Computer Fraud and Security
ISSN (online): 1873-7056

Identity and Access Management in Distributed Enterprise Applications: A Role-Based and Attribute-Based Access Control Framework

Shubh Prabhat

Project Leader, Wipro Limited, Cary, North Carolina, USA

E-mail: sprabhat.research@gmail.com

Research area: Cybersecurity

Abstract

Enterprise applications are typically deployed as distributed services in multiple locations, including co-located and partner data centers. All distributed services require a mechanism to enforce access control for authenticated users. RBAC is a simpler access control mechanism; however, if there are clearance and other location-based and time-based access restrictions, it may require many roles. ABAC is a more efficient access control mechanism for implementing clearance and other location-based and time-based restrictions but is a less convenient mechanism for managing access policies because it lacks other organizational features of access policies. This paper proposes an IAM architecture that integrates RBAC and ABAC features by adding attributes to RBAC role permissions. A two-step evaluation process that includes opportunities for caching and no attribute queries for environment attributes provides efficient access enforcement. The framework includes duplicate policy enforcement systems and features in deployed applications and event notifications to quickly remove user access. A Python prototype implementation and simulation of an organization with 5,000 employees, 20,000 objects, and 200,000 session accesses showed that a combined role-based and attribute-based access control policy requires 274 policy elements, while a pure role-based policy requires 6,912 roles. A pure attribute-based policy would provide 22.2% more access than necessary. The combined policy allows more accurate access decisions, requiring one policy rule check per transaction, compared with 30.4 policy rule checks under a pure attribute-based policy. Attribute screening was reduced from 2.00 checks per transaction with no caching to 1.23 checks per transaction using a decision cache of 4,096 entries. System transaction volume increased almost linearly with the number of decision point replicas, to 16 replicas. Changes to the employee workforce require 33% fewer administrative tasks than changes under a role-based system.

Keywords: Identity and access management; Role-based access control; Attribute-based access control; Distributed enterprise applications; Policy decision point; Authorization caching

1. Introduction

Enterprise applications that once operated in one data center now include services operating in private and public clouds owned by the enterprise and its partners. Services used by people and other software applications must apply enterprise security policies. Identity and access management products and tools can be used to apply enterprise security policies by authenticating users to verify their identity and authorizing or denying their access requests. Role-based access control is widely used in enterprises to simplify user administration and auditing by granting access based on job roles [1]. Other models related to role-based access control have been developed for use among multiple enterprises that share data and resources [2-4].

The role abstraction is a disadvantage when other considerations beyond job title are relevant to access decisions. Considerations relevant to a finance officer's ability to approve payments may include the officer's security clearance and the security clearance and department of the record to be paid for, as well as the time the payment request was received. Multiple roles needed to model all relevant considerations for a finance officer can lead to role explosion [5]. Attribute-based access control models can capture these and other considerations directly as attributes of the user, resource, and environment. Simple lists of attribute rules are more computationally intensive to evaluate because all attribute rules must be checked each time to see whether they apply to a particular request [6] and harder to administer because user permissions

are not tied directly to the user [5]. Some proposed hybrid role-based and attribute-based models are being implemented in multi-tenant cloud and multimodal settings [7,8].

A second challenge applies to distributed systems. Having multiple services share an authorization service can create performance and failure risks for the shared service. Previous authorization decisions shared among services have been proposed as a solution to these problems for hierarchical RBAC [9]. Environmental attributes in authorization decisions for services based on attribute-based authorization typically vary among requests, preventing or complicating safe sharing of cached authorization decisions that include environmental attributes. Revocation challenges also prevent some sharing of authorization decisions, the permit stored in a cache needs to be revoked as role assignments or other request environmental attributes change for a request. Previous research on hybrid models has explored policy issues and complexity but provided few details on solutions for fast, consistent access to a shared authorization service in distributed environments other than role-based systems.

This paper offers one solution to two problems. The policy solution stores policy in role assignments and constraints on role rights, not user rights, saving space in the policy store by an additive rather than multiplicative factor related to the number of attribute types. The policy decision process looks up rights by role, saving a large number of policy checks per request. A two-stage policy decision process caches results by subject and resource attributes and performs a second policy decision check for each request by environment attributes, enabling broad policy caching without weakening the context conditions. Application deployment includes a replicated policy decision service and embedded policy enforcement points in each application, plus services to audit and revoke cached policy decisions as needed. The paper investigates three research questions. RQ1 investigates policy size and accuracy differences between the hybrid model and models based on RBAC and ABAC as the number of attribute conditions grows. RQ2 investigates decision cost differences between the hybrid model with and without a two-stage cache, such as rule and attribute lookups and other delays. RQ3 investigates decision service and administrative cost differences between the hybrid model and other models as employees are added to or removed from the workforce in a distributed application environment. Contributions of the paper include a formal specification of a hybrid policy and two-stage decision process, a distributed application design with invalidation features, and a Python prototype evaluated for policy size, accuracy, and decision and administrative costs compared with RBAC and ABAC models.

2. Related work

2.1. Role-based access control in distributed and multi-domain systems

An object-oriented RBAC model for distributed systems was developed to address access issues for shared resources managed by different organizations and used for access in multiple domains [2]. The paper presents a linear-time algorithm for role assignment. An access control model originally developed for internal CORBA applications was adapted for Internet-based web services for internal businesses and used to enforce a unified policy across web services and internal middleware interfaces [10]. Tools and infrastructure for implementing and managing RBAC policies using XML were developed to support the model. Policy harmonization issues among multiple organizations using RBAC policies were addressed as a policy integration problem and resolved using integer programming techniques to bound loss of autonomy for the involved organizations [3]. Problems with role mapping in multiple RBAC systems were identified, and a method for identifying such problems that reduces the number of roles involved in the analysis in a distributed environment was presented [11]. The X-FEDERATE framework adds new XML and UML capabilities and enforcement infrastructure to RBAC to enable information sharing among organizations. A digital library federation was demonstrated with the framework [4].

Enterprise collaboration research at the organizational level explored access control mechanisms using an organizational-level model implemented as a service in a service-oriented architecture and a pattern language approach to organizational modeling [12]. Inconsistencies between the semantics of authorization and protected information resulted in development of an authorization system based on Semantic Web technologies [13]. Features of the authorization system include hierarchical and conditional RBAC and object hierarchies, as well as an authorization architecture for distributed systems with privacy and trust management. Research and modeling of RBAC-based authority delegation explored a function-based model allowing users to define their own delegation requirements and authentication and other requirements using quantifiers [14]. Feature modeling tools and UML were used to develop a tailored RBAC solution with selected features applicable to specific applications, including partial inheritance and compatibility rules to support composition and verification [1].

2.2. Performance and verification of distributed authorization

A centralized authorization service creates a single point of failure for distributed applications. Authorization recycling methods for hierarchical RBAC allow enforcement points to recycle authorization data to make precise or approximate authorization decisions and provide other benefits, including protection from single points of failure related to the authorization server. These methods have been tested analytically and through prototype and simulation studies [9]. Verification methods include an abductive analysis method for answering questions about users' permission access in rule-based access control systems overseen by multiple administrators, even when some policy information is incomplete [15]. A model checking method that includes RBAC reasoning has been developed and tested for use in verifying security properties of inter-operating policies in multi-domain cloud systems [16].

2.3. Attribute-based and hybrid models for cloud and multi-tenant systems

A study comparing discretionary, mandatory, role, attribute, and usage control models against cloud computing requirements identified issues with current access control mechanisms in cloud computing and proposed access control extensions [5]. Testing of a semantic-access control system for multi-tenant clouds that included role, hierarchical, and conditional role access control schemes and hierarchical objects, as well as a trust model to help control tenant access to information, was conducted with OpenStack [17]. Multi-tenant access control using role-based access control mechanisms with trust relationships between cooperating tenants was formally described and enhanced to include finer-grained trust relationships and an administrative model [18]. A prototype of the system had an average policy response time of 12 ms. One study suggested a system that uses both RBAC and ABAC to provide users with access to the correct tenant in a multitenant cloud application [7]. Another study suggested a multitenant ABAC system for cloud infrastructure services that has been modified for use in Intercloud applications through techniques such as token exchange to simplify logical policies and save evaluation time. This system was implemented and tested in the GEYSERS project [6]. Another study suggested an extensible access control system for cloud applications with features to provide multiple access control options and dynamic updates to resource attributes. The DSpace application was used to test this access control system [19]. For multicloud and IoT applications, a case study of multimodal systems compared a role, attribute, access mode, and resource type hybrid system with extended attribute models of RBAC and ABAC [8]. Advantages of the hybrid approach include simplified security enforcement and a reduced number of rule and role definitions compared with RBAC and ABAC. Other researchers have suggested a capability-based alternative to RBAC and ABAC for securing an IoT system using signed, revocable tokens with delegation permissions to address scalability and other issues with RBAC and ABAC in multidevice systems [20].

2.4. Comparison of the reviewed studies and research gap

Table 1 compares the papers reviewed by five criteria. Comparison criteria include whether a specific policy model was presented, whether distributed or multidomain policy deployment was covered, whether attributes or context conditions were supported in policies, whether policy caching or replication was covered, and the type of evaluation described in the paper.

Table 1. Comparison of the reviewed access control studies.

Study	Policy basis	Distributed or multi-domain	Attribute or context conditions	Caching or replication	Evaluation reported
[2]	Object-oriented RBAC	Yes, multiple domains	No	No	Algorithm complexity
[10]	RBAC for web services	Yes, applications coupled over the Internet	No	No	Enforcement infrastructure and tools
[3]	RBAC policy integration	Yes, multidomain	No	No	Integer programming formulation
[4]	Extended RBAC (X-FEDERATE)	Yes, federated	No	No	Prototype in a digital library
[12]	Organisation-model based	Collaborative enterprise, SOA	No	No	Conceptual architecture

[11]	RBAC with role mappings	Yes, interoperating systems	No	No	Detection method analysis
[14]	RBAC delegation	Not specific	Delegation constraints	No	Model analysis
[13]	Semantic RBAC, conditional RBAC	Yes, distributed architecture	Conditions	No	Implementation description
[9]	Hierarchical RBAC	Yes, distributed applications	No	Decision recycling	Analysis, simulation, prototype
[1]	RBAC feature model	Not specific	No	No	Banking application, tool
[20]	Capability-based	Yes, IoT services	Token conditions	No	Project implementation
[17]	Semantic RBAC, hRBAC, cRBAC	Yes, multi-tenant cloud	Conditions	No	OpenStack proof of concept
[15]	Rule-based with administration	Not specific	Facts and rules	No	Abductive analysis algorithm
[16]	RBAC reasoning	Yes, multi-domain cloud	No	No	Verification performance tests
[7]	RBAC combined with ABAC	Yes, multi-tenant cloud	Attributes	No	Mechanism description
[18]	RBAC with cross-tenant trust	Yes, multi-tenant cloud	No	No	Prototype, 12 ms decision overhead
[5]	Survey of five models	Cloud requirements	Discussed	No	Comparative review
[6]	ABAC	Yes, multi-tenant, Intercloud	Attributes	Decision diagrams	Prototype performance evaluation
[19]	Multiple models combined	Cloud SaaS	Dynamic attributes	No	DSpace case study
[8]	Hybrid roles and attributes	Cloud and IoT applications	Attributes	No	Empirical policy complexity comparison

Two weaknesses of the table are described. Modeling and analysis studies of role combinations with other policy features [7,8,19] include models and analyzes of policy complexity but do not consider reuse and invalidation issues of authorization policies used in distributed systems with many distributed applications sharing one decision service. Reuse and invalidation issues of authorization policies in distributed systems with many distributed applications sharing one decision service include authorization delay and consistency problems. Research on authorization decision reuse in distributed systems [9] includes studies of RBAC policies with hierarchical structures and cached authorization decisions that do not depend on contextual information about the user's environment. The findings cannot be applied to other types of policies that depend on network and device conditions and time of day. None of the cited research includes findings about policy size, decision accuracy, and decision costs with a hybrid approach and decision caching in distributed applications, topics of this paper.

3. Problem formulation

A distributed enterprise application system consists of applications that perform operations on resources for authenticated users. Authorization for a distributed enterprise application system involves deciding whether to allow users to access resources in a manner consistent with the organization's security policy for the enterprise. Security policy goals for the enterprise applications addressed in this article include limiting users' access to appropriate job-related operations, limiting users' access to resources with matching departmental, sensitivity, and location attributes, and limiting access to sensitive operations to trusted networks and devices and specific times during the workday. The first goal includes a role requirement. The other two goals include user and resource and request environment attribute requirements.

Let U be the set of users, R the set of roles, O the set of resources, T the set of types, and A the set of actions. Let type: O → T be a type function. Users have attributes att(u) (department, location, clearance), and resources have attributes att(o) (owning department, location, sensitivity). Environments have attributes e (network, device, time) associated with

requests. User-role assignments UA ⊆ U × R and role hierarchy RH ⊆ R × R are defined as in the standard RBAC model. The set of roles available to a user includes all inherited roles:

$$roles^{*}(u) = \{ r \in R : (u, r) \in UA \} \cup \{ r' \in R : \exists\, r, (u, r) \in UA \wedge (r, r') \in RH^{*} \} \quad (1)$$

where RH* is the transitive closure of the role hierarchy. There are five requirements for the access control framework: The evaluation algorithm must take attribute values into account (R1). The size of the framework's policies must grow in proportion to the number of attribute types (R2). The time required to evaluate access decisions at enforcement points must be proportional to a small constant multiple of the number of policy and attribute queries made during evaluation (R3). Changes to role assignments or attribute values made for an individual must have a near-instantaneous effect on future access decisions made at enforcement points for that individual (R4). The time required for an administrator to make a change to a role assignment or attribute value for an individual must be proportional to the number of authority changes made, not the total number of permissions held by the individual (R5).

4. Proposed role-based and attribute-based access control framework

4.1. Architecture

Fig. 1 shows a three-layer framework model. The identity and governance layer includes the identity provider, role store, attribute provider, and policy administration point. The identity provider verifies a user's identity and sends a signed bearer token containing user identity and role information from UA and RH to applications. Applications do not need to access the role store for user identity and role information for every user request. The attribute provider provides subject and resource information. The policy administration point supplies permission assignments and the attribute constraints attached to them.

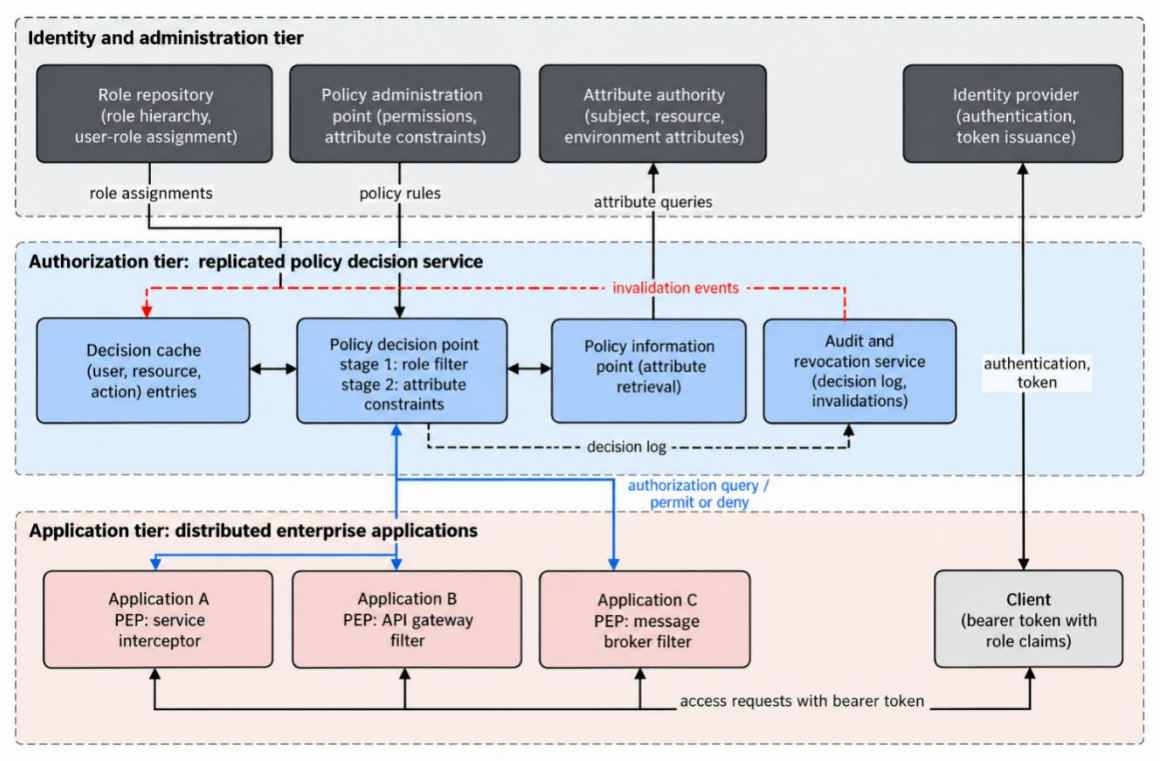


Fig. 1. Architecture of the proposed framework across the identity, authorization and application tiers.

The authorization tier is a replicated policy decision service. Each node of the replicated service includes four elements. The policy decision point (PDP) includes two processing steps, one of which is stored in a decision cache. The policy information point (PIP) communicates with the attribute authority to standardize attribute data. The audit and revocation service records authorizations and revocations. The application tier includes distributed enterprise applications that contain embedded policy enforcement points (PEPs) at the request entry point within the application. PEPs are configured as service interceptors, API gateway filters, or message broker filters based on the application technologies used. PEPs extract user ID and role information from a bearer token to identify the resource and action being requested. The PEP also collects

other information about the request environment and forwards an authorization request to the nearest PDP node. Once a PDP response of permit or deny is received, the PEP enforces that response before the request reaches the business logic layer.

4.2. Policy model

The policy is a role permission assignment PA ⊆ R × T × A that defines which roles have access to which resource types with which actions and a constraint function C that maps each permission to a pair of predicates. One predicate Cs(r, t, a) over subject and resource attributes is a static predicate, and another predicate Ce(r, t, a) over environmental attributes related to the access request is an environmental predicate. Access request (u, o, a, e) is permitted if the user has a role with access to the requested resource type and action for which both predicates evaluate to true.

$$permit(u, o, a, e) = 1 \iff \exists\, r \in roles^{*}(u) : (r, type(o), a) \in PA \wedge C_s(r, type(o), a)(att(u), att(o)) \wedge C_e(r, type(o), a)(e) \quad (2)$$

Constraints are imposed on permissions rather than users or other objects. The static predicate used in this research has three elements: Tests for matching departments when the permission is specific to a department. Tests for matching or higher sensitivity clearance for the user than for the object. Tests for matching physical locations of the user and object when the object is the most sensitive. The environmental predicate used in this research requires access to an internal or virtual private network and a managed device for write, approve, and delete access and matching or overlapping work hours for approve and delete access. Other predicates with the same elements could be substituted without making changes to the architecture.

The size of a policy is the total number of roles and permissions and number of constraint terms. In a policy with J job roles, M permissions, and k attribute dimensions, each having at most one constraint term per permission, the size of the policy is:

$$N_{hybrid} = J + M + k \cdot M \quad (3)$$

A pure RBAC model that produces the same decisions as some other model needs a role for every set of attribute values, since roles do not have conditions attached to them. With k attribute types and c1, …, ck possible values for each attribute type, the number of roles needed is:

$$N_{RBAC} = J \cdot c_1 \cdot c_2 \cdot \ldots \cdot c_k \quad (4)$$

A flat ABAC model with no roles has M policies with subject constraints that would be provided by roles in other models and k attribute value constraints. NABAC, the number of elements in a flat ABAC model, is M·(1 + k). Equations (3) and (4) show that the hybrid and full-context RBAC models grow linearly with k. The RBAC model grows with the product of the sizes of all attribute value sets and satisfies requirement R2.

4.3. Two-stage decision procedure

The decision process uses Eq. (2) in two steps, as outlined in Algorithm 1. In step 1, role assertions are used to index into PA, and subject and resource attribute information is obtained from the PIP. Static evaluation is then performed. The result of step 1 depends only on subject, object, and action information and is cached in the decision cache under key (u, o, a). For step 2, evaluation of the environment predicate is performed on the environment vector in the request message. No additional attribute information is required for this step. The final decision result is the logical AND of the results of steps 1 and 2. Using role assertions as indexing keys in step 1 ensures that only one rule needs to be evaluated for any request, regardless of how many policy rules there are, which meets requirement R3. In a flat ABAC implementation, all rules must be scanned to find a matching rule. As the number of policy rules grows, so does the number of rule evaluations.

Algorithm 1. Two-stage access decision at a policy decision point replica.

Input: user u with role claims roles^*(u), resource o, action a, environment vector e

Output: permit or deny

1: key ← (u, o, a)

2: if key ∈ cache then

3: s ← cache[key] ▷ stage 1 result reused

```
4:  else
5:      s ← false
6:      for each r ∈ roles^*(u) do
7:          if (r, type(o), a) ∈ PA then                ▷ role used as index into PA
8:              att_u ← PIP.subject(u); att_o ← PIP.resource(o)
9:              s ← C_s(r, type(o), a)(att_u, att_o); break
10:     cache[key] ← s; evict least recently used entry if |cache| > capacity
11: if s = false then return deny
12: if C_e(r, type(o), a)(e) = false then return deny      ▷ stage 2, no attribute retrieval
13: log(u, o, a, e, permit); return permit
```

Let h be the hit rate in the decision cache. Suppose that a noncached decision fetches one subject and one resource record. The average number of attribute fetches for each request is:

$$L = 2 \cdot (1 - h) \tag{5}$$

and the average number of rule checks per request is 1 − h. The hit ratio depends on workload characteristics and the size of the cache. The hit ratio is measured in experiments.

4.4. Revocation and cache invalidation

Cached first-stage outputs are invalidated by changes to a user's roles or to user and resource attributes and policies. The audit and revocation system is notified of changes by the role repository and user and resource attribute and policy systems and alerts all cache nodes that cached outputs are invalid. Alerts about user changes cause all cached outputs related to the changed user to be deleted. Alerts about resource changes cause all cached outputs related to the changed resource to be deleted. Alerts about changes to policies result in deletion of all cached outputs. Environmental data are not cached, so changes in environmental data, including location and time of day, take effect immediately the next time any service is used and do not require invalidation messages. Bearer tokens have short expiration times and must be refreshed regularly to account for current role assignments. This helps ensure that former role assignments are not used by PEPs for a limited period of time. These measures satisfy requirement R4.

4.5. Administration

Administrators manage three kinds of objects: roles in a role store, attributes in an attribute authority, and permissions and other policies in a policy administration point. A change in a user's job department or security clearance can be handled by changing an attribute for the user. A change in job function for a user can be handled by assigning the user to a new role. Neither kind of change requires changes to permissions or the creation of new roles corresponding to new combinations of attribute values to meet requirements R5 and R6. In a context-based RBAC implementation, changes to a user's job department or security clearance can be addressed by reassigning the user to an existing role representing the attribute combination or by creating a new role and reassigning the user. Changes to a user's job function can be addressed by reassigning the user to an existing role representing the attribute combination or by creating a new role and reassigning the user.

5. Experimental evaluation

5.1. Synthetic enterprise dataset

No publicly available datasets provide data on organization access requests and other attributes that can be used for testing access control systems. A synthetic organization, as described in Table 2, was built for testing the three access control models described in this paper. Attributes for users were assigned randomly to represent job role, department, location, and security clearance. Attributes for resources were assigned randomly to represent type, department, location, and sensitivity. Other attributes for users and resources in each access request in the organization's access log were assigned randomly for session-based access requests. User popularity followed a Zipf distribution with an exponent of 1.0 based on the number of users in the organization. Each user worked with 30 resources, 70% of which were in the same department

as the user. Resource popularity for each user followed a Zipf distribution with an exponent of 0.8. Common actions included read, write, approve, and delete, occurring with probabilities of 60%, 25%, 10%, and 5%, respectively. Other environmental attributes were assigned independently to each access request. Decisions on access requests were based on the organization's full security policy, including job role access and other subject and resource attributes and environmental conditions on sensitive actions. The three models were evaluated against reference decisions.

Table 2. Characteristics of the synthetic enterprise dataset.

Characteristic	**Value**
Users	5,000 (4,992 appear in the trace)
Resources	20,000 (18,057 appear in the trace)
Access requests	200,000
Job functions	8 (analyst, developer, finance officer, HR officer, manager, auditor, administrator, contractor)
Departments / locations	6 / 3
Resource types / actions	6 / 4
Clearance and sensitivity levels	4
Permissions in the policy	38 (role, type, action) triples
Action mix in the trace	read 59.9%, write 25.0%, approve 10.0%, delete 5.1%
Environment mix in the trace	working hours 75.1%; internal 60.0%, VPN 25.0%, external 15.0%; managed device 80.0%
Reference permit rate	16.2% of requests

5.2. Implementation and experimental setup

The three policy and request-generating programs were implemented in Python using standard libraries and the matplotlib package for generating graphs. Roles and permissions were implemented for the RBAC baseline to match each job role and organizational unit combination and were implemented without attributes. In the ABAC baseline, one rule was implemented for each permission, incorporating attribute conditions, and policies implemented in this way were tested until a matching policy was found. The hybrid model was implemented according to Algorithm 1 with a least recently used cache. Attributes were stored in a serialized database in JSON format to model the overhead of accessing attributes stored in a directory service. All three policy implementations used the same attribute and trace data. Table 3 reports parameter values used for the implementations. Measurements of time were taken on two virtual cores of an Intel Xeon 2.1-GHz processor with 8 GB of memory. Standard deviations and medians of replicated measurements are reported.

Table 3. Implementation parameters of the evaluated models.

Parameter	**Value**
Decision cache capacity (default)	4,096 entries, least recently used eviction
Cache capacities in the sensitivity study	256 to 16,384 entries
PDP replicas in the replication study	1, 2, 4, 8, 16 (trace partitioned round robin, one cache per replica)
Enterprise sizes in the scalability study	1,000 to 50,000 users with four resources per user, 50,000 requests each
Repetitions per timing measurement	7 (main comparison), 5 (scalability), 3 (cache sensitivity)

Workforce events in the administration study	1,000 (329 transfers, 343 job changes, 328 clearance changes)
Random seed	7

5.3. Evaluation metrics

Decision accuracy is compared to a reference decision. Let TP be the number of requests granted by both the model and the reference, TN the number of requests rejected by both the model and the reference, FP the number of requests granted by the model but rejected by the reference, and FN the number of requests rejected by the model but granted by the reference. Decision accuracy is (TP + TN) / (TP + TN + FP + FN). The over-privilege rate is the percentage of requests rejected by the reference and granted by the model:

$$OP = FP / (FP + TN) \quad (6)$$

and the under-privileged rate is FN / (FN + TP). Decision costs include the number of rules evaluated per query, the number of attributes retrieved per query, the median decision time in microseconds, and the number of decisions per second. Policy size includes the number of policy elements, as defined in Eq. (3) and Eq. (4). Administrative costs include the number of operations required to carry out a series of workforce actions, where each change to an attribute, role revocation, role assignment, or role creation is one operation.

5.4. Policy size

Fig. 2 shows the number of policy elements in a policy with six security requirement attribute dimensions. With zero attribute dimensions, the number of elements in the three policies is about the same: 8 for RBAC, 38 for ABAC, and 46 for a hybrid policy. Adding one attribute dimension adds the number of attribute dimension values to the number of RBAC roles. Adding all six attribute dimensions increases the number of RBAC roles to 6,912. Adding all six attribute dimensions increases the number of elements in ABAC and hybrid policies by one element per permission to 266 and 274 elements, respectively. With only the security requirement attribute for department, the RBAC policy has 48 roles and 738 role-permission relationships, compared with 8 roles and 38 constrained permissions in the hybrid policy. The hybrid policy has a similar number of policy elements to the ABAC policy, with only a 3% difference in the number of roles for the policy with only the security requirement attribute for department.

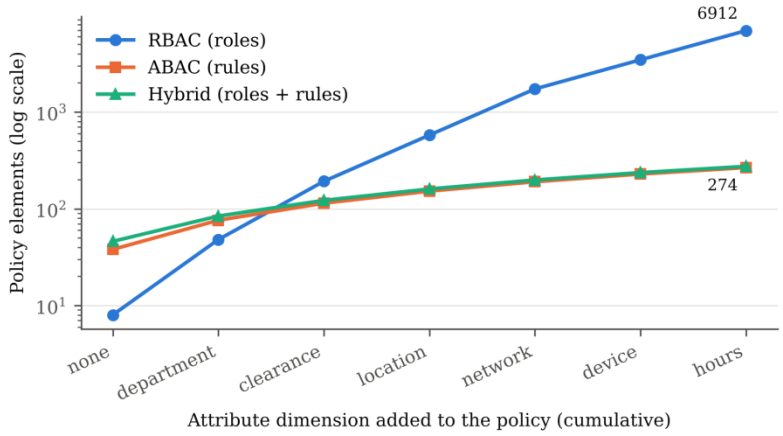


Fig. 2. Policy elements required by each model as attribute dimensions are added.

5.5. Decision correctness and decision cost

Table 4 presents decision accuracy and cost results for a data set with 200,000 requests. RBAC has a decision accuracy of 0.814 and 22.2% over-privilege for requests containing claims with the same value as role attributes. Other information, including geographic location, network, device type, and time of request, is not considered. No under-privilege occurs using RBAC. Decisions for other models, including the ABAC baseline and hybrid model, have 0% over- and under-privilege and match reference decisions for all requests. Other differences exist between the two models. The ABAC model evaluates 30.4 rules per request because it evaluates rules in sequence until a match is found. The hybrid model evaluates one rule per request because role claims are directly linked to their associated permissions. Median latency for the hybrid

model with no cache is 6.33 µs, close to the 6.28 µs median latency for the RBAC model and less than the 8.43 µs median latency for the ABAC model. A cache with 4,096 entries can be used to reuse first-stage results for 38.5% of requests. The average number of attribute calls and rule evaluations per request decreases to 1.23 and 0.62, respectively. Median latency with the cache is 5.12 µs, and decision throughput is 195,000 decisions per second.

Table 4. Decision correctness and decision cost on the 200,000-request trace.

Metric	RBAC	ABAC	Hybrid, no cache	Hybrid, cache 4,096
Accuracy	0.814	1.000	1.000	1.000
Precision of permits	0.455	1.000	1.000	1.000
Over-privilege rate (%)	22.2	0.0	0.0	0.0
Under-privilege rate (%)	0.0	0.0	0.0	0.0
Rule evaluations per request	1.00	30.39	1.00	0.62
Attribute retrievals per request	2.00	2.00	2.00	1.23
Cache hit ratio (%)	n/a	n/a	n/a	38.5
Median latency (µs), SD	6.28, 1.41	8.43, 1.22	6.33, 1.63	5.12, 0.72
Throughput (decisions/s)	159,300	118,600	157,900	195,500

Fig. 3(a) shows the error costs per query for the four models. Fig. 3(b) shows more details about the over-privilege of the role-based access control model by query type. Read queries have the highest over-privilege at 35.3% because the role-based model permits most read queries and lacks security and location constraints for read reference checks. Write, approve, and delete queries have lower over-privilege rates of 9.3%, 3.6%, and 1.8%, respectively, because most of these queries fail security-based reference checks but pass role-based access control checks based on role. The hybrid model has no over-privilege for any query type.

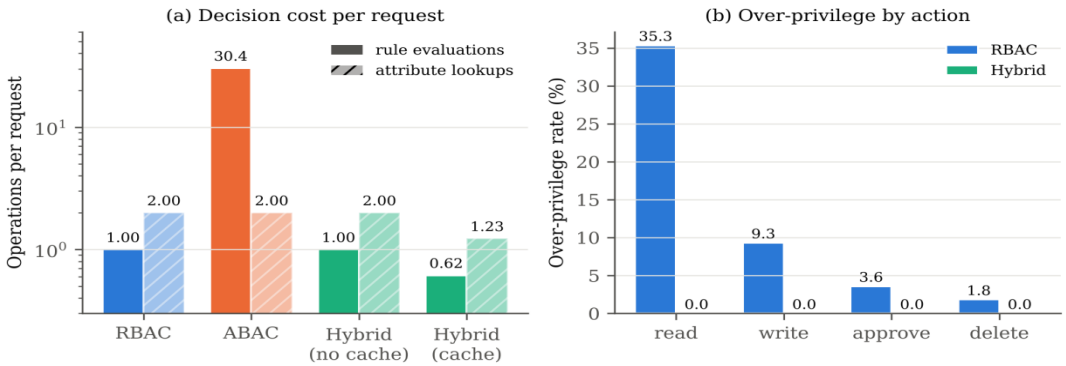


Fig. 3. Decision cost per request for the evaluated models and over-privilege rate of RBAC and the hybrid model by action.

5.6. Scalability and replication

Table 5 provides median decision-making times for all three models at enterprise-level scales of 1,000 to 50,000 users, with four resources assigned to each. Decision-making time does not increase with user and resource scale for any of the models tested, given that the three key data structures used in the models (the role index, attribute data store, and cache) all have lookup times that are not affected by user or resource scale. Variations in decision-making time across scales are

within the repetition noise of the timing environment. The ABAC model is slower than the others because it requires rule evaluation for all rules, and the hybrid model with cache is equal to or faster than the other models at all tested scales.

Table 5. Median decision latency in microseconds for increasing enterprise size.

Users	Resources	RBAC	ABAC	Hybrid, cache 4,096
1,000	4,000	7.72	9.23	4.83
2,000	8,000	4.59	7.23	3.46
5,000	20,000	5.03	6.54	3.48
10,000	40,000	4.34	7.01	3.99
20,000	80,000	4.64	10.07	5.20
50,000	200,000	4.34	9.44	4.61

Fig. 4(a) shows the hit rate for the hybrid approach as a function of cache size. The hit rate rises from 10.0% at 256 entries to 55.1% at 16,384 entries. The number of attributes requested per access falls from 1.80 at 256 entries to 0.90 at 16,384 entries, as expected from Eq. (5). The hit rate for the hybrid approach does not top out at 16,384 entries because the trace includes 4,992 active users, each with 30 resources and four actions. A 16,384-entry cache cannot store all active (user, resource, action) combinations. Fig. 4(b) shows system throughput with one to 16 PDP replicas, each with its own 4,096-entry cache. System throughput is 0.26 million decisions/sec with a single replica and 3.70 million decisions/sec with 16 replicas. Test results fall somewhat short of linear scaling due to the lower local cache hit rate per replica, 34.1%, compared with 38.5%, resulting from splitting the trace across replicas. There is no shared state among replicas other than invalidation messages. More local replicas can be deployed to support other applications making requests.

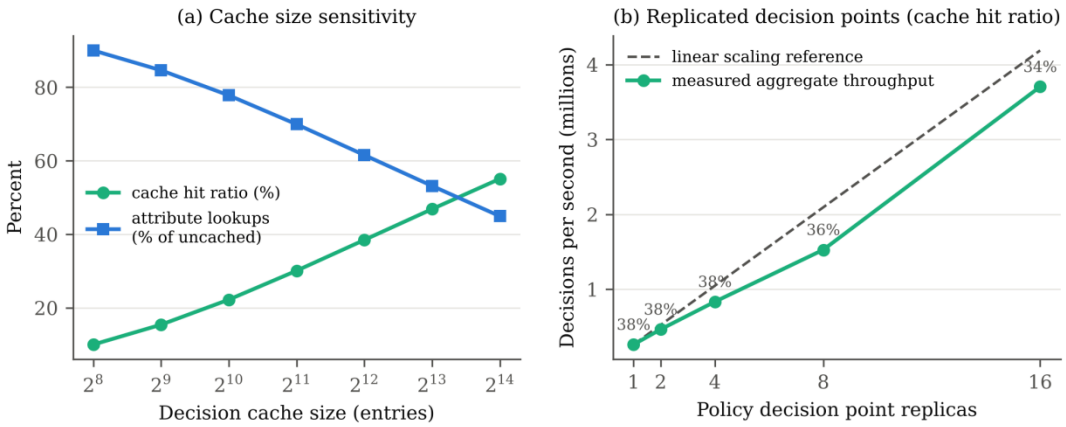


Fig. 4. Cache capacity sensitivity of the hybrid model and aggregate throughput of replicated decision points.

5.7. Administrative cost

A sequence of 1,000 HR actions was simulated for a user population of 5,000, 329 of whom changed departments, 343 changed job functions, and 328 changed security clearances. For a complete RBAC implementation with one role per combination of job function, department, and security clearance, all 1,000 HR actions required one revocation and one grant, for a total of 2,000 actions. No new role definitions were needed because all 5,000 users represented all combinations of the three attributes. Under the hybrid model, one department or security clearance change required changing one attribute, and one job function change required a revocation and a grant, for a total of 1,343 actions, 33% fewer than under RBAC. Benefits of the hybrid model grow with the number of attributes. Additional attribute dimensions require additional RBAC roles for reassignment of users. Additional attribute dimensions under the hybrid model require changes to just one attribute.

6. Discussion

Results validate all three research questions. Results for RQ1 show that partially assigning attributes to roles in the hybrid policy increases policy size by less than 3% compared with a full ABAC policy and substantially reduces policy size compared with a full RBAC policy of thousands of roles that include contextual information. All access decisions made under the hybrid policy agreed with a benchmark policy's access decisions on decision requests, and over 20% of denied access requests were granted access under a full RBAC policy. Results for RQ2 show that accessing roles during policy evaluation reduces policy evaluation to one rule per decision request. Using a two-component policy structure can allow caching to decrease attribute queries while keeping environmental information in policy decisions. Performance differences among architectures are small for a single user session. Hardware-independent measures of policy rule evaluation and attribute queries show potential to greatly reduce network traffic for attribute queries. Results for RQ3 show that read-only policy replicas sharing only policy invalidation messages provide near-linear scaling of transaction processing. Administrative tasks related to staff changes are reduced by two-thirds compared with full RBAC policies.

The design choices mitigate problems identified in previous work in various ways. Use of a decision cache implements the reuse pattern of authorization recycling [9] and provides it for the first phase of decision making only, as required for reuse when environmental attributes are present in the policy. Use of a role index implements design choices to improve policy evaluation performance in a multitenant cloud [6] and applies them to the role component of a hybrid policy. Additivity of the policy is consistent with previous findings that policies do not experience exponential growth in the number of roles and policies when hybrid policies are deployed [8]. Findings of this work include information on the cost of decision making using a hybrid policy and deployment approaches for hybrid policies in distributed applications. Bearer tokens that include role information have a similar use in policy-based federated access management systems [4], helping to avoid calls to other applications to obtain user data.

The study has limitations. Data used in the study are simulated. The trace used in the study to represent locality of use and calculate cache hit rates was simulated, not generated from a business application. Results for cache hit rates in the study should not be applied directly to other business applications but only to applications with similar levels of locality of use. Time measurements reported in the study were taken for operations performed by a single process and do not reflect communications times between PEPs and PDP replicas and an attribute authority. Real-world implementations are likely to have higher total times. The PDP replica study time measurements were based on splitting the trace between replicas. Time costs of invalidation messages were not considered in the study. Outdated first-stage information might be provided by a PDP replica to users. The framework described in the study protects interfaces to applications running within the business environment but does not protect data stored outside the business environment.

7. Conclusions

This paper presents an identity and access management solution for distributed enterprise applications that combines role-based and attribute-based access control strategies. Benefits of role-based access control are preserved by maintaining roles as the primary administrative and policy-granting entities. Attribute rules are introduced as part of role-based policies. A two-phase policy decision process combines a cacheable phase based on subject and resource attributes with a phase for individual access requests involving environment attributes. Multiple instances of decision service and agent components embedded in enterprise applications offer notifications about policy changes. A test with simulated enterprise application data and 200,000 session requests found that the combined approach used 274 role- and attribute-based policies compared with 6,912 roles for full-context role-based access control. All access requests were granted or denied according to reference policies. Role-based access control without full context granted access to 22.2% of requests that would have been denied under other policies. The combined approach provided accurate access decisions using a single policy rule evaluation per request compared with 30.4 evaluations per request for attribute-based access control. Attribute queries per request were reduced to 1.23 by using a cache with 4,096 entries. System throughput increased almost linearly with 16 replicas of the decision service. Policy changes involving workforce data required 33% fewer administrative actions than role-based access control. Future research includes tests with real-world session request data from an enterprise application implementation, measurements of end-to-end latencies involving remote attribute authorities and PDP replicas, and other evaluations of the usefulness of invalidation notifications.

References

[1] S. Kim, D.K. Kim, L. Lu, S. Kim, S. Park, A feature-based approach for modeling role-based access control systems, Journal of Systems and Software 84 (12) (2011) 2035–2052. https://doi.org/10.1016/j.jss.2011.03.084

[2] C.N. Zhang, C.G. Yang, Designing a complete model of role-based access control system for distributed networks, Journal of Information Science and Engineering 18 (6) (2002) 871–889.

[3] B. Shafiq, J.B.D. Joshi, E. Bertino, A. Ghafoor, Secure interoperation in a multidomain environment employing RBAC policies, IEEE Transactions on Knowledge and Data Engineering 17 (11) (2005) 1557–1577. https://doi.org/10.1109/TKDE.2005.185

[4] R. Bhatti, E. Bertino, A. Ghafoor, X-FEDERATE: a policy engineering framework for federated access management, IEEE Transactions on Software Engineering 32 (5) (2006) 330–346. https://doi.org/10.1109/TSE.2006.49

[5] M. Beraka, J. Al-Muhtadi, Critical comparison of access control models for cloud computing, Journal of Internet Technology 16 (3) (2015) 431–442. https://doi.org/10.6138/JIT.2015.16.3.20130124

[6] C. Ngo, Y. Demchenko, C. de Laat, Multi-tenant attribute-based access control for cloud infrastructure services, Journal of Information Security and Applications 27–28 (2016) 65–84. https://doi.org/10.1016/j.jisa.2015.11.005

[7] N.W. Lo, T.C. Yang, M.H. Guo, An attribute-role based access control mechanism for multi-tenancy cloud environment, Wireless Personal Communications 84 (3) (2015) 2119–2134. https://doi.org/10.1007/s11277-015-2515-y

[8] H. Ben Attia, L. Kahloul, S. Benharzallah, A new hybrid access control model for security policies in multimodal applications environments, Journal of Universal Computer Science 24 (4) (2018) 392–416.

[9] Q. Wei, J. Crampton, K. Beznosov, M. Ripeanu, Authorization recycling in hierarchical RBAC systems, ACM Transactions on Information and System Security 14 (1) (2011) Article 3. https://doi.org/10.1145/1952982.1952985

[10] T. Fink, M. Koch, C. Oancea, Specification and enforcement of access control in heterogeneous distributed applications, in: Web Services, ICWS-Europe 2003, Lecture Notes in Computer Science, vol. 2853, Springer, 2003, pp. 88–100.

[11] X.Y. Wang, J.L. Sun, X.H. Yang, C. Huang, D. Wu, Security violation detection for RBAC based interoperation in distributed environment, IEICE Transactions on Information and Systems E91-D (5) (2008) 1447–1456. https://doi.org/10.1093/ietisy/e91-d.5.1447

[12] F.L.G. Vela, J.L.I. Montes, P.P. Rodríguez, M.S. Román, B.J. Valverde, An architecture for access control management in collaborative enterprise systems based on organization models, Science of Computer Programming 66 (1) (2007) 44–59. https://doi.org/10.1016/j.scico.2006.10.005

[13] J.M. Calero, G.M. Perez, A.F.G. Skarmeta, Towards an authorisation model for distributed systems based on the Semantic Web, IET Information Security 4 (4) (2010) 411–421. https://doi.org/10.1049/iet-ifs.2009.0260

[14] M. Lee, N.D. Cho, T.M. Gatton, A function-based user authority delegation model, Information Sciences 180 (5) (2010) 765–775. https://doi.org/10.1016/j.ins.2009.11.009

[15] P. Gupta, S.D. Stoller, Z.Y. Xu, Abductive analysis of administrative policies in rule-based access control, IEEE Transactions on Dependable and Secure Computing 11 (5) (2014) 412–424. https://doi.org/10.1109/TDSC.2013.42

[16] A. Gouglidis, I. Mavridis, V.C. Hu, Security policy verification for multi-domains in cloud systems, International Journal of Information Security 13 (2) (2014) 97–111. https://doi.org/10.1007/s10207-013-0205-x

[17] J.B. Bernabe, J.M.M. Perez, J.M.A. Calero, F.J.G. Clemente, G.M. Perez, A.F.G. Skarmeta, Semantic-aware multi-tenancy authorization system for cloud architectures, Future Generation Computer Systems 32 (2014) 154–167. https://doi.org/10.1016/j.future.2012.05.011

[18] B. Tang, R. Sandhu, Q. Li, Multi-tenancy authorization models for collaborative cloud services, Concurrency and Computation: Practice and Experience 27 (11) (2015) 2851–2868. https://doi.org/10.1002/cpe.3446

[19] F. Mehak, R. Masood, M.A. Shibli, I. Elgedway, EACF: extensible access control framework for cloud environments, Annals of Telecommunications 72 (5–6) (2017) 307–323. https://doi.org/10.1007/s12243-016-0548-1

[20] S. Gusmeroli, S. Piccione, D. Rotondi, A capability-based security approach to manage access control in the Internet of Things, Mathematical and Computer Modelling 58 (5–6) (2013) 1189–1205. https://doi.org/10.1016/j.mcm.2013.02.006