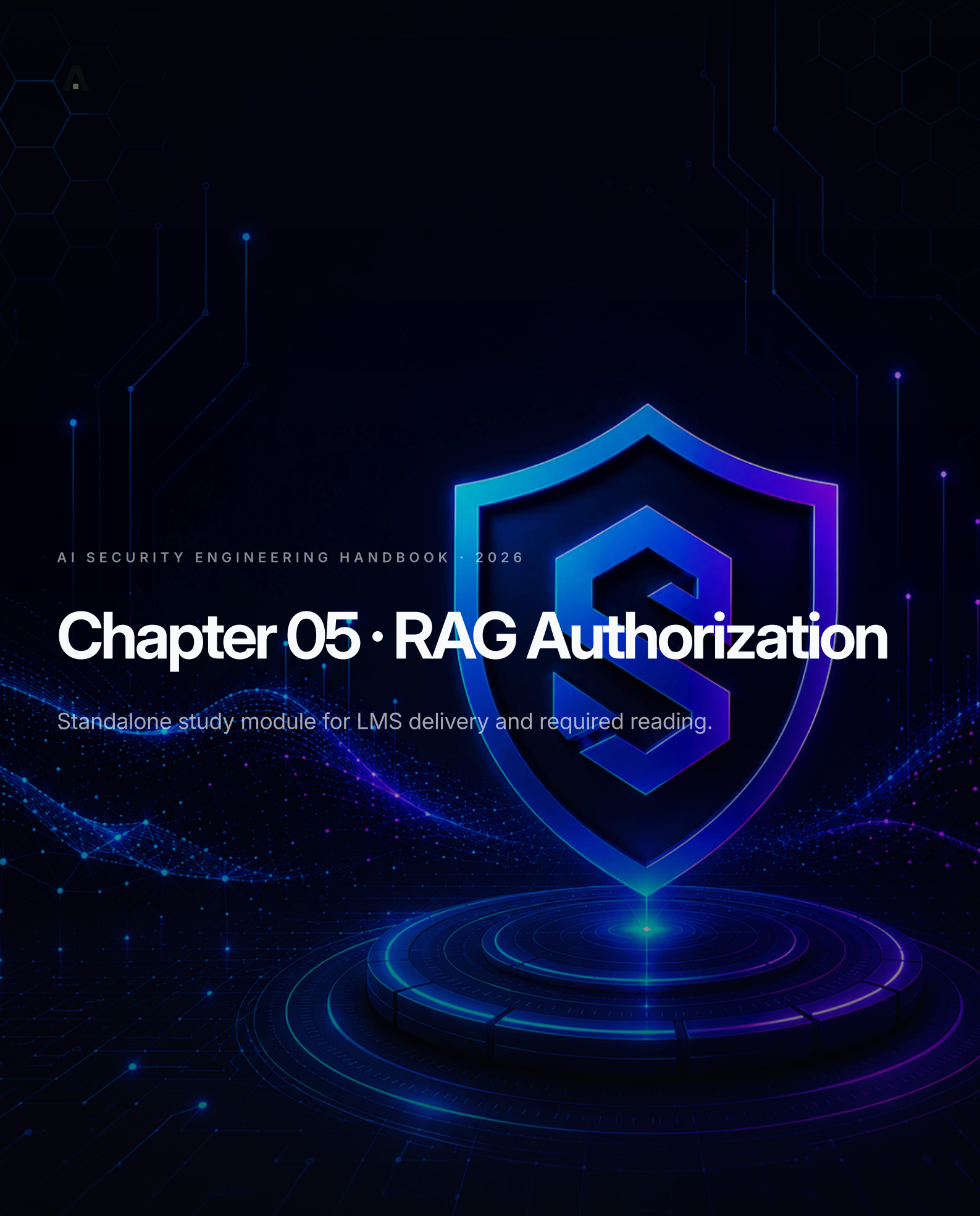
AI SECURITY ENGINEERING HANDBOOK · 2026

Chapter 05 · RAG Authorization

Standalone study module for LMS delivery and required reading.



FORMAT	USE	SCOPE	AUDIENCE
Standalone PDF	Study module	Single chapter	Learners

CHAPTER 05

RAG Authorization

HANDBOOK STUDY COMPANION: STUDY FRAME

Use this chapter to build vocabulary, judgment, and role-readiness. Pair it with the Field Guide when you need applied actions, checklists, and control execution.

STUDY FOCUS

STUDY FOCUS	WHY IT MATTERS
Retrieval authorization, tenant filtering, chunk metadata, permission propagation, citation integrity, and retrieval evidence.	RAG systems fail when retrieval is treated as search rather than an authorization and provenance boundary.

Study Outcomes

- Explain why authorization must happen before context assembly.
- Reason about stale permissions, poisoning, tenant isolation, and citations.
- Identify retrieval evidence needed for assurance and incident response.

DOMAIN MAPPING

RELATED AIPSA DOMAINS	APPLIED NEXT STEP	WORKBENCH INSTRUMENTS	RELATED SERVICES
RAG Security	RAG security	RAG Test Harness, Runtime Proxy	AI Product Security Assessment

CERTIFICATION AND ASSESSMENT BOUNDARY

This chapter supports training, diagnostic preparation, scorecards, interviews, and role-readiness evaluation. It does not guarantee credential outcomes.

Retrieval-augmented generation changes the data access model in ways that most security programs have not caught up with. The retrieval layer is not search. It is a data access path that assembles context for the model, and it must be authorized with the same rigor applied to any other path that returns sensitive data. The failure mode that teams discover in production is that they built authorization for the answer — checking what the model is allowed to say — while leaving the retrieval layer functionally permissionless. By the time the model generates its answer, the authorization failure has already occurred.

The failure mode that teams discover in production is that they built authorization for the answer — checking what the model is allowed to say — while leaving the retrieval layer functionally permissionless.

HANDBOOK

The authorization boundary must be enforced before results enter the model's context. A filter that runs after similarity ranking has already violated that boundary — the model processes whatever the index returns, regardless of what the output shows. The mandatory filter gate is not an optimization; it is the control.

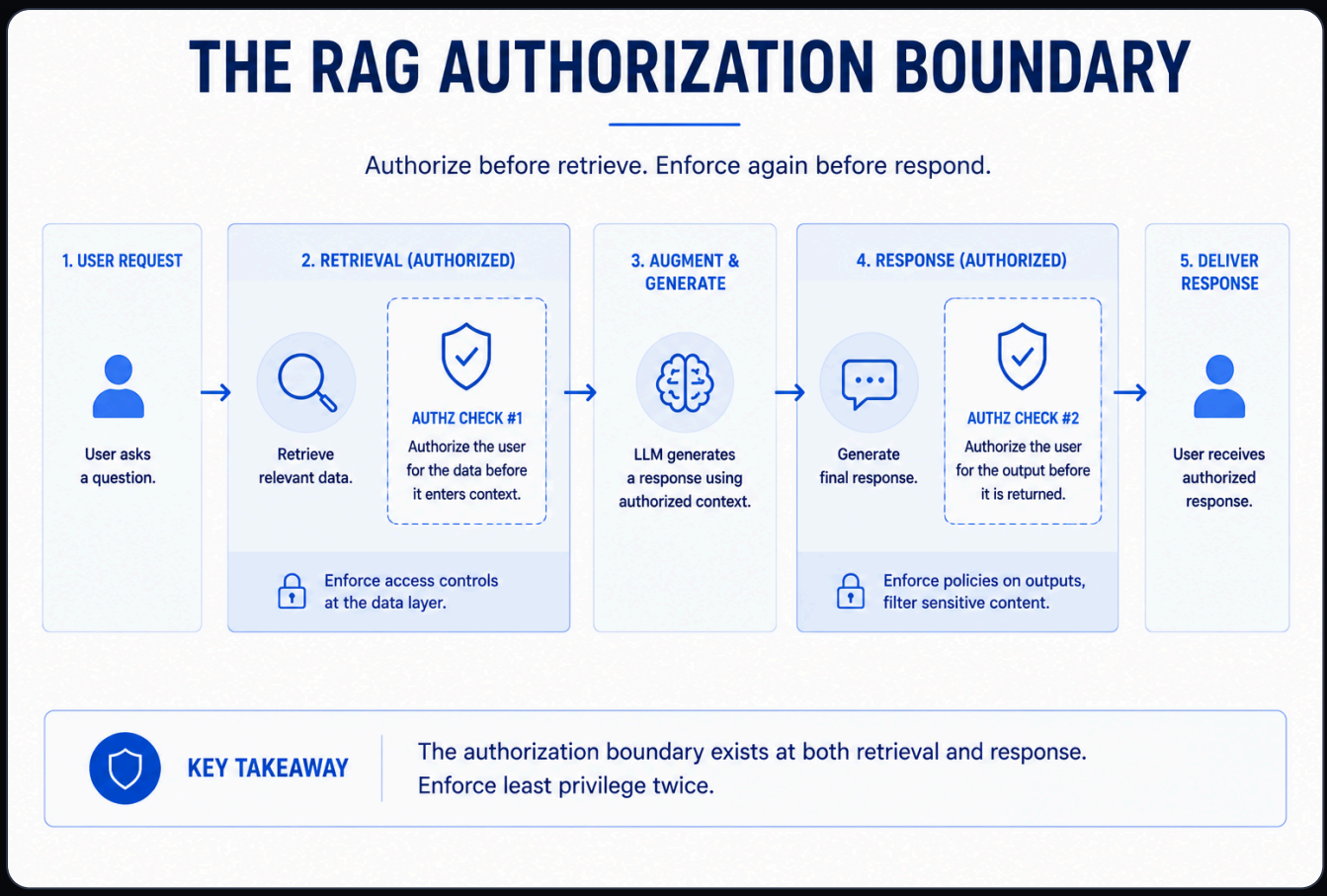


FIGURE 1: FIGURE 9: RAG AUTHORIZATION BOUNDARY — USER QUERY ENTERS A MANDATORY FILTER GATE BEFORE REACHING THE VECTOR DATABASE, WITH UNAUTHORIZED CONTENT BLOCKED AT THE RETRIEVAL STAGE AND ONLY AUTHORIZED RESULTS FLOWING TO THE PROMPT BUILDER

CORE CONCEPTS

RETRIEVAL-TIME AUTHORIZATION

Authorization must be applied before retrieval results enter the model's context window. Post-generation output filtering cannot repair a retrieval authorization failure because the model has already processed the unauthorized content. The retrieval layer must apply user identity, tenant, role, document permissions, classification label, and retrieval purpose as hard filters that execute before similarity ranking. The ranked results must be drawn from the authorized set, not from the full corpus with filtering applied afterward.

CHUNK METADATA AS AUTHORIZATION STATE

Retrieval authorization depends on metadata that must survive from the source document through every stage of the ingestion pipeline. Each chunk in the index must carry source identifier, document owner, tenant identifier, classification label, access policy or ACL reference, ingestion timestamp, version, and deletion status. If this metadata is incomplete or missing — which commonly happens when ingestion is treated as an ML preprocessing task — the retrieval layer cannot make accurate authorization decisions.

VECTOR STORE TENANCY MODELS

Vector stores can be designed with different tenancy models: shared index with metadata filters, tenant-namespaced indexes, or physically separated indexes per tenant or classification zone. Each model has different isolation guarantees and different failure modes. A shared index with metadata filters is the most common and most failure-prone: filters must be mandatory, correctly populated, and enforced before ranking. A namespace-separated or physically separated design provides stronger isolation at higher operational cost. The tenancy decision should match the confidentiality requirements of the data being stored.

INGESTION PIPELINE SECURITY

The ingestion pipeline is where retrieval authorization is either set up to work or set up to fail. The pipeline must: preserve source document permissions and classification labels through chunking and embedding, propagate deletion and permission changes from source systems to the index, apply ingestion review controls for sources that allow user-generated content, and verify that metadata fields required for authorization are populated before adding a chunk to the index. Ingestion is a security-sensitive data pipeline, not an ML preprocessing job.

CITATION INTEGRITY AS FORENSIC EVIDENCE

Source attribution — the record of which documents contributed to a generated answer — is an incident response requirement before it is a usability feature. When a retrieval authorization failure occurs, citation records determine scope: which users received which documents in context during which time window. When a generated answer contains incorrect information, citations determine whether the failure is a generation problem, a retrieval accuracy problem, or a retrieval

authorization problem. Citation records should be stored with retrieval traces, not just displayed in the UI.

THE PRACTITIONER'S CHALLENGE

The political challenge is that retrieval authorization is often discovered as a gap after the system is built and working. The demo works. Users can ask questions and get answers. The relevant documents appear. Adding retrieval-time authorization at that stage requires changes to the retrieval query path, metadata schema, and possibly the ingestion pipeline – all of which feel like regressions to a team that is satisfied with the current relevance. The practitioner must reframe the conversation: the system does not work correctly if it finds the right answer for the wrong user.

The structural challenge is ownership fragmentation. Search or AI engineering may own embedding quality and retrieval ranking. Data platform may own source system permissions. Identity engineering may own ACL structures. Security may own the threat model. Product may own the user experience. A RAG authorization failure can emerge in the gap between any two of these teams. The security review must explicitly define which team owns retrieval-time authorization enforcement and what the interface is between source system permissions and chunk-level metadata.

The technical challenge is that relevance and authorization pull in different directions. Retrieval optimization wants broad semantic recall; authorization wants narrow filtering that may exclude high-relevance but unauthorized documents. Chunking strategies that improve answer quality can fragment the metadata that authorization requires. The practitioner must specify authorization requirements as constraints on the retrieval design, not as additions to it.

HOW TO APPROACH IT

- Start with the source systems, not the vector store. Identify every corpus feeding the RAG system: documents, wikis, tickets, email, customer records, code repositories, policy documents, uploaded files, and vendor content. For each source, record the owner, classification, tenant model, permission model, deletion behavior, and update cadence. These properties must be preserved through ingestion; if you do not understand them at the source, you cannot verify they were preserved in the index.

- Map the ingestion pipeline and identify where metadata is populated, transformed, or lost. Trace a specific document from source through chunking, embedding, and index entry. Verify that every metadata field required for authorization is present in the index record. Verify that deletions and permission changes in the source system propagate to chunk records with bounded latency. If metadata gaps exist, fix them in the ingestion pipeline before building retrieval authorization logic on top of missing data.

- Design the retrieval query as an authorization workflow. The query must carry user identity, tenant identifier, role, classification floor, purpose, and request context into the retrieval layer. The retrieval implementation must apply these as mandatory filters before ranking, not as hints that can be omitted when metadata is incomplete. Define explicit failure behavior: if required authorization metadata is missing for a chunk, the chunk is excluded from retrieval results regardless of its relevance score.

- Test retrieval authorization independently of output filtering. Retrieval authorization tests verify that unauthorized chunks do not enter context; they do not verify what the model says. The test procedure: authenticated as a low-privilege user, submit queries that would retrieve high-privilege documents if authorization were absent. Verify that the retrieval layer returns no high-privilege chunks. This test is completely separate from testing whether the model refuses to display sensitive information.

- Build deletion propagation tests as part of the security testing suite. The test procedure: ingest a document, verify it is retrievable, trigger deletion in the source system, wait for propagation, verify that the document no longer appears in retrieval results. Record propagation latency. If propagation latency is too long for the system's risk tier, implement immediate index invalidation for deletions rather than waiting for the next ingestion cycle.

OUTPUTS AND DELIVERABLES

- The design artifacts are the **RAG authorization data-flow map**, **chunk metadata schema**, and **vector store tenancy decision record**. The data-flow map shows how source permissions travel through ingestion into the index and how they are applied during retrieval. The metadata schema defines the required fields for each chunk, with descriptions of how each field is populated, validated, and used in retrieval filters. The tenancy decision record documents the tenancy model chosen, the isolation guarantees it provides, and the failure modes that must be monitored.

- The enforcement artifacts are the **retrieval authorization policy**, **ingestion security checklist**, and **deletion propagation specification**. The authorization policy defines which filters must execute before ranking, what happens when required metadata is missing, and who can modify filter behavior. The ingestion checklist verifies metadata population, permission propagation, and deletion handling for each new source system added to the corpus. The deletion propagation specification defines maximum acceptable latency and the immediate invalidation procedure for high-sensitivity deletions.

- The testing and evidence artifacts are the **retrieval authorization test suite**, **cross-tenant retrieval test report**, and **citation integrity validation record**. The authorization test suite covers unauthorized chunk retrieval, cross-tenant access attempts, stale permission state, and deletion propagation timing. The cross-tenant test report documents the tenant isolation model and test results. The citation validation record documents how generated answers are traced to source chunks and how citation accuracy is measured.

COMMON FAILURE MODES

- **Output-Layer Authorization:** The team tests whether the model refuses to display sensitive information rather than testing whether unauthorized chunks enter context. The authorization failure occurs silently while the output test passes. Fix: build retrieval authorization tests that verify chunk retrieval results independently of model output.

- **Metadata Stripping in Ingestion:** The ingestion pipeline drops permission labels or ACL references during chunking because they were not part of the original ML preprocessing design. The retrieval layer is built on top of incomplete metadata and produces structurally incorrect authorization behavior. Fix: treat metadata preservation as a required engineering constraint for the ingestion pipeline from the beginning.

- **Shared Index Default:** The team uses a shared vector index for all tenants because it is the default configuration, without specifying mandatory metadata filters as hard enforcement. Tenant isolation depends on correctly populated filter values and consistent filter application. When either fails, cross-tenant retrieval occurs. Fix: specify tenancy model and isolation requirements before selecting the vector store configuration.

- **Deletion Propagation Gap:** Source records are deleted but corresponding chunks remain in the index. The propagation job runs on a batch schedule, and the gap is treated as an operational detail rather than a privacy or security risk. Fix: specify maximum acceptable deletion propagation latency as a security requirement; implement immediate invalidation for high-sensitivity deletions.

IMPLEMENTATION CHECKLIST

- [ ] Map every source system feeding the RAG corpus with its permissions, classification, and deletion behavior.
- [ ] Specify the chunk metadata schema with all fields required for retrieval authorization.
- [ ] Verify that metadata is preserved through every ingestion pipeline stage.
- [ ] Design retrieval queries to apply authorization filters as mandatory constraints before ranking.
- [ ] Define and implement fail-closed behavior when required authorization metadata is missing.
- [ ] Build retrieval authorization tests that verify chunk exclusion independently of model output.
- [ ] Build cross-tenant retrieval tests and run them before launch and after index configuration changes.
- [ ] Specify deletion propagation latency requirements and test propagation timing.

RELATED READING

- Handbook chapters: Chapter 2 (Architecture and Trust Boundaries) for data plane authorization design; Chapter 4 (Prompt Injection) for context authority tier enforcement; Chapter 7 (Data Exposure and Privacy) for deletion propagation and purpose limitation.
- Field Guide: RAG Security for retrieval authorization tests, chunk metadata review, tenant-boundary checks, and leakage evidence.