International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

Governance at Scale: Managing IAM and Policy Enforcement across AWS and GCP

Harish Govinda Gowda
Devops Engineer, Promates Technologies LLC

Abstract- As enterprises accelerate their multi-cloud strategies, managing Identity and Access Management (IAM) and enforcing governance policies across platforms like AWS and GCP has become a top priority. These cloud providers offer distinct IAM models, policy enforcement tools, and logging mechanisms, creating complexity for organizations seeking consistent security, compliance, and operational control. This article explores a comprehensive governance framework for managing IAM and policy enforcement at scale in a dual-cloud environment. It examines core architectural principles, identity federation strategies, scalable IAM design, and automation practices using infrastructure-as-code and policy-as-code tools. Additionally, it highlights native policy enforcement mechanisms such as AWS Service Control Policies and GCP Organization Constraints, while outlining approaches for centralized monitoring, auditing, and anomaly detection. Through a real-world case study of a financial services platform, the article illustrates how cross-cloud governance can be automated, monitored, and evolved to meet business and regulatory demands. The piece concludes with lessons learned, technical recommendations, and a blueprint for sustainable cloud governance in large-scale environments.

Index Terms- Multi-cloud governance, AWS IAM, GCP IAM, identity federation, single sign-on (SSO).

I. INTRODUCTION TO CLOUD GOVERNANCE

As enterprises accelerate their adoption of cloud technologies, managing governance at scale becomes a cornerstone of maintaining security, compliance, and operational integrity. Governance in the cloud is not merely about managing users and permissions—it's about establishing clear policies that control who can access what resources, under what conditions, and with what visibility. In multi-cloud environments, particularly those spanning AWS and GCP, governance presents both an opportunity for robust security design and a challenge due to the differing models, terminology, and enforcement mechanisms each provider uses.

AWS and GCP offer mature Identity and Access Management (IAM) solutions, yet they approach governance from different architectural philosophies. AWS follows a more granular and resource-scoped approach with policies attached to users, roles, or services. GCP, on the other hand, emphasizes hierarchy-based inheritance, where IAM policies are applied to resources such as projects, folders, and organizations and then flow downward.

As organizations scale up their cloud usage—spanning multiple accounts, projects, and teams—these differences in policy structure and enforcement become more pronounced, and the need for consistent, auditable governance intensifies.
One of the major drivers for strong governance is compliance with industry standards and regulations such as HIPAA, PCI-DSS, and GDPR. Failing to manage IAM properly can lead to excessive privileges, orphaned access, data leaks, and audit failures. Moreover, with the advent of infrastructure-as-code and self-service cloud provisioning, cloud environments are growing faster than traditional security teams can manage manually. This necessitates automated, policy-driven governance models that scale with the business while still enforcing best practices and guardrails.

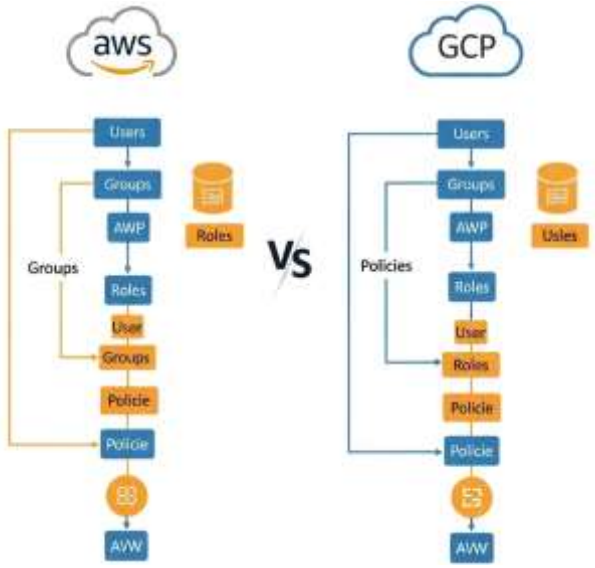
Governance also extends beyond identity and access to include tagging policies, resource organization, data sovereignty, and usage controls. Effective cloud governance requires collaboration between security, DevOps, platform, and compliance teams. It also demands the right tools and frameworks to audit access, detect policy drift, and respond to violations proactively.

This article explores the practical implementation of scalable IAM and policy enforcement across AWS and GCP. By analyzing each cloud provider's IAM architecture, policy enforcement capabilities, and automation tools, we provide a comprehensive guide for enterprises to build resilient governance frameworks that work seamlessly in a multi-cloud world.

© 2024 IJSRET

International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

II. IAM ARCHITECTURE IN AWS AND GCP



IAM architecture in AWS and GCP

Identity and Access Management (IAM) is the foundation of security and governance in cloud environments. Both AWS and GCP offer powerful IAM systems, but they differ significantly in structure, nomenclature, and management paradigms. Understanding these differences is key to designing scalable and maintainable access control models in multi-cloud organizations.

In AWS, IAM revolves around users, groups, roles, and policies. Permissions are defined using JSON-based policies that are either identity-based (attached to users or roles) or resource-based (attached to resources like S3 buckets or Lambda functions). AWS also offers service-linked roles and session policies for temporary access. For organizations managing multiple AWS accounts, AWS Organizations provides Service Control Policies (SCPs), which act as permission boundaries at the account or organizational unit level. SCPs do not grant permissions themselves but set limits on what permissions IAM identities can be granted, making them a powerful tool for governance at scale.

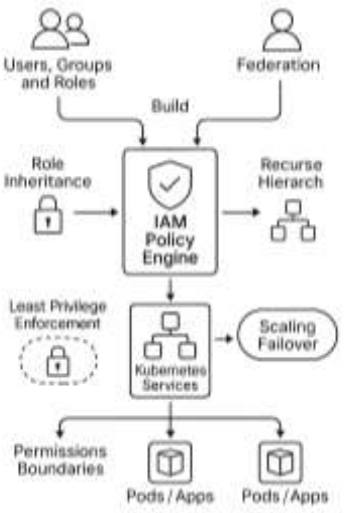
GCP takes a different approach with a hierarchical IAM model. Resources are organized under organizations, folders, and projects, and permissions are inherited down the hierarchy. Instead of explicit users and groups at every level, GCP defines roles (primitive, predefined, or custom) and binds them to members (users, groups, or service accounts) at a given resource level. GCP IAM policies are also defined in JSON but are usually shorter and more focused on binding roles to principals. Organization Policies in GCP allow enforcement of constraints, such as allowed regions or restrictions on external IP addresses, further tightening governance.

One of the architectural contrasts is how AWS and GCP manage trust and delegation. AWS uses AssumeRole and policy conditions for cross-account or federated access. GCP uses workload identity federation, service account impersonation, and context-aware access to manage similar scenarios. While both clouds support integration with identity providers using SAML or OIDC, the implementations differ, and harmonizing identity across both platforms often requires an external IdP and custom federation logic.

In multi-cloud organizations, these architectural differences present both challenges and opportunities. While AWS offers more granular control at the resource level, GCP provides easier top-down management via hierarchy. Effective governance requires embracing these paradigms while enforcing consistency through IaC tools, policy-as-code frameworks, and standardized role definitions. Aligning IAM models with business structure, compliance goals, and operational workflows is critical to scaling governance across AWS and GCP environments.

III. DESIGNING SCALABLE IAM MODELS

Designing IAM (Identity and Access Management) models that scale effectively across cloud environments is a complex task requiring a balance between operational efficiency, security, and usability. As cloud usage expands across teams, departments, and global regions, the IAM model must support growth while minimizing risk and complexity. Scalability in IAM doesn't just mean supporting more users or services—it means doing so without policy sprawl, permission overlaps, or unmanaged access.



Access Management (IAM) model for cloud environments

© 2024 IJSRET

One of the key principles in scalable IAM design is least privilege, where users and services are granted only the permissions necessary to perform their tasks. Implementing least privilege requires deep visibility into resource usage patterns and a clear understanding of access requirements. Tools like AWS Access Analyzer and GCP Recommender help identify over-permissive roles and unused privileges, enabling rightsizing over time.

Another cornerstone of scalable IAM is role-based access control (RBAC), which simplifies permission management by grouping actions into roles and assigning those roles to users or service accounts. In AWS, roles can be assumed by services or users, enabling temporary access and reducing long-lived credentials. GCP binds roles to members at the project or organizational level, making inheritance a powerful mechanism for wide-scale access management. Creating reusable role templates and mapping them to job functions helps maintain consistency across accounts, projects, and clouds.

Organizations must also decide between centralized and decentralized IAM management. Centralized models offer control and consistency but may slow down provisioning and limit team autonomy.

Decentralized models empower teams but risk drift and inconsistency without strong governance controls. Hybrid approaches—such as allowing teams to manage IAM within pre-approved guardrails enforced by SCPs or GCP constraints—strike a balance between security and agility.

Inheritance and nested resource hierarchies further complicate IAM design. In AWS, permissions do not inherit across accounts unless configured explicitly via AssumeRole and trust relationships.

In GCP, resource hierarchy allows IAM bindings at the organization, folder, or project level, cascading downward. This feature makes it easier to apply global controls, but it also introduces the risk of unintentionally granting broad access if roles are bound too high in the hierarchy.

Scalable IAM design also benefits from infrastructure as code (IaC) practices. Using tools like Terraform or Pulumi, teams can codify IAM roles, bindings, and policies, review changes through pull requests, and integrate access provisioning into CI/CD pipelines. This makes IAM auditable, version-controlled, and reproducible across environments.

Ultimately, a scalable IAM model is proactive, standardized, and enforced through automation—supporting the pace of cloud adoption without compromising security or compliance.

IV. CROSS-CLOUD IDENTITY FEDERATION AND SSO

In multi-cloud environments, maintaining consistent user identity and access control across AWS and GCP is one of the most challenging aspects of governance. Cross-cloud identity federation and single sign-on (SSO) offer a solution to unify identity management, reduce credential sprawl, and provide a consistent access experience for users and services. With the right federation architecture, organizations can centralize identity control while still allowing cloud-native workloads to function securely and independently.

At the foundation of identity federation is the integration between the cloud provider's IAM system and an external identity provider (IdP)—such as Okta, Azure AD, Google Workspace, or an enterprise LDAP/SAML directory. This setup enables organizations to authenticate users via SAML or OpenID Connect (OIDC), allowing centralized control over authentication flows and user lifecycles. Both AWS and GCP support federation with external IdPs, but their implementation details differ.

In AWS, federation typically involves IAM roles that can be assumed by federated users. The IdP issues a SAML assertion that maps the user to an IAM role with defined permissions. This approach enables just-in-time access and removes the need to manually manage IAM users in AWS. For workloads, AWS now supports OIDC-based workload identity federation, allowing external identities—such as GitHub Actions or other CI/CD systems—to assume roles securely without creating static credentials.

GCP handles identity federation through Google Cloud Identity and workload identity federation. For user access, GCP supports SAML and OIDC-based federation to allow users from external IdPs to access GCP resources via mapped groups or roles. For services, GCP's workload identity federation lets non-Google workloads authenticate to GCP APIs using external identity tokens, such as OIDC tokens from AWS or GitHub, without storing long-lived service account keys.

These federation models also support SSO, enabling users to log into both AWS and GCP consoles using the same enterprise credentials. This simplifies user onboarding and offboarding and enhances compliance by centralizing identity lifecycle management.

Cross-cloud federation introduces trust boundaries that must be tightly managed. This includes validating IdP certificates, ensuring token expiration is short, and auditing federated access regularly. Additionally, context-aware access controls,

© 2024 IJSRET

such as device trust, location, or IP, can be applied to further harden the trust model.

Ultimately, identity federation and SSO unify the authentication layer across clouds, reduce operational overhead, and enhance security—especially when combined with conditional access, audit logging, and policy enforcement layers.

V. POLICY ENFORCEMENT MECHANISMS

Effective policy enforcement is at the core of scalable cloud governance. As cloud environments grow more dynamic, manually managing identity and access control becomes impractical. To ensure consistency, prevent configuration drift, and maintain compliance, enterprises must implement automated, enforceable guardrails across all cloud accounts and projects. AWS and GCP both provide policy enforcement mechanisms, but they differ in their structure, scope, and level of control.

In AWS, governance begins with Service Control Policies (SCPs), which are part of AWS Organizations. SCPs allow administrators to define a set of maximum permissions that apply to accounts or organizational units. These policies restrict what IAM identities in those accounts can do, regardless of what their attached IAM policies say. This makes SCPs powerful for enforcing organizational-wide security constraints, such as denying access to specific services (e.g., disabling IAM user creation) or enforcing usage of specific regions. SCPs act as a permissions boundary and do not grant access by themselves—they only limit what is possible.

AWS also provides tagging policies, which ensure resources are tagged consistently across accounts. This supports billing, automation, and access controls. Another key feature is permissions boundaries, which let you define the maximum permissions an IAM entity can have, serving as a fine-grained policy enforcement layer for delegated access.

In GCP, policy enforcement is achieved through the Organization Policy Service. This service allows organizations to set constraints on GCP resources and APIs. Constraints include enforcing resource locations, disabling external IPs, requiring CMEK encryption, or limiting which services can be enabled. These policies can be applied at the organization, folder, or project level and inherit downward. GCP also offers Policy Library, a collection of pre-defined constraints and templates that simplify enforcement and compliance.

A growing trend in both clouds is the use of policy-as-code frameworks that allow enforcement of custom policies using tools like Open Policy Agent (OPA) and Rego. These tools integrate with CI/CD pipelines and can validate infrastructure definitions before deployment. For Kubernetes environments, platforms like Gatekeeper and Config Sync (in GCP) extend this model to Kubernetes-native resources.

Together, these enforcement mechanisms form a multi-layered governance model that prevents unauthorized access, enforces compliance, and enables teams to operate within safe, predefined boundaries without slowing innovation or requiring constant manual oversight.

VI. MONITORING, AUDITING, AND GOVERNANCE TOOLING

In any cloud governance framework, visibility is essential. Without robust monitoring and auditing, it's impossible to know who accessed what, when, and under what conditions—leaving gaps that can lead to security incidents, compliance failures, or policy violations. AWS and GCP both provide comprehensive tooling to support continuous auditing and monitoring of IAM and policy enforcement activities across their respective environments.

AWS CloudTrail is the cornerstone of audit logging in AWS. It records all API calls and management operations across AWS services and accounts, including those performed via the console, SDKs, and command-line tools. CloudTrail logs can be sent to S3 for long-term storage and analysis or integrated with Amazon CloudWatch for real-time alerting. For deeper insights into IAM usage and permission analysis, AWS provides IAM Access Analyzer, which helps identify unused permissions, overly permissive roles, and resources shared publicly or with external accounts.

Complementing CloudTrail is AWS Config, a service that tracks configuration changes and resource compliance against defined rules. Config allows teams to define custom or managed rules to continuously evaluate AWS resources and flag drift or misconfigurations. Combined with AWS Security Hub, organizations can aggregate security findings and prioritize remediation efforts using a centralized dashboard.

In GCP, the equivalent to CloudTrail is Cloud Audit Logs, which offer detailed visibility into admin activity, data access, and system events across all GCP services. These logs are categorized into admin activity logs, data access logs, and system logs, and can be exported to Cloud Logging, BigQuery, or Pub/Sub for analysis and monitoring. GCP also offers Policy Analyzer, a tool that helps audit IAM policies and detect overbroad or unintended access across projects and organizations.

For continuous policy validation, tools like Config Validator and Forseti Security in GCP allow scanning for violations against predefined policy sets. Forseti, for example, can detect

© 2024 IJSRET

misconfigured firewall rules, public storage buckets, and identity misconfigurations. Additionally, Chronicle and Security Command Center provide advanced threat detection and risk analysis capabilities.

To manage visibility across both AWS and GCP in a multi-cloud environment, organizations often integrate audit logs with SIEM platforms like Splunk, Datadog, or Elastic, where events are normalized, correlated, and monitored in real time. This unified observability layer helps security teams maintain governance at scale, detect anomalies quickly, and ensure continuous compliance with internal and external policies.

VII. AUTOMATING IAM AND POLICY DEPLOYMENT

As cloud environments grow in complexity and scale, manual IAM and policy configuration quickly becomes unsustainable. Automation is essential not only for efficiency but also for consistency, repeatability, and security. Automated IAM and policy deployment allows organizations to manage identity and access across multi-account and multi-cloud environments without the risk of human error or configuration drift. It also enables teams to adopt DevSecOps practices, where security and compliance are integrated directly into the software development and infrastructure provisioning lifecycle.

One of the primary enablers of automation is Infrastructure as Code (IaC). Tools like Terraform, Pulumi, and the AWS Cloud Development Kit (CDK) allow IAM roles, policies, resource hierarchies, and permissions to be defined as code, version-controlled, peer-reviewed, and deployed through pipelines. This makes IAM changes auditable and traceable, reducing the likelihood of unauthorized or accidental modifications. For example, Terraform's modules can encapsulate reusable IAM configurations for consistent role assignments across AWS accounts and GCP projects.

To automate governance, organizations often use policy-as-code frameworks such as Open Policy Agent (OPA) with Rego for defining fine-grained policies that can be evaluated during CI/CD workflows. These policies can enforce conditions like preventing wildcard permissions, blocking external IPs, or disallowing public S3 buckets. Integration with tools like Conftest, Checkov, or Terraform Sentinel allows enforcement of these policies before infrastructure is deployed.

In GCP, Policy Controller (based on OPA Gatekeeper) can be used to automatically enforce constraints at the Kubernetes and infrastructure level. AWS provides native guardrails using Service Control Policies (SCPs) and permissions boundaries, which can be applied automatically when accounts or resources are provisioned via automation templates.

GitOps is another automation pattern gaining traction, where all IAM configurations and policies are stored in Git repositories and deployed via continuous deployment pipelines. This provides a declarative, versioned source of truth for governance. Tools like ArgoCD, Flux, or Jenkins pipelines can apply IAM changes in response to approved pull requests, allowing security teams to collaborate with developers in a streamlined and auditable workflow.

By automating IAM and policy deployment, organizations not only reduce operational overhead but also improve their security posture. Automation enforces consistency, accelerates compliance readiness, and enables dynamic environments to remain secure and governed as they evolve.

VIII. CASE STUDY: MANAGING GOVERNANCE FOR A MULTI-CLOUD FINANCIAL PLATFORM

A large financial services company operating globally needed to modernize its cloud strategy by adopting a multi-cloud architecture based on both AWS and GCP. The firm was subject to strict regulatory requirements, including GDPR, PCI-DSS, and internal compliance frameworks, which required tight control over access, encryption, auditability, and data residency. As part of the transformation, the company needed to implement a unified governance strategy to manage IAM, enforce policies, and maintain compliance across both cloud providers without slowing down innovation or developer productivity.

Initially, the company relied on manual IAM role assignments and ad hoc policy management, which quickly became unsustainable as teams scaled across projects and business units. To address this, they designed a centralized governance model that integrated with their enterprise Identity Provider (IdP) using SAML for user authentication and workload identity federation for CI/CD pipelines. This allowed them to eliminate local IAM users and rely solely on federated identities across both AWS and GCP.

They adopted Terraform as their infrastructure provisioning tool, building reusable modules for IAM roles, groups, service accounts, and policy bindings. These modules were stored in a centralized Git repository, version-controlled, and consumed by various teams via GitOps workflows. For policy enforcement, they used OPA Rego policies embedded in their CI pipelines to block high-risk IAM configurations, such as overly permissive wildcard roles or policies granting administrative access.

© 2024 IJSRET

On AWS, they leveraged Service Control Policies (SCPs) to restrict dangerous services (like disabling direct IAM user creation) and enforced strict tagging requirements for all resources. On GCP, they implemented Organization Policies to block the use of unmanaged services, enforce encryption using customer-managed keys, and prevent deployment to non-compliant regions.

For monitoring and auditing, they centralized logs from AWS CloudTrail and GCP Cloud Audit Logs into a SIEM platform, enabling real-time anomaly detection and policy violation alerts. This setup provided their security and compliance teams with a unified view of IAM activity across both clouds.
The implementation of this multi-cloud governance strategy resulted in a 60% reduction in IAM misconfigurations, faster onboarding of development teams, and improved audit readiness. More importantly, it established a repeatable framework for governance that aligned with their long-term digital transformation goals, ensuring scalability, security, and compliance across environments.

Challenges and Lessons Learned
Implementing governance at scale across AWS and GCP comes with a unique set of challenges, especially in large enterprises where diverse teams, evolving business needs, and strict compliance requirements intersect. One of the most common challenges is managing complexity. Each cloud provider has distinct IAM architectures, APIs, and governance tools. Teams must learn to navigate these differences while maintaining consistency, which often leads to duplication of effort or misaligned policies unless careful abstraction and tooling are introduced.

Cultural resistance can also hinder governance efforts. Developers and application teams may perceive guardrails and centralized IAM processes as barriers to productivity. Without early buy-in, governance policies risk being bypassed or misused. Overcoming this requires fostering a security-as-enabler mindset, where governance is framed not as control, but as an accelerator that protects both users and customers. Embedding security champions within dev teams and providing self-service governance tools can help bridge this gap.

Tooling fragmentation is another major issue. Many enterprises initially lack unified tooling to deploy, monitor, and audit IAM and policies across clouds. Inconsistent logging formats, different compliance control sets, and incompatible APIs make it difficult to build centralized dashboards and reporting. Investing in cloud-agnostic platforms, custom automation layers, or third-party governance suites becomes essential for visibility and policy consistency.

Policy drift is a persistent concern. As environments change rapidly with infrastructure-as-code, automation pipelines, and team handovers, access controls can drift from intended baselines. Without automated compliance checks and remediation, this drift accumulates over time and weakens the organization's security posture. Incorporating policy-as-code and continuous validation tools into CI/CD pipelines has proven effective in detecting and correcting drift early.

Some technical lessons include the importance of limiting IAM granularity where appropriate. Trying to micromanage every resource with custom roles often backfires, leading to unmaintainable sprawl. Using predefined roles, layered with organization-level constraints, strikes a better balance between flexibility and control. Also, centralizing identity through federated authentication simplifies lifecycle management and reduces the risk of orphaned access.

Ultimately, success comes from viewing governance as a continuous process, not a one-time project. Iteration, collaboration across teams, and investment in automation are key to adapting to ever-evolving cloud platforms, business requirements, and threat landscapes.

IX. CONCLUSION AND RECOMMENDATIONS

Effective governance at scale is a foundational requirement for any organization operating across AWS and GCP. As cloud adoption accelerates and multi-cloud strategies become more prevalent, managing IAM and enforcing policies consistently across platforms is critical for security, compliance, and operational efficiency. This article has explored the key concepts, architectural differences, and practical implementation strategies required to build and sustain a scalable governance framework across both cloud providers.
A successful governance model begins with a strong understanding of each platform's IAM architecture. While AWS offers granular, resource-based control, GCP's hierarchical model provides top-down visibility and policy inheritance. Bridging these differences requires a deliberate approach to IAM design that emphasizes standardization, least privilege, and modularity. Role-based access control, coupled with clearly defined identity federation mechanisms, forms the backbone of a unified access model.

Automation is essential at every layer of governance. By embracing infrastructure-as-code, policy-as-code, and GitOps practices, organizations can codify, validate, and deploy IAM roles and policies with confidence and repeatability. Automation also supports rapid growth and reduces the risk of configuration drift or manual error. Tools like Terraform, OPA, AWS SCPs, and GCP Organization Policies should be

© 2024 IJSRET

leveraged to enforce guardrails and ensure consistent behavior across clouds.

Monitoring and auditing complete the governance lifecycle. Organizations must invest in centralized logging, real-time alerting, and compliance reporting to detect anomalies, validate policies, and support internal and external audits. Integrating logs into a unified SIEM platform ensures that visibility extends across both AWS and GCP, enabling security teams to respond quickly and effectively to risks.

From our case studies and analysis, several key recommendations emerge:

- Standardize IAM and policy deployment using IaC and GitOps.
- Establish centralized identity and federated access across clouds.
- Enforce baseline policies using native tools like SCPs and GCP Constraints.
- Continuously monitor and audit for drift, misconfigurations, and violations.
- Foster a governance culture that prioritizes security while empowering teams.

By following these principles, enterprises can build governance systems that are scalable, secure, and adaptable. In an era of increasing complexity and regulatory scrutiny, such systems are not just beneficial—they are essential for long-term cloud success.

REFERENCES

1. Paidy, P. (2019). Hardening AWS Infrastructure after Capital One: IAM, S3, and Network Security. JOURNAL OF RECENT TRENDS IN COMPUTER SCIENCE AND ENGINEERING.
2. Heyns, C. (2016). Human Rights and the use of Autonomous Weapons Systems (AWS) During Domestic Law Enforcement. Human Rights Quarterly, 38, 350 - 378.
3. Zahoor, E., Asma, Z., & Perrin, O. (2017). A Formal Approach for the Verification of AWS IAM Access Control Policies. European Conference on Service-Oriented and Cloud Computing.
4. Ahmad, T., Morelli, U., Ranise, S., & Zannone, N. (2018). A Lazy Approach to Access Control as a Service (ACaaS) for IoT: An AWS Case Study. Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies.
5. Fleury-Steiner, B.D. (2019). Deportation Platforms: The AWS-ICE Alliance and the Fallacy of Explicit Agendas. Surveillance & Society.
6. Anand, A. (2017). Managing Infrastructure in Amazon using EC2, CloudWatch, EBS, IAM and CloudFront. International Journal of Engineering Research and, 6.
7. Morelli, U., & Ranise, S. (2017). Assisted Authoring, Analysis and Enforcement of Access Control Policies in the Cloud. IFIP International Information Security Conference.
8. Shackelford, A. (2015). Working with AWS OpsWorks.
9. Spagnolo, A. (2017). Human Rights Implications of Autonomous Weapon Systems in Domestic Law Enforcement: Sci-fi Reflections on a lo-fi Reality.
10. Wadia, Y.R. (2016). AWS Administration - The Definitive Guide.
11. Iams, D.M., & O'Brien, J. (1985). TRAFFIC ENFORCEMENT PROGRAM: BASIC EVALUATION HANDBOOK.
12. Spagnolo, A. (2017). Human Rights Implications of Autonomous Weapon Systems in Domestic Law Enforcement: Sci-fi Reflections on a lo-fi Reality.
13. Chengeta, T. (2014). Can Robocop 'Serve and Protect' within the Confines of Law Enforcement Rules?
14. Alpernas, K., Flanagan, C., Fouladi, S., Ryzhyk, L., Sagiv, S., Schmitz, T., & Winstein, K. (2018). Secure serverless computing using dynamic information flow control. Proceedings of the ACM on Programming Languages, 2, 1 - 26.
15. Bradlow, D., & Naude Fourie, A. (2011). The Evolution of Operational Policies and Procedures at International Financial Institutions: Normative Significance and Enforcement Potential. PSN: International Institutions: Origins.

© 2024 IJSRET