Research on Policy-as-Code for Implementation of Role-based and Attribute-based Access Control*

Oleksandr Vakhula[1,†], Ivan Opirskyy[1,*,†], Pavlo Vorobets[1,†], Orest Bobko[1,†] and Oleh Kulinich[2,†]

[1] *Lviv Polytechnic National University, 12 Stepan Bandera str., 79000 Lviv, Ukraine*
[2] *National University of Life and Environmental Sciences of Ukraine, 19/1 Horikhuvatskyi Shliakh str., 03041 Kyiv, Ukraine*

Abstract

The growing adoption of cloud-native and microservices architectures has revolutionized how organizations deploy and manage applications, bringing enhanced scalability, agility, and speed. However, these advancements also introduce significant challenges in managing access control in dynamic, distributed environments. Traditional access control models like Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) are foundational to securing sensitive resources but often struggle to scale and adapt to the complex, rapidly evolving demands of cloud infrastructures. RBAC systems face issues such as "role explosion", while ABAC systems can become unmanageable due to the increasing complexity of attribute combinations. These challenges necessitate innovative solutions for consistent, scalable, and automated access control management. Policy-as-Code (PaC) has emerged as a transformative approach to address these challenges by codifying access control policies, enabling automation, scalability, and real-time adaptability. PaC integrates seamlessly into modern DevOps practices, allowing RBAC and ABAC policies to be managed, tested, and enforced as code within CI/CD pipelines. This approach ensures consistent policy enforcement across diverse environments, improves transparency and auditability through version control, and facilitates compliance with regulatory standards like GDPR, HIPAA, and PCI DSS. Tools such as Open Policy Agent (OPA) and HashiCorp Sentinel play pivotal roles in implementing PaC, providing granular control and automation for complex policy requirements. This research explores the integration of PaC with RBAC and ABAC in cloud-native infrastructures, addressing key challenges such as scalability, compliance, and cross-cloud interoperability. It includes a comprehensive review of recent literature, practical implementation guides, and real-world case studies demonstrating the application of PaC for automated access control. The findings highlight how organizations can achieve continuous compliance, operational efficiency, and enhanced security by adopting PaC. For instance, the study outlines strategies for reducing operational overhead through automated policy validation and enforcement, mitigating risks of misconfigurations, and ensuring dynamic, context-aware access control. The research also identifies challenges associated with PaC adoption, including initial setup complexity, performance overhead in attribute-heavy environments, and managing policy sprawl in large-scale systems. It proposes future research directions, such as leveraging AI for adaptive policy optimization, enhancing real-time policy evaluation, and developing unified frameworks for multi-cloud environments. In conclusion, this study establishes PaC as a robust framework for implementing scalable and adaptive RBAC and ABAC in cloud-native and multi-cloud environments. By automating policy management and integrating access control into the software development lifecycle, PaC empowers organizations to meet the demands of modern infrastructures while ensuring security, compliance, and operational agility. This work provides actionable insights for practitioners and researchers seeking to leverage PaC for effective access control management.

Keywords

policy-as-code, PaC, role-based access control, attribute-based access control, Kubernetes, continuous integration, continuous development, security-as-code, open policy agent, gatekeeper

1. Introduction

Organizations are increasingly adopting cloud-native and microservices architectures to enhance scalability, agility, and speed of deployment. However, as these environments

CPITS 2025: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2025, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
oleksandr.p.vakhula@lpnu.ua (O. Vakhula); ivan.r.opirskyi@lpnu.ua (I. Opirskyi); pavlo.a.vorobets@lpnu.ua (P. Vorobets); orest.bobko.mkbas.2024@lpnu.ua (O. Bobko); o.kulinich@nubip.edu.ua (O. Kulinich)
0009-0008-5367-3344 (O. Vakhula); 0000-0002-8461-8996 (I. Opirskyy); 0009-0007-3870-829X (P.Vorobets); 0009-0005-4224-1448 (O. Bobko); 0000-0002-0643-6898 (O. Kulinich)

growmailto:orest.bobko.mkbas.2024@lpnu.ua, the challenge of managing and enforcing access control becomes more complex. Established access control models, such as RBAC and ABAC, remain crucial for ensuring that sensitive resources are accessible only to authorized users and services.

Yet, implementing these models in distributed, dynamic infrastructures poses significant challenges, particularly in terms of scalability, granularity, and adaptability.

RBAC has long been the foundation of access control, assigning permissions based on predefined roles. However, in large-scale, distributed environments, the sheer volume of users, roles, and permissions can overwhelm conventional RBAC systems. Meanwhile, ABAC introduces a more granular approach, allowing access decisions based on user attributes, resource attributes, and environmental factors. While ABAC provides flexibility, it also increases complexity, as access policies must account for a wide range of attributes and conditions.

To address these challenges, PaC has emerged as a transformative approach, enabling the automation and codification of access control policies within development pipelines. PaC allows RBAC and ABAC policies to be defined, enforced, and managed as code, providing consistent, scalable, and adaptive access control across cloud environments. Through PaC, organizations can implement access control policies that are versioned, tested, and deployed alongside application code, ensuring alignment with changing application requirements and compliance standards.

In addition to enabling automation, PaC facilitates continuous compliance and auditability. By maintaining a versioned history of policy changes, organizations can ensure that access control decisions are transparent, traceable, and easy to review for compliance with regulations such as GDPR [1], HIPAA [2], and PCI DSS [3]. Tools like Open Policy Agent (OPA) and HashiCorp Sentinel have become popular for implementing PaC, offering the flexibility to enforce both RBAC and ABAC in dynamic environments.

This research aims to explore how PaC can streamline and strengthen access control in distributed environments, with a specific focus on RBAC and ABAC implementations. The paper will investigate the key challenges in applying these access control models within cloud-native infrastructures, evaluate the benefits of using PaC to manage and enforce these policies and provide a practical implementation guide using tools like OPA and Sentinel. By examining real-world case studies and assessing the limitations and advantages of PaC, this research seeks to establish a framework for implementing scalable, compliant, and adaptive access control through PaC.

2. Literature and recent research review

The fast shift toward cloud-native environments has brought increased attention to the scalability and flexibility of access control models. PaC is now widely discussed in the literature as an innovative solution for implementing complex access control policies in distributed systems [4, 5]. This section provides an overview of foundational research and recent and older studies, that explore PaC, RBAC, and ABAC.

PaC emerged from the DevOps and Infrastructure-as-Code (IaC) paradigms, which advocate for the codification of infrastructure and configurations to enable versioning, automated testing, and consistent deployments [6].

PaC extends these principles to security and access control, allowing policies to be codified, tested, and managed as part of the software lifecycle. PaC has since gained popularity as a method for defining and enforcing security controls across distributed environments, particularly for organizations adopting microservices and serverless architectures.

RBAC was formally introduced in the 1990's as a solution to simplify access management by assigning permissions based on user roles [7]. RBAC has since been widely adopted in enterprise environments due to its simplicity and ease of implementation. However, recent studies point out limitations in scalability and flexibility when applied to dynamic, cloud-native environments.

ABAC, on the other hand, offers a more flexible approach by basing access decisions on a set of attributes related to the user, resource, and environment [8].
The integration of PaC with RBAC and ABAC models offers promising solutions to address the challenges associated with access control in cloud-native environments. Open Policy Agent (OPA) has become a leading tool in this area, providing a declarative language, Rego, for defining PaC.

Moreover, Gartner Research identifies PaC as a critical component for achieving continuous compliance, especially for organizations operating in regulated industries. Their report suggests that by integrating PaC with identity and access management (IAM) solutions, organizations can achieve a unified and automated approach to compliance that reduces the risk of human error and regulatory violations [9].

While the benefits of PaC for implementing RBAC and ABAC are well-documented, researchers have identified several limitations and problems. Languages that can simplify the management of attribute-based policies in large-scale environments can have some weaknesses. The paper on the Tenable blog highlights vulnerabilities associated with domain-specific languages (DSLs), using Open Policy Agent (OPA) as a case study, and explores how attackers can exploit poorly implemented policies or misconfigurations in these systems [10].

Future research should focus on developing adaptive PaC frameworks that can dynamically adjust policies based on changing contextual factors, such as user behavior or network conditions. Additionally, the integration of machine learning techniques with PaC holds promise for automatically detecting and mitigating potential policy misconfigurations or security threats, an area that is still underexplored.

3. Challenges in Implementing RBAC and ABAC in dynamic environments

In distributed, cloud-native, and microservices architectures, managing access control presents a complex set of challenges. RBAC and ABAC have been widely adopted to ensure that users can access only the resources they are authorized to use. However, when applied to dynamic and large-scale environments, traditional implementations of RBAC and ABAC face several significant limitations [11].

3.1. Scalability and complexity

- **RBAC Complexity in Large-Scale Systems**: As organizations grow, the number of roles and permissions required to control access can become overwhelming. In dynamic environments, where services, users, and resources frequently change, managing a vast set of static roles can lead to a "role explosion." This situation occurs when the system requires an excessive number of roles to accommodate diverse and evolving access needs. For example, in large organizations, a single user might belong to multiple departments, projects, or teams, each with distinct access requirements [12].
- **ABAC Attribute Management**: ABAC is designed for flexibility, allowing access based on a combination of attributes such as user role, location, time, and device type. However, as attribute-based policies grow more complex, they can become difficult to manage. The number of potential combinations of attributes can quickly escalate, making it challenging to track, update, and validate all possible policies. This complexity can lead to administrative overhead and increased risk of misconfiguration, especially in highly dynamic environments like microservices architectures [13].

3.2. Dynamic policy requirements

- **Frequent Policy Changes**: Cloud-native environments are highly dynamic, with frequent changes in user roles, application states, and resource configurations. In such

environments, access policies must be regularly updated to reflect current conditions. Traditional RBAC systems, which rely on static roles and permissions, struggle to adapt to these changes without significant manual intervention. ABAC is more adaptable, but managing attribute changes manually across distributed services is both time-consuming and error-prone.

- **Context-Aware Policies**: Dynamic environments often require context-aware policies that adapt based on real-time factors, such as a user's location, device, or time of access. For example, an ABAC policy may allow access to sensitive data only if the request is made within a specific network range or during certain hours. Managing these conditional policies at scale is challenging, as it requires continuous monitoring of contextual factors and prompt enforcement of access rules based on changes in the environment.

3.3. Auditability and compliance

- **Lack of Policy Transparency**: Ensuring compliance with industry standards and regulatory requirements, such as GDPR, HIPAA, or PCI DSS, demands a high level of transparency in access control policies. Traditional RBAC and ABAC implementations may not provide adequate visibility into policy changes, making it difficult to demonstrate compliance or to perform audits. PaC offers some relief by enabling version control and audit trails for policy changes, but implementing this at scale remains complex.
- **Audit and Reporting Overhead**: In distributed systems, access control policies often need to be enforced across multiple services and locations, each with its own logging and auditing mechanisms. Gathering and consolidating these logs to produce a unified access report is challenging. The lack of centralized visibility can make it difficult to identify unauthorized access or policy violations, increasing the risk of non-compliance with regulations and standards. Implementing a comprehensive audit trail requires additional resources and tools, which can be costly and time-consuming.

3.4. Performance and resource constraints

- **Latency in Policy Enforcement**: Dynamic environments require that access policies be enforced in real-time. However, implementing complex RBAC and ABAC policies across distributed systems can introduce latency, especially when policy evaluation involves numerous attributes or complex conditional logic. This latency can impact the user experience or slow down application performance, which is a significant concern for high-demand environments.
- **Resource Consumption**: The enforcement of attribute-based policies, particularly those with context-aware conditions, requires continuous monitoring of environmental factors and frequent evaluation of access rules. This can consume significant computational resources, especially in large-scale environments where multiple attributes must be evaluated for each access request. High resource consumption not only affects performance but also increases operational costs, making it challenging to implement ABAC effectively in cost-sensitive environments.

3.5. Complexity of cross-platform and multi-cloud environments

- **Inconsistent Policy Management**: Organizations often use multiple cloud providers and platforms to meet their business needs. Implementing RBAC and ABAC across these platforms introduces complexity, as each provider may have different access control mechanisms, policy languages, and enforcement strategies. Coordinating policies across platforms can lead to inconsistencies, where policies are properly enforced on one platform but not on others.

- **Interoperability Challenges**: In multi-cloud or hybrid cloud environments, ensuring that access control policies work seamlessly across different platforms is a significant challenge. For instance, a policy defined in one cloud provider's format may not be directly translatable to another provider's system. The lack of interoperability between platforms can make it difficult to implement consistent RBAC or ABAC policies, increasing the risk of unauthorized access or inconsistent policy enforcement.

Implementing RBAC and ABAC in dynamic environments is a complex task that requires addressing scalability, adaptability, and transparency issues, among others. PaC offers promising solutions to many of these challenges by allowing policies to be codified, versioned, and deployed across distributed systems in an automated and consistent manner. However, as discussed, the effective integration of PaC with RBAC and ABAC models remains a challenge in dynamic, multi-cloud environments. The next sections of this paper will explore how PaC can help overcome these challenges, providing an adaptable and scalable approach to access control in modern infrastructures [14].

4. Policy-as-code for RBAC and ABAC

PaC enables organizations to define, manage, and enforce access control policies in a code-based format, offering consistency, scalability, and automation in modern infrastructures [15]. When applied to RBAC and ABAC, PaC brings automation and flexibility to these traditional access models, addressing many of the challenges associated with managing access in dynamic, distributed environments. This section explores how PaC transforms RBAC and ABAC into more adaptable and manageable models and discusses the primary benefits of this approach.

4.1. Automated policy enforcement

- **Consistency Across Environments**: In distributed and multi-cloud environments, maintaining consistent access control policies can be difficult. By defining RBAC and ABAC policies as code, PaC enables organizations to apply the same policies across diverse environments, ensuring that access controls are consistently enforced. This automation also reduces the risk of discrepancies or misconfigurations between environments, a common issue in large-scale deployments.
- **Continuous Policy Enforcement**: PaC integrates with CI/CD pipelines, ensuring that access policies are continuously validated and deployed alongside application code. Any changes to access control policies can be tested and enforced automatically, providing real-time policy updates without manual intervention. This continuous enforcement is especially beneficial for ABAC, where contextual attributes may change frequently, requiring policies to adapt dynamically [16].

4.2. Policy versioning and change management

- **Version Control for Policies**: Storing RBAC and ABAC policies as code allows them to be managed in version control systems (e.g., Git), enabling teams to track and audit changes over time. Versioning provides a historical record of policy updates, allowing organizations to roll back to previous policy states if needed. This capability is crucial for compliance, as it ensures that organizations can document access control changes and demonstrate adherence to regulatory requirements.
- **Collaborative Change Management**: With PaC, policy updates follow the same processes as software development, including code review and testing. This enables collaborative change management, where security teams, developers, and DevOps teams can work together to define and refine access policies. Through this approach,

organizations can ensure that policies are well-tested and approved before being deployed, reducing the risk of unintended access control changes.

4.3. Tools for policy-as-code in RBAC and ABAC

- **Open Policy Agent (OPA)**: OPA is a popular open-source tool for defining and enforcing policies in cloud-native environments. Using its declarative policy language, Rego, OPA enables organizations to write RBAC and ABAC policies as code, which can be integrated across various services (e.g., Kubernetes, CI/CD pipelines). OPA's flexibility makes it suitable for both RBAC and ABAC, allowing fine-grained control over access decisions based on user roles and attributes [17].
- **HashiCorp Sentinel**: HashiCorp Sentinel is a policy as code framework designed to enforce policies in HashiCorp's suite of products, such as Terraform and Vault. Sentinel supports both RBAC and ABAC implementations, making it ideal for organizations using HashiCorp's ecosystem. Sentinel enables administrators to define access control policies in a code-based format, which can be applied across infrastructure resources, providing consistent and automated access control [18].
- **AWS IAM Policies**: For organizations using AWS, Identity and Access Management (IAM) offers built-in support for PaC through JSON-based policy documents. These policies can be defined for both RBAC and ABAC models, allowing administrators to set role- and attribute-based permissions for users and services within AWS. IAM policies integrate well with AWS's ecosystem and provide a straightforward way to enforce access control in cloud-native applications [19].

4.4. Policy validation and testing

- **Automated Policy Testing in CI/CD Pipelines**: One of the key benefits of PaC is the ability to test access policies as part of CI/CD workflows. By embedding policy tests in the pipeline, organizations can validate that RBAC and ABAC policies are configured correctly before deployment. For example, policies can be tested to ensure that only authorized users have access to specific resources or that attribute-based conditions are applied correctly [20].
- **Policy Simulation and Validation**: Tools like OPA offer simulation capabilities that allow administrators to test policies in a sandbox environment. This feature is particularly useful for ABAC, where policies can be complex due to the combination of multiple attributes. By simulating policies before they are applied, organizations can verify that access controls work as intended, reducing the risk of misconfigurations.

4.5. Compliance and auditability

- **Auditable Policy History**: PaC enables organizations to maintain a comprehensive record of policy changes, which is essential for demonstrating compliance with regulatory standards. By version-controlling access policies, organizations can provide auditors with a traceable history of policy updates, showing how access decisions were made over time. This level of transparency is beneficial for meeting compliance requirements, especially in regulated industries like finance and healthcare.
- **Automated Compliance Checks**: PaC facilitates automated compliance validation, where policies are continuously checked against regulatory standards. By integrating compliance checks into CI/CD pipelines, organizations can ensure that their RBAC and ABAC policies meet industry standards before deployment. For example, policies can be validated to confirm that access restrictions align with GDPR, HIPAA, PCI DSS, or ISO 27001 requirements, reducing the risk of non-compliance [21].

4.6. Real-time policy enforcement and adaptability

- **Dynamic Policy Adaptation**: In environments where user roles or attributes frequently change, PaC enables real-time policy adaptation to ensure that access controls remain relevant. For ABAC, which depends on attributes like user location, device type, or time of day, PaC provides the flexibility to enforce policies based on current conditions. This real-time adaptability reduces the administrative burden of manually updating policies in response to changing conditions.
- **Proactive Incident Response**: PaC allows organizations to enforce security policies proactively, triggering incident response protocols when policy violations occur. For example, if an ABAC policy detects an access attempt from an unauthorized location, the system can automatically revoke access or alert the security team. This proactive enforcement strengthens security by enabling rapid responses to potential threats.

PaC transforms RBAC and ABAC from static, manually managed models into flexible, automated systems capable of adapting to dynamic, cloud-native environments. By enabling versioning, testing, and real-time enforcement, PaC addresses many of the scalability and compliance challenges inherent in traditional access control implementations. The integration of tools like Open Policy Agent, HashiCorp Sentinel, and AWS IAM further enhances the ability to manage and enforce access control policies as code, bringing consistency, auditability, and scalability to access control in distributed environments [22].

5. Case studies and tools

This section presents real-world examples of implementing PaC for RBAC and ABAC using leading tools. Each case study provides technical insights into how PaC helps automate, enforce, and manage access control policies within dynamic environments. Additionally, we explore specific tools such as Open Policy Agent (OPA), HashiCorp Sentinel, and AWS IAM, detailing their setup and configurations for RBAC and ABAC.

5.1. Case study 1: RBAC implementation in Kubernetes with open policy agent

- **Context**: A financial services organization deployed a multi-tenant Kubernetes environment to support microservices. They needed a secure and scalable way to enforce RBAC policies, ensuring that each service could only access resources within its designated namespace.
- **Challenge**: Kubernetes's native RBAC lacked the granularity required for fine-tuned role management across tenants. Additionally, managing roles at scale for hundreds of services presented a risk of misconfiguration and permission sprawl.
- **Solution**: The organization integrated **Open Policy Agent (OPA)** with Kubernetes, defining policies using the Rego language to enforce role-based permissions on resources within each namespace.

Technical Details:

- **Policy Definition**: RBAC policies were written in OPA's Rego language. Each policy was stored as code in a Git repository and managed through CI/CD pipelines.

- **Sample Rego Policy**:

```
package kubernetes.authz

allow {
    input.user.roles[_] == "dev-role"
    input.resource.namespace == "development"
    input.verb == "get"
    input.resource.kind == "pod"
}
```

This policy allows users with the dev-role to perform get operations on pod resources within the development namespace.

- **OPA Integration with Kubernetes**: OPA was deployed as an admission controller, intercepting requests to the Kubernetes API server. When a request was made, OPA evaluated the Rego policies and allowed or denied access based on the defined RBAC rules [23, 24].
- **Automated Testing and Validation**: Policy changes were version-controlled, with automated tests in CI/CD to validate access rules before deployment. This setup allowed the organization to detect and fix policy misconfigurations before applying them to the live environment.

5.2. Case study 2: ABAC implementation for data access control with HashiCorp Sentinel

- **Context**: A healthcare provider requires ABAC to secure patient data based on user attributes, such as job role, department, and access time. This setup ensured that only authorized healthcare personnel could view sensitive information under specific conditions.
- **Challenge**: Ensuring fine-grained control over data access in a multi-cloud environment, while enabling real-time access adjustments based on changing attributes (e.g., location).
- **Solution**: The organization used **HashiCorp Sentinel** within their IaC setup, managing access control for data stored in AWS S3 and GCP buckets.

Technical Details:

- **Policy Definition**: Sentinel policies were written to define conditions for accessing sensitive resources based on user attributes. For example, access to patient records was restricted to healthcare professionals during their shift hours and only within designated facilities.
- **Sample Sentinel Policy**:

```
import "strings"

main = rule {
    // Allow access if the requestor's job role includes "nurse" or "doctor"
    strings.contains(requestor.job_role, "nurse") or
    strings.contains(requestor.job_role, "doctor")
    // Restrict access to specific IP ranges for hospital facilities
    and requestor.ip_address in ["10.0.0.0/24", "10.1.0.0/24"]
    // Ensure access during working hours (8am - 8pm)
    and time.now.hour >= 8 and time.now.hour <= 20
```

```
}
```

This policy restricts access based on role, IP address, and time, allowing healthcare personnel to access records within working hours and specific network locations [25].

- **Real-Time Enforcement**: The Sentinel policy engine was integrated into the provider's cloud environment using HashiCorp's Terraform for IaC, enforcing policies each time infrastructure or IAM roles were provisioned or updated. Changes to user attributes (e.g., IP address) triggered an automatic re-evaluation of access permissions.
- **Logging and Auditing**: Sentinel logs were stored centrally and integrated with the healthcare provider's logging infrastructure, providing a complete audit trail for all access decisions based on ABAC policies.

5.3. Case study 3: Multi-cloud ABAC for dynamic data access using AWS IAM and OPA

- **Context**: A global e-commerce company needed to secure customer data across AWS and Azure. The organization required ABAC to restrict access based on attributes such as department, project, and user location, with automated enforcement across both clouds.
- **Challenge**: Coordinating access policies across multiple cloud environments while ensuring consistency and security in data access.
- **Solution**: The company used **AWS IAM** for defining JSON-based policies for ABAC within the AWS environment, while **Open Policy Agent (OPA)** managed cross-cloud access policies. OPA acted as a centralized policy decision point for all data access requests [26].

Technical Details:

- **AWS IAM Policy for ABAC**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::customer-data/*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": "customer-support",
          "aws:PrincipalTag/location": "us-west-2"
        }
      }
    }
  ]
}
```

This IAM policy restricts S3 access based on department and location attributes, allowing only customer support staff in the US West region to access customer data.

- **Cross-Cloud Enforcement with OPA**: OPA was deployed as a centralized policy engine that processed ABAC policies for resources across AWS and Azure. OPA's Rego policies were stored in a Git repository and versioned, allowing the organization to enforce the same access controls on both platforms.

- **Automated Attribute Synchronization**: Attributes were synchronized across clouds using AWS Lambda functions and Azure Logic Apps, which updated user attributes in real time to ensure consistency. This enabled OPA to evaluate access decisions based on up-to-date attribute information from both clouds.
- **Policy Evaluation and Monitoring**: OPA's decision logs were integrated with the company's monitoring tools (e.g., Datadog), allowing the security team to track policy enforcement and respond to any anomalies in access patterns.

5.4. Tools Overview and Setup

The following tools are widely used for implementing PaC in RBAC and ABAC across dynamic environments:

Table 1
Comparison of policy-as-code systems parameters

	Open Policy Agent	Hashicorp Sentinel	AWS IaM Policies
Integration	OPA integrates with Kubernetes as an admission controller, as well as with CI/CD pipelines and APIs in multi-cloud environments	Sentinel works seamlessly with HashiCorp products like Terraform and Vault, making it ideal for organizations using HashiCorp's ecosystem.	AWS IAM enables JSON-based policy documents for fine-grained control over resources in AWS.
Policy language	Rego, a declarative language allowing detailed rule definitions for RBAC and ABAC.	Sentinel's language supports logical operators and imports, making it highly configurable for ABAC.	JSON format with built-in support for both RBAC and ABAC policies.
Setup	OPA can be deployed in Kubernetes using Helm charts or manually configured with YAML files for fine-tuning admission control policies. Rego policies are stored in Git for version control and integrated with CI/CD for automated validation.	Sentinel policies are defined as .hcl files within Terraform modules. By integrating Sentinel with Terraform's workflows, organizations can enforce policies whenever infrastructure changes are provisioned.	Policies are defined in AWS IAM with conditions that reference user attributes and are automatically enforced across AWS resources. Policies can also be combined with Lambda functions for real-time policy adaptation based on attribute changes.

These case studies demonstrate the versatility and effectiveness of PaC for implementing RBAC and ABAC in cloud-native environments. By leveraging tools like OPA, Sentinel, and AWS IAM, organizations can achieve consistent, scalable, and real-time access control enforcement across distributed systems. In the following section, we present a practical implementation guide, detailing step-by-step instructions for applying PaC in RBAC and ABAC models [27, 28].

6. Practical implementation guide

Implementing PaC for RBAC and ABAC involves defining, testing, deploying, and managing policies within an automated workflow. This section provides a step-by-step guide for setting up PaC, including policy definition, CI/CD integration, testing, and real-time enforcement across dynamic environments. The guide focuses on using Open Policy Agent (OPA) and HashiCorp Sentinel as examples but can be adapted for other tools.

Fig. 1 illustrates a general overview of the implementation process for PaC in validating RBAC and ABAC policies.

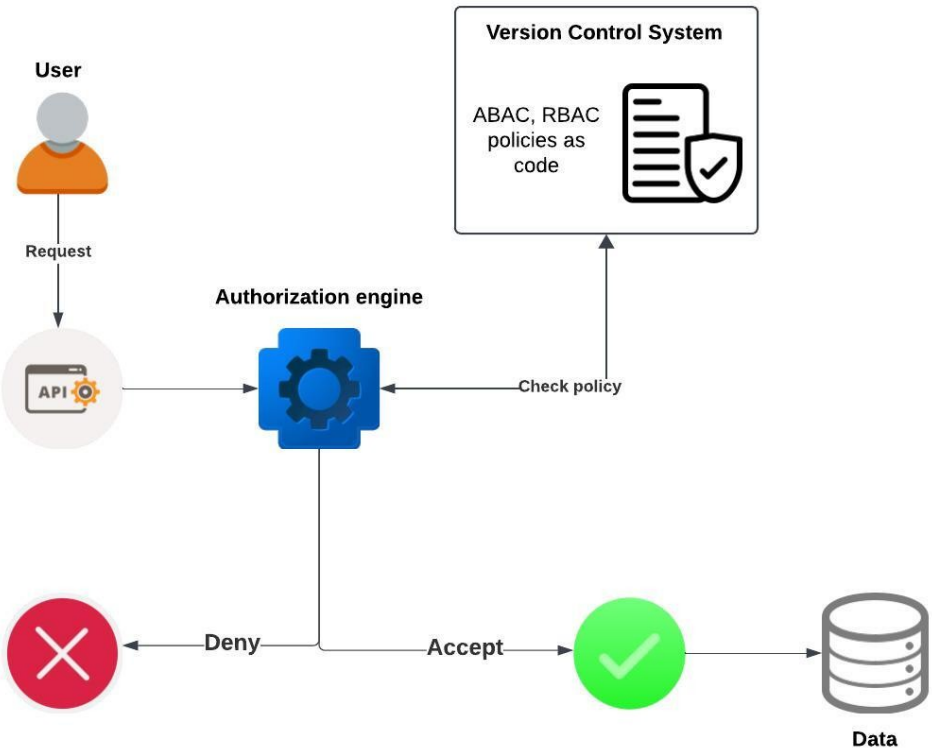

Figure 1: Policy as Code in validating Attribute-Based Access Control and Role-Based Access Control policies

6.1. Defining and managing policies-as-code

Step 1: Define Policies using Rego or Sentinel Language

- For RBAC, start by defining roles and associated permissions in a policy file. For ABAC, define attribute-based rules to allow or deny access based on user and resource attributes.

Example RBAC Policy in Rego (Open Policy Agent):

```
package kubernetes.authz

allow {
    input.user.roles[_] == "admin"
    input.verb == "create"
    input.resource.kind == "pod"
}
```

This policy grants the admin role permission to create pod resources.

- Example ABAC Policy in Sentinel:

```
import "strings"

main = rule {
    // Permit only if the requestor's department is "HR" and the access time is within working hours.
    strings.contains(requestor.department, "HR") and
    time.now.hour >= 9 and time.now.hour <= 17
}
```

Step 2: Version Control Policies with Git

- Store policy files in a Git repository, allowing version control for tracking changes and enforcing review processes. Set up branching for development, testing, and production stages to prevent accidental deployment of untested policies [29].

6.2. Integrating PaC with CI/CD pipelines

Step 1: Set Up Policy Validation in CI/CD

- Configure automated checks in your CI/CD pipeline to validate policies before deployment. For example, add a policy validation step in GitHub Actions, GitLab CI/CD, or Jenkins.
- OPA Example (using opa test command):

```
opa test ./policies/
```

This command runs tests on all policies within the ./policies directory. Failures will prevent the policy from moving to the next stage.

Step 2: Policy Testing and Simulation

- Define unit tests for policies to verify expected behavior under various scenarios. This includes tests for both allowed and denied actions based on specified roles or attributes.

Example Test for OPA (using Rego test files):

```
package kubernetes.authz

test_admin_can_create_pod {
  allow with input as {
    "user": {"roles": ["admin"]},
    "verb": "create",
    "resource": {"kind": "pod"}
  }
}
```

- Sentinel Test File:

```
test "HR access during work hours" {
  input = {
    requestor = {
```

```
        "department": "HR",
        "access_time": "10:00"
      }
    }
    assert main
}
```

Step 3: Automate Deployment of Policies

- Use CI/CD to automatically deploy validated policies to the environment. Policies can be loaded to OPA as bundles or pushed directly to Sentinel's policy repository.

6.3. Configuring real-time enforcement with OPA

Step 1: Deploy OPA as an Admission Controller (Kubernetes)

- In Kubernetes, OPA can be deployed as an admission controller to enforce policies in real time. Use Helm to install OPA:

```
helm install opa stable/open-policy-agent
```

Step 2: Define Admission Controller Configuration

- Configure OPA to intercept specific Kubernetes API calls by defining admission controller configurations. Specify which operations to monitor, such as creating or deleting resources, and which roles or attributes are required.

Example Kubernetes Admission Controller Config:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: opa-validation
webhooks:
- name: validation.openpolicyagent.org
  clientConfig:
    service:
      name: opa
      namespace: opa
      path: "/v1/data/kubernetes/authz/allow"
  rules:
  - operations: ["CREATE", "UPDATE"]
    apiGroups: ["*"]
    apiVersions: ["*"]
    resources: ["pods"]
```

Step 3: Test Real-Time Policy Enforcement

- Use kubectl or other clients to create resources and test the real-time application of policies. Policies should automatically permit or deny requests based on defined rules, such as user roles or attributes [30].

6.4. Implementing sentinel for ABAC in multi-cloud

Step 1: Define Sentinel Policies for ABAC

- In multi-cloud environments, Sentinel policies can enforce ABAC based on conditions like department, job role, or environment type. Use logical expressions to define complex ABAC rules that account for multiple attributes.

Step 2: Integrate Sentinel with Terraform for Infrastructure Provisioning

- Configure Sentinel policies to run whenever Terraform resources are provisioned or modified, ensuring that access controls are applied consistently across resources.

Example Sentinel Policy to Enforce Departmental Access:

```
main = rule {
  requestor.department == "Engineering"
}
```

Step 3: Use Sentinel to Enforce Attribute-Based Controls Across Clouds

- Apply Sentinel policies across AWS and Azure environments by integrating Sentinel with Terraform providers for each cloud. This allows organizations to enforce uniform ABAC policies across platforms.

6.5. Logging and auditing for compliance

Step 1: Enable Centralized Logging of Policy Decisions

- Integrate OPA and Sentinel with a centralized logging solution, such as Elasticsearch or AWS CloudWatch, to track all access requests and policy decisions. This setup provides an auditable trail of access control decisions.

Step 2: Configure Alerting for Policy Violations

- Set up alerts to notify security teams of any unauthorized access attempts or policy violations. For example, integrate OPA logs with Datadog or Prometheus to monitor access control patterns in real-time and alert on suspicious activity.

Step 3: Maintain a Versioned History of Policies

- Version control all policy changes and keep records of updates. This history is essential for auditability, allowing organizations to demonstrate how access policies evolved and to investigate potential security incidents related to policy changes.

6.6. Best practices for PaC in RBAC and ABAC

- **Separate Policy Definitions for Different Environments**: Keep separate policies for development, staging, and production to prevent test policies from affecting live environments.
- **Use Least Privilege**: Design RBAC and ABAC policies based on the principle of least privilege, granting the minimum necessary access required by roles or attributes.

- **Automate Policy Updates**: Establish a CI/CD pipeline to automate policy validation and deployment. This automation minimizes manual errors and enforces consistent security controls across environments.
- **Regularly Test Policies**: Schedule regular policy tests and simulations to ensure that RBAC and ABAC controls remain effective as environments and access requirements change over time.

Implementing PaC for RBAC and ABAC allows organizations to manage access control in a scalable, automated, and auditable way. By defining PaC, integrating them into CI/CD pipelines, enforcing them in real-time, and maintaining logs for compliance, PaC provides a robust framework for secure access management in dynamic environments. The next section will explore the outcomes, benefits, and limitations of implementing PaC for access control, based on case studies and recent research findings [26].

7. Results and discussion

Implementing PaC for RBAC and ABAC offers measurable improvements in security, operational efficiency, and compliance. By automating access control management, PaC enables organizations to enforce policies consistently across dynamic environments, such as multi-cloud and Kubernetes infrastructures. This section examines the outcomes of PaC implementations in terms of security, scalability, compliance, and adaptability, as well as challenges and limitations identified during the research [31].

7.1. Enhanced security and consistency

Results: With PaC, organizations achieved more consistent and reliable enforcement of access control policies. The automation provided by PaC reduced manual interventions, minimizing the risk of misconfiguration and unauthorized access.

Impact: Implementing PaC in dynamic environments like Kubernetes and multi-cloud setups has significantly improved access security by ensuring that RBAC and ABAC policies are enforced in real time. Organizations observed fewer access control incidents, such as privilege escalation and policy drift, due to the version-controlled and testable nature of PaC.

Discussion: PaC's versioning and automated validation allows for precise and transparent control over policy changes. By deploying policies alongside application code, organizations maintain security alignment with changing application requirements, creating a more agile security approach.

7.2. Scalability and reduced operational overhead

Results: PaC enabled scalable management of RBAC and ABAC policies, even in environments with a high volume of resources, roles, and users. Teams noted a significant reduction in the operational burden associated with managing access controls manually.

Impact: By using PaC, organizations could scale their access control systems in line with their infrastructure growth, without a corresponding increase in administrative overhead. Automated policies reduced the need for constant manual updates and mitigated the risks of "role explosion" in RBAC and attribute sprawl in ABAC.

Discussion: PaC frameworks, like Open Policy Agent (OPA) and HashiCorp Sentinel, allowed teams to define and manage policies in a consistent, centralized format, making it easier to scale access control policies across complex architectures. However, some organizations noted initial setup challenges as they transitioned to code-based access management, particularly in aligning policies across multiple cloud providers.

7.3. Continuous compliance and auditing

Results: PaC's integration with CI/CD pipelines and centralized logging systems provided organizations with a continuous compliance framework, automating audit trails and policy validation processes. Real-time logging of access requests and policy enforcement decisions enabled comprehensive auditing and traceability.

Impact: By maintaining a versioned history of all policy changes, PaC facilitated audit readiness and regulatory compliance. Organizations operating in regulated industries, such as healthcare and finance, observed improved compliance outcomes, as policies were automatically validated against industry standards like GDPR and HIPAA.

Discussion: The ability to store policies in version control systems such as Git provided transparency, traceability, and ease of auditing. PaC's compatibility with CI/CD workflows ensured that access control configurations were validated continuously, minimizing the risk of non-compliance and allowing organizations to address regulatory requirements proactively.

7.4. Improved adaptability and real-time enforcement

Results: PaC empowered organizations to adopt policies in real-time, applying attribute-based access controls that responded to contextual conditions, such as user location, device type, and time. This adaptability allowed organizations to meet dynamic access control needs effectively.

Impact: Real-time enforcement was especially beneficial in environments where attributes change frequently. For instance, in ABAC implementations, PaC allowed policies to adapt to changes in user roles, environmental factors, and other attributes, enabling more granular and responsive access control.

Discussion: PaC's real-time adaptability is an advantage in cloud-native environments, where rapid changes in infrastructure demand a flexible approach to access control. However, some organizations encountered latency issues with attribute-heavy ABAC policies, indicating a need for optimization to maintain performance at scale.

7.5. Limitations and challenges

While PaC significantly enhances access control capabilities, challenges remain in its implementation and scalability. Key limitations encountered include:

7.5.1. Complexity in policy management

I**ssue**: For organizations implementing attribute-based access control, the management of attributes can become complex as the number of attributes grows. High attribute diversity often requires highly specific policies, which can lead to "policy sprawl".

Discussion: Although PaC offers tools for versioning and automation, managing numerous detailed policies and attributes can be challenging in large-scale environments. Simplifying attribute management, possibly through consolidated or hierarchical policies, could help mitigate complexity and improve operational efficiency.

7.5.2. Initial setup and integration effort

Issue: Integrating PaC into existing CI/CD pipelines and infrastructure required a significant initial setup effort, particularly for organizations new to policy automation. Setting up and configuring tools like OPA and Sentinel involved custom configurations and, in some cases, additional training for teams.

Discussion: The initial setup and learning curve may present challenges, especially for teams that are less experienced in DevSecOps practices. Providing training or building templates for common RBAC/ABAC configurations could streamline the adoption of PaC for access control.

7.5.3. Performance overhead in real-time policy evaluation

Issue: Organizations using attribute-heavy ABAC policies noted increased latency during policy evaluations, especially in environments with real-time enforcement requirements. This latency may impact user experience or system performance, particularly in high-demand applications.

Discussion: Optimizing policy evaluation processes, such as caching frequently accessed policies or balancing attribute granularity, can help reduce latency. Additionally, some organizations may need to evaluate the trade-offs between detailed, attribute-rich policies and simpler, role-based policies for performance-critical applications.

7.6. Future directions and improvements

To address these challenges, future research and development in PaC should focus on:

Adaptive Policy Management: Developing methods to dynamically adjust policies based on changing organizational and environmental contexts can improve the scalability and flexibility of PaC for RBAC and ABAC.

Enhanced Integration with AI for Policy Optimization: Leveraging machine learning to optimize policy decisions and identify anomalies can further enhance PaC's ability to scale effectively in complex environments. Also we can use methods of context analysis for optimizing policies [32].

Unified Policy Frameworks for Multi-Cloud Environments: Establishing unified policy standards for multi-cloud setups would simplify policy management and enforcement across diverse platforms, reducing integration overhead and ensuring consistent access control.

PaC provides a powerful framework for implementing RBAC and ABAC in distributed, cloud-native, and multi-cloud environments. By automating policy management, ensuring real-time enforcement, and enabling continuous compliance, PaC improves security, scalability, and suitability for access control systems. While challenges related to complexity, integration, and performance persist, ongoing developments in PaC tools and methods offer promising solutions for addressing these limitations. The final section will summarize the main findings, highlighting the advantages and future potential of PaC for access control in dynamic environments [33].

Conclusions

As organizations adopt increasingly complex, cloud-native architectures, the need for scalable, flexible, and automated access control mechanisms has become paramount. PaC represents a transformative approach to access control, enabling organizations to define, enforce, and manage RBAC and ABAC policies through automated, code-driven processes. This research has explored the benefits of implementing PaC for RBAC and ABAC, emphasizing its impact on security, scalability, compliance, and adaptability in dynamic environments.

PaC's ability to integrate with CI/CD pipelines has made continuous compliance a reality, providing automated policy validation and real-time policy enforcement across cloud and multi-cloud environments. By managing policies in code, organizations gain the advantages of version control, collaborative change management, and comprehensive audit trails, all of which contribute to a more transparent and compliant access control framework. Additionally, the real-time adaptability offered by PaC allows organizations to implement context-aware policies that respond dynamically to changes in user roles, attributes, and environmental conditions, further enhancing security in high-demand applications.

Despite its advantages, PaC presents challenges that organizations must address, such as managing the complexity of attribute-heavy policies in ABAC, optimizing policy evaluation for high-performance applications, and overcoming the initial integration effort required for CI/CD environments. As PaC continues to evolve, future developments—such as adaptive policy management and AI-enhanced policy optimization—promise to address these limitations, making PaC even more accessible and effective in a wider range of use cases.

In conclusion, PaC offers a robust framework for modern access control, enabling organizations to enforce RBAC and ABAC policies at scale, with enhanced security, compliance, and efficiency. As organizations continue to navigate the demands of cloud-native infrastructures, PaC provides a scalable solution to automate, simplify, and strengthen access control, laying the groundwork for a more secure and adaptable future in access management.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] M. Iavich, et al., Classical and post-quantum encryption for GDPR, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3829 (2024) 70–78.

[2] S. Shevchenko, et al., Protection of information in telecommunication medical systems based on a risk-oriented approach, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 158–167.

[3] P. Skladannyi, et al., Improving the security policy of the distance learning system based on the zero trust concept, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 97–106.

[4] O. Vakhula, I. Opirskyy, O. Mykhaylova, Research on security challenges in cloud environments and solutions based on the "Security-as-Code" approach, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3550, 2023, 55–69.

[5] O. Vakhula, et al., Security-as-code concept for fulfilling ISO/IEC 27001:2022 requirements, in: in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3654, 2024, 59–72.

[6] Palo alto what is secuirty as a code, 2024. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-policy-as-code

[7] D. F. Ferraiolo, D. R. Kuhn, Role-based access controls, arXiv, 2009. doi:10.48550/arXiv.0903.2171

[8] E. Yuan, J. Tong, Attributed based access control (ABAC) for Web services, in: IEEE International Conference on Web Services (ICWS'05), 2005. doi:10.1109/ICWS.2005.25

[9] Gartner Research 2022, Policy as Code' to Secure Application Deployments and Enforce Compliance, 2024.

[10] S. Raban, The Dark Side of Domain-Specific Languages: Uncovering New Attack Techniques in OPA and Terraform, 2024.

[11] T. Baumer, M. Mueller, G. Pernul, System for Cross-domain Identity Management (SCIM): Survey and Enhancement with RBAC, IEEE Access 11 (2023) 86872–86894. doi:10.1109/ACCESS.2023.3304270

[12] S. Aboukadri, A. Ouaddah, A. Mezrioui, Boosted-3R: Towards a novel framework for inferring ABAC policies, 2024.

[13] A. Biswas, G. Baranwal, A. Tripathi, ABAC: Alternative by alternative comparison based multi-criteria decision making method, Expert Syst. Appl. 208 (2022). doi:10.1016/j.eswa.2022.118174

[14] B. Lee, Using open policy agent (OPA) to apply policy-as-code to infrastructure-as-code, 2022. URL: https://cloudsecurityalliance.org/blog/2020/04/02/using-open-policy-agent-opa-to-apply-policy-as-code-to-infrastructure-as-code/

[15] M. Sánchez-Gordón, R. Colomo-Palacios, Security as culture: A systematic literature review of DevSecOps, in: ICSEW'20: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, 2020, 266–269. doi:10.1145/3387940.3392233

[16] Guest Expert on GitGuardian, What is policy-as-code? An introduction to open policy agent, 2022. URL: https://blog.gitguardian.com/what-is-policy-as-code-an-introduction-to-open-policy-agent/
[17] W. Salami, HashiCorp Sentinel: An introduction, 2024. URL: https://www.globallogic.com/uki/insights/blogs/hashicorp-sentinel-an-introduction/
[18] X. Zhang, Cloud governance and compliance on AWS with policy as code, 2021. URL: https://aws.amazon.com/ru/blogs/opensource/cloud-governance-and-compliance-on-aws-with-policy-as-code/
[19] M. Ferris, A Comprehensive Guide to Automated Testing for CI/CD Pipelines, 2023. URL: https://qameta.io/blog/automated-testing-ci-cd-guide/
[20] Y. Kurii, I. Opirskyy, L. Bortnik, ISO/IEC 27001:2022. Analysis of changes and compliance features of the new version of the standard, in: 9th International Scientific and Technical Conference Information Protection and Information Systems Security, 2023, 15–17.
[21] O. Vakhula, I. Opirskyy, Research on security as code approach for cloud-native applications based on Kubernetes clusters, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3800, 2024, 58–69.
[22] Policy as code in Kubernetes: security with seccomp & network policies, ArmoSec, 2024. URL: https://www.armosec.io/blog/policy-as-code-in-kubernetes-security-seccomp-and-network-policies/
[23] R. Ferreira, Policy design in the age of digital adoption: Explore how PolicyOps can drive policy as code adoption in an organization's digital transformation 1st Edition, Packt Publishing, 2022.
[24] Hashicorp Sentinel Documentation, 2024. URL: https://developer.hashicorp.com/sentinel/docs
[25] V. Khoma, et al., Comprehensive Approach for Developing an Enterprise Cloud Infrastructure, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3654, 2024, 201–215.
[26] S. I. Shamim, F. A. Bhuiyan, A. Rahman, XI Commandments of Kubernetes security: A systematization of knowledge related to Kubernetes security practices, in: 2020 IEEE Secure Development (SecDev), 2020, 58–64. doi:10.1109/SecDev45635.2020.00025
[27] Y. Martseniuk, et al., Shadow IT risk analysis in public cloud infrastructure, in: Cyber Security and Data Protection, vol. 3800, 2024, 22–31.
[28] Y. Martseniuk, et al., Universal centralized secret data management for automated public cloud provisioning, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3826, 2024, 72–81.
[29] B. Burns, et al., Borg, Omega, and KUBERNETES, Queue 14(1) (2016) 70–93. doi:10.1145/2898442.2898444
[30] Y. Martseniuk, et al., Automated conformity verification concept for cloud security, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3654, 2024, 25–37.
[31] Y. Dreis, et al., Restricted Information Identification Model, in: Cybersecurity Providing in Information and Telecommunication Systems Vol. 3288 (2022) 89–95.
[32] S. Yevseiev, et al., Development of a method for determining the indicators of manipulation based on morphological synthesis, Eastern-European J. Enterp. Technol. 3(9(117)) (2022) 22–35. doi:10.15587/1729-4061.2022.258675
[33] D. Shevchuk, et al., Designing secured services for authentication, authorization, and accounting of users, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3550, 2023, 217–225.