Rillor

EXPLAINER

Permission-aware retrieval for internal assistants

RAG access control for internal assistants: how retrieval returns only what the asking user may read, from per-document filters to delegated tokens, and what published leaks show.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Permission-aware retrieval keeps an internal assistant inside the asking user's access: the index stores each source's permissions, every query filters on a verified identity, and retrieval runs with that user's delegated rights [1, 2, 3].

- An assistant built on retrieval-augmented generation (RAG) answers from passages it retrieves from an index. Three cloud vendors' documentation for their enterprise assistant and search services sets the same rule: answers draw only on content the asking user may already access [4, 5, 6].
- An assistant that inherits each user's access also inherits sharing mistakes, so a file shared too widely can surface for everyone it is shared with. One vendor's governance reports find sites open to thousands of users, to external guests or to everyone except external users [4, 7].
- In two disclosed cases, a 2025 enterprise copilot flaw and a 2024 workplace chat flaw, retrieval inside the victim's own permissions carried an attacker's instructions into the answer [8, 9]. Both vendors deployed fixes [8, 10].
- The common control stores each document's access list in the index and filters every query on an identity the server has verified. An ID passed as plain filter text proves nothing, and a search engine's per-user restrictions can still let users count how many documents they cannot open contain a term [11, 12].
- A delegated token carries one user's rights and names the assistant as the acting party, as the actor claim of RFC 8693 does. OWASP's Top 10 for Agentic Applications warns against agents that let other users act through their maker's identity [2, 13].
- Embeddings, the vectors an index stores for each passage, can give the text back. One inversion method recovered 92% of 32-token inputs exactly for one embedding model in its authors' tests, and a 2024 paper showed a deleted document still appearing in answers until the next indexing [14, 15].
- Permission changes reach the index only at the next sync, crawl or refresh, and one vendor's temporary discovery restriction, which changes no permissions, can take more than a week to process on sites with more than 500,000 items [3, 5, 16].
- A log of every passage retrieved, cited or not, is what shows an injected message, because in one disclosed case the answer's citation did not point to the attacker's message [1, 9].

Background

An assistant sees what its index holds

OWASP defines RAG as "a model adaptation technique that enhances the performance and contextual relevance of responses from LLM Applications, by combining pre-trained language models with external knowledge sources" [1]. In the standard design, the system retrieves passages once and the model answers from them, as our explainer on agentic RAG and standard RAG (https://rillor.com/insights/agentic-rag-vs-rag) sets out. For an internal assistant, the external knowledge is company content, and whatever the index holds can reach an answer unless the query filters it out.

Vendors that sell assistants over company content describe the same rule. One enterprise copilot service "only surfaces organizational data to which individual users have at least view permissions", and its semantic index "honors the user identity-based access boundary so that the grounding process only accesses content that the current user is authorized to access" [4]. A second cloud's enterprise search uses the customer's identity provider "to identify the end user performing a search and determine if they have access to the documents that are returned as results" [6]. A third vendor's connectors "index access control list (ACL) information that's attached to a document along with the document itself" and use it to "filter chat responses based on the end user's access to documents" [5]. The permissions themselves stay with the customer: the copilot vendor asks customers to use its permission models "to help ensure the right users or groups have the right access to the right content within your organization" [4].

Terms used here

NIST SP 800-162 describes attribute-based access control (ABAC) as a method in which authorization "is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships" [17]. In the assistant and search documentation reviewed here, a retrieval filter typically reads the users and groups on each item's access list, and one search service can also read sensitivity labels [3, 5]. OWASP asks teams to "Only grant access to data that is necessary for the specific user or process" [18].

RAG access control, in this report, is the set of checks that decides which passages a retrieval step may return for a given person. Tool calls, their scopes and their audit logs are covered in our explainer on MCP access control (https://rillor.com/insights/mcp-access-control). This report is for security architects and IT leads rolling out an assistant over company documents.

Method

What was examined. How leaks through retrieval have happened, where the access check runs, how an assistant acts for a user, whether embeddings carry their source's sensitivity, how permissions are fixed before and after indexing, and what a retrieval log records.

Sources. OWASP's risk lists [1, 13, 18, 19], NIST's ABAC guide and the IETF token exchange standard [2, 17], vendor documentation [3, 4, 5, 6, 7, 11, 12, 16, 20, 21, 22, 23, 24, 25, 26, 27], a vulnerability record, two disclosures and one vendor statement [8, 9, 10, 28], and research papers [14, 15, 29, 30, 31, 32, 33, 34, 35].

How. Each claim was read in the source's own text, and quotes are copied from it. Commercial products are described by kind and named only in Sources. Figures from papers hold for the authors' setups, and vendor figures are the vendor's own examples. Statements that join two sources are marked as reasoning. All pages were viewed on 8 November 2026, and [6] again on 12 November 2026 after its update date changed.

Findings

1. How leaks happen: shared indexes, oversharing and injection

A shared index with loose partitions. OWASP's entry on vector and embedding weaknesses lists unauthorized access first: "Inadequate or misaligned access controls can lead to unauthorized access to embeddings containing sensitive information" [1]. Its second item covers shared stores: "In multi-tenant environments where multiple classes of users or applications share the same vector database, there's a risk of context leakage between users or queries" [1]. Its stated fix is that "A permission-aware vector database should be implemented to restrict access and ensure that only authorized groups can access their specific information" [1]. OWASP's agentic list adds cross-tenant vector bleed, in which "Near-duplicate content seeded by an attacker exploits loose namespace filters, pulling another tenant's sensitive chunk into retrieval by high cosine similarity", and advises per-tenant namespaces for shared vector or memory stores [13]. Memory that "is not segmented or cleared between tasks or users" lets attackers "leak data from a prior secure session into a weaker one" [13].

Oversharing. The copilot vendor's statement that its service shows only what a user can view [4] implies that a file shared too widely can surface in answers for everyone it is shared with (reasoning from [4]). The same vendor publishes governance reports that "let you discover sites that contain potentially overshared or sensitive content" [7]. A site permissions report finds "sites with the broadest user access, such as sites with thousands of users, external guests, or 'Everyone except external users' permissions", and activity reports cover the last 28 days [7]. In that vendor's connector API, "A third access control entry type Everyone represents all the users in the tenant", and "The accessType value deny takes precedence over grant" [22].

Instructions inside retrieved content. OWASP's prompt injection entry says that "Indirect prompt injections occur when an LLM accepts input from external sources, such as websites or files", gives a scenario in which an attacker modifies a document that a RAG application retrieves, and notes that RAG and fine-tuning "do not fully mitigate prompt injection vulnerabilities" [19]. Greshake and colleagues showed in 2023 that adversaries can exploit such applications "by strategically injecting prompts into data likely to be retrieved", because they "blur the line between data and instructions" [30].

Case	Attacker's access	What retrieval did	Response	Sources
Enterprise copilot service, CVE-2025-32711, published 11 June 2025	Sending one email	Pulled the email in through normal retrieval, and its hidden instructions put sensitive context into an outbound link	Server-side fix in May 2025, no customer action needed	[8, 28]
Workplace chat platform's assistant, disclosed 20 August 2024	An account in the same workspace and a public channel with only the attacker in it	Put the attacker's message and the victim's private message into one context	Patch deployed 20 August 2024	[9, 10]

Table 1. Two disclosed cases of injection through retrieval.

The National Vulnerability Database records CVE-2025-32711 as command injection in an enterprise copilot service that "allows an unauthorized attacker to disclose information over a network" [28]. The reporting organization, the vendor, scored it 9.3 Critical under CVSS 3.1, and NVD's own assessment scored it 7.5 High [28]. The EchoLeak paper describes "remote, unauthenticated data exfiltration via a single crafted email" [8]. The email was "pulled in via the normal retrieval process", and its hidden instructions made the assistant "embed the most sensitive details from the user's context into an outbound reference link" [8]. The chain evaded a cross-prompt injection classifier and link redaction, and relied on images fetched automatically [8]. The vendor deployed a server-side fix in May 2025 and "stated that no customer action was required and there was no evidence of in-the-wild exploitation" [8].

In the chat case, a research firm's write-up of 20 August 2024 describes an attacker who posts an instruction in a public channel of which only the attacker is a member [9]. When the victim asks the assistant for an API key the victim had put in a private channel, retrieval "pulls both their message and the attacker's message into the same 'context window'", and the assistant renders a link that carries the key [9]. The answer's citation did not refer to the attacker's channel [9]. The researchers quote the vendor's reply that public channel messages "can be searched for and viewed by all Members of the Workspace, regardless if they are joined to the channel or not" [9]. The vendor's statement of 21 August 2024 says it deployed a patch on 20 August 2024 and had "no evidence at this time of unauthorized access to customer data" [10].

In both cases the attacker's text was retrieved inside the victim's own permissions. The EchoLeak authors write: "Least privilege also limits blast radius: the AI can only exfiltrate what it can see" [8]. Permission-aware retrieval bounds what an injected instruction can reach, and it does not stop the injection. The paper proposes "Context segregation: Partition retrieval into trust tiers and force single-tier answers", and notes that "If emails from unknown external senders are never used to answer questions about internal projects, EchoLeak fails by design" [8].

Writes and extraction. Who may write to the index is an access question too. PoisonedRAG showed that "an attacker could inject a few malicious texts into the knowledge database of a RAG system to induce an LLM to generate an attacker-chosen target answer for an attacker-chosen target question" [32]. In the authors' experiments, five malicious texts per target question in a database of millions of texts reached a 90% attack success rate [32]. OWASP advises teams to "Accept data only from trusted and verified sources" [1]. On the read side, Qi and colleagues extracted datastore text verbatim through prompt injection [31]. Against custom assistants on one commercial platform, their attack caused datastore leakage in all 25 randomly selected assistants with at most 2 queries, and extracted 41% of a 77,000-word book and 3% of a 1,569,000-word corpus verbatim with 100 queries [31]. Read together, these results argue for treating anything placed in a retrieval store as readable by whoever can query it (reasoning from [31, 32]).

2. Access rules applied at query time

The common pattern. One cloud search service documents it step by step. A string field holds user or group identities, documents are loaded with their ACLs, and at query time the application must "get the identity of the caller" and pass it in as the filter, so that results "are trimmed to exclude any matches that fail to include the user or group identity string" [3]. The page lists four approaches, three of them marked preview [3].

A plain string authenticates nothing: "There's no authentication or authorization through the security principal. The principal is just a string, used in a filter expression, to include or exclude a document from the search results" [11]. A token-checked variant moves the check into the service, which "checks for user or group permissions on documents that are returned in search results, excluding any that don't match" [3]. In one cloud's worked example for its knowledge-base service, a backend function looks up stored doctor-patient associations before it queries, a step that "validates that the doctor is authorized to access the requested patient or list of patient's information" [25]. In each design the server establishes who is asking before results are filtered.

Document-level security in a search engine. One search engine's documentation says that "Document level security restricts the documents that users have read access to", with a role query templated per user, such as a term query on `acl.username` filled from `{{_user.username}}` [12]. Its undated page lists the caveats:

- If one role grants access to an index without document level security and another with it, "document level security is not applied" [12]. The page advises "splitting documents by index" where both documents and fields need restricting [12].
- A restricted user "could still learn about field names and terms that only exist in inaccessible documents, and count how many inaccessible documents contain a given term" [12].
- Relevance scores are "computed without taking the role query into account" [12].
- "Document level security doesn't apply to write APIs" [12].

Row security in a relational database. In one open-source relational database, row security means "all normal access to the table for selecting rows or modifying rows must be allowed by a row security policy", with "a default-deny policy" when none exists [23]. "Superusers and roles with the BYPASSRLS attribute always bypass the row security system", and table owners normally do too unless the table forces row security [23]. An assistant's backend that connects as the table owner therefore sees every row (reasoning from [23]).

Tenant partitions. One vector database's namespaces mean that "all upserts, searches, and other data read and write operations always target one namespace", and in its serverless architecture they provide "physical isolation of data between tenants" [24]. In the vendor's pricing example, with 100 tenants of 1 GB each, searching one tenant's namespace costs 1 read unit (RU), against 100 RUs for the same search with metadata filtering in one 100 GB namespace, "because it scans all data regardless of filters" [24]. A namespace separates tenants, and users inside one company still need per-document checks (reasoning from [24]).

Per-user indexes or one shared index. The HONEYBEE paper frames the trade-off: "dedicated per-user indexes minimize query latency but incur high memory redundancy, while shared indexes with post-search filtering reduce memory overhead at the cost of increased latency" [33]. In the authors' experiments, even per-role partitions had memory overheads 7 times larger than a unified index, and post-filtering lost recall and speed "when filters are highly selective" [33]. Their method reached "up to 13.5x lower query latency than row-level security with only a 1.24x increase in memory usage" [33]. A June 2026 vision paper finds fine-grained access control "not fully supported in modern vector databases" [35].

An authorization service. An open-source authorization project notes that "the user's access to each resource now has to be validated before the resource can be shown" [26]. It documents three options: search and then check each result, build a local index from a feed of changes and then search and check, or build a list of permitted IDs and then search, which suits a low count, about 1,000 objects, and becomes "impractical" as the count grows [26]. For retrieval, search-then-check means fetching more candidates than the prompt needs (reasoning from [26]).

Shared conversations. A September 2025 position paper argues that "any content used in training, retrieval, or generation by an LLM is explicitly authorized for all users involved in the interaction", and that defenses such as prompt sanitization and output filtering are "fundamentally probabilistic" [34]. On that position, an assistant answering in a shared channel or meeting is limited to what every participant may read.

Product limits. One cloud's enterprise search allows 3,000 readers per document, and access control must be chosen when the data store is created: "You can't turn this setting on or off for an existing data store" [6]. Another vendor states that "turning off ACLs and identity crawling once they have been enabled are no longer supported" [5].

Approach	What it checks	Caveat in the source	Sources
String security filter	The caller's user and group IDs, matched as strings	Authenticates nothing, so the server supplies the value	[11, 25]
Token-checked query	The caller's token against each result	Marked preview	[3]
Document-level security role	A role query per user	Roles combine permissively, and aggregates and scores see every document	[12]
Row security policy	A policy per row	Owners and superusers bypass it by default	[23]
Tenant namespace	Which partition is searched	Separates tenants, not users inside one (reasoning from [24])	[24]

Approach	What it checks	Caveat in the source	Sources
Local index, search, then check	An authorization service per result	The local index built from the change feed can lag revoked rights, so each result is checked	[26]

Table 2. Where the check can sit, drawn from the sources cited.

3. Acting for a user: delegated tokens and the actor claim

RFC 8693 separates two ways one party acts for another. Under impersonation, A "is indistinguishable from B in that context" [2]. Under delegation, "principal A still has its own identity separate from B", and "any actions taken are being taken by A representing B. In a sense, A is an agent for B" [2]. Token exchange can "allow a resource server A to make calls to a backend service C on behalf of the requesting user B" [2].

"The act (actor) claim provides a means within a JWT to express that delegation has occurred and identify the acting party to whom authority has been delegated" [2]. The RFC's example has a subject of `user@example.com` and an act claim naming `admin@example.com` [2]. Nested act claims record a chain, and "the consumer of a token MUST only consider the token's top-level claims and the party identified as the current actor by the act claim" [2].

One identity platform's on-behalf-of flow is a deployed form of the idea: "Referred to as delegation in OAuth, the intent is to pass a user's identity and permissions through the request chain" [27]. It "only uses delegated scopes and not application roles", and "Roles remain attached to the principal (the user) and never to the application operating on the user's behalf" [27].

The contrast matters for retrieval (reasoning from [2, 27]). An assistant that reads the index with its own broad service credential and filters afterwards holds every user's access at once. A delegated token carries one user's rights and the assistant's identity, so the retrieval layer checks the user and the log can name both.

OWASP's agentic list describes what goes wrong without this. Under "Un-scoped Privilege Inheritance", "A narrow worker then receives excessive rights" [13]. Under "Workflow Authorization Drift", "the user's spending limit is reduced, but the workflow proceeds with the old authorization token" [13]. Under "Identity Sharing", an agent "gains access to systems on behalf of a user, often their maker", then "allows other users to leverage that identity implicitly" [13]. Its mitigations include "Issue short-lived, narrowly scoped tokens per task" and "Re-verify each privileged step with a centralized policy engine" [13].

4. Embeddings are sensitive data too

OWASP warns that "Attackers can exploit vulnerabilities to invert embeddings and recover significant amounts of source information, compromising data confidentiality" [1]. Morris and colleagues showed at EMNLP 2023 that "a multi-step method that iteratively corrects and re-embeds text is able to recover 92% of

32-token text inputs exactly" for one open embedding model, recovered 61% exactly for a second, commercial model, and recovered full names from a dataset of clinical notes [15]. The 92% covers 32-token inputs and that one model [15].

OWASP asks for "permission-aware vector and embedding stores" and "strict logical and access partitioning of datasets in the vector database" [1]. A vector store and every copy or cache made from it therefore needs the permissions of the documents it came from (reasoning from [1, 15]).

5. Fixing permissions before indexing

The source system decides. The three assistant vendors reviewed enforce the permissions they find in the source [4, 5, 6], so fixes belong there. One vendor recommends deleting or re-creating a group of the same name with different members in the data source, instead of in the assistant's own user store [5].

Temporary discovery restrictions. One vendor's restricted content discovery hides chosen document sites from organization-wide search and assistant answers while their permissions are reviewed, and is "designed as a temporary governance control" [16]. It "doesn't change existing permissions", leaves content in the search index, and on sites with more than 500,000 items an update "could take more than a week to fully process" [16]. The same vendor's restricted search allow-list is "a short-term solution that gives your organization's administrators time to review and audit site and file permissions" [20]. It "isn't a security boundary and doesn't change any permissions", holds up to 100 sites, and still shows sites a user recently opened or that were shared with them in chat or mail [20].

The index lags the source. In one cloud search service, permission changes "are only reflected in search results after that metadata is synchronized to the index", and changes inherited on the vendor's document sites "require an explicit refresh" [3]. Another vendor's connectors update ACLs "each time that your data source content is crawled", and it advises customers to "re-sync your data source regularly" [5]. When an email address passes to a new employee, "you must delete the original user from the User Store", and a re-created group of the same name means "document ACLs which contain this group may be impacted" [5].

Caches and deleted documents. ConfusedPilot, a 2024 paper, found that an assistant which indexes and caches documents in the background keeps presenting a deleted document until the next indexing, and that its compliance check "does not check whether the information is from a deleted document" [14].

Order of changes. The Zanzibar paper, on a system handling "trillions of access control lists and millions of authorization requests per second" in 2019, names the "new enemy" problem, which "can arise when we fail to respect the ordering between ACL updates or when we apply old ACLs to new content" [29]. In its example, Bob is removed from a folder before new documents move in, and he "should not be able to see the new documents, but may do so if the ACL check neglects the ordering between the two ACL changes" [29]. A check at query time closes part of the gap, filtering out "any resource with permissions revoked but whose authorization data has not made it into your index yet" [26].

6. Logging what was retrieved and for whom

OWASP asks teams to "Maintain detailed immutable logs of retrieval activities to detect and respond promptly to suspicious behavior" [1]. One vendor's audit records for its assistant show what such a log can hold: which user interacted with the assistant, when and where, and "references to files, sites, or other resources" accessed to generate the response [21]. Each accessed resource carries its ID, type, name, sensitivity label, action, any policy under which access "was blocked or restricted", and a flag for "whether there was an XPIA (Cross Prompt Injection Attack) detected from a particular resource" [21].

A log limited to cited passages would have missed the chat case, because the answer's citation did not point to the attacker's message [9]. A record of every passage retrieved for each request, with the user, the time and the acting assistant, is what shows the injected message afterwards (reasoning from [1, 9]). Audit fields for tool calls are in our explainer on MCP access control (https://rillor.com/insights/mcp-access-control).

Implications

Keep the source's permissions with each document, and filter on a verified identity. Every query filters on an identity the server looked up or verified [3, 25], because the filter string itself authenticates nothing [11]. Check how document-level security combines roles and what aggregates expose [12], and connect to a database as a role that row security binds [23].

Retrieve with the user's delegated rights. A delegated token carries one user's rights and records the assistant as the actor [2, 27]. Short-lived tokens per task and a re-check at each privileged step address the inherited-rights and stale-token cases on OWASP's agentic list [13].

Check every participant in shared spaces, as the participant-aware position proposes [34].

Close the sync gap at query time. Permissions reach the index after a crawl, a sync or a refresh [3, 5]. A query-time check catches rights revoked since indexing [26], caches need the same treatment [14], and ACL changes need ordering against new content [29].

Treat vectors and caches as copies of the documents, with the permissions of their sources and partitions in shared stores [1, 15]. A namespace per tenant separates customers [24]. Inside a tenant, the per-document checks above still apply.

Expect injection anyway. Permission-aware retrieval bounds what an injected instruction can reach and does not stop it [8, 9]. Trust tiers that one answer never mixes address the path EchoLeak used [8]. Our explainer on prompt injection testing (https://rillor.com/insights/prompt-injection-testing) covers how to test an agent, and write access to the index needs the same care as read access [1, 32].

Use discovery restrictions to buy time. They hide content while owners review permissions, change no permissions and can take more than a week on large sites [16, 20]. The governance reports find the sites to review [7].

Log every retrieval, cited or not, with the user, the time and the acting assistant [1, 9, 21].

Where Rillor works. Rillor builds internal assistants that work inside a client's systems with each user's own permissions and keep a log of every action they take.

Limits of this analysis

- This is an explainer of published documentation and research. It ranks no products and tests none, and no source reviewed measures how often files are overshared.
- Vendor documentation changes. Four pages carry no date [5, 12, 23, 24], and several features are marked preview [3, 6].
- Research figures hold for the authors' setups: HONEYBEE's workloads [33], the 92% inversion result for 32-token inputs and one of the two models tested (61% for the other) [15], PoisonedRAG's 90% [32] and the extraction rates of Qi and colleagues [31]. The 1 RU against 100 RUs is the vector database's own pricing example [24].
- Both disclosed cases were fixed by their vendors and are described as past cases [8, 10].
- RFC 8693 is a standard and the on-behalf-of flow is one vendor's design [2, 27]. No claim is made that any assistant product uses either.
- Access control for models fine-tuned on restricted data is outside the scope of this report.
- The healthcare example in one vendor's post is that vendor's illustration [25]. Nothing here is legal or compliance advice.

Sources

1. LLM08:2025 Vector and Embedding Weaknesses, OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, published 10 April 2024, modified 28 April 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm082025-vector-and-embedding-weaknesses/

2. RFC 8693: OAuth 2.0 Token Exchange. M. Jones, A. Nadalin, B. Campbell, J. Bradley and C. Mortimore, Internet Engineering Task Force, January 2020, viewed 8 November 2026. https://datatracker.ietf.org/doc/html/rfc8693

3. Document-Level Access Control, Azure AI Search. Microsoft Learn, updated 17 September 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/azure/search/search-document-level-access-overview

4. Data, Privacy, and Security for Microsoft Copilot. Microsoft Learn, updated 4 November 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/copilot/microsoft-365/microsoft-365-copilot-privacy

5. Data source connector concepts. Amazon Q Business User Guide, Amazon Web Services, undated, viewed 8 November 2026. https://docs.aws.amazon.com/amazonq/latest/qbusiness-ug/connector-concepts.html

6. Set up data source access control. Google Cloud Documentation, Agent Search, last updated 10 November 2026, viewed 12 November 2026. https://docs.cloud.google.com/generative-ai-app-builder/docs/data-source-access-control

7. Data access governance reports for SharePoint sites. Microsoft Learn, updated 30 July 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/sharepoint/data-access-governance-reports

8. EchoLeak: The First Real-World Zero-Click Prompt Injection Exploit in a Production LLM System. Pavan Reddy and Aditya Sanjay Gujral, arXiv:2509.10540, 6 September 2025 (AAAI Fall Symposium Series 2025), viewed 8 November 2026. https://arxiv.org/abs/2509.10540

9. Data Exfiltration from Slack AI via indirect prompt injection. PromptArmor, 20 August 2024, viewed 8 November 2026. https://promptarmor.substack.com/p/data-exfiltration-from-slack-ai-via

10. Slack Security Update. Slack, 21 August 2024, viewed 8 November 2026. https://slack.com/blog/news/slack-security-update-082124

11. Security filters for trimming results in Azure AI Search. Microsoft Learn, updated 24 August 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/azure/search/search-security-trimming-for-azure-search

12. Controlling access at the document and field level. Elastic Docs, undated, viewed 8 November 2026. https://www.elastic.co/docs/deploy-manage/users-roles/cluster-or-deployment-auth/controlling-access-at-document-field-level

13. OWASP Top 10 for Agentic Applications for 2026. OWASP Gen AI Security Project, Agentic Security Initiative, 9 December 2025, viewed 8 November 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

14. ConfusedPilot: Confused Deputy Risks in RAG-based LLMs. Ayush RoyChowdhury, Mulong Luo, Prateek Sahu, Sarbartha Banerjee and Mohit Tiwari, arXiv:2408.04870, version 5, 23 October 2024, viewed 8 November 2026.

https://arxiv.org/abs/2408.04870

15. Text Embeddings Reveal (Almost) As Much As Text. John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov and Alexander M. Rush, arXiv:2310.06816, 10 October 2023 (EMNLP 2023), viewed 8 November 2026. https://arxiv.org/abs/2310.06816

16. Restrict discovery of SharePoint sites and content. Microsoft Learn, updated 31 October 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/sharepoint/restricted-content-discovery

17. Guide to Attribute Based Access Control (ABAC) Definition and Considerations, NIST SP 800-162. Vincent C. Hu, David Ferraiolo, Rick Kuhn and others, National Institute of Standards and Technology, January 2014, updated 2 August 2019, viewed 8 November 2026. https://csrc.nist.gov/pubs/sp/800/162/upd2/final

18. LLM02:2025 Sensitive Information Disclosure, OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, published 10 April 2024, modified 5 May 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm022025-sensitive-information-disclosure/

19. LLM01:2025 Prompt Injection, OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, published 10 April 2024, modified 17 April 2025, viewed 8 November 2026. https://genai.owasp.org/llmrisk/llm01-prompt-injection/

20. Restricted SharePoint Search. Microsoft Learn, updated 18 August 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/sharepoint/restricted-sharepoint-search

21. Audit logs for Copilot and AI applications. Microsoft Learn, Microsoft Purview, updated 26 August 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/purview/audit-copilot

22. Create, update, and delete items in a Microsoft Graph connection. Microsoft Learn, updated 6 August 2025, viewed 8 November 2026. https://learn.microsoft.com/en-us/graph/connecting-external-content-manage-items

23. 5.9 Row Security Policies, PostgreSQL 18 Documentation. The PostgreSQL Global Development Group, undated, viewed 8 November 2026. https://www.postgresql.org/docs/current/ddl-rowsecurity.html

24. Implement multitenancy. Pinecone Docs, undated, viewed 8 November 2026. https://docs.pinecone.io/guides/index-data/implement-multitenancy

25. Access control for vector stores using metadata filtering with Amazon Bedrock Knowledge Bases. AWS Machine Learning Blog, Amazon Web Services, 2 July 2024, viewed 8 November 2026. https://aws.amazon.com/blogs/machine-learning/access-control-for-vector-stores-using-metadata-filtering-with-knowledge-bases-for-amazon-bedrock/

26. Search With Permissions. OpenFGA documentation, last modified 28 September 2026, viewed 8 November 2026. https://openfga.dev/docs/interacting/search-with-permissions

27. Microsoft identity platform and OAuth 2.0 On-Behalf-Of flow. Microsoft Learn, updated 15 June 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-on-behalf-of-flow

28. CVE-2025-32711. NIST National Vulnerability Database, published 11 June 2025, last modified 17 June 2026, viewed 8 November 2026. https://nvd.nist.gov/vuln/detail/CVE-2025-32711

29. Zanzibar: Google's Consistent, Global Authorization System. Ruoming Pang, Ramon Caceres, Mike Burrows and others, 2019 USENIX Annual Technical Conference, July 2019, viewed 8 November 2026. https://www.usenix.org/conference/atc19/presentation/pang

30. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz and Mario Fritz, arXiv:2302.12173, version 2, 5 May 2023, viewed 8 November 2026. https://arxiv.org/abs/2302.12173

31. Follow My Instruction and Spill the Beans: Scalable Data Extraction from Retrieval-Augmented Generation Systems. Zhenting Qi, Hanlin Zhang, Eric Xing, Sham Kakade and Himabindu Lakkaraju, arXiv:2402.17840, version 3, 6 October 2024, viewed 8 November 2026. https://arxiv.org/abs/2402.17840

32. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. Wei Zou, Runpeng Geng, Binghui Wang and Jinyuan Jia, arXiv:2402.07867, version 3, 13 August 2024 (USENIX Security Symposium 2025), viewed 8 November 2026. https://arxiv.org/abs/2402.07867

33. HONEYBEE: Efficient Role-based Access Control for Vector Databases via Dynamic Partitioning (Technical Report). Hongbin Zhong, Matthew Lentz, Nina Narodytska, Adriana Szekeres and Kexin Rong, arXiv:2505.01538, version 3, 7 January 2026 (SIGMOD 2026), viewed 8 November 2026. https://arxiv.org/abs/2505.01538

34. Enterprise AI Must Enforce Participant-Aware Access Control. Shashank Shreedhar Bhatt, Tanmay Rajore, Khushboo Aggarwal and others, arXiv:2509.14608, 18 September 2025, viewed 8 November 2026. https://arxiv.org/abs/2509.14608

35. Policy-aware Vector Search: A Vision for Fine Grained Access Control in Vector Databases. Lakshmi Sahithi Yalamarthi and Primal Pappachan, arXiv:2606.19803, 18 June 2026 (SeQureDB 26, SIGMOD 2026), viewed 8 November 2026. https://arxiv.org/abs/2606.19803

About this report

Published by Rillor on 13 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Permission-aware retrieval for internal assistants. 13 November 2026. https://rillor.com/insights/permission-aware-retrieval-assistants

Notices

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Tell us the workflow, the sources it draws on and the decisions it supports, and we scope the agents, tools and tests around them. Contact us at rillor.com/contact?type=agentic-systems or contact@rillor.com. More on agentic systems: rillor.com/si/agentic-systems