VERTEX FRONTIER

FIELD GUIDE · 2026 EDITION

The Enterprise RAG Security Field Guide

A practical playbook for building zero-trust retrieval pipelines before your next audit finds the gap.

Most companies treat Retrieval-Augmented Generation like a smarter search bar. That assumption is the single biggest security mistake in enterprise AI today. This guide compresses the architecture decisions, access-control mechanics, and common failure patterns into a portable reference you can hold in one hand while reviewing your own pipeline with the other.

PUBLISHED BY VERTEX FRONTIER
VERTEXFRONTIER.COM

FREE DOWNLOAD

01 · EXECUTIVE SUMMARY

RAG is a security problem, not just an accuracy problem.

A Retrieval-Augmented Generation system is not a search engine wearing a chat interface. It is a new **identity and authorization layer** sitting on top of every document your company owns. The moment you connect it to real company data, you inherit real company access rules — and if you do not design it that way from day one, the system will eventually hand someone a file they were never supposed to see.

The cost of getting this wrong is not theoretical. In 2023, Samsung's semiconductor division allowed engineers to use ChatGPT for day-to-day work. Within twenty days, three separate incidents occurred — proprietary source code, defect-identification logic, and an entire confidential meeting transcript were pasted into a tool with **no segmentation, no retention controls, and no audit trail**. Samsung banned generative AI company-wide within weeks, and competitors including Apple, JPMorgan Chase, and Verizon followed. The lesson is not "ban AI." It is that a properly architected internal RAG, with document-level ACLs and logging, is the alternative that lets a company capture the productivity benefit without the exposure.

> THE CORE PROMISE
>
> *If someone cannot open a document in SharePoint, they should not be able to extract its contents through your chatbot. That is not a minor edge case — it is the entire reason an enterprise RAG exists.*

THE GUARD FRAMEWORK

Five pillars to hold in your head while making architecture decisions.

Generic security checklists read like compliance forms nobody applies. GUARD is built specifically around how RAG systems fail in production. Each pillar maps directly to a section of an audit and to a layer of your pipeline — keep the acronym in mind and the difference between "we added some access control" and "we built a system that was secure by design" becomes structural, not aspirational.

G	U	A	R	D
GOVERNANCE	**USER IDENTITY**	**ACCESS INHERITANCE**	**REDACTION**	**DETECTION**
Every deployment maps to GDPR, SOC 2, or the EU AI Act before go-live — not after an audit finds the gap.	No query reaches the retriever without a verified identity via SSO/SAML/OAuth. Never an anonymous or shared service account.	Permissions are pulled from the source system at query time, not reinvented inside the RAG layer.	Data is encrypted at rest and in transit, and PII is masked before it ever reaches the LLM's context window.	Every query, retrieved source, and generated answer is logged — so an incident is a five-minute investigation, not a mystery.

02 · FILTERING ARCHITECTURE

Pre-filtering vs. post-filtering — and the tradeoff vendors won't name.

Once access control is mandatory, you still have to choose *where in the pipeline* it gets enforced. Pre-filtering narrows the vector database down to only the chunks a user is allowed to see, then searches within that restricted subset. Post-filtering runs the semantic search across the entire database first, then strips out anything the user isn't permitted to see. The choice has real consequences for both security and search quality, and most teams pick the wrong one for the wrong reason.

DIMENSION	PRE-FILTERING	POST-FILTERING
How it works	Filter by permission metadata first, then run semantic search on the narrowed set.	Run semantic search on everything, then strip unauthorized results.
Security boundary	Strong — unauthorized chunks never enter the candidate pool.	Weaker — unauthorized chunks are scored, then removed.
Search relevance	Can exclude best matches early if the allowed subset is small or fragmented.	Preserves maximum relevance — but may silently return fewer results.
Implementation effort	Harder. Not all vector DBs support it cleanly with HNSW indexes.	Simple. Bolts onto any existing vector database.
Latency profile	Fast on small allowed subsets; can stall on fragmented permission clusters.	Consistent, but adds an extra verification step after retrieval.

THE CONTRARIAN TRUTH

Post-filtering is the industry default not because it is more secure, but because it is easier to bolt onto an existing vector database. It quietly trades security rigor for implementation speed. If your retrieval pipeline filters permissions after the semantic search instead of before it, you are carrying a design compromise that most teams never actually audit — and it is worth asking your vendor which one they use, in those words.

DECISION MATRIX

Which method for which data class?

DATA SENSITIVITY	RECOMMENDED FILTER	BEST VECTOR DB PATTERN	RATIONALE
M&A, legal, HR, trade secrets	PRE-FILTER	pgvector + RLS, or Milvus partitions	Any leak is unacceptable; hard boundary required.
Financial reports, customer PII	PRE-FILTER	Pinecone namespaces, pgvector	Strict tenant separation; compliance-mandated.
Internal docs, policies, wikis	HYBRID	Milvus / Qdrant with partition keys	Pre for sensitive namespaces, post for the rest.
Public marketing, support KB	POST-FILTER	Any — relevance matters more than boundary.	Pair with strong logging so filtered results stay auditable.

03 · FAILURE PATTERNS

The seven deadliest RAG security mistakes.

Most RAG security failures trace back to a small set of repeated mistakes. They are not exotic — they are the predictable outcome of treating retrieval as a search problem instead of an authorization problem. Each one below is paired with the fix that turns it from a liability into a deliberate, monitored control.

1

Treating the knowledge base as a monolith

Indexing everything into one undifferentiated vector store with no metadata for classification or ownership. The retriever has no way to know which chunks are sensitive.

FIX Tag ACLs at ingestion time, not after the fact. Every chunk carries its source-system permissions from the moment it is embedded.

2

Choosing post-filtering by default without evaluating the tradeoff

It is the path of least implementation effort, not necessarily the right security posture for your most sensitive content. Teams inherit it without questioning the compromise.

FIX Use pre-filtering specifically for high-sensitivity data categories — legal, HR, M&A, trade secrets — even if the rest of the corpus stays on post-filtering.

3

Letting permission drift accumulate

Source-system access changes that never propagate to the vector index. An employee removed from a confidential project at 9:05 AM may still query its contents 15 hours later during the next nightly batch.

FIX Build permission sync as a first-class, event-driven pipeline (webhooks + Kafka + targeted metadata updates) — not a nightly cron job nobody watches.

4

Skipping the audit log until compliance asks for one

Retrofitting logging after go-live means losing months of unrecoverable history. The first regulator question becomes unanswerable.

FIX Audit logging ships with version one, not version two. Capture user, query, retrieved chunk IDs, generated answer, timestamp, IP, and model version for every interaction.

5

No environment separation

Testing new retrieval logic or prompt changes directly against production data. A regression in access-control logic silently affects real users in real time.

FIX Enforce dev / test / prod separation with genuinely different data access levels — not just different URLs pointing at the same index.

6

Assuming model quality solves security

A more capable LLM does not compensate for a retrieval pipeline that returns unauthorized documents — it just writes a more convincing answer from them. Better models raise the stakes, they do not lower them.

FIX Security lives in the retrieval and orchestration layer, not the model choice. Treat the LLM as untrusted by default; enforce boundaries upstream.

7

Unprotected semantic caches

Teams deploying Redis or GPTCache key cache hits on query embeddings alone. A junior analyst asking "show me Q3 M&A targets" can pull a cached executive-level answer, bypassing the entire retrieval pipeline.

FIX Every cache key must be a compound hash of the query vector and the requesting user's verified permission groups: hash(query_vector + user_acl_token).

04 · QUANTIFY YOUR EXPOSURE

The Data Exposure Risk Score.

One of the most useful early diagnostics before building or auditing a RAG system is a simple exposure metric: what share of your indexed documents are accessible to more people than

they should be? Security teams call this an **over-permissioning ratio**, and it is worth calculating before you index a single document. There is no universal safe threshold — a five-person startup and a 50,000-employee bank tolerate very different numbers — but tracking this ratio over time tells you whether your access-control discipline is improving or eroding as your document base grows.

```
Risk Score (%) = ( Over-permissioned documents ÷ Total indexed documents )
                                                × 100
```

Worked example. A 50,000-document corpus where 4,500 files are accessible to more users than their classification allows produces a risk score of **(4,500 ÷ 50,000) × 100 = 9.0%** — moderate exposure. A targeted review of your highest-sensitivity folders should be your next move before wider rollout.

0–5%	5–15%	> 15%
LOW	MODERATE	HIGH
Maintain your current review cadence.	Targeted review of highest-sensitivity folders.	Run a permission audit before any wider rollout.

THE SECURITY STACK

Battle-tested tools for each layer.

You do not need to build every control from scratch. Production deployments typically combine open-source and commercial frameworks to secure each layer of the pipeline. The list below is what working architects actually reach for — not a wishlist.

INPUT / OUTPUT GUARDRAILS

NVIDIA NeMo Guardrails · Lakera Guard

Detects prompt injections, topic drift, and toxic outputs in real time at the orchestration boundary.

PII MASKING

Microsoft Presidio · AWS Comprehend

Automated entity detection and token replacement before embedding creation. Re-hydration happens locally after generation.

ACCESS-AWARE VECTOR DB

pgvector + RLS · Pinecone · Milvus · Qdrant

Permission metadata bound to each chunk. Choice depends on scale, tenant model, and compliance regime.

AUDIT & OBSERVABILITY

LangSmith · Arize Phoenix · OpenTelemetry

Feeds query, retrieval, and generation traces into your SIEM (Splunk, Datadog) for compliance and incident response.

05 · PRE-LAUNCH REVIEW

The 30-point pre-launch security checklist.

G · GOVERNANCE

- [ ] Compliance framework (GDPR / SOC 2 / EU AI Act) explicitly mapped before go-live.
- [ ] Data classification policy documented and enforced at ingestion.
- [ ] Retention and deletion policies defined per data category.
- [ ] Cross-functional review (legal, security, engineering) sign-off recorded.
- [ ] Vendor data-processing agreements and ZDR terms in place.
- [ ] Incident response runbook drafted and tested with a tabletop exercise.

U · USER IDENTITY

- [ ] SSO via SAML 2.0 / OAuth 2.0 enforced — no anonymous access paths.
- [ ] No shared service accounts for user-facing queries.
- [ ] JWT / OAuth token validated at the Authorization Proxy Layer on every request.
- [ ] Group memberships resolved in real time from the enterprise directory.
- [ ] Session timeout and token refresh policies enforced.
- [ ] Rate limits per user per time window configured (e.g., 50 queries/hour).

A · ACCESS INHERITANCE

- [ ] ACL metadata captured at indexing time, not retroactively.
- [ ] Source-system permissions treated as single source of truth — no parallel copy.
- [ ] Event-driven ACL propagation (webhooks + Kafka) replaces nightly batch sync.
- [ ] Pre-filtering enforced for legal, HR, M&A, and trade-secret categories.
- [ ] Hybrid search applies filters upstream of the reranker, not after.
- [ ] Permission revocations reach the vector index within minutes, not hours.

R · REDACTION & ENCRYPTION

- [ ] AES-256 encryption at rest for vectors, source fragments, and audit logs.
- [ ] TLS 1.3 enforced on every hop: client → app → vector DB → LLM API.
- [ ] Keys managed in a dedicated KMS (Azure Key Vault / AWS KMS) with rotation.
- [ ] PII redaction pipeline (Presidio / Comprehend) runs at ingestion.
- [ ] Sensitive fields tokenized before reaching the LLM context window.
- [ ] Re-hydration of tokens happens locally, never inside the LLM provider.

D · DETECTION

- [ ] Every query logged with user, query text, retrieved chunk IDs, and answer.
- [ ] Timestamp, IP address, and model version captured per interaction.
- [ ] Semantic cache keyed on hash(query_vector + user_acl_token) — never query alone.
- [ ] Volume-based alerts fire when one user pulls unusual unique-chunk counts.
- [ ] Enumeration patterns ("all Q1", "all Q2"...) trigger human review before continuation.
- [ ] Traces exported to SIEM (Splunk / Datadog) via OpenTelemetry or LangSmith.

ENV · SEPARATION

- [ ] Dev environment uses anonymized or synthetic data only — no prod access.
- [ ] Staging runs representative but non-sensitive data subsets.
- [ ] Access-control regression tests run on every retrieval-logic change.
- [ ] Production enforces full security policy + continuous monitoring.
- [ ] Hash-based integrity verification blocks poisoned documents at ingestion.
- [ ] Source attribution surfaces which chunks informed every generated answer.