International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

Governed Retrieval Augmented Generation with Guardrails Role Based Access Control and Audit Logging

Lohith Reddy Kalluru

Cloud Developer III, Hewlett Packard Enterprise, Plainfield, Illinois, USA
Email: *kallurulohithreddy[at]gmail.com*

Abstract: ***Enterprise retrieval-augmented generation (RAG) must preserve source permissions while controlling unsafe content and retaining audit evidence. This paper integrates permission-tagged ingestion, authorization-aware retrieval, guardrails, and correlated logging, and evaluates the authorization boundary in a reproducible synthetic diagnostic. Five retrieval strategies were compared across eight corpus randomizations, each containing 750 documents, 12 principals, and 360 queries under five policy conditions. SQLite FTS5 supplied exact lexical ranking; no language model or Amazon Bedrock deployment was evaluated. Indexed filtering alone admitted unauthorized context in an average 99.5% of role-revocation queries and 57.7% of document-revocation queries. Adding a current-permission recheck produced zero observed context violations in the tested conditions. With 10% of private index metadata removed, authorized recall was 92.1%; policy-service outages forced all protected methods to abstain. An independent recomputation verified the saved retrieval and policy decisions. These results illustrate a known stale-permission failure mode and its mitigation under a trusted policy model. They do not establish production security, generated-answer safety, or regulatory compliance.***

Keywords: retrieval-augmented generation, role-based access control, guardrails, audit logging, AI governance

1. Introduction

Retrieval-augmented generation (RAG) supplies a large language model (LLM) with passages retrieved from an external corpus, allowing an application to ground responses in organizational information [1], [2]. However, relevance and source attribution do not establish that a user is authorized to access the retrieved information. An enterprise assistant may summarize a restricted compensation document accurately while still violating the source system's permissions.

The security boundary therefore includes every component that handles retrieved content: the retriever, reranker, model context, cache, citation renderer, and logging pipeline. Unauthorized passage exposure to a model is a control failure even when the final answer does not visibly reproduce the passage. Conversely, a source citation or cached answer can disclose information even when the current retrieval step is correctly filtered.

RAG also remains vulnerable to unsupported generation [3], indirect prompt injection [4], and corpus poisoning [5]. Governance frameworks emphasize documenting and managing AI risks [6], [7]. These concerns motivate an integrated design in which authorization, safety checks, and decision evidence are explicit parts of the request lifecycle.

This paper proposes four coordinated control points: permission-preserving ingestion, authorization-aware retrieval, guardrails around generation, and correlated audit logging. Its contribution is a design synthesis and a reproducible authorization diagnostic that tests known enforcement strategies under controlled permission changes. It does not claim a new access-control model, a production deployment, or measured improvements in latency or compliance-review effort.

2. Literature Survey

2.1 Retrieval-augmented generation

Lewis et al. introduced RAG as a way to combine a parametric language model with a non-parametric retrieval index, improving performance on knowledge-intensive tasks and allowing knowledge to be updated without retraining [1]. Subsequent work has refined chunking, embedding, re-ranking, query rewriting, and multi-step retrieval; Gao et al. survey these developments [2]. Managed services can package ingestion and retrieval behind an API; Section 4.6 discusses one proposed integration.

2.2 Risks specific to RAG

Hallucination is one relevant risk. Ji et al. survey hallucination in natural language generation and distinguish intrinsic errors, which contradict the source, from extrinsic errors, which add unsupported content [3]. Retrieval supplies evidence but does not guarantee faithful generation [2]. Greshake et al. showed that instructions embedded in retrieved content can hijack an LLM application, a threat known as indirect prompt injection [4]. Zou et al. demonstrated that injecting a small number of crafted passages into a knowledge base can steer RAG answers toward attacker-chosen targets [5]. The OWASP Top 10 for LLM Applications lists prompt injection, sensitive information disclosure, and vector and embedding weaknesses among the leading risks for such systems [8].

2.3 Access control models

Role-based access control assigns permissions to roles and roles to users, simplifying administration in large organizations [9], [10]. Attribute-based access control generalizes this by evaluating policies over attributes of the subject, resource, action, and environment [11]. These models can also govern mediated access. For RAG, their policies must be enforced before passages are supplied to downstream components. The proposed service uses RBAC for role assignment and ABAC-style resource constraints; describing the entire policy as pure RBAC would omit those attributes.

2.4 Governance frameworks

The NIST AI Risk Management Framework organizes AI risk management into govern, map, measure, and manage functions [6], and its Generative AI Profile identifies information security, data privacy, and confabulation among the risks specific to generative systems [7]. The EU Artificial Intelligence Act includes record-keeping provisions for high-risk AI systems [12]. Applicability depends on the system and its intended use; an audit trail alone does not establish legal compliance. These frameworks describe what organizations should achieve but leave the engineering to implementers. The present paper maps these governance concerns to RAG control points without claiming that earlier work lacks access-control mechanisms.

2.5 Closest authorization aware RAG work

Chen et al. implemented permission checks between retrieval and model context assembly in a patient-profile proof of concept [13]. Chen and Taipalus subsequently combined metadata filtering with fine-grained role-based permissions [14]. Jeong and Lee studied permission-aware retrieval across independently governed resources through native IAM validation [15]. These studies establish that access-controlled RAG predates the present work; post-retrieval checking before context assembly must not be conflated with output-only redaction.

OpenFGA documents a local authorization index followed by a final check for revocations that have not yet reached that index [16]. The present diagnostic tests this established mechanism using preserved corpus, permission, and ranking records. Its contribution is reproducibility and explicit boundary measurement under a declared workload, rather than a new authorization defense.

3. Problem Definition

3.1 Requirements

G1, Entitlement fidelity. No chunk outside the current policy decision may enter downstream model context, reranking, citations, or a user-visible response. Source-level permission semantics, including explicit denies and tenant boundaries, must be preserved.

G2, Safe interaction. The application must evaluate inputs and outputs against an explicit safety policy and handle detected violations consistently; detection is not assumed to be complete.

G3, Privacy protection. Personal and regulated identifiers must be blocked or masked in responses according to policy.

G4, Groundedness. The application must assess support for generated claims, use calibrated rejection criteria, and return an insufficient-evidence response when authorized sources are inadequate.

G5, Accountability. Each request must retain a correlated record of identity, policy and source versions, retrieved context, safeguard outcomes, and final status, subject to privacy and retention rules.

3.2 Threat model and trust assumptions

The design considers authenticated users who request information beyond their permissions, contributors who place malicious instructions in permitted sources, and stale or incomplete metadata. It assumes that the identity provider, policy service, ingestion credentials, and enforcement code are protected. Compromise of these trusted components, model-training-data disclosure, and low-level side channels require additional controls outside this proposal.

Table 1: Threats and primary controls

ID	Threat	Primary control
R1	Unauthorized context	Mandatory policy filter and ACL recheck
R2	Stale permissions	Revocation handling and bounded cache age
R3	Sensitive output	Output checks and restricted release
R4	Unsupported claims	Grounding assessment and abstention
R5	Direct injection	Input screening and constrained orchestration
R6	Indirect injection	Source screening and tool isolation
R7	Corpus poisoning	Trusted ingestion and source review
R8	Log disclosure	Minimization and restricted access
R9	Missing evidence	Durable correlated audit events

Authorization is an enforcement decision, whereas guardrail outcomes are detector assessments. A safe-looking response cannot compensate for unauthorized context exposure. The design objective is to prevent that exposure under its stated assumptions and test the implementation for violations.

4. Methodology and Approach

The method combines architecture design with an executed local authorization diagnostic. Literature and service documentation inform the proposed integrated workflow in Figure 1. The experiment isolates retrieval and context admission; guardrail effectiveness, generation, audit durability, and the proposed Bedrock integration remain unevaluated.

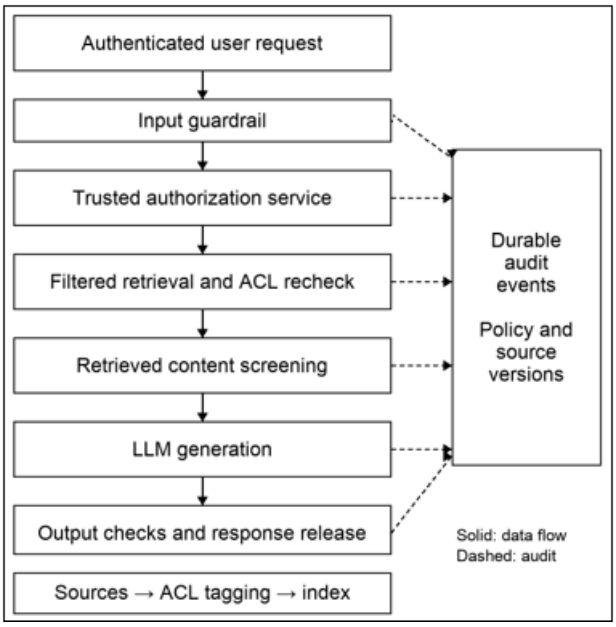


Figure 1: Proposed governed RAG workflow. Ingestion supplies the permission-tagged index used by filtered retrieval. Dashed paths represent audit events.

4.1 Permission Preserving Ingestion

The ingestion pipeline reads source access-control lists and maps them to normalized metadata such as tenant, permitted groups, classification, source version, and policy version. Every chunk inherits the applicable permissions. Documents with unknown permissions are quarantined. Mixed-permission documents require passage-level controls or exclusion; copying a document-level label is insufficient when sections have different audiences.

Permission synchronization uses source change events and periodic reconciliation. Metadata updates must be propagated through the supported connector and index update process; avoiding re-embedding cannot be assumed for every backend. For revocations that must take effect immediately, an authoritative check or quarantine of affected content is required while the index catches up.

4.2 Identity and Policy Evaluation

The application authenticates users through the enterprise identity provider, for example using OpenID Connect, and validates signed token issuer, audience, expiry, and signature. OAuth 2.0 supplies delegated authorization rather than authentication by itself. A trusted service resolves current groups and roles and evaluates access for the requested tenant and resource class.

The policy service returns a constrained retrieval predicate and a policy version. The application never accepts a client-supplied authorization filter or allows a model-generated query filter to widen that predicate. Missing identity, unresolved groups, policy errors, and unsupported filter expressions produce denial rather than an unfiltered retry. Explicit denies and source-specific rules take precedence over simplified group membership.

4.3 Authorization Aware Retrieval

The application attaches the mandatory predicate to every retrieval call. Retrieval is restricted to eligible passages using supported backend filtering semantics [17]. A trusted ACL check on returned chunks precedes reranking and context assembly, providing defense against stale metadata or translation errors. If no authorized evidence remains, the assistant returns an insufficient-evidence response without revealing whether restricted documents exist.

Logical eligibility filtering does not establish how a vector database internally orders approximate search and filtering. Ranking quality and filter behavior must be tested for the selected backend. The model receives only checked chunks; service credentials do not confer the service's broader permissions on the end user. Cached passages, response caches, conversation history, and citation links must be scoped to the principal or equivalent entitlement set and invalidated when policy changes.

4.4 Guardrails and response release

Input screening runs before retrieval. Retrieved passages are treated as untrusted data, separated from system instructions, and screened separately where required. Source content cannot grant tool privileges or alter the authorization policy. Output assessment checks sensitive information, scope, and support in the authorized reference context [18]. Detectors can miss violations and must be evaluated against benign and adversarial examples.

Table 2: Guardrail placement and purpose

Control	Placement	Purpose
Content and prompt-attack filters	Input; content checks on output	Detect unsafe content and attacks
Denied topics and word filters	Input and output	Enforce approved scope
Sensitive-information filters	Output; input if required	Block or mask configured entities
Source screening	Retrieved passages	Assess indirect injection risk
Grounding and relevance checks	Generated output with sources	Assess support and responsiveness

A grounding score measures agreement with supplied sources, not the truth or authority of those sources [19]. Poisoned but well-matched evidence may still produce a high score. Source provenance and corpus review therefore remain necessary. The application buffers output until the required checks pass, or uses a validated streaming policy that prevents unchecked text from reaching the user; later blocking cannot retract text already displayed.

4.5 Audit design

Each request creates correlated, append-only events for receipt, authorization, retrieval, guardrail decisions, and delivery. Retention controls, separation of duties, integrity verification, and durable ingestion protect these records. Storage is not intrinsically immutable merely because the application only appends. Audit guarantees require enforced retention or equivalent tamper-resistant controls.

Table 3: Minimum audit evidence

Field group	Recorded information
Identity and time	Request ID, timestamp, pseudonymous principal, tenant
Authorization	Policy version, predicate, entitlement snapshot
Context provenance	Document/chunk IDs, content versions, integrity references
Safeguards	Guardrail version, actions, reason codes and scores
Delivery	Delivered, masked, blocked, error or abstained
Operations	Model/version, configuration, latency and audit status

Raw prompts and responses are omitted by default. A keyed digest can help correlate queries without making a predictable query hash easy to enumerate, but it does not reconstruct the query. Exact reconstruction requires an approved protected payload store, source-version retention, and access-controlled links. These create additional privacy and retention obligations. Durable queuing and reconciliation handle asynchronous writes; audit receipt must be persisted before response release when complete evidence is a release condition.

4.6 Proposed Amazon Bedrock implementation

The proposed implementation would use Amazon S3 documents and associated metadata with Amazon Bedrock Knowledge Bases. The trusted application calls Retrieve with an explicit retrieval Configuration. Vector Search Configuration. filter, validates returned chunks, screens source content, invokes the model, and assesses output before release [17]. This decoupled path exposes the control points needed for authorization and source inspection.

For illustration, a simplified predicate requires membership in either grp-platform-eng or grp-all-employees AND a classification of public or internal. Bedrock operators such as listContains, in, and andAll can express these conditions where supported [17]. This example is not a complete policy: production rules also require tenant isolation, explicit-deny semantics, valid metadata types, and backend capability tests. Empty or oversized group sets must never cause the filter to be dropped.

ApplyGuardrail can evaluate input before retrieval and output independently of a model invocation [20]. Contextual grounding assessment must receive the authorized reference source, user query, and generated response in the required API structure [19]. The integration pins a guardrail version and defines handling for intervention, masking, service errors, and timeouts. It does not silently retry without the required checks.

A critical service boundary is that guardrails on RetrieveAndGenerate apply to the input and generated response, not automatically to retrieved references [21]. Attaching a guardrail to that operation therefore does not establish that source passages have been screened. The separate retrieval path makes this missing step explicit. Regional, model, filter, and backend support must be verified and recorded for the actual deployment.

4.7 Protocol for subsequent integrated evaluation

The integrated architecture requires validation beyond the authorization diagnostic reported here. A future deployment study should use independently labeled source permissions and held-out benign and adversarial requests. Keep model, embedding, reranker, corpus, prompts, sampling settings and resource budgets fixed for paired comparisons, and ablate input, source and output checks separately. Include stale caches, cross-tenant citations, mid-session revocation, indirect injection, poisoning, and guardrail or audit-service failure.

Table 4: Additional deployment evaluation measures

Dimension	Required evidence
Safety and privacy	Attack block rate, false blocks, prohibited identifier escape
Answer quality	Faithfulness, relevance and justified abstention
Accountability	Complete correlated audit events and integrity checks
Operations	Latency percentiles, cost and throughput under failures

Automated quality tools such as RAGAs [22] can supplement human review but cannot establish authorization correctness. Report metric definitions, denominators, uncertainty and evaluator settings. Audit completeness must include blocked and failed requests. Zero observed violations in a finite test set cannot prove zero production risk. These deployment outcomes were not measured by the local experiment below.

4.8 Executed local diagnostic

A local protocol was frozen and hashed before substantive runs. The primary hypothesis was that an indexed authorization filter followed by a current-source permission recheck would prevent unauthorized context admission in the declared workload, whereas stale indexed permissions alone would admit revoked content. Any unauthorized context admitted after the recheck would falsify the primary safety claim. A separate preflight seed was used to verify retrieval, metric calculation, and artifact writing and was excluded from all reported results.

The eight preregistered seeds were 4101 through 4108. Each generated 750 synthetic documents: 720 private documents across three tenants, 30 topics and eight variants, plus 30 globally public documents. Twelve principals represented four roles per tenant. Every principal issued one query per topic, yielding 360 queries per seed. Document text, lexical repetition, role sets, and perturbation selections varied reproducibly by seed. All topic-matching documents permitted by the current source policy were labeled relevant; this controlled labeling does not model real semantic relevance.

SQLite FTS5 ranked matching text using BM25 with document ID as a deterministic tie break and an output limit of eight contexts. The five methods were unfiltered context (U), current-ACL postfilter of the global top eight (P), indexed prefilter (I), indexed prefilter plus current-ACL recheck (IR), and a current-ACL prefilter oracle (O). The

query and scoring function were identical; permission predicates changed eligibility before or after the limit. P had no overfetch, so the comparison does not represent an optimized postfilter baseline. U did not execute or test a system prompt.

The five conditions were fresh permissions; role revocation with stale cached role membership; source-side revocation of 20% of private documents with stale indexed ACLs; removal of metadata from 10% of private documents; and an unavailable policy service. Revocations were applied before each condition's requests. Missing index metadata was denied by I and IR. All policy-enforcing methods, including O, returned no context during the simulated outage. There were 72,000 query-method-condition observations, nested within eight corpus seeds, rather than 72,000 independent trials.

A context violation was a request admitting any document disallowed by the current source policy. Authorized recall was the number of admitted authorized topic-matching documents divided by the smaller of eight and the number of authorized topic-matching corpus documents, averaged over requests and then seeds. This coverage metric does not penalize additional unauthorized contexts, so violation rate must be reported alongside it. Empty-context rate measured abstention at the retrieval boundary, not a generated refusal.

Results are means and minimum-to-maximum ranges across the eight corpus randomizations, not confidence intervals over enterprise traffic. The paired comparison used the same seed under I and IR. Source-policy evaluation was shared by the simulated recheck and its initial scorer; separate audit code independently reconstructed ranking and policy eligibility from saved corpora and ACL snapshots and reconciled all 72,000 observations. The implementation used Python 3.12.14 and SQLite 3.53.1 on local CPU. No remote experimental service, language model, paid compute, or Bedrock API was used.

5. Results and Discussion

5.1 Observed context violations

Table 5 reports the primary outcome. I produced mean violation rates of 99.5% (range 99.2–99.7%) under role revocation and 57.7% (range 55.3–61.1%) under document revocation. IR, P, and O admitted no unauthorized contexts in any tested condition. The paired mean reductions from I to IR were therefore 99.5 and 57.7 percentage points. These values characterize the declared synthetic perturbations, not the prevalence of enterprise leakage.

Table 5. Mean context violation rate in percent

Condition	U	P	I	IR	O
Fresh	100	0	0	0	0
Role revoked	100	0	99.5	0	0
Documents revoked	100	0	57.7	0	0
Metadata missing	100	0	0	0	0
Policy outage	100	0	0	0	0

U admitted unauthorized context on every query in this workload. Each topic was deliberately represented across tenants and permission groups, and high-ranking private variants competed with public guidance. The 100% result reflects this adversarial diagnostic construction. It is not a measured user-visible disclosure rate: no generator or response text was evaluated.

5.2 Authorized retrieval and abstention

Table 6 reports authorized recall alongside the violation results. Under fresh permissions, P attained 26.0% recall (range 24.5–26.8%), while I and IR attained 100.0%. Global top-k truncation removed candidates before P could recover authorized evidence. Because overfetch was excluded, this result supports a fixed-candidate-budget effect rather than general superiority over tuned postfiltering.

Table 6: Mean authorized recall in percent

Condition	U	P	I	IR	O
Fresh	26	26	100	100	100
Role revoked	0	0	99.9	99.9	100
Documents revoked	23.8	23.8	100	100	100
Metadata missing	26	26	92.1	92.1	100
Policy outage	26	0	0	0	0

With missing metadata, IR recall decreased to 92.1% (range 91.5–92.5%), whereas the source-policy oracle retained 100.0%. The recheck cannot recover documents excluded earlier because their index metadata is unknown. After document revocation, IR recall was 99.97%, slightly below the oracle because no refill followed removal of revoked candidates. Rounded one-decimal cells can conceal this small gap; unrounded values remain in the evidence package. All four protected methods returned empty context on 100.0% of policy-outage requests. This is the declared fail-closed behavior, not evidence of service reliability. P also returned empty context for all role-revocation requests in this corpus, whereas IR retained almost all available public evidence. High authorized recall under I did not imply safety: the metric counts authorized coverage but does not subtract unauthorized passages.

5.3 Authorization placement

Prompt-only restrictions leave restricted content in model context and provide no dependable authorization boundary. Post-retrieval filtering can protect the model when applied before every downstream consumer, but the retrieval service still handles unauthorized candidates. With a fixed retrieval budget, removing those candidates may leave too little authorized evidence. Retrieval-time filtering narrows eligibility earlier; its security depends on correct metadata and policy translation, while its recall depends on backend search behavior.

A returned-chunk authorization recheck remains useful even when retrieval filters are present. It addresses stale index labels and implementation mistakes without treating post-filtering as a replacement for the mandatory predicate. None of these mechanisms alone prevents disclosure through poorly scoped caches or overprivileged citation links.

5.4 Blocking masking and grounding

Blocking is appropriate when the request or answer violates scope. Masking can preserve useful surrounding text when an identifier is incidental, but removing identifiers does not necessarily prevent re-identification through context. Both actions depend on detector coverage. Grounding thresholds trade unsupported-answer acceptance against rejection of valid paraphrases; thresholds require held-out calibration and domain-specific error analysis.

5.5 Relation to prior work

The proposal complements retrieval-centered designs [1], [2] by making permission decisions and audit evidence explicit. Indirect-injection research [4] motivates isolating source content from instructions, while poisoning attacks [5] show why grounded output can still be untrustworthy. Established RBAC and ABAC models [9], [10], [11] supply policy concepts; the paper's contribution lies in coordinating their enforcement with generation and response release.

5.6 Operational implications and limitations

Authorization calls, content checks, source validation, and durable audit receipts add processing and storage costs. Short-lived entitlement caching can reduce lookup overhead but increases the revocation window. Query embedding may run concurrently with approved preliminary checks, provided retrieval and content release remain gated. No claim of modest overhead is justified until latency, cost, and failure behavior are measured.

Central policy and guardrail configuration may reduce duplicated implementation work, but that benefit remains a hypothesis. A deployment study should compare engineering effort, review duration, support outcomes, and incidents against documented baselines. The architecture cannot guarantee compliance, factual truth, or immunity to prompt injection. Its most consequential limitations are permission fidelity, synchronization delays, probabilistic detectors, trusted-service compromise, and the operational burden of retaining enough evidence without creating a sensitive secondary corpus.

The local results are conditional on a small synthetic corpus, exact lexical ranking, topic-based relevance labels, and a trusted source-policy function. They do not test approximate vector search, within-request revocation races, real caches, service consistency, source compromise, model answer disclosure, guardrail detectors, or audit persistence. No runtime latency or cost comparison is claimed, and per-resource check counts are bookkeeping rather than a fair proxy for performance. Local preregistration hashes establish consistency with retained artifacts, not an externally timestamped registry.

6. Conclusion

Enterprise RAG governance requires permission-preserving context assembly as well as controls on generated text. This paper proposes an architecture that combines access-control tagging, trusted authorization-aware retrieval, input and output safeguards, separate source screening, and correlated audit evidence. Its central requirement is that every path by which source information reaches a downstream component or user must preserve the relevant entitlement decision.

The proposed Amazon Bedrock integration illustrates how managed retrieval and guardrail services can be coordinated with a custom policy layer, while recognizing their service boundaries. The local diagnostic observed that stale indexed permissions can admit revoked content, while a current-source recheck removed those violations under the stipulated policy model. The utility results also expose fixed-top-k losses, metadata-related exclusions, and outage abstention. These findings validate known engineering failure modes in a reproducible fixture; production effectiveness and compliance require separate deployment evidence.

7. Future Scope

The next step is an independently implemented deployment study with real source-system permission labels, approximate vector retrieval, overfetch sensitivity, and concurrent revocation schedules, building on the broader protocol in Section 4.7. Further work should measure revocation propagation, validate passage-level permissions in mixed-sensitivity documents, evaluate source-integrity controls against poisoning, and compare filtered approximate retrieval across backends. Multi-turn tests should examine whether earlier authorized context remains accessible after a role change. Audit research should determine the minimum retained evidence needed for incident reconstruction under defined privacy and retention constraints.

Declarations

The views expressed are the author's own and do not represent the position of Hewlett Packard Enterprise. The author declares no competing interests. The integrated architecture is proposed; the authorization diagnostic was executed locally on synthetic data. The accompanying evidence package contains the frozen protocol, generated corpora and queries, source code, raw context-admission records, run manifests, claim ledger, and independent audit. No human-subject or private enterprise data were used. OpenAI Codex assisted with research, implementation, and manuscript preparation; author review and responsibility remain with the named author.

References

[1] Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems. 2020; 33: 9459–9474.
[2] Gao Y, Xiong Y, Gao X, et al. Retrieval-augmented generation for large language models: a survey. arXiv:2312.10997. 2023 [revised 2024]. Available from: https://arxiv.org/abs/2312.10997
[3] Ji Z, Lee N, Frieske R, et al. Survey of hallucination in natural language generation. ACM Computing Surveys. 2023; 55(12): Article 248, 38 p. doi:10.1145/3571730.

[4] Greshake K, Abdelnabi S, Mishra S, Endres C, Holz T, Fritz M. Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security. 2023. p. 79–90. doi:10.1145/3605764.3623985.

[5] Zou W, Geng R, Wang B, Jia J. PoisonedRAG: knowledge corruption attacks to retrieval-augmented generation of large language models. In: 34th USENIX Security Symposium. USENIX Association; 2025. p. 3827–3844.

[6] National Institute of Standards and Technology. Artificial intelligence risk management framework (AI RMF 1.0). NIST AI 100-1. 2023. doi:10.6028/NIST.AI.100-1.

[7] National Institute of Standards and Technology. Artificial intelligence risk management framework: generative artificial intelligence profile. NIST AI 600-1. 2024. doi:10.6028/NIST.AI.600-1.

[8] OWASP Foundation. OWASP Top 10 for Large Language Model Applications 2025 [Internet]. [cited 2026 Oct 31]. Available from: https://genai.owasp.org/llm-top-10/

[9] Ferraiolo DF, Kuhn DR. Role-based access controls. In: Proceedings of the 15th National Computer Security Conference. 1992. p. 554–563.

[10] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. Computer. 1996;29(2):38–47.

[11] Hu VC, Ferraiolo D, Kuhn R, et al. Guide to attribute based access control (ABAC) definition and considerations. NIST SP 800-162. 2014 [updated 2019]. doi:10.6028/NIST.SP.800-162.

[12] European Parliament, Council of the European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. Official Journal of the European Union. 2024 Jul 12. Available from: https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[13] Chen B, Tackman J, Setälä M, Poranen T, Zhang Z. Integrating access control with retrieval-augmented generation: a proof of concept for managing sensitive patient profiles. In: Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing. 2025. p. 915–917. doi:10.1145/3672608.3707848.

[14] Chen B, Taipalus T. Metadata-aware RAG for enforcing access control and metadata-based filtering: proof of concept and evaluation. In: RACS 25: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. 2026. Article 20, p. 1–7. doi:10.1145/3769002.3769952.

[15] Jeong J, Lee SG. Permission-aware RAG: identity and access management (IAM)-based access filtering in multi-resource environments. IEEE Access. 2025;13:192819–192835. doi:10.1109/ACCESS.2025.3628960.

[16] OpenFGA. Search with permissions [Internet]. [cited 2026 Oct 31]. Available from: https://openfga.dev/docs/interacting/search-with-permissions

[17] Amazon Web Services. Configure and customize queries and response generation [Internet]. Amazon Bedrock User Guide. [cited 2026 Oct 31]. Available from: https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html

[18] Amazon Web Services. Detect and filter harmful content by using Amazon Bedrock Guardrails [Internet]. [cited 2026 Oct 31]. Available from: https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails.html

[19] Amazon Web Services. Use contextual grounding check to filter hallucinations in responses [Internet]. [cited 2026 Oct 31]. Available from: https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-contextual-grounding-check.html

[20] Amazon Web Services. Use the ApplyGuardrail API in your application [Internet]. [cited 2026 Oct 31]. Available from: https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-use-independent-api.html

[21] Amazon Web Services. Query a knowledge base and generate responses based off the retrieved data [Internet]. [cited 2026 Oct 31]. Available from: https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-retrieve-generate.html

[22] Es S, James J, Espinosa Anke L, Schockaert S. RAGAs: automated evaluation of retrieval augmented generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics; 2024. p. 150–158. doi:10.18653/v1/2024.eacl-demo.16.