

# RAG Access Control Audit Kit

## A practical checklist and test pack for document-level authorization

**Version:** 1.0 | **Use with:** enterprise and multi-tenant RAG systems

Semantic retrieval answers “what is relevant?” Authorization answers “what may this principal see?” A production RAG system must answer both separately.

### 1. What this kit helps you review

This kit helps a team inspect whether a RAG pipeline prevents unauthorized source documents and chunks from reaching retrieval results, rerankers, prompt builders, model context, caches, logs, or downstream operators. It covers tenant isolation, document-level permissions, policy freshness, fail-closed behavior, auditability, and adversarial testing.

It does not certify security or compliance. It does not replace a threat-led security assessment, provider review, penetration test, privacy review, or deployment-specific benchmark.

### 2. Threat model worksheet

Complete this section before choosing a vector-store pattern.

| Item                                           | Your answer |
|------------------------------------------------|-------------|
| Protected business data                        |             |
| Authenticated principal types                  |             |
| Tenant or organization boundary                |             |
| Source systems of record                       |             |
| Permission models in use (ACL/RBAC/ABAC/ReBAC) |             |
| Vector store and exact version                 |             |
| Embedding and reranking providers              |             |
| Model provider and region                      |             |
| Maximum acceptable permission-staleness window |             |
| Cache and log recipients                       |             |
| Required deletion and revocation behavior      |             |
| Highest-impact unauthorized disclosure         |             |

## Protected data-flow

Document the path explicitly:

```
identity provider → authorization decision → source permission metadata →
document/chunk record → filtered retrieval → candidate verification →
model context → response
```

For each arrow, identify the component, trust boundary, data retained, and failure behavior.

## 3. Production readiness checklist

Mark each item in the spreadsheet as **Pass**, **Needs work**, **Not applicable**, or **Unknown**.

### Identity and policy

- The principal is authenticated before retrieval.

- Tenant, user, group, role, and attribute values are derived from trusted server-side identity context.
- Client-supplied values cannot widen the authorization scope.
- One authoritative policy decision path exists for `can_read(principal, resource)`.
- Policy revisions or permission snapshots are identifiable.

## **Ingestion and metadata**

- Every chunk maps to a canonical source-document ID.
- Source version, tenant, and permission metadata are recorded.
- Missing or malformed authorization metadata excludes the record.
- Source deletions and revocations propagate to chunks and downstream caches.
- Permission synchronization has a documented and tested freshness window.

## **Retrieval and enforcement**

- Tenant isolation and document authorization are treated as separate boundaries.
- Authorization is enforced before unauthorized content reaches a model, reranker, cache, trace, or analytics system.
- The vector store's actual filter execution semantics are documented for the deployed version.
- Highly selective filters are benchmarked for recall, latency, and cost.
- There is no unfiltered fallback when policy, identity, metadata, or filter construction fails.

## **Caching, logging, and residual risk**

- Cache keys bind the principal or authorization scope, tenant, policy revision, and document version as appropriate.
- Allow and deny decisions are auditable without copying sensitive document text unnecessarily.
- Similarity scores, ranks, and error messages do not expose avoidable corpus side channels.
- Logs, traces, backups, support tools, and model APIs have explicit access and retention rules.
- The design documents what it does not guarantee.

## 4. Adversarial test plan

Use near-duplicate documents across tenants, departments, and clearance levels. Execute the same queries under multiple principals. The expected result is not only a safe final answer; it is that unauthorized content never reaches an untrusted or generative component.

| Test family                 | Expected outcome                                                              |
|-----------------------------|-------------------------------------------------------------------------------|
| Cross-tenant near duplicate | Only the authorized tenant's chunks are eligible and returned.                |
| Revocation after indexing   | The document is excluded within the documented propagation window.            |
| Missing tenant context      | Retrieval is denied; no default or global scope is used.                      |
| Policy service timeout      | Retrieval fails closed or returns a controlled unavailable response.          |
| Malformed filter            | The request is denied; no unfiltered retry occurs.                            |
| Missing ACL metadata        | The record is excluded and the ingestion gap is observable.                   |
| Cross-principal cache reuse | The cached result cannot be reused under a different authorization scope.     |
| Side-channel probing        | Scores/ranks are not exposed; repeated probing is rate-limited and monitored. |

## 5. Choosing an enforcement pattern

| Requirement                                  | Starting pattern                                  | Verify before adoption                                      |
|----------------------------------------------|---------------------------------------------------|-------------------------------------------------------------|
| Strong customer-to-customer separation       | Namespace, dedicated index, or dedicated resource | Provider's exact isolation guarantee and operational limits |
| Different visibility inside one tenant       | Server-derived metadata or policy filters         | Filter semantics, stale metadata, selectivity, and recall   |
| Nested or delegated relationships            | External policy engine or relationship graph      | Consistency, revision semantics, latency, and auditability  |
| Source permissions must remain authoritative | Live source-system authorization/retrieval        | Availability, latency, rate limits, and data-flow exposure  |

## 6. Audit-log minimum fields

Record enough to answer “what was actually enforced?” without turning logs into a second sensitive data store. Suggested fields include request ID, principal ID, tenant, resolved groups/attributes, resource IDs, policy revision, allow/deny outcome, enforced-scope hash, retrieved chunk IDs, timestamp, denial reason category, and error state. Define retention, access, redaction, and replay procedures separately.

## 7. Final review questions

1. Can the system prove which policy revision was used for a past retrieval?
2. What happens when identity, policy, metadata, filter construction, or the vector store is unavailable?
3. Can a revoked document remain in a cache, trace, reranker, or model context?
4. Are tenant isolation and within-tenant document authorization independently tested?
5. Which external providers receive document text, embeddings, metadata, logs, or backups?
6. What is the measured unauthorized-retrieval rate in adversarial tests?
7. Which claims remain assumptions rather than verified deployment behavior?

## References

- [1] [OWASP LLM08:2025 Vector and Embedding Weaknesses](#)
- [2] [PostgreSQL Row Security Policies](#)
- [3] [Pinecone Filter by Metadata](#)
- [4] [Qdrant A Complete Guide to Filtering in Vector Search](#)
- [5] [Milvus Filtered Search](#)
- [6] [Microsoft Learn: Document-level access control in Azure AI Search](#)
- [7] [OWASP RAG Security Cheat Sheet](#)