SANS | CLOUD SECURITY

Stop Playing Security Whack-a-Mole: Enforcing Cloud Security with Organizational Controls

Kenneth G. Hartman

TUESDAY, OCTOBER 21, 2025

About Me

LUCID TRUTH TECHNOLOGIES
LucidTruthTechnologies.com

Human Resilience Project
humanresilienceproject.org

Head in the Clouds YouTube Video Series
headintheclouds.site

Kenneth G. Hartman

- Owner – Lucid Truth Technologies, a Digital Forensics Firm
- BS Electrical Engineering, Michigan Technological University
- MS Information Security Engineering, SANS Technology Institute
- Multiple Security Certifications: CISSP, GIAC Security Expert, etc.
- SANS Instructor – SEC502: Cloud Security Tactical Defense
  & SEC510: Cloud Security Engineering and Controls

www.kennethghartman.com
@kennethghartman

The content and opinions in this presentation are my own and do not necessarily reflect the positions, strategies, or opinions of any current client or previous employer.


WHAC-A-MOLE
30
10 20 50 60 80

The Case for Organizational Controls

Why shift to preventive, org-level controls?

The Problem:

- **Alert Fatigue:** Security teams are overwhelmed playing Whack-a-Mole with misconfigurations
- **Resource Drain:** Time spent on basic violations instead of sophisticated threats
- **Compliance Gaps:** Inconsistent security postures across cloud environments

The Solution:

- **Block Before Deploy:** Prevent misconfigurations at the source
- **Scale Automatically:** Policies inherit across entire organizational hierarchies
- **Reduce Noise:** Fewer alerts mean higher priority alerts get attention
- **Multi-Cloud Consistency:** Same governance approach across AWS, Azure, GCP

Organization-level Controls Comparison

Cloud Service Provider	Name of Org-Level Control
Amazon Web Services	Service Control Policies
Microsoft Azure	Azure Policy
Google Cloud	Organization Constraints + Deny Policies

AWS Service Control Policies (SCPs) – Overview

What are SCPs?

- Org-wide permission guardrails in AWS Organizations
- Define *maximum* available permissions (permission ceilings)
- Apply to Organization → Organizational Units → Accounts
- Work alongside IAM policies (SCPs restrict, IAM grants)
- **Key Concept:** *SCPs don't grant permission; they define the ceiling of what's possible*

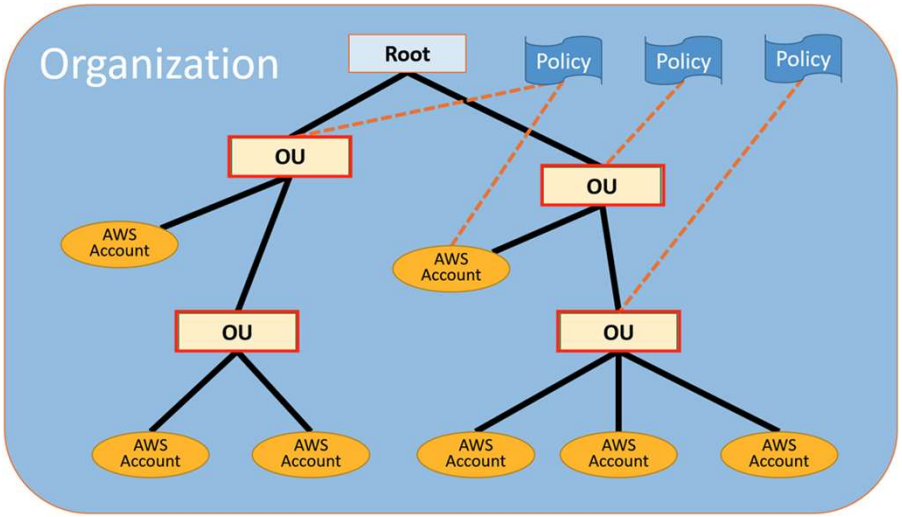
AWS SCPs – Key Mechanics & Use-Cases

How SCPs Work:

- **Evaluation Logic:** Request must be allowed by IAM *AND* not denied by SCP
- **Inheritance:** SCPs attached to parent OUs apply to all child accounts
- **Multiple SCPs:** All attached SCPs must allow the action (most restrictive wins)

Common Use Cases:

- Deny root user login (or require MFA)
- Prevent public S3 buckets
- Force encryption on EBS volumes and S3
- Restrict deployment to approved regions
- Block access to specific AWS services



AWS SCPs – Testing & Validation

Testing Strategy:

- **Sandbox First:** Test SCPs in an isolated OU with non-production accounts
- **Real Action Testing:** Test SCPs by attempting actual actions that should be denied
- **Gradual Rollout:** Start with audit mode, monitor impact, then enforce
- **Documentation:** Maintain clear evaluation logic and exemption procedures

Validation Methods:

- **Real Action Testing:** Test SCPs by attempting actual actions
- **Service Last Accessed Data:** Identify which services need SCP restrictions
- **CloudTrail Analysis:** Review actual API calls to understand impact
- **Break-Glass Procedures:** Document emergency SCP bypass processes

Common Pitfalls to Avoid:

- Deploying SCPs directly to production accounts
- Not testing with real user scenarios
- Over-restrictive policies that break legitimate operations
- Lack of exemption procedures for emergency situations

Azure Policy – Overview

What is Azure Policy?

- **Governance Service:** Define, assign, and manage policy definitions across Azure resources
- **Multi-Purpose:** Addresses resource consistency, compliance, cost, and security
- **Hierarchical:** Policies apply from Management Groups → Subscriptions → Resource Groups → Resources
- **Policy-as-Code:** Define policies using JSON, ARM templates, Bicep, or Terraform

Key Capabilities:

- **Preventive Controls:** Block non-compliant resources before creation
- **Audit Mode:** Log violations without blocking (unlike AWS SCPs)
- **Auto-Remediation:** Automatically fix non-compliant resources
- **Built-in Policies:** 200+ pre-built policy definitions available

Azure Policy – Effects & Deny Mechanism

Policy Effects Overview:

- **Deny:** Blocks non-compliant resource creation/updates (403 error)
- **Audit:** Logs violations without blocking (Azure's advantage over AWS SCPs)
- **Modify:** Automatically adds required properties to resources
- **DeployIfNotExists:** Deploys remediation resources when conditions aren't met
- **Append:** Adds additional properties to existing resources

The Deny Effect in Detail:

- **Evaluation:** Happens during resource creation/update requests
- **Blocking:** Returns 403 Forbidden before resource is created
- **Scope:** Can target specific resource types, locations, or conditions
- **Exemptions:** Management Group admins can create exemptions for legitimate cases

Azure Policy – Built-in Definitions & Implementation

Built-in Policy Library:

- **200+ Pre-built Definitions:** Covering security, compliance, cost, and governance
- **Categories:** Security Center, Storage, Compute, Network, Identity, and more
- **Policy Initiatives:** Grouped sets of related policies for comprehensive coverage
- **Regular Updates:** Microsoft continuously adds new policies based on best practices

Implementation Best Practices:

- **Start with Audit Mode:** Deploy policies in audit mode first to understand impact
- **Gradual Rollout:** Begin with non-critical resources, then expand scope
- **Policy-as-Code:** Define policies using ARM templates, Bicep, or Terraform
- **Continuous Validation:** Integrate with CI/CD pipelines for ongoing compliance

Integration with Third-Party Tools:

- **CSPM Platforms:** Azure Policy complements Cloud Security Posture Management tools
- **SIEM Integration:** Policy compliance data feeds into security monitoring
- **Compliance Reporting:** Automated compliance dashboards and reports

GCP Org Constraints – Overview

What are GCP Organization Policy Service / Org Constraints?

- **Hierarchical Control:** Define allowed/denied configurations at org/folder/project levels
- **Limited Scope:** Only ~30 built-in constraints available (unlike Azure's 200+ policies)
- **Organization Requirement:** Projects must be part of a GCP Organization to use constraints
- **Inheritance Model:** Policies inherit downward from Organization → Folders → Projects

Key Constraint Examples:

- **External IP Access:** `compute.vmExternalIpAccess` - restrict VM external IPs
- **Resource Locations:** `gcp.resourceLocations` - limit where resources can be created
- **Domain Restrictions:** `iam.allowedPolicyMemberDomains` - control who can access resources
- **Service Account Creation:** `iam.disableServiceAccountCreation` - prevent service account sprawl

GCP Deny Policies – Overview

What are GCP IAM Deny Policies?

- **Deny-First Evaluation:** Conditional deny rules evaluated before allow policies
- **Hierarchical Attachment:** Attach at org, folder, or project levels
- **Conditional Logic:** Support complex conditions using Common Expression Language (CEL)
- **Flexible Targeting:** Can target specific principals, permissions, or resources

Key Advantages over Org Constraints:

- **Custom Conditions:** Create complex conditional logic (unlike Org Constraints)
- **Granular Control:** Target specific users, groups, or service accounts
- **Resource-Specific:** Apply policies based on resource attributes
- **Comprehensive Coverage:** Work with any GCP service (not limited like Org Constraints)

GCP Deny Policies – Example

```
{
 "denyRules": [
   {
     "denyCondition": {
        "expression": "request.auth.claims.email_verified == false ||
!request.auth.claims.email.matches('.*@yourcompany\\.com$')"
     },
     "deniedPrincipals": ["allUsers"],
      "deniedPermissions": [
        "storage.objects.get",
        "storage.objects.create",
        "storage.objects.update",
        "storage.objects.delete"
      ],
      "deniedResourceNames": [
        "projects/*/buckets/sensitive-data-*"
      ]
   }
 ]
}
```

Example: Prevent External Users from Accessing Sensitive Resources

What This Policy Does:

- **Email Verification:** Blocks users with unverified email addresses
- **Domain Restriction:** Only allows users from your company domain
- **Sensitive Bucket Protection:** Applies to buckets with "sensitive-data-" prefix
- **Comprehensive Storage Control:** Blocks all object-level storage operations

GCP Deny Policies – Best Practices

- **Start with Dry-Run Mode:** Test policies in dry-run mode (audit mode) before enforcement
- **Use Specific Resource Names:** Target specific resources rather than broad patterns
- **Test with Real Users:** Validate policies with actual user scenarios
- **Document Exemptions:** Maintain clear procedures for legitimate exceptions

When you are so busy fighting alligators, it is hard to drain the swamp

Integrating Prevention & Detection

The Synergy Model:

- **Prevention Reduces Noise:** Organizational controls eliminate entire classes of misconfigurations
- **Detection Validates Prevention:** Monitoring confirms controls are working and catches bypasses
- **Signal Amplification:** Fewer false positives mean higher priority alerts get attention
- **Resource Optimization:** Security teams can focus on sophisticated threats, not basic misconfigurations

Real-World Integration Examples:

- **AWS SCP + CloudTrail:** SCP prevents public S3 buckets, CloudTrail alerts if SCP is bypassed
- **Azure Policy + Defender for Cloud:** Policy enforces encryption, Defender for Cloud monitors compliance
- **GCP Deny + Security Command Center:** Deny Policy blocks external access, SCC monitors compliance and policy violations

Implementation Strategy: Start to Scale

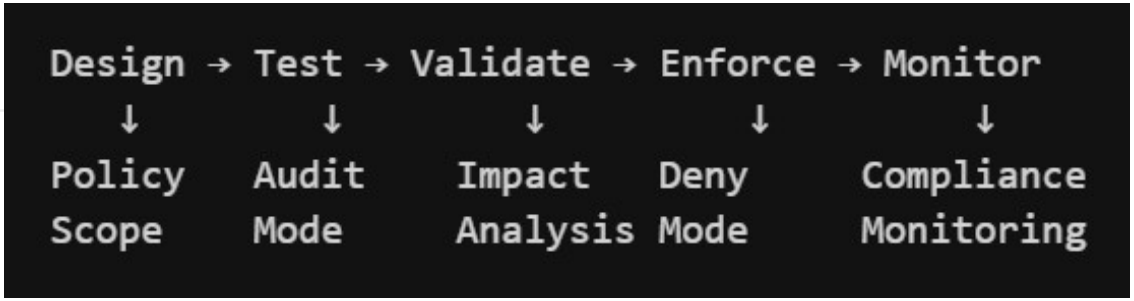
The Incremental Rollout Approach:

- **Phase 1: Start Small:** Begin with one high-impact control in non-production environment
- **Phase 2: Test & Validate:** Use audit mode (Azure/GCP) or sandbox testing (AWS) to understand impact
- **Phase 3: Gradual Enforcement:** Move from audit to deny mode, expanding scope incrementally
- **Phase 4: IaC Integration:** Embed policies in Infrastructure as Code for consistent deployment

Key Success Factors:

- **Dedicated Security Teams:** 51% already have multicloud security teams (p.7)
- **Cross-Functional Collaboration:** Include DevOps, compliance, and business stakeholders
- **Success Metrics:** Track alert reduction, compliance improvement, and operational impact

The Preventive Policy Lifecycle



Key Lifecycle Components:

- **Design:** Define policy scope, conditions, and exemptions
- **Test:** Use audit mode or sandbox testing to understand impact
- **Validate:** Analyze results and refine policy before enforcement
- **Enforce:** Deploy policy in deny mode with proper scope
- **Monitor:** Track compliance, violations, and operational impact

Maturity Progression:

- **Intermediate:** Reactive security with some preventive controls
- **Advanced:** Proactive prevention with comprehensive organizational controls
- **Lifecycle Discipline:** Systematic approach to policy management
- **Continuous Improvement:** Regular review and refinement of policies

AWS Service Control Policies:

- AWS SCPs:
  https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html
- SCP Syntax:
  https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps_syntax.html

Azure Policy:

- Overview:
  https://learn.microsoft.com/en-us/azure/governance/policy/overview
- Deny Effect:
  https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effect-deny

GCP Deny Policies:

- Overview:
  https://cloud.google.com/iam/docs/deny-overview
- Deny Access:
  https://cloud.google.com/iam/docs/deny-access

Conclusion

Ready to Deep-Dive into Preventive Cloud Security?

➔Enroll in **SEC510: Cloud Security Engineering and Controls**.

- Hands-on labs for AWS, Azure, GCP org-level controls.
- Learn to design, test, and maintain preventive controls at scale.
- Move from reactive to proactive defense.

https://www.sans.org/cyber-security-courses/cloud-security-engineering-controls