Beyond Service Control Policies (SCPs)

Achieve more granular control in AWS

Author: Christophe PARISEL

Introduction

In AWS, Service Control Policies (SCPs) provide a powerful way to enforce governance across multiple accounts within AWS Organizations. Their preventive nature ensures that permissions never exceed prescribed limits—but this “hard cap” approach can miss the nuance needed for context-sensitive control.

In contrast, Azure offers finer grain permissions management services: the well-known preventive Azure Policy, and a lesser-known built-in hierarchical model for resource management, naturally enforcing structured access controls [1]

What if we could apply lessons from Azure to enhance AWS control plane security? We explore how a detective approach using Attribute-Based Access Control (ABAC) and hierarchical clustering can provide deeper visibility and context for AWS operations beyond what SCPs offer alone.

Problem statement

Consider an organization with three separate AWS accounts for different banking departments: one for Forex, one for Bonds, and one for Private Banking. Using native SCPs, you can restrict each account's actions, but SCPs operate only at the account level—they don't distinguish resource relationships within or across accounts.

For example, suppose you want to allow the Forex department (Account 3) to access some resources of the Bonds department (Account 2) but not to Private Banking resources (Account 1). In Azure, you can inspect activity logs and calculate the distance (derived from Azure's intrinsic hierarchical resource IDs) between any source and target account pairs. If Forex and Bond resources reside in the same local "Trading" hierarchy, cross-account activity involving Forex and Bonds will be closer (from a distance perspective) than activity involving Forex and Private Banking.

In AWS, SCPs alone cannot make such distinctions because they lack visibility into resource relationships. While IAM policies and resource-based policies can restrict cross-account access, they require manual, error-prone configuration:

- IAM Roles: Forex (Account 3) can assume an IAM role in Bonds (Account 2) with policies scoped to Bond-tagged resources.
- Resource Policies: S3 bucket policies in Bonds (Account 2) can explicitly allow access to Forex (Account 3).

Complexity and lack of visibility on actual permissions is further increased by the transitive nature of roles assumption.

Azure's Advantage: A Strong Hierarchical Model

Azure organizes resources through a clear, structured hierarchy:

1. Management Groups – High-level grouping for multiple subscriptions.
2. Subscriptions – Billing and administrative boundaries.
3. Resource Groups – Logical containers for resources with shared lifecycles.

This built-in hierarchy acts as a ground truth for governance. A rigorously enforced tag taxonomy (or intrinsic resource ID structure) can serve as a reliable source for:

- Risk assessments: For example, "Is this cross-department modification expected?"
- Compliance audits: Such as verifying that all production Finance resources have aligned controls.

Furthermore, Azure's structure naturally enables a distance between resources—allowing one to measure the distance between any two resources and placing explicit upper bounds not to be trespassed. This mathematical distance is ultrametric: it satisfies a strong form of the triangle inequality: for any three points A, B, C,

$$d(A,C) \leq \max(d(A,B), d(B,C))$$

In Azure, this means the distance between resources is determined by their *lowest common ancestor* in the hierarchy: a very desirable feature for scalability and automatability.

Applying Azure's Lessons to AWS

While AWS lacks an intrinsic hierarchy like Azure, disciplined tagging can emulate it. By superimposing a clustering hierarchy using ABAC and AWS Resource Groups, organizations can:

- Establish Hierarchical Tagging: Define relationships explicitly via tags (e.g., mirroring AWS Organizations' OUs such as OU=Finance, OU=HR).
- Compute Custom Distance Metrics: Use these tags to calculate the ultrametric distance between resources.
- Enable Detective Controls: Integrate custom tools with AWS Config, CloudTrail, and IAM Access Analyzer to flag unexpected or "high distance" interactions, which can then be subject to post-hoc analysis.

For example:

- Forex (Tags: Division=Trading/Dept=Forex)
- Bonds (Tags: Division=Trading/Dept=Bonds)
- Private Banking (Tags: Division=Retail/Dept=PrivateBanking)

The (ultrametric) distance between Forex and Bond accountq is 1 (shared Division=Trading), while Forex and Private Banking accounts stand 2 units apart (the lowest common ancestor is the root).

In practice, this emulated hierarchy is extended to principals and resources:

A trader assigned to Division=Trading/Dept=Forex can be allowed to assume a role in Division=Trading/Dept=Bonds to get or put data in a given S3 bucket named "Accounting" tagged Division=Retail/Dept=PrivateBanking

The extension can be horizontal, have we've just seen, but also vertical: for example, we may add an extra Activity tag to the hierarchy:

In this case, the distance between trader's principal ID tagged Division=Trading/Dept=Forex/Activity=Arbitration and the Accounting's ARN tagged Division=Retail/Dept=PrivateBanking/Activity=Arbitration becomes 3.

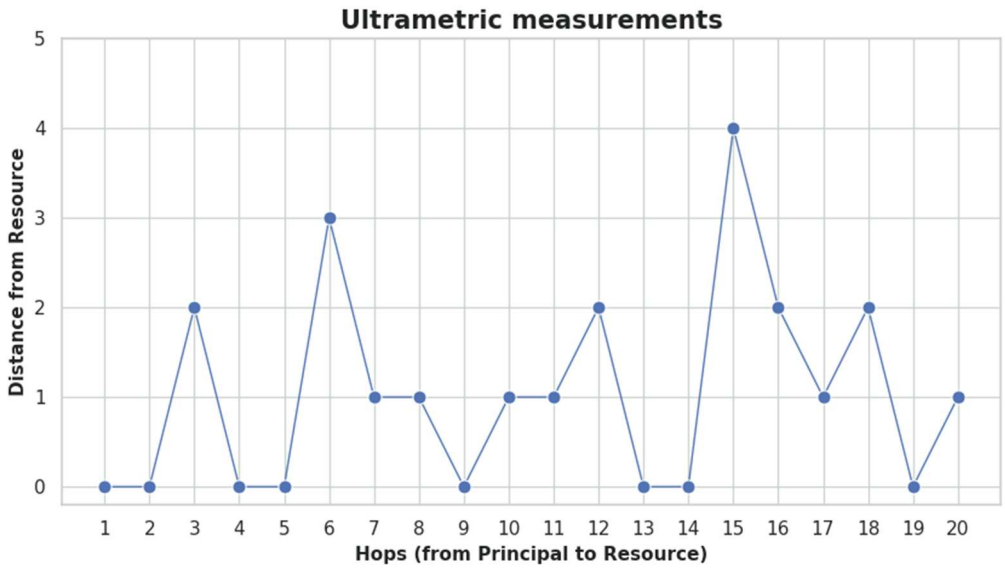
Lastly, and crucially, to consider the graph nature of transitive roles, we calculate the distances of all intermediate data points to determine the maximum distance from the target (the requested resource):

In our example, suppose the intermediate role is now attached to Division=Retail/Dept=PrivateBanking/Activity=Accounting. Therefore, it stands 1 unit apart from the target bucket.

In the trader => intermediate role => bucket path, the maximum distance to the bucket never exceeds 3.

Whether this distance of 3 is acceptable or not from a risks perspective depends on two factors: 1) the max() value of the strong triangle inequality, 2) the compliance thresholds which depend on your local use cases.

Here is a theoretical example showing a *deep path* involving 20 hops between a source principal and a target resource:

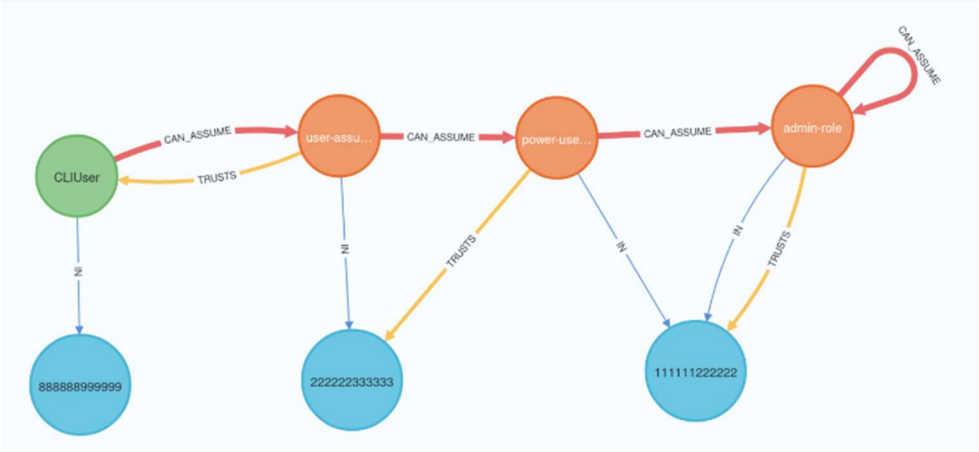


We see that the ultrametric distance to the target never exceeds 3 unit, except at hop number 15 which might be suspicious.

Ultrametry calculation

One might be tempted to generate the transitive closure of role assumptions and calculate the ultrametric distance of all role pairs, once and for all. This, however, dismisses the fact that measurements must be made *to the target resource*, so it is not local: it is global, for all resources.

To help in this task, graph queries can identify all intermediate points in the path. This "reachability analysis" can be done either out-of-band, using a tool like pmapper[2], IAMGraph[3] or in-band, feeding from "live" AWS CloudTrail logs and using custom scripts.



IAMGraph (source: labs.withsecure.com/tools/iamgraph)

The latter approach has two important benefits:

1. the actual resource ARN of the target asset is pre-defined, unlike the former which relies on the famous and all-to-common wildcard conditions in IAM configurations.
2. computation only happens when required ("backed by evidence"), and the ultrametric measurement can be cached if similar events are excepted, which is often the case. This greatly reduces calculation times.

Conclusion

This detective approach augments—but does not replace—existing AWS preventive ABAC policies. While detective controls identify fine-grain cross-hierarchy activity, keep in mind that preventive ABAC policies can block unauthorized actions upfront, hence they remain the prime way to go.

[1]: https://www.linkedin.com/pulse/adversarial-lateral-motion-azure-paas-we-prepared-christophe-parisel-hyrpe/

[2]: https://github.com/nccgroup/PMapper

[3]: https://github.com/WithSecureLabs/IAMGraph