SPECTEROPS

Apeman

Get in loser, we’re doing cloud stuff

Why?

Current Analysis


```
{
    "UserDetailList": [
        {
            "Path": "/",
            "UserName": "Administrator",
            "UserId": "AIDAZ2VU5FUQLGMVGYLVD",
            "Arn": "arn:aws:iam::675763006752:user/Administrator",
            "CreateDate": "2020-04-03T17:15:50+00:00",
            "UserPolicyList": [
                {
                    "PolicyName": "all-ec2-inline",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Sid": "VisualEditor0",
                                "Effect": "Allow",
                                "Action": "ec2:*",
                                "Resource": "*"
                            }
                        ]
                    }
                }
            ],
            "GroupList": [
                "Admins",
                "Administrators"
            ],
            "AttachedManagedPolicies": [
                {
                    "PolicyName": "assume_role_1",
                    "PolicyArn": "arn:aws:iam::675763006752:policy/assume_role_1"
                }
            ],
            "Tags": [
```


✖ Up to 300

Example: Client A

- 50 AWS accounts
- Customer stated they had roughly 10 “Org Admins”
  - Manual analysis found that it was closer to 200
- What does it mean to be an admin?

Apeman Goals

1. To effectively analyze an AWS environment and identify identity attack paths
2. What are your “Tier 0” attack paths?
   - “Tier 0”™

Why is this difficult?

1. Resultant Set of Policy
    1. A principal may have multiple statements that contradict each other
2. Conditions and Policy Variables
    1. Some conditions may be resolvable
    2. Others not so much

Hasn’t this been done?

- AWS will let you simulate access from a single principal with a single action to a single resource
  - We want all principles for all actions on all resources
- Zelkova
  - Automated Reasoning
- AWS Access Advisor will show what actions a principal can perform
  - Doesn’t show on what resources they can perform actions on
- Policy Simulator not quite what we want
  - More on this later

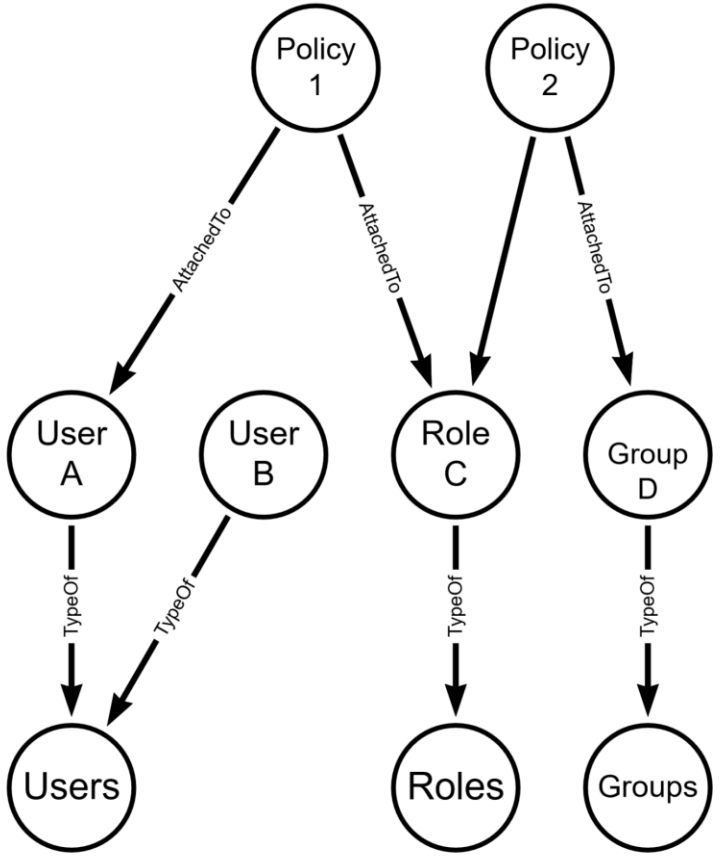
Three Main Questions

More like 6

1. Given a resource, what principals can perform actions on that resource? Given a principal, what actions can it take?
2. What roles can a principal assume? What principals can assume a role?
3. What principals are tier 0? What identity paths are tier 0?

AWS Permissions

- User A has 1 policy attached
- Role C has 2 policies attached
- User B has 0 policies attached
  - Cannot do anything
- Deny overrides Allow



Anatomy of an AWS Policy

```
policy = {
        <version_block?>
        <id_block?>
        <statement_block>
}
```

```
statement_block = {
        <sid_block?>
        <principal_block?>
        <effect_block>
        <action_block>
        <resource_block>
        <condition_block>
}
```

Anatomy of an AWS Policy

```
statement_block = {
        <sid block?>
        <principal block?>
        <effect block>
        <action block>
        <resource block>
        <condition block>
}
```

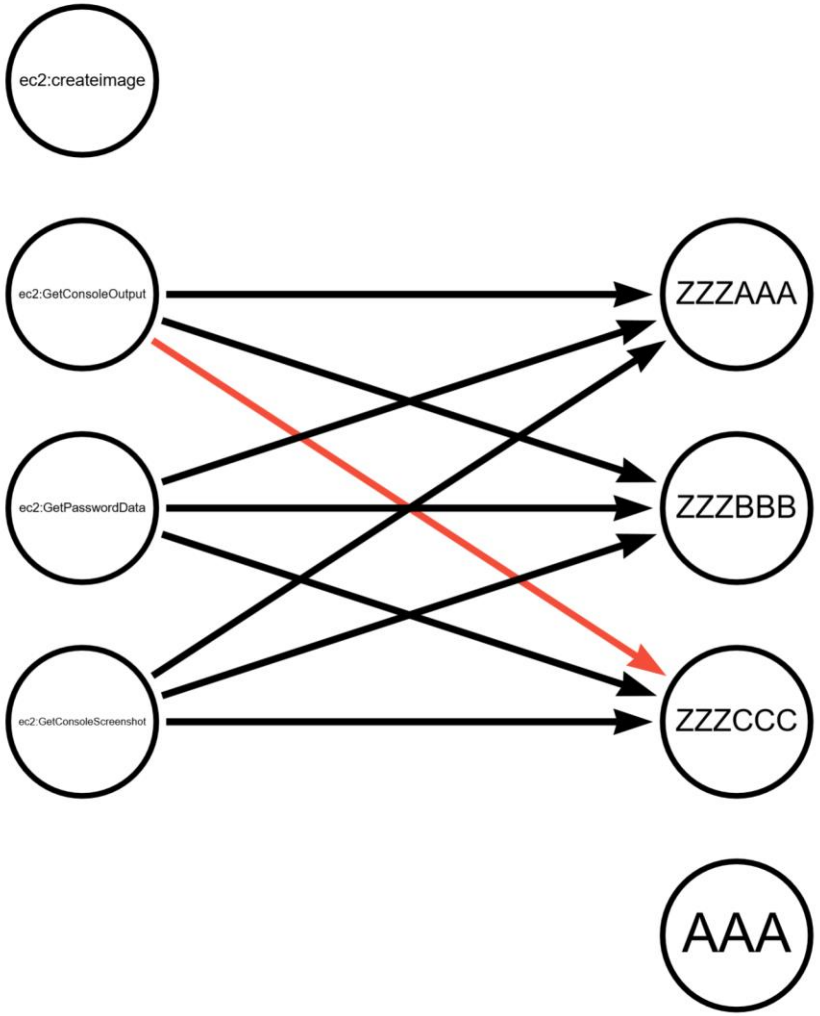
```
"Statement": [
{
        "Effect": "Allow",

        "Action": "ec2:Get*”,
                "Resource":"arn:aws:ec2:*:*:instance/*ZZZ*"
}
]
```

Anatomy of an AWS Policy

```
"Statement": [
{
        "Effect": "Allow",
        "Action": "ec2:Get*”,
                "Resource":"arn:aws:ec2:*:*:instance/*ZZZ*"
},
{
        "Effect": “Deny",
        "Action": "ec2:GetConsoleOutput”,
                "Resource":"arn:aws:ec2:*:*:instance/*ZZZCCC*"
}
]
```

Resultant Set Of Policy (RSOP)



The 5 Stages of Apeman

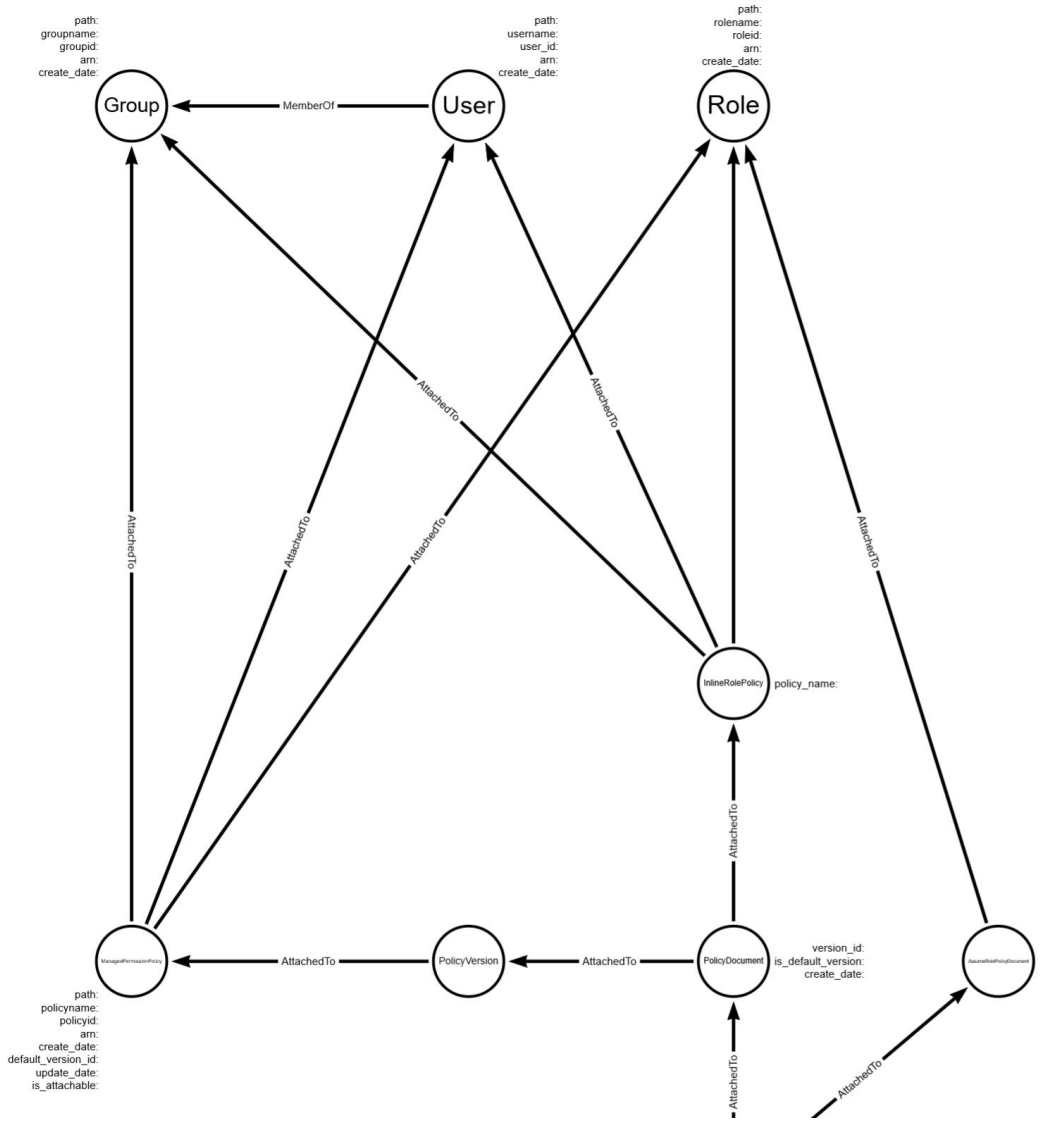
1. Denial
   - Grepping through this JSON is not that bad
2. Anger
   - What do you mean you can specify ‘NotAction’ and ‘NotResource’?
3. Bargaining
   - What if we consider the use of conditions a finding so that we don’t need to evaluate it
4. Depression
   - AWS is not even that popular
5. Creating a new Github Project and giving it a silly name
   - We do it not because it is easy, but because we thought it would be easy

We don’t even need a collector!

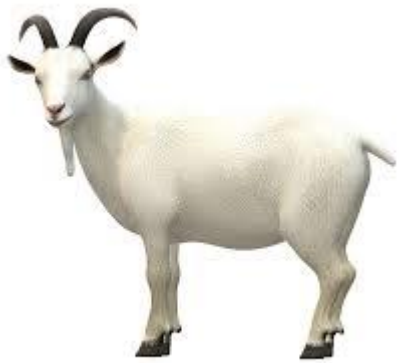
aws iam get-account-authorization-details

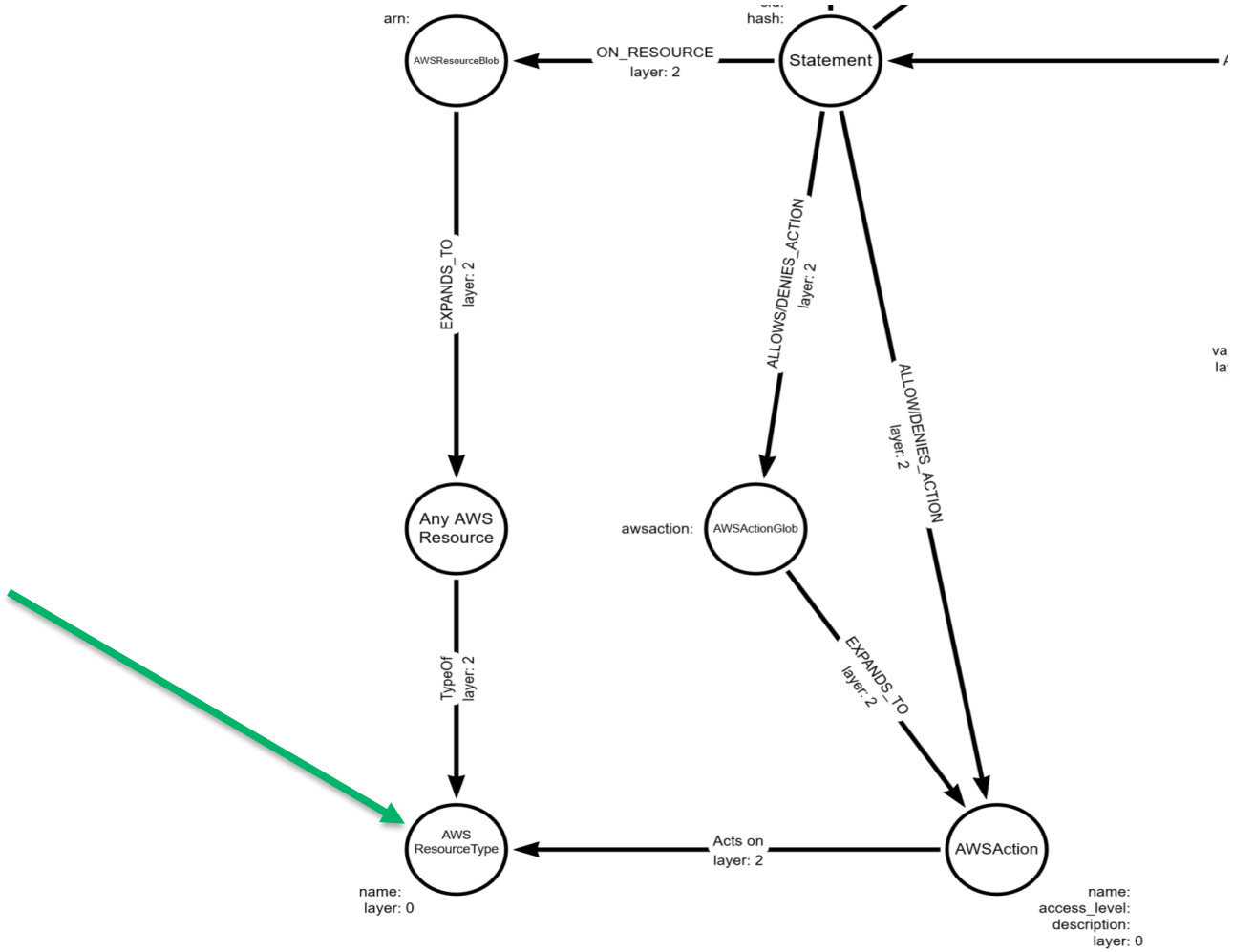
- Gets every Role, Group, User, and Policy
- Enough to model role assumptions
- Additional resources can be ingested for more accurate details

When your only tool is a graph shaped hammer

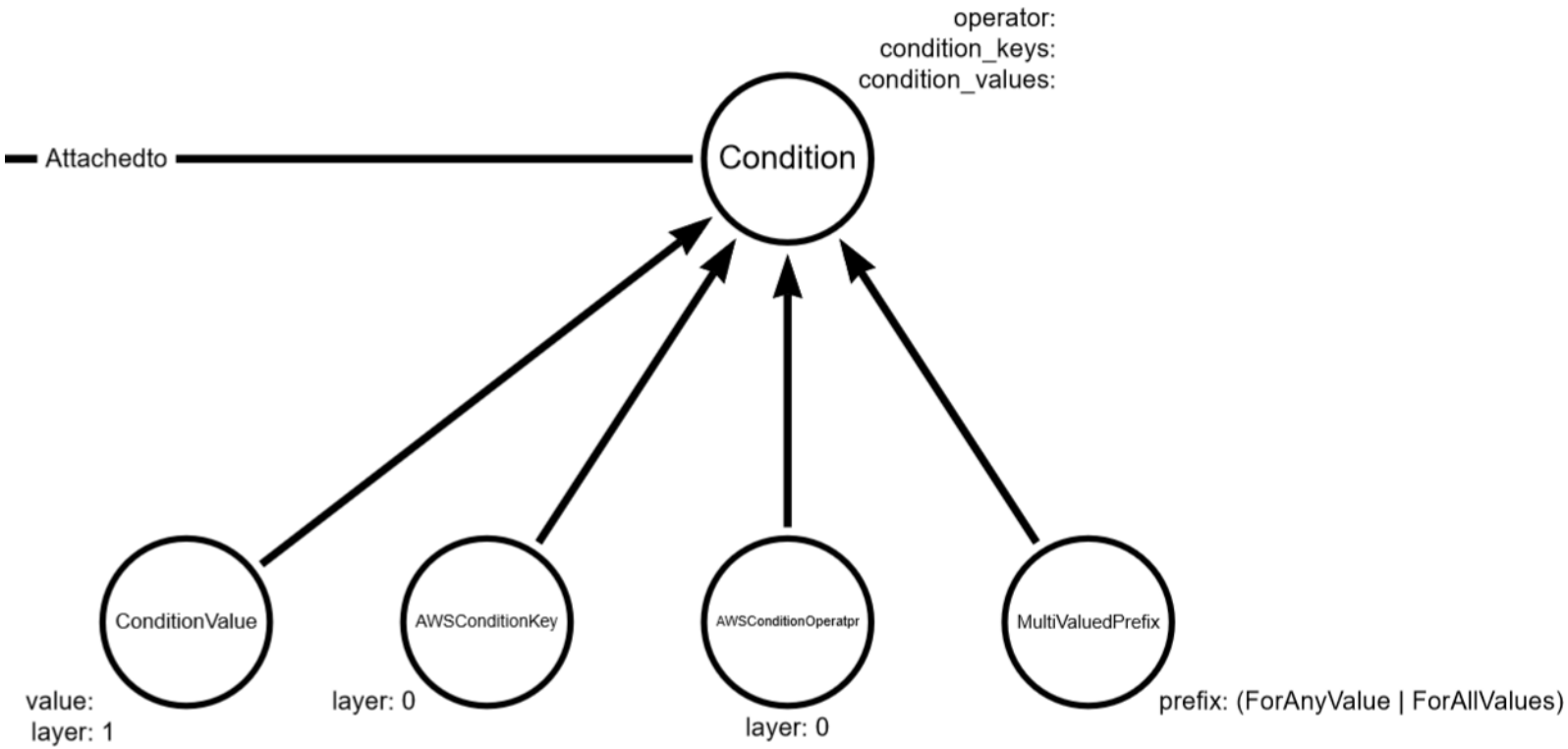


When your only tool is a graph shaped hammer



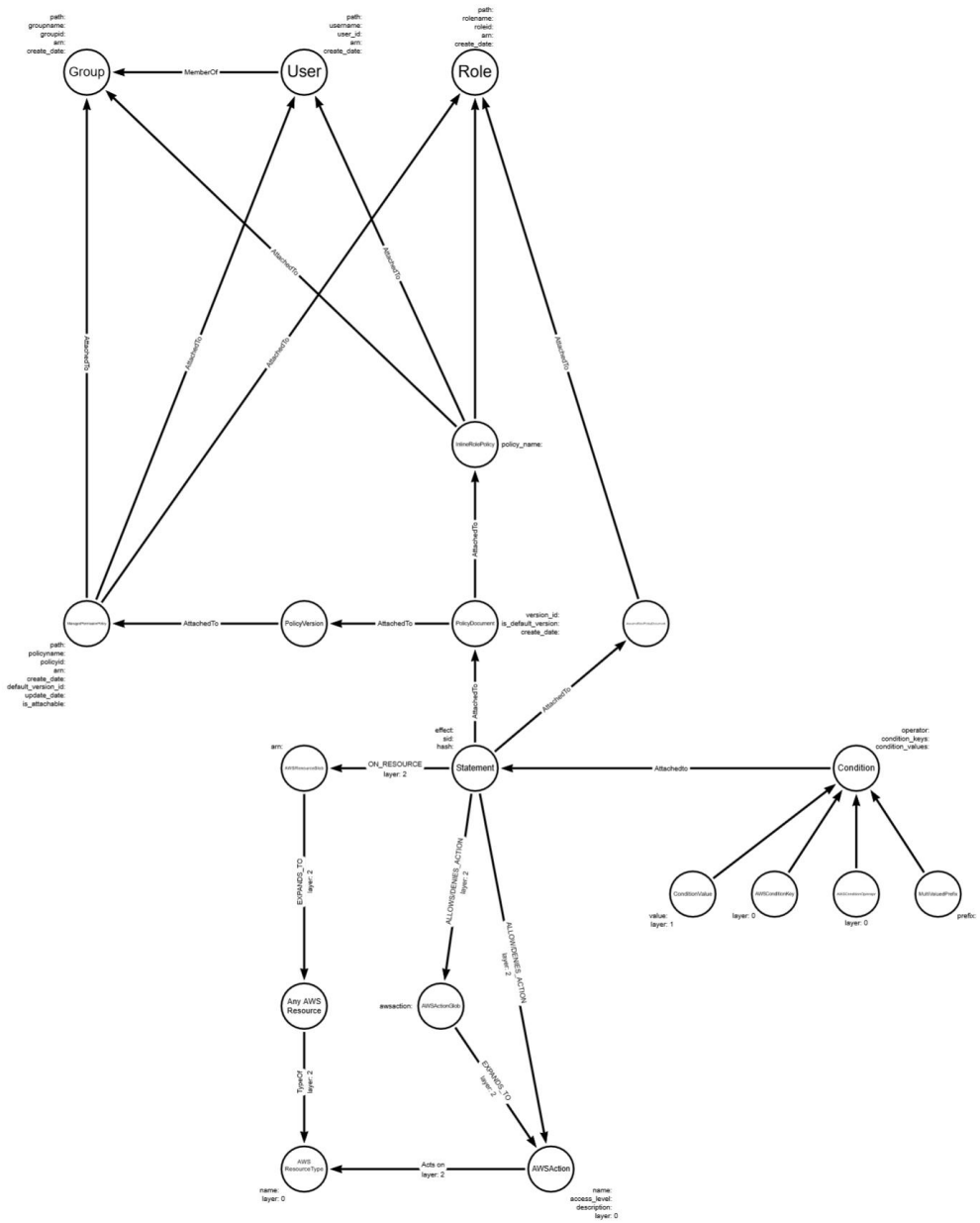


When your only tool is a graph shaped hammer

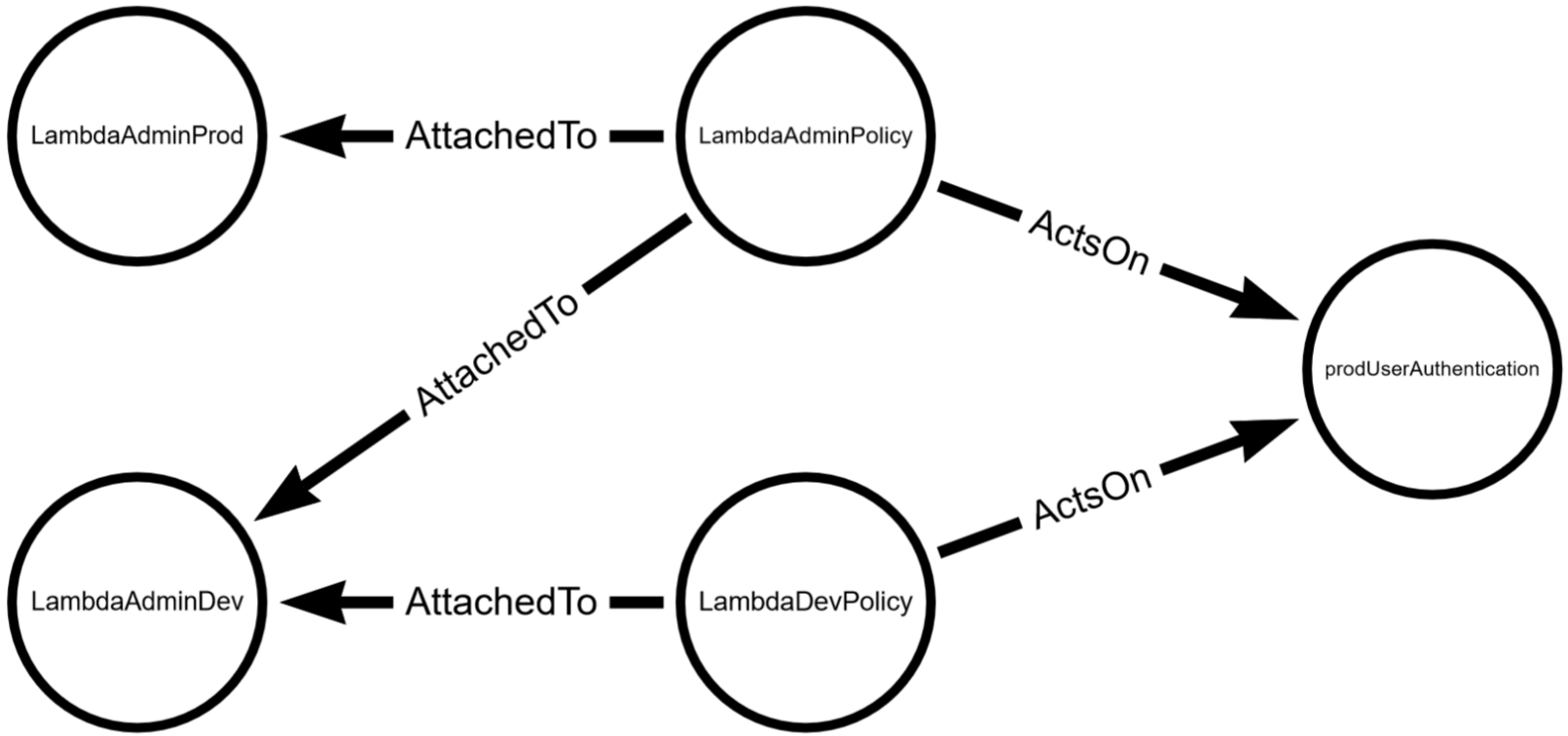


All Together

- We ***could*** stop here
- But we won't stop here
- Not everyone should need to learn cypher



What principals can act on a resource?



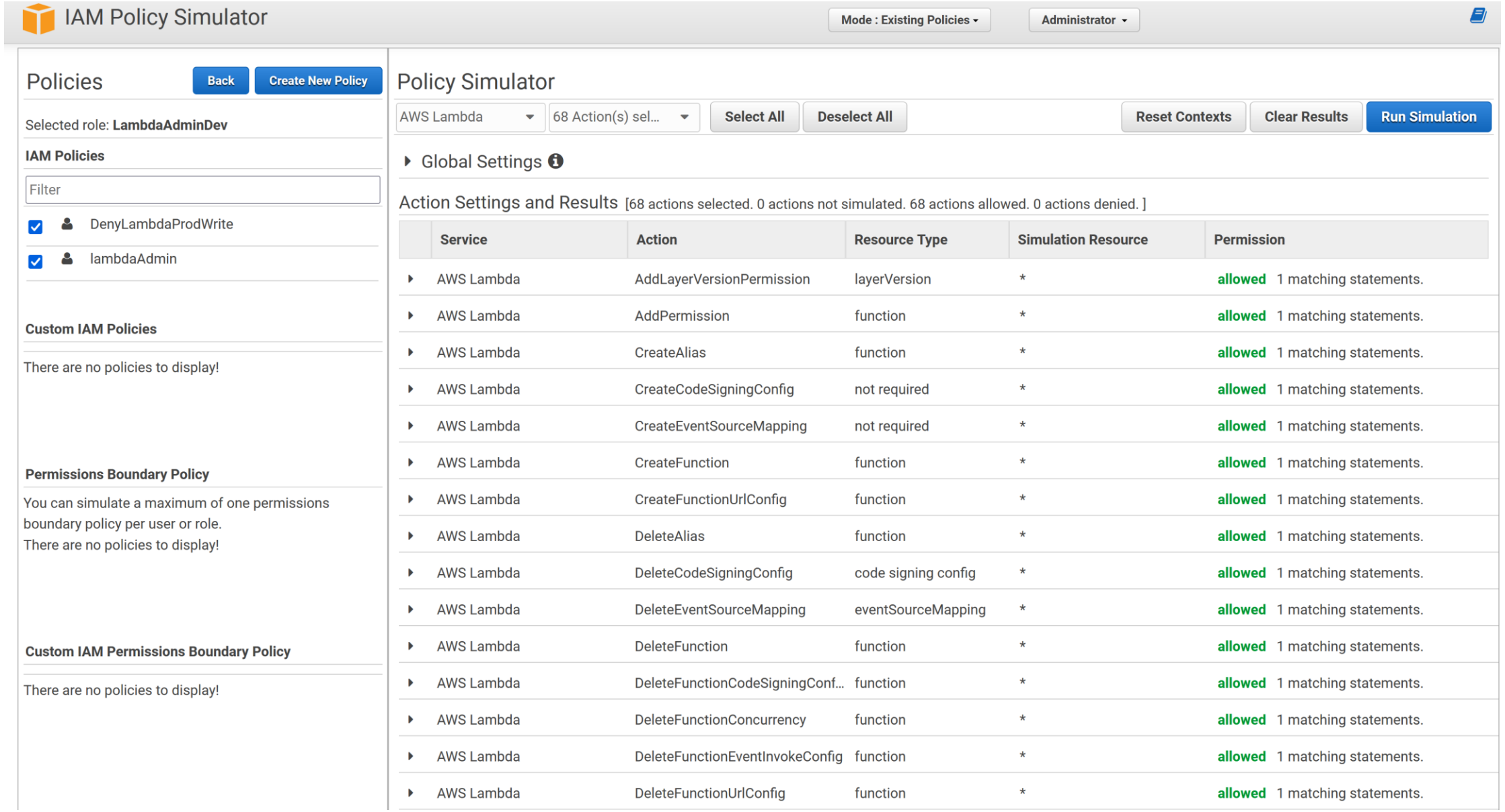
LambdaAdminDev policies

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lambda:*"
            ],
            "Resource": "*"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Deny",
            "Action": [
                "lambda:CreateFunction",
                "lambda:PublishLayerVersion",
                "lambda:DeleteProvisionedConcurrencyConfig",
                "lambda:InvokeAsync",
                "lambda:CreateEventSourceMapping",
                "lambda:PutFunctionConcurrency",
                "lambda:DeleteCodeSigningConfig",
...
                "lambda:PublishVersion",
                "lambda:DeleteFunctionConcurrency",
                "lambda:DeleteEventSourceMapping",
                "lambda:DeleteFunctionUrlConfig",
                "lambda:CreateAlias"
            ],
            "Resource": "arn:aws:lambda:*:*:function:prod*"
        }
    ]
}
```

Resource RSOP Psuedo Query

- Get all statements that have the target resource in scope
- Of those statements, get all effective statements
  - Effective statements contain an action that can act on that resource
  - {Action: *, Resource: s3:*} doesn't mean that one can call lambda:CreateFunction on an s3 object
- Of effective statements, resolve which policies they are attached to
- Resolve which principals have the policies attached to them
- For each principal, calculate RSOP for all effective policies attached to them
- Flag if a statement has a condition


IAM Policy Simulator
Mode : Existing Policies
Administrator
Policies
Back
Create New Policy
Selected role: LambdaAdminDev
IAM Policies
Filter
DenyLambdaProdWrite
lambdaAdmin
Custom IAM Policies
There are no policies to display!
Permissions Boundary Policy
You can simulate a maximum of one permissions boundary policy per user or role.
There are no policies to display!
Custom IAM Permissions Boundary Policy
There are no policies to display!
Policy Simulator
AWS Lambda
68 Action(s) sel...
Select All
Deselect All
Reset Contexts
Clear Results
Run Simulation
Global Settings
Action Settings and Results [68 actions selected. 0 actions not simulated. 68 actions allowed. 0 actions denied.]
Service
Action
Resource Type
Simulation Resource
Permission
AWS Lambda AddLayerVersionPermission layerVersion * allowed 1 matching statements.
AWS Lambda AddPermission function * allowed 1 matching statements.
AWS Lambda CreateAlias function * allowed 1 matching statements.
AWS Lambda CreateCodeSigningConfig not required * allowed 1 matching statements.
AWS Lambda CreateEventSourceMapping not required * allowed 1 matching statements.
AWS Lambda CreateFunction function * allowed 1 matching statements.
AWS Lambda CreateFunctionUrlConfig function * allowed 1 matching statements.
AWS Lambda DeleteAlias function * allowed 1 matching statements.
AWS Lambda DeleteCodeSigningConfig code signing config * allowed 1 matching statements.
AWS Lambda DeleteEventSourceMapping eventSourceMapping * allowed 1 matching statements.
AWS Lambda DeleteFunction function * allowed 1 matching statements.
AWS Lambda DeleteFunctionCodeSigningConf... function * allowed 1 matching statements.
AWS Lambda DeleteFunctionConcurrency function * allowed 1 matching statements.
AWS Lambda DeleteFunctionEventInvokeConfig function * allowed 1 matching statements.
AWS Lambda DeleteFunctionUrlConfig function * allowed 1 matching statements.

Demo 1

Introduction to Role Chaining

- Users, Roles, and IdentityProviders can assume roles in AWS
- When an assume role operation occurs, temporary credential are issued to authenticate as the destination role
- **Can be cross account**

arn:aws:iam::123456789012:role/RoleA

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::123456789012:role/RoleB"
        }
    ]
}
```

arn:aws:iam::123456789012:role/RoleB

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/RoleA"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

arn:aws:iam::123456789012:role/RoleA

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::210987654321:role/RoleB"
        }
    ]
}
```

arn:aws:iam::210987654321:role/RoleB

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/RoleA"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Demo 2

We have arrived at the start

What is Tier 0?

- Microsoft
  - Focused on identity infrastructure
- AWS is ***policy*** centric
  - If an attacker compromises an identity provider, they don’t have complete control over the account

Tier 0

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Sid": "Statement2",
            "Effect": "Deny",
            "Action": "braket:*",
            "Resource": "*"
        }
    ]
}
```

Tier 0 – A Loose Definition

- What are the bare minimum permissions needed for ***guaranteed*** control in ***any*** AWS account
  - (iam:attach*policy AND iam:detach*policy) AND (The ability to attach and detach a policy to a principal to which you can authenticate)
- There will be more definitions
- Every AWS environment can define their own Tier 0

Self Contained Tier 0

A subset of the Tier 0 definition

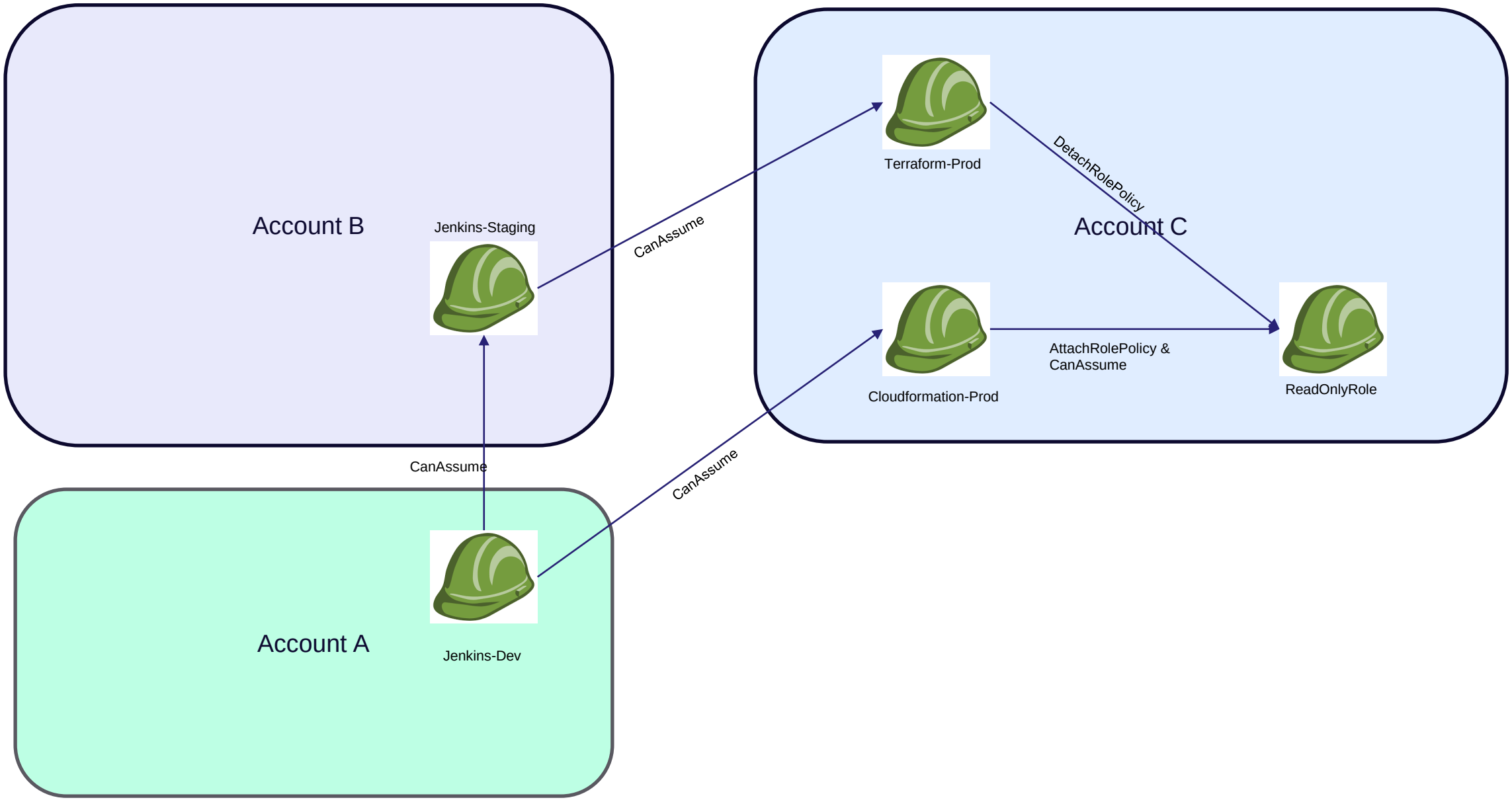
- A principal can attach and detach policies on themselves
  - Have the privileges to take over the account directly or indirectly without changing identity

Example: Client B

- Had an Administrator role that was explicitly blocked from performing attachrolepolicy on itself
  - It was named “Administrator”
- Over a dozen other roles could attach and detach role policies on themselves
- Identified with Apeman prototype

Tier 0 Paths

- iam:attach*policy, iam:detach*policy on a particular principal for the entirety of the path
- A collection of principals can also be tier 0, in aggregate
  - We can query for this, but it’s ugly
- For each principal in my path:
  - For all RSOP in the path
    - Do all the actions combined form a Tier 0 privilege set?


Account B
Jenkins-Staging
CanAssume
Terraform-Prod
DetachRolePolicy
Account C
Cloudformation-Prod
AttachRolePolicy &
CanAssume
ReadOnlyRole
CanAssume
CanAssume
Account A
Jenkins-Dev

Demo 3

Real Work Begins

- Providing the means to categorize the tiers of access in an account
- Identifying more absolute Tier 0 definition sets
- Common context templates
  - Context for common users in an Org
- Expand vocabulary around discussing AWS attack paths

Next Steps

Release before August

- Policy Evaluation
  - SCPs / Session Policies / Permission Boundaries
- Complete condition resolver engine
- Overhaul UI
- Testing / Productizing

SPECTEROPS

Questions

Daniel Heinsen | @hotnops | dheinsen@specterops.com

SPECTEROPS

Presentation Overview

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Praesent vehicula aliquam magna sed ornare.