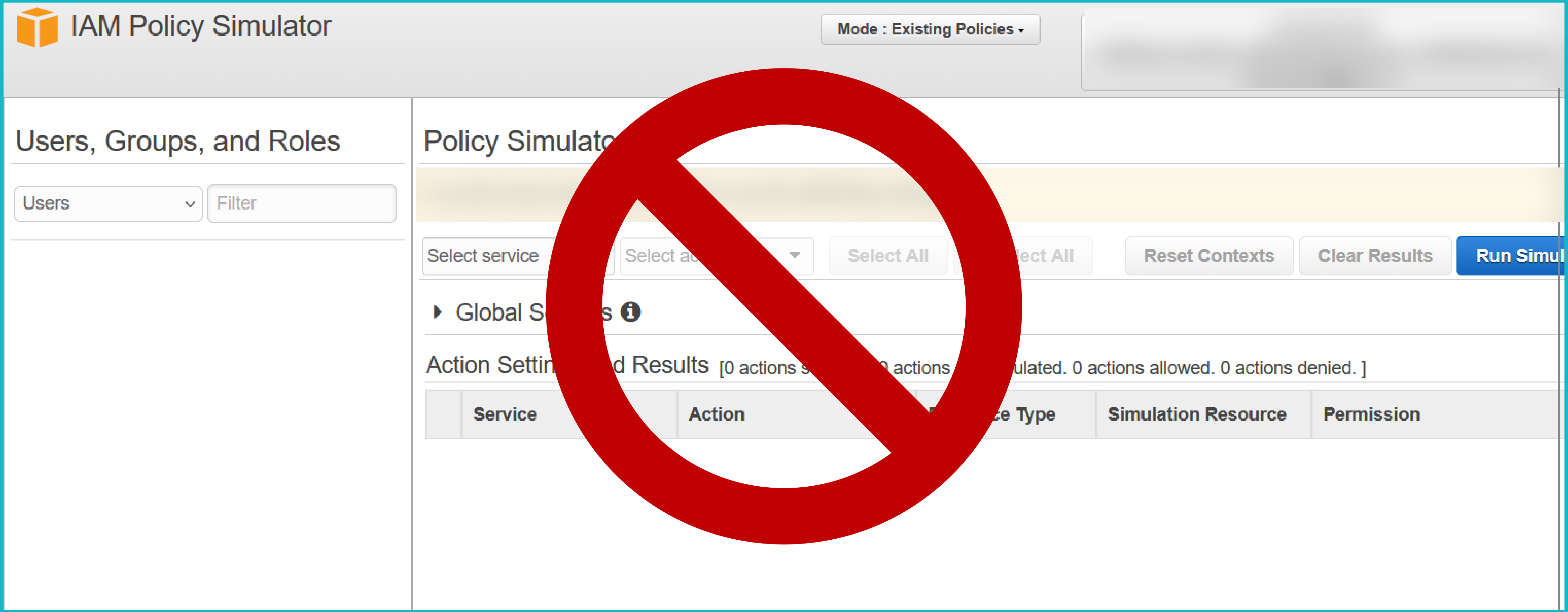
Architecture pt 3

- How to handle IAM?
- Simulate it!

IAM Permissions

Driving question:

How to do you represent permissions?



IAM Policy Simulator

Mode : Existing Policies

Users, Groups, and Roles

Users

Filter

Policy Simulato

Select service

Select ac

Select All

ect All

Reset Contexts

Clear Results

Run Simul

Global S s

Action Settin d Results [0 actions s 0 actions ulated. 0 actions allowed. 0 actions denied.]

Service	Action	ce Type	Simulation Resource	Permission

https://policysim.aws.amazon.com

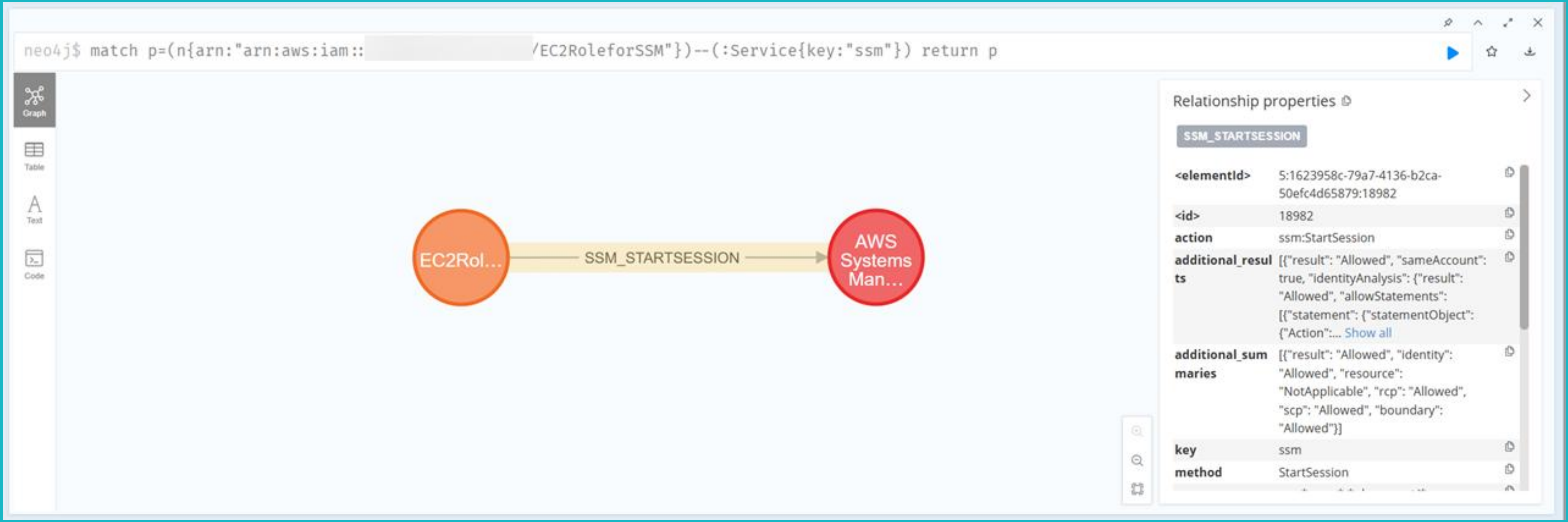
Component: IAM simulator

- Local IAM simulator
- github.com/cloud-copilot/iam-simulate

```
request: {
  action: 's3:GetObject',
  principal: 'arn:aws:iam::123456789012:user/username',
  resource: {
    accountId: '123456789012',
    resource: 'arn:aws:s3:::mybucket/file.txt'
  },
  contextVariables: {
    'aws:PrincipalOrgID': 'o-123456789012'
  }
}
```

+ policies

```
{
  effect: 'Allow',
  identifier: '1',
  matches: true,
  actionMatch: true,
  principalMatch: 'Match',
  resourceMatch: true,
  conditionMatch: true,
  resources: [
    {
      resource: 'arn:aws:s3:::mybucket/*',
      matches: true,
    }
  ],
  actions: [ { action: 's3:GetObject', matches: true } ],
}
```



```
neo4j$ match p=(n{arn:"arn:aws:iam::[redacted]/EC2RoleforSSM"})--(:Service{key:"ssm"}) return p
```

Relationship properties

SSM_STARTSESSION

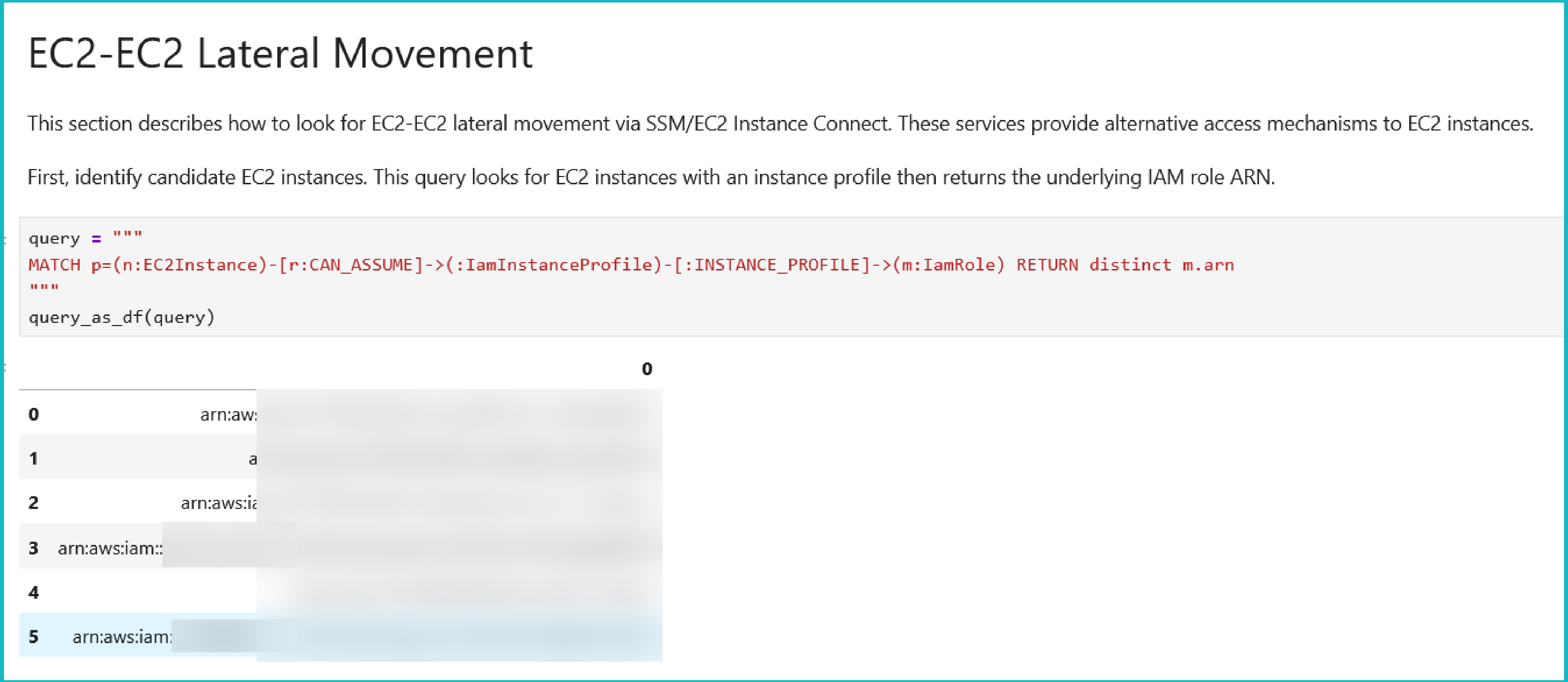
<elementId>	5:1623958c-79a7-4136-b2ca-50efc4d65879:18982
<id>	18982
action	ssm:StartSession
additional_results	[{"result": "Allowed", "sameAccount": true, "identityAnalysis": {"result": "Allowed", "allowStatements": [{"statement": {"statementObject": {"Action":... Show all
additional_summaries	[{"result": "Allowed", "identity": "Allowed", "resource": "NotApplicable", "rcp": "Allowed", "scp": "Allowed", "boundary": "Allowed"}]
key	ssm
method	StartSession

```
$ neph sim --principal <> --action ssm:StartSession --resource <> --write …
```

Limitation: IAM ✓

Architecture pt 4

- How to reproduce?
- Notebooks!



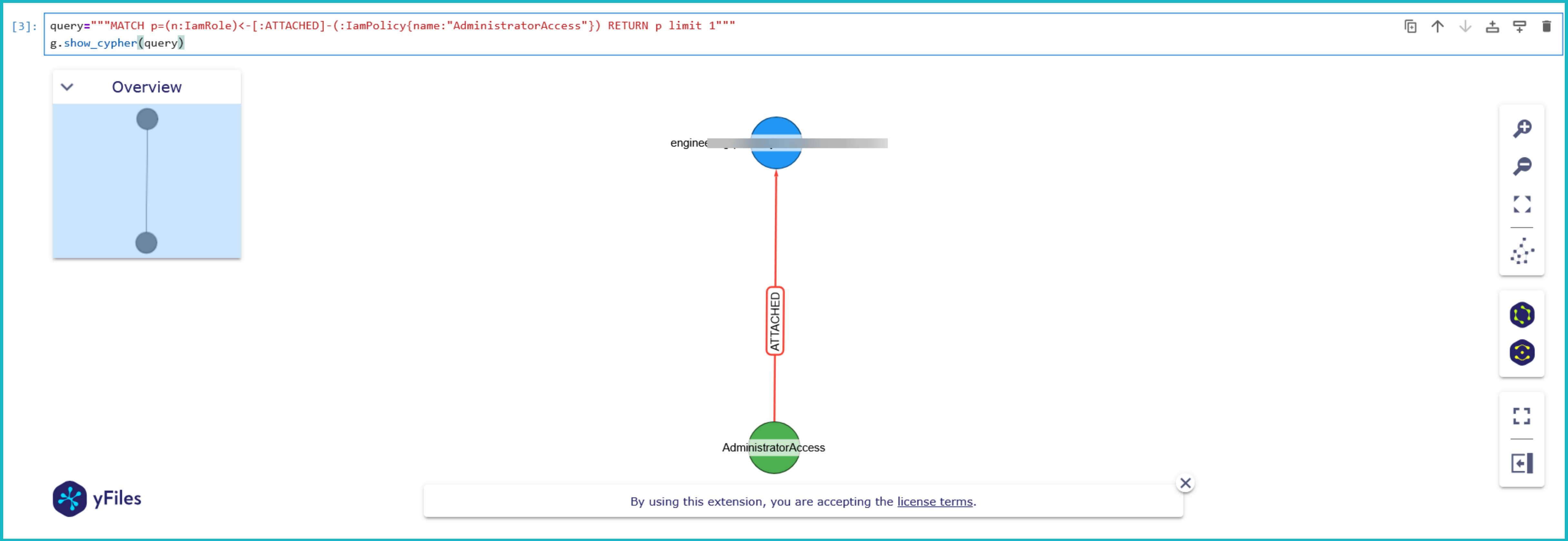
EC2-EC2 Lateral Movement

This section describes how to look for EC2-EC2 lateral movement via SSM/EC2 Instance Connect. These services provide alternative access mechanisms to EC2 instances.

First, identify candidate EC2 instances. This query looks for EC2 instances with an instance profile then returns the underlying IAM role ARN.

```
query = """
MATCH p=(n:EC2Instance)-[r:CAN_ASSUME]->(:IamInstanceProfile)-[:INSTANCE_PROFILE]->(m:IamRole) RETURN distinct m.arn
"""
query_as_df(query)
```

	0
0	arn:aws:[redacted]
1	a[redacted]
2	arn:aws:ia[redacted]
3	arn:aws:iam::[redacted]
4	[redacted]
5	arn:aws:iam:[redacted]



```
[3]: query="""MATCH p=(n:IamRole)<-[:ATTACHED]-(:IamPolicy{name:"AdministratorAccess"}) RETURN p limit 1"""
     g.show_cypher(query)
```

By using this extension, you are accepting the license terms.

Limitation: Analysis ✓

Analysis

Attack Paths

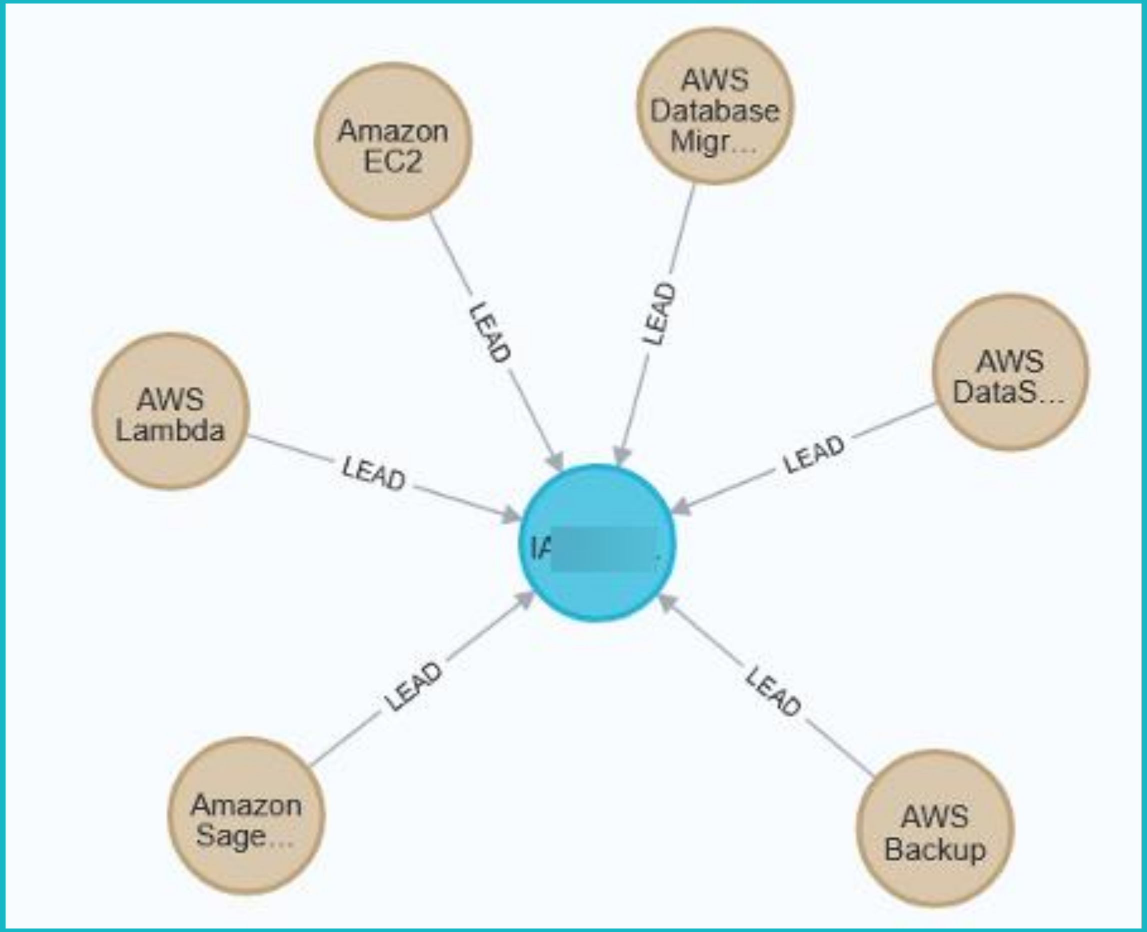
How can we apply these building blocks to identify attack paths?

Inbound Service Trust

- More inbound paths = likely riskier
- More services = more permissions

Inbound Service Trust

- **Schematically**: Check roles assumable by **>1** service



```
MATCH p=(n:Service)-[r:LEAD{type:"CAN_ASSUME"}]->(m:IamRole)
WITH m, collect(r) as rels
WHERE size(rels) > 1
…
```

Inbound Account Access

- AWS Organizations master account manages all other accounts (~tier zero)
  - Similar to a domain controller

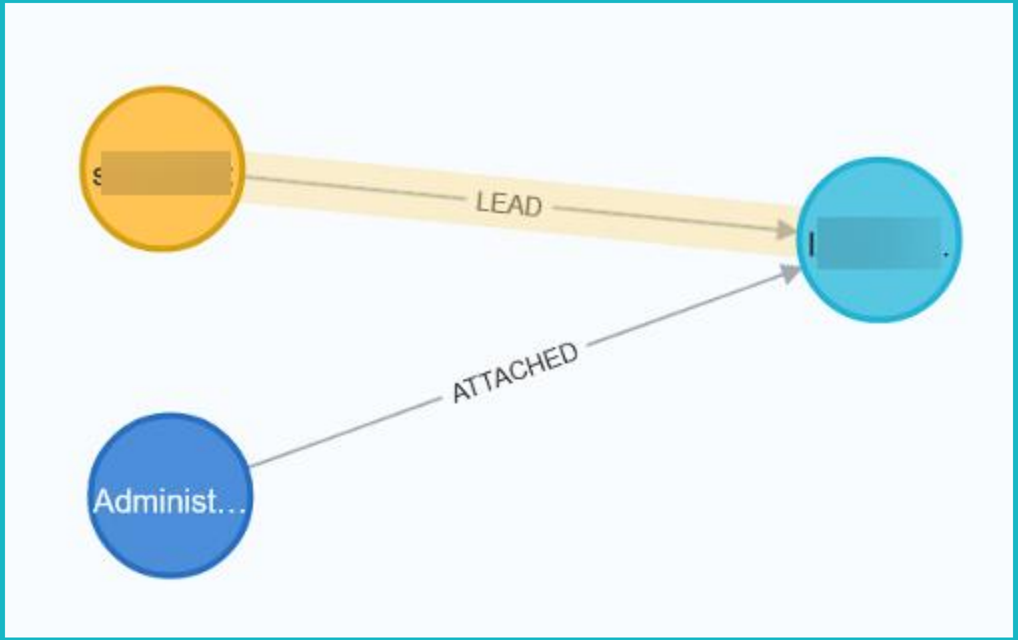
- Master account typically **trusted** by child accounts

- Paths inbound to master account = risky

Inbound Account Access

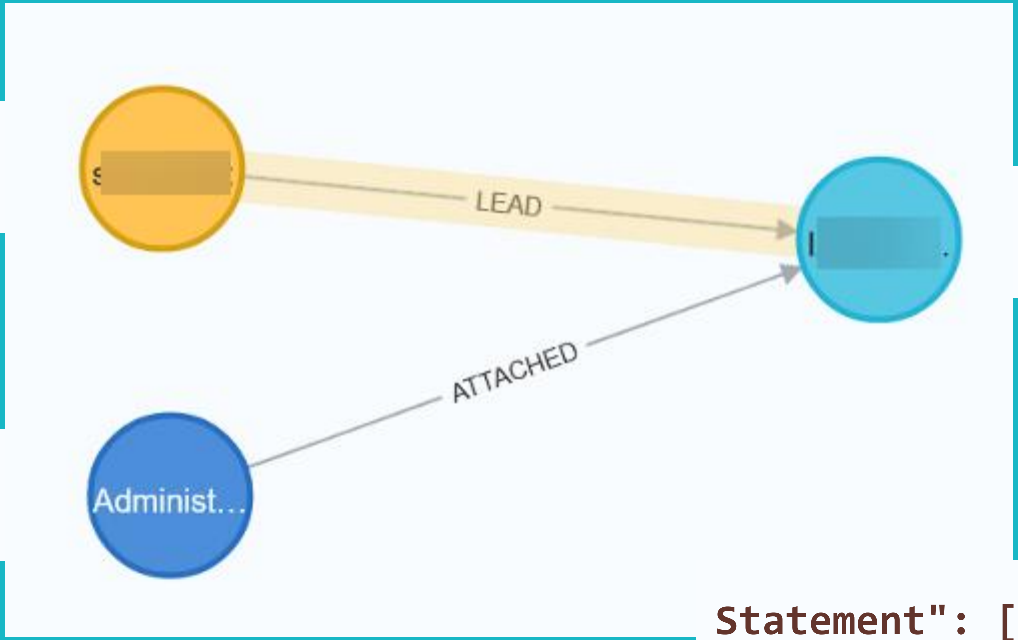
- Child -> Master -> Child
- **Schematically**: inbound paths to master from non-master
  - Special risk from out-of-org resources

```
MATCH p=(src)-[]->(dst)
WERE dst.account_id = "<master account>"
     and src.account_id <> "<master account>"
…
```



```
MATCH p=(n)-[]->(m)
WERE m.account_id = "<master account>"
     and n.account_id <> "<master account>"
…
```

Not Master Account ->



<- Role in
Master Account

Admin Policy->

Permissive Trust Policy->

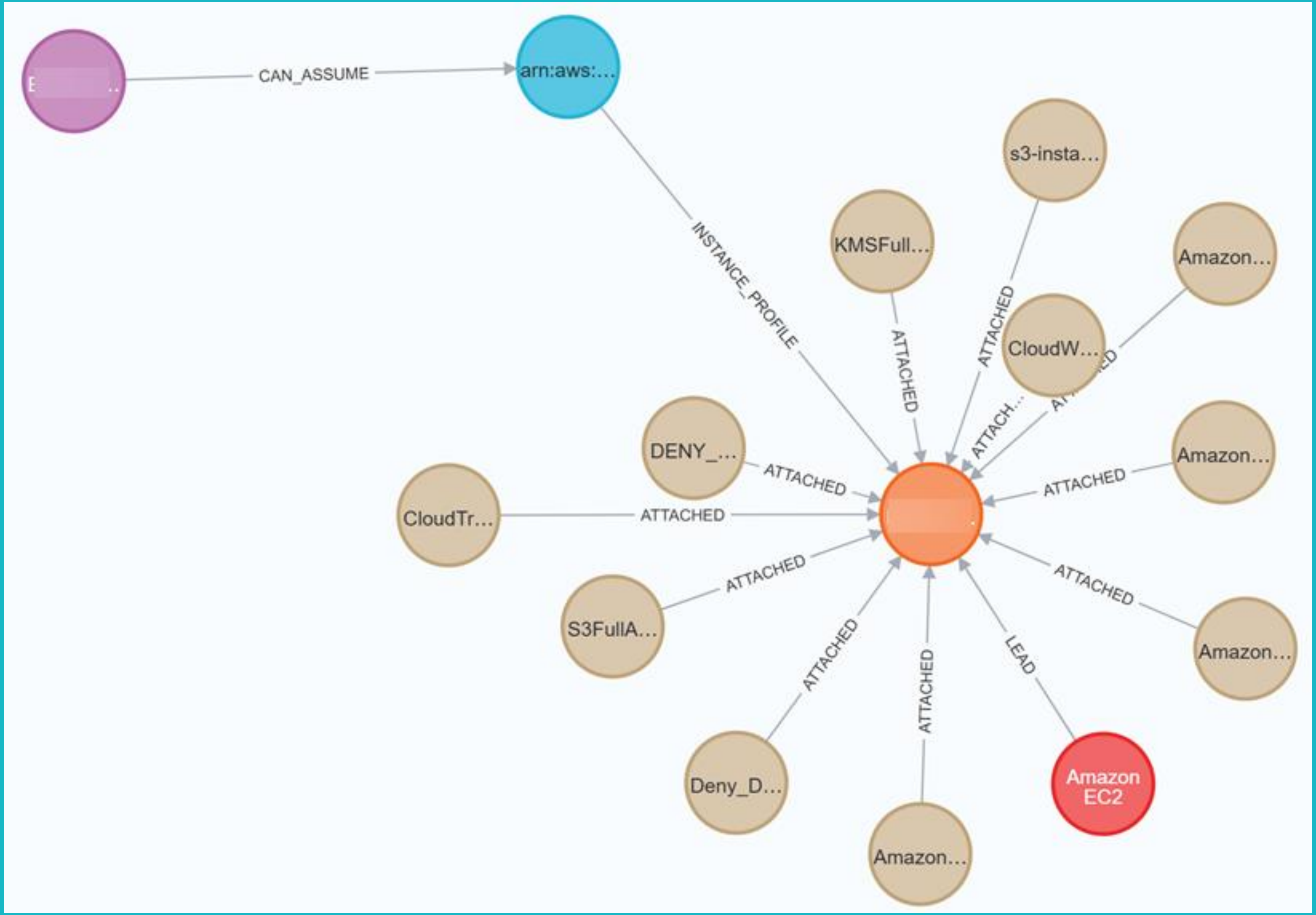
```
Statement": [{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456:root"}
   }
]}
```

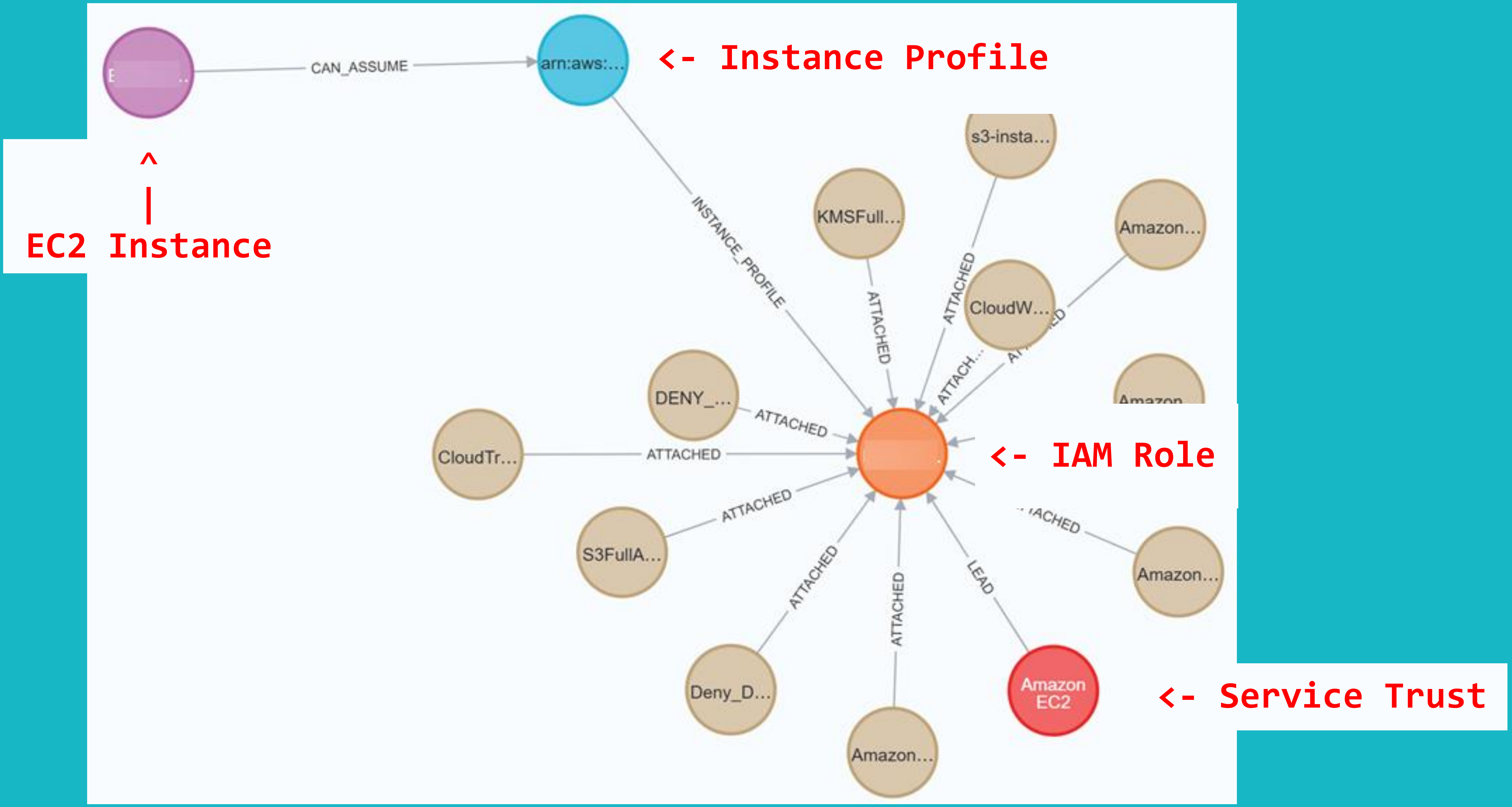
EC2 Lateral Movement

- EC2 instances can have IAM roles
- EC2 roles = common exploitation path
- SSM & Instance Connect (EIC) both allow **alternative** access
- Principals with SSM/EIC = risky

EC2 Lateral Movement

- **Schematically**:
  - Identify roles used by instances
  - Filter for SSM/EIC permissions
  - Simulate access to *

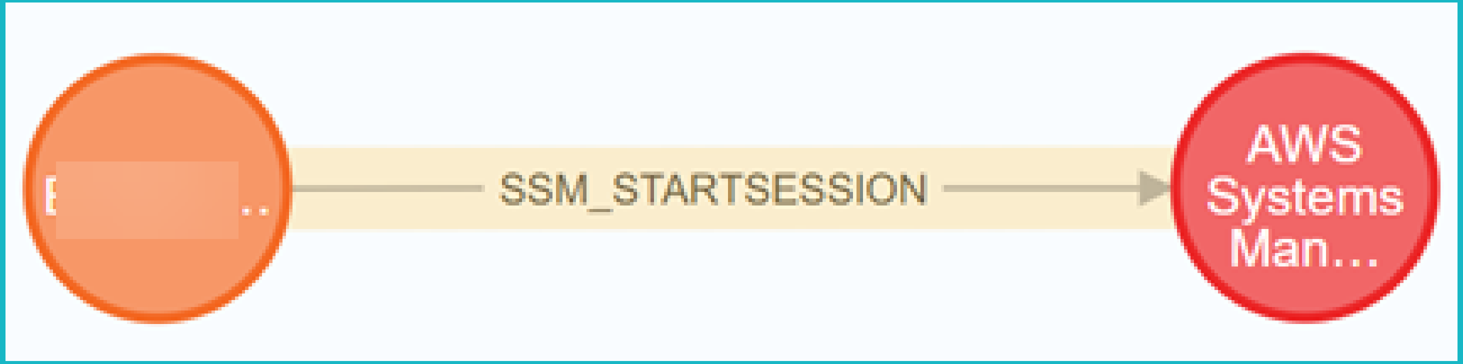

CAN_ASSUME
arn:aws:…
INSTANCE_PROFILE
s3-insta…
KMSFull…
Amazon…
CloudW…
ATTACHED
ATTACH…
DENY_…
CloudTr…
Amazon…
S3FullA…
Amazon…
LEAD
Deny_D…
Amazon…
Amazon EC2



CAN_ASSUME
arn:aws:...
<- Instance Profile
^
|
EC2 Instance
INSTANCE_PROFILE
s3-insta...
KMSFull...
Amazon...
CloudW...
ATTACHED
ATTACH...
Amazon...
DENY_...
CloudTr...
<- IAM Role
S3FullA...
Deny_D...
Amazon...
LEAD
Amazon EC2
Amazon...
<- Service Trust

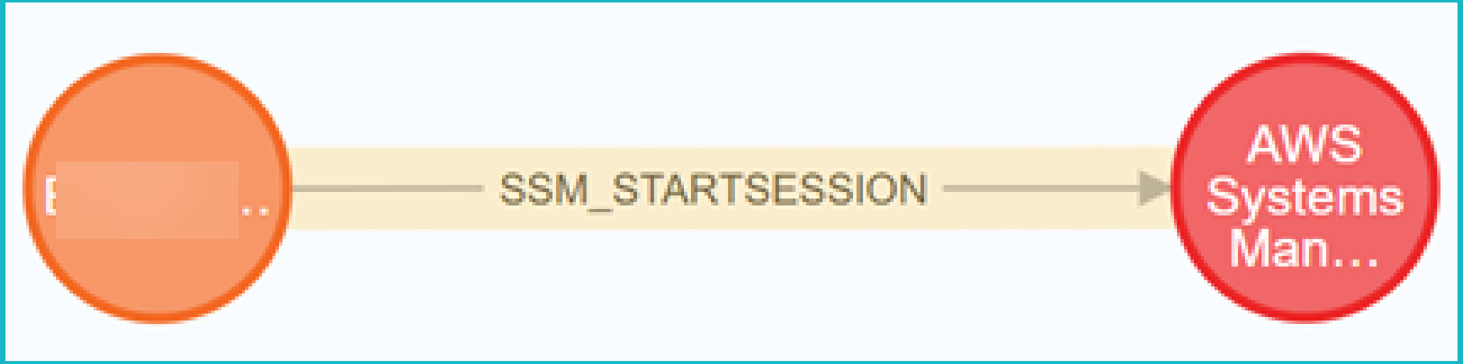
```
$ neph fanout --arn “<EC2 ROLE>” --strategy EC2AccessFanout
```

```
$ neph sim --principal “<EC2 ROLE>” --action ssm:StartSession
--resource * --write …
```



```
$ neph fanout --arn “<EC2 ROLE>” --strategy EC2AccessFanout
```

```
$ neph sim --principal “<EC2 ROLE>” --action ssm:StartSession
--resource * --write …
```



Traditional Priv Esc

- Many abusable permissions in IAM
- Ex: Put X policy, attach X policy, create X
- How to differentiate from **legitimate** admins?

Traditional Priv Esc

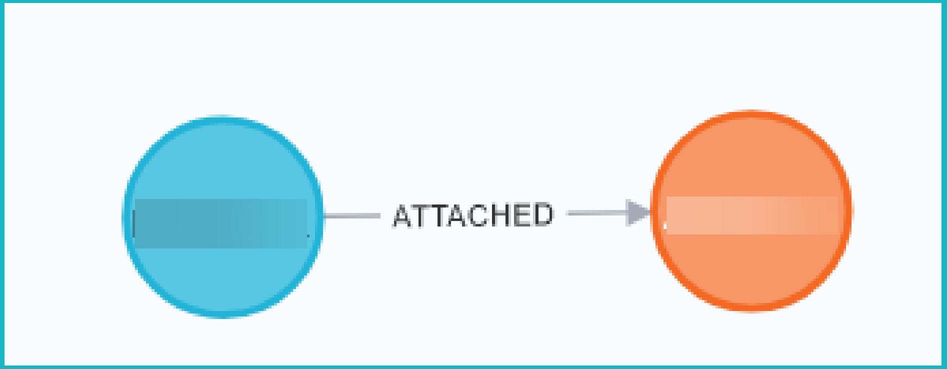
- Schematically (**targets**)
    - Find principals with attached policies
    - Filter for ones with abusable permissions
    - Simulate action for *

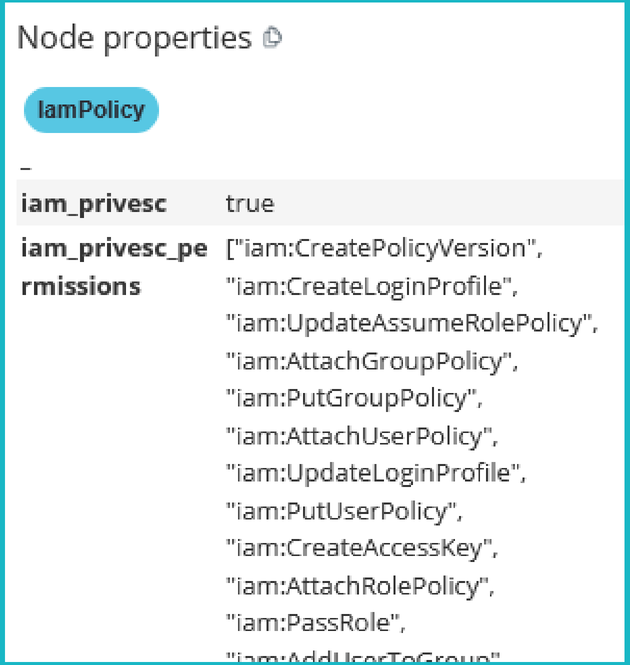
Traditional Priv Esc

- Schematically (**sources**)
  - Find principals with attached policies
  - Filter for ones without abusable permissions
  - Filter for ones with paths to targets
  - Also look at inbound paths to sources

- Represents **increase** in privilege

```
MATCH p=(n)<-[:ATTACHED]-(m:IamPolicy|IamInlinePolicy)
WHERE toBoolean(m.iam_privesc) = TRUE
      and not m.iam_privesc_permissions ='["iam:PassRole"]'
…
```




Node properties

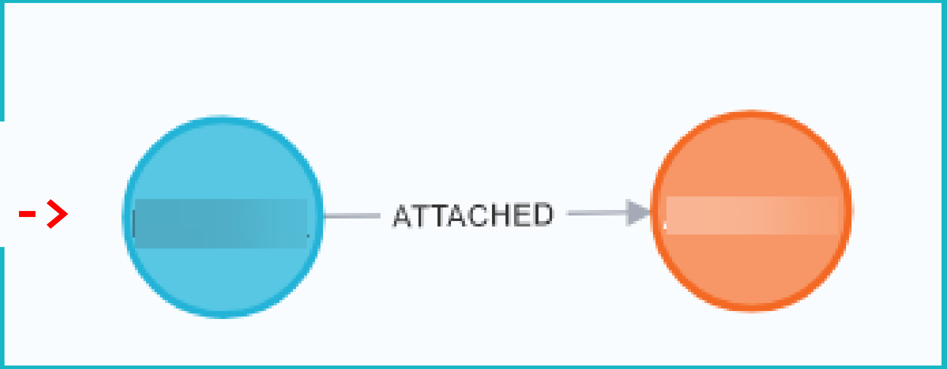
IamPolicy

iam_privesc	true
iam_privesc_permissions	["iam:CreatePolicyVersion", "iam:CreateLoginProfile", "iam:UpdateAssumeRolePolicy", "iam:AttachGroupPolicy", "iam:PutGroupPolicy", "iam:AttachUserPolicy", "iam:UpdateLoginProfile", "iam:PutUserPolicy", "iam:CreateAccessKey", "iam:AttachRolePolicy", "iam:PassRole", "iam:AddUserToGroup"

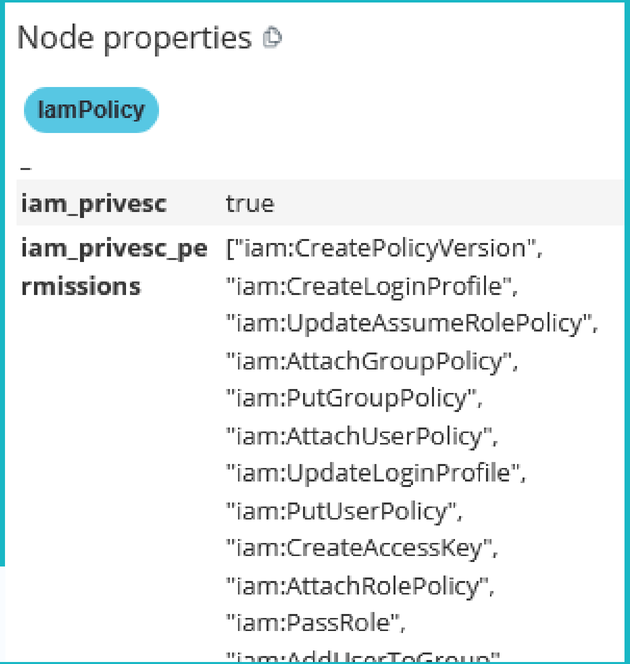
```
$ neph sim --principal “<TARGET>” --action <ABUSABLE> --
resource * …
```

```
MATCH p=(n)<-[:ATTACHED]-(m:IamPolicy|IamInlinePolicy)
WHERE toBoolean(m.iam_privesc) = TRUE
      and not m.iam_privesc_permissions ='["iam:PassRole"]'
…
```

Admin Policy on Role ->



Bad Permissions in Policy ->

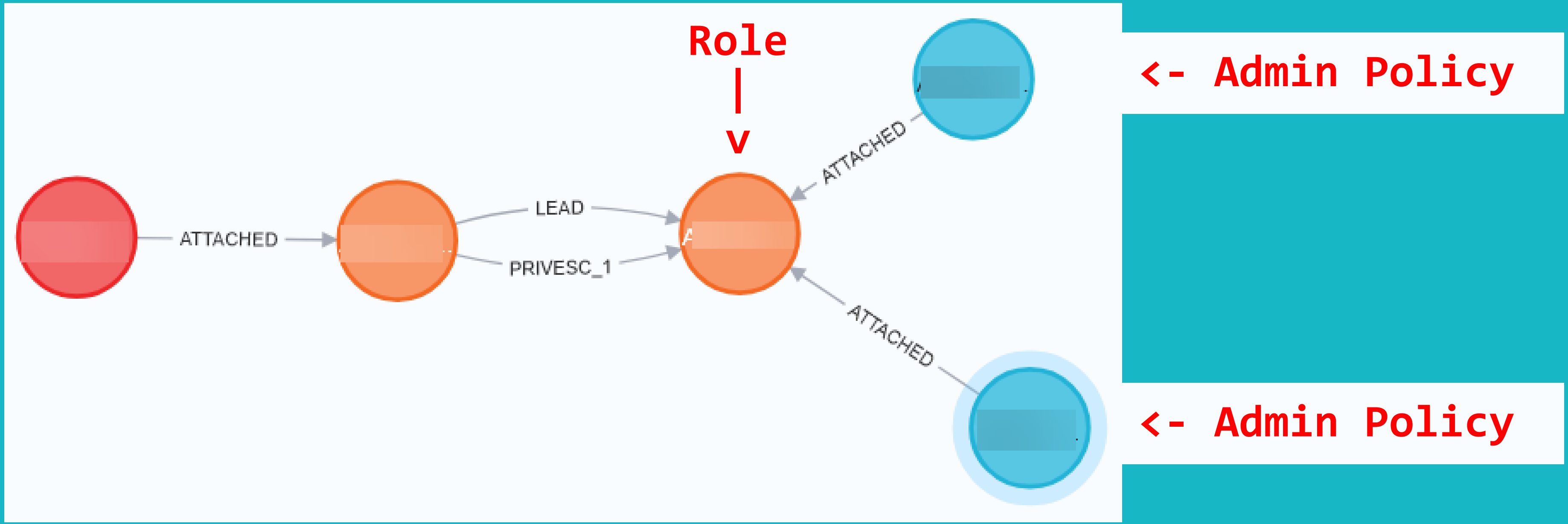

Node properties

IamPolicy

iam_privesc	true
iam_privesc_permissions	["iam:CreatePolicyVersion", "iam:CreateLoginProfile", "iam:UpdateAssumeRolePolicy", "iam:AttachGroupPolicy", "iam:PutGroupPolicy", "iam:AttachUserPolicy", "iam:UpdateLoginProfile", "iam:PutUserPolicy", "iam:CreateAccessKey", "iam:AttachRolePolicy", "iam:PassRole", "iam:AddUserToGroup"

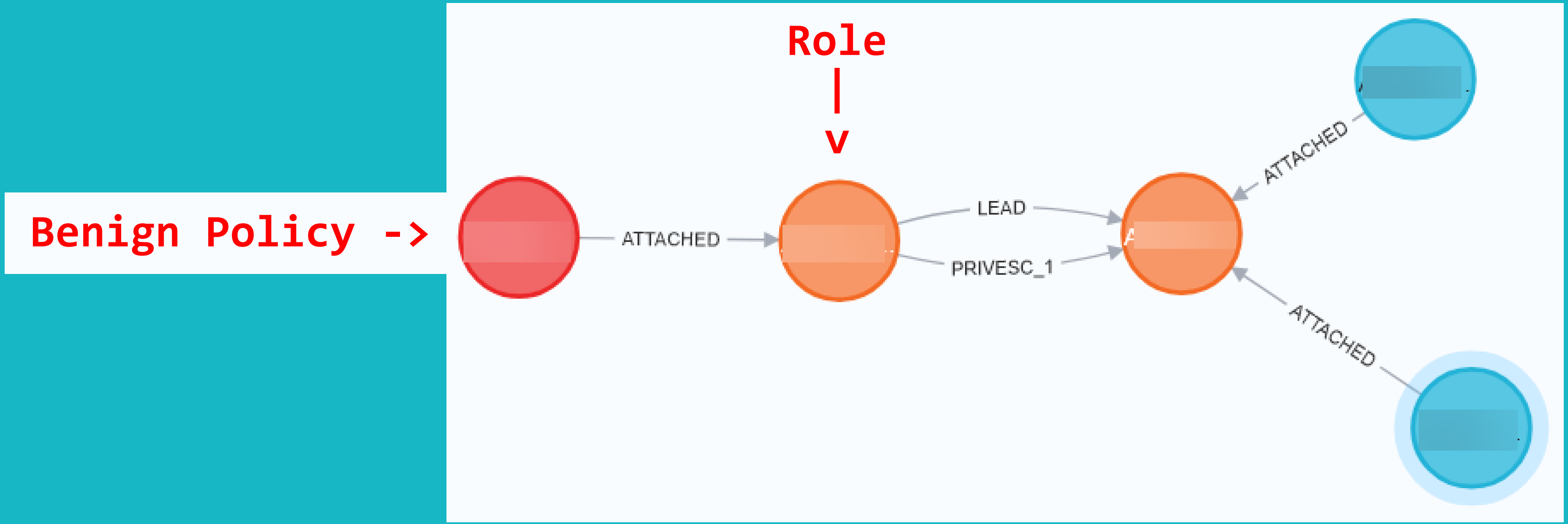
```
$ neph sim --principal "<TARGET>" --action <ABUSABLE> --resource * …
```

```
MATCH (z:IamRole|IamUser)
      WHERE count{(z)<-[:ATTACHED]-(pol{iam_privesc:"true"})} = 0

MATCH (n)<-[:LEAD{type:"CAN_ASSUME"}]-(z)
      WHERE n.arn in $dest_roles
```



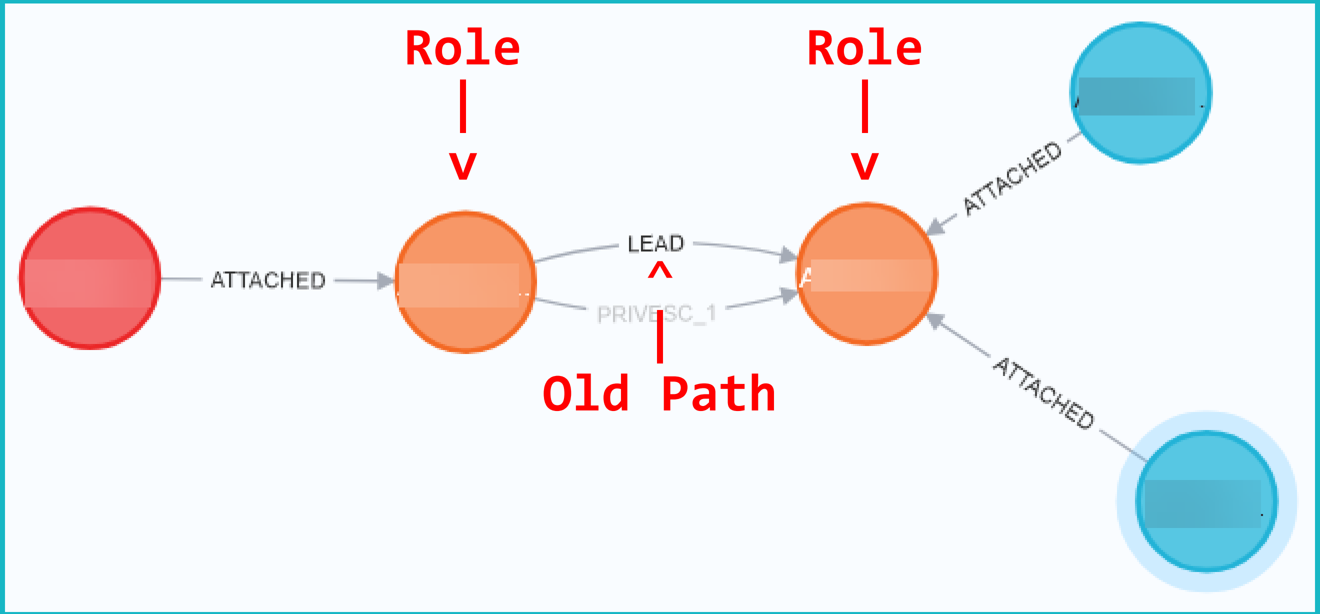
```
$ neph sim --principal “<SRC>” --action sts:AssumeRole --
resource “<TARGET>” …
```

```
MATCH (z:IamRole|IamUser)
     WHERE count{(z)<-[:ATTACHED]-(pol{iam_privesc:"true"})} = 0

MATCH (n)<-[:LEAD{type:"CAN_ASSUME"}]-(z)
     WHERE n.arn in $dest_roles
```



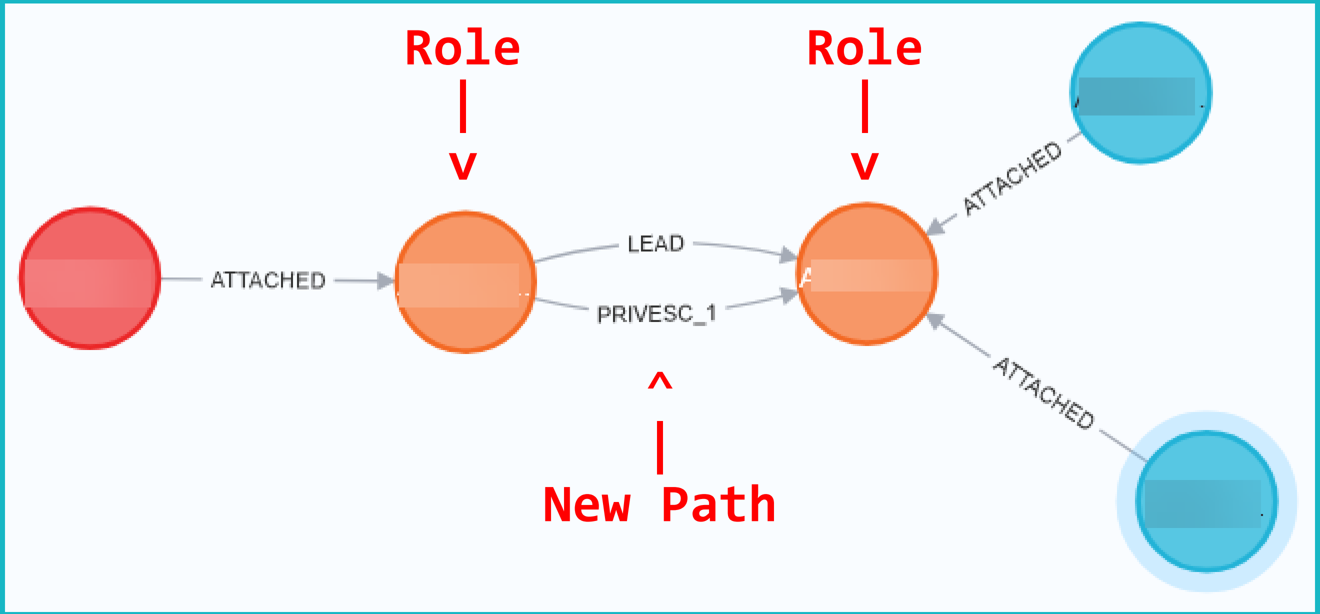
```
$ neph sim --principal “<SRC>” --action sts:AssumeRole --
resource “<TARGET>” …
```

```
MATCH (z:IamRole|IamUser)
     WHERE count{(z)<-[:ATTACHED]-(pol{iam_privesc:"true"})} = 0

MATCH (n)<-[:LEAD{type:"CAN_ASSUME"}]-(z)
     WHERE n.arn in $dest_roles
```

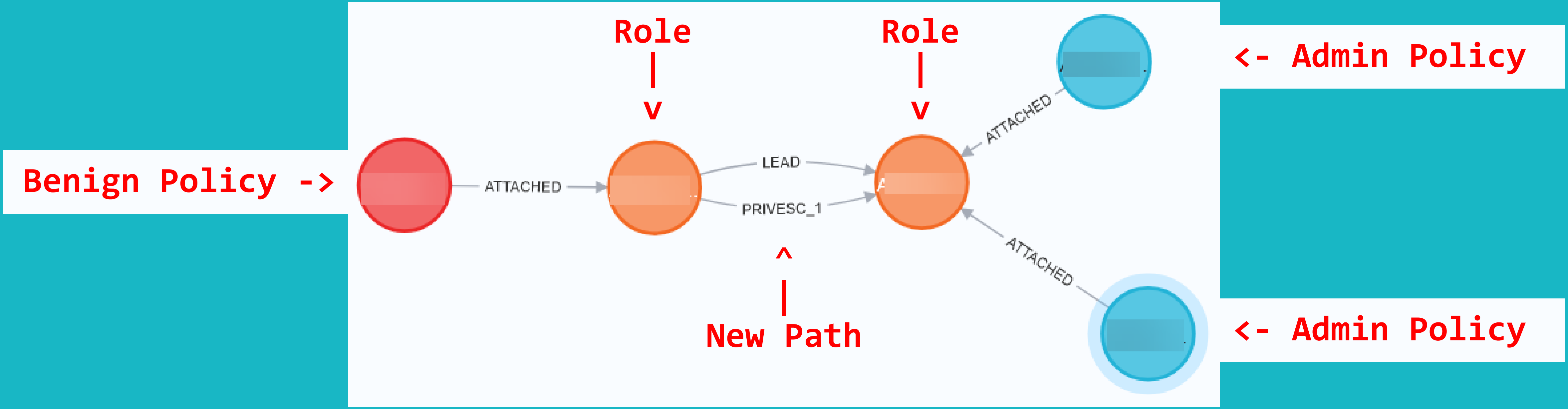


```
$ neph sim --principal “<SRC>” --action sts:AssumeRole --
resource “<TARGET>” …
```

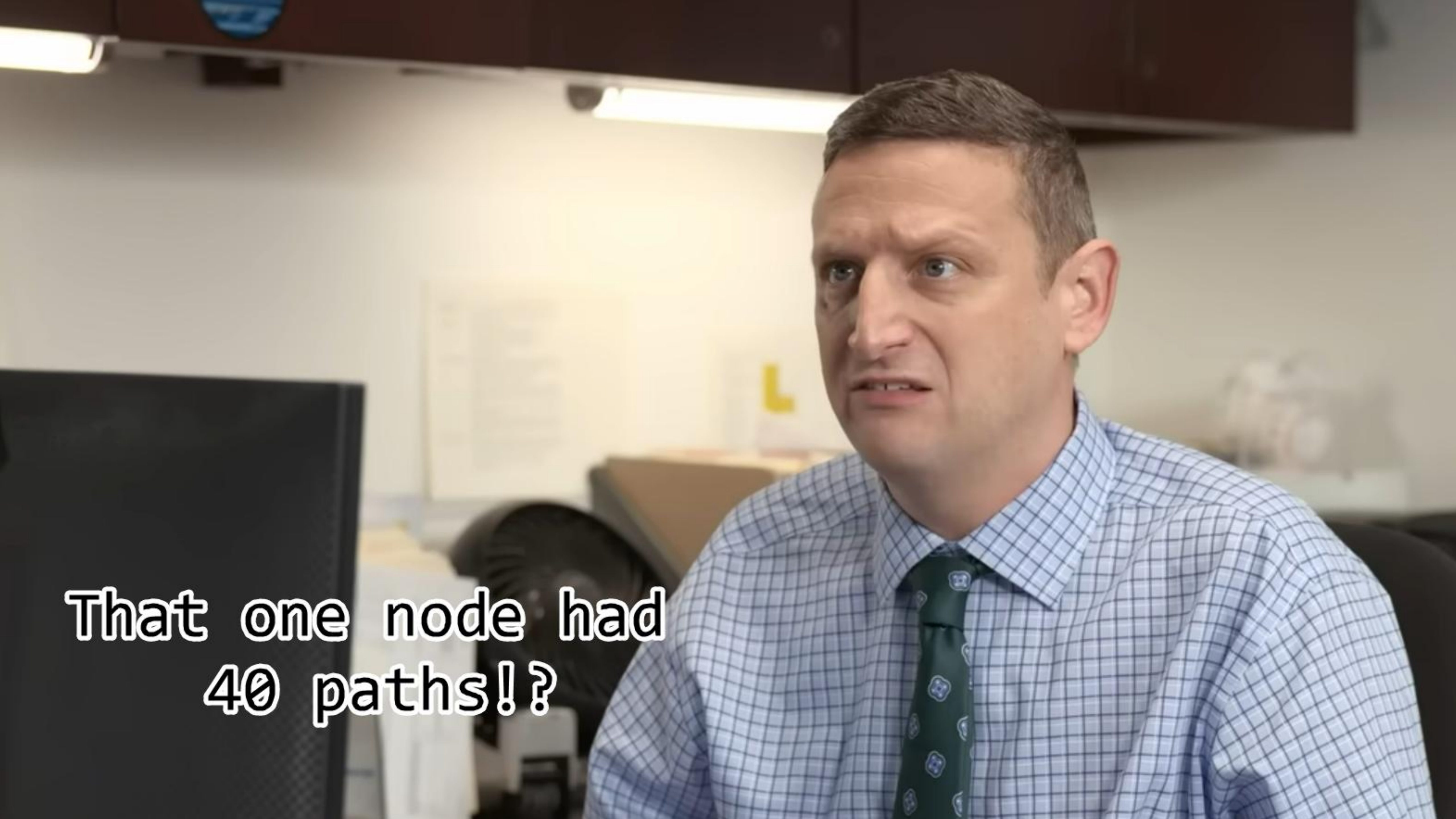
```
MATCH (z:IamRole|IamUser)
     WHERE count{(z)<-[:ATTACHED]-(pol{iam_privesc:"true"})} = 0

MATCH (n)<-[:LEAD{type:"CAN_ASSUME"}]-(z)
     WHERE n.arn in $dest_roles
```



```
$ neph sim --principal “<SRC>” --action sts:AssumeRole --
resource “<TARGET>” …
```


Benign Policy ->
Role
Role
<- Admin Policy
<- Admin Policy
ATTACHED
LEAD
PRIVESC_1
ATTACHED
ATTACHED
New Path

Closing


That one node had
40 paths!?

Limitations + Next Steps

- Neph is still **~alpha**, active dev ongoing
- Expect bugs
- Requires **manual** analysis (intentionally)

- OpenGraph

Links

- Slides
  github.com/SecurityRiskAdvisors/public-assets
- Neph
  github.com/SecurityRiskAdvisors/neph
- Me
  @2xxeformyshirt
  2xxe.com

Questions