Computer Fraud and Security
ISSN (online): 1873-7056

Secure Retrieval-Augmented Generation: Preventing Data Leakage with Provenance and Policy Enforcement

Narendra Bhargav Boggarapu

Wells Fargo, USA

Abstract

Retrieval-Augmented Generation (RAG) has become a paradigm architectural design to implement large language models in industries that face some form of regulation, but its probabilistic nature of retrieval creates a high risk of data leaks where access controls are not enforced or are used improperly. Secure RAG must have a comprehensive governance posture covering ingestion, indexing, retrieval, and generation mediated by identity-aware filtering, attribute-based access control, and policy-as-code frameworks that are version-controlled and auditably separate. Threats in this list are prompt injection, entitlement bypass, index poisoning, and generation-stage inferential disclosure, which need different prevention measures at the correct stage of the pipeline. The structural basis of the consistent enforcement is a two-plane architecture between the data plane and the control plane, where Open Policy Agent is the common decision-making and verifiable evidence bundle that meets the traceability considerations of the IEEE P7001 transparency standard. Assessment should be ongoing as opposed to periodic, and in terms of leakage rate, entitlement violation rate, provenance fidelity, and refusal correctness against a security harness that is consistent with the four core governance functions of the NIST AI RMF. The outcome is a deployment model where the enforcement of policy can be measured, provenance can be demonstrated to regulatory inquisitions, and leakage risk is structurally constrained as opposed to the widely held belief that this risk is mitigated.

Keywords: Retrieval-Augmented Generation, Attribute-Based Access Control, Policy-as-Code, Data Provenance, Zero Trust Architecture

1. Introduction

Generative artificial intelligence has swiftly advanced from experimental prototype to production-grade implementation in regulated sectors, especially in financial services. Retrieval Augmented Generation (RAG) is one of the architectural patterns that have been developed to bring large language models (LLMs) enterprise-ready through its practicality to base model outputs on verified, up-to-date, and domain-specific knowledge without storing proprietary data directly in model weights. RAG makes the break between the knowledge store and the model to allow dynamic updating of knowledge, source attribution and selective access control on sensitive content. This capability at scale in the initial RAG architecture presented by Lewis et al. (2020) was achieved by indexing 21 million document chunks, created by a Wikipedia corpus, searched at inference time with a dense passage retriever implementation of the Maximum Inner Product Search (MIPS) [2]. It consisted of a BERT-based bi-encoding retriever and a BART-large generator with 400 million parameters fine-tuned end-to-end on natural language processing tasks that are knowledge-intensive [2]. The empirical findings proved the feasibility of the non-parametric memory retrieval as a production-scale alternative to purely parametric methods.

Nonetheless, a similar increase in the attack surface is presented by the architectural convenience of RAG. Within the setting when data classification is warranted over regulatory filings that are available to the public, customer-level personally identifiable information (PII) and confidential operating processes, the retrieval pipeline is a possible high-throughput exfiltration channel. Every query is inserted into a vector index, which can hold millions of embedded document fragments, and the process of retrieving the similarity-based retrieval will result in the appearance of semantically relevant data without regard to the level of entitlement of the requester unless specific access controls are enforced on the index layer. In the absence of identity-conscious filtering, a similarity score is the sole determinant of what sensitive content is promoted into a context window of the model, a structural vulnerability that can not be entirely addressed by application-layer controls [2].

Published in January 2023, the NIST Artificial Intelligence Risk Management Framework (AI RMF 1.0) dictates that AI systems should meet seven characteristics of trustworthiness, including validity and reliability, safety, security and resilience, accountability and transparency, explainability and interpretability, privacy enhancement, and fairness with

harmful bias managed [1]. The structure systematizes the risk management activity around four main core functions, namely, GOVERN, MAP, MEASURE, and MANAGE, and places the continuous monitoring as a structural necessity in place of a periodic activity. These principles, when applied to RAG deployments in controlled settings, can be simply translated into verifiable provenance to retrieve auditable policy implementation at each pipeline stage and quantifiable leakage controls verifiable to regulators and internal audit functions [1]. The AI RMF clearly acknowledges that AI risks are socio-technical in nature and manifest not solely due to the behavior of the models but also as a result of the system design choices, data governance activities, and organizational settings in which AI is implemented.

The Natural Questions benchmark had a maximum possible match score of 44.5, which was achieved by RAG-Sequence models, which are significantly better than much bigger purely parametric models because of the use of retrieved non-parametric context during inference time [2]. This retrieval capability should be controlled in an enterprise implementation: the retriever has to be identity-conscious, the index has to store the access control tags at the chunk level, and each decision to retrieve has to produce an evidence trail that can be reconstructed. A secure RAG deployment in controlled settings is characterized by three properties: leakage protection at both retrieval and generation phases, provability under legal examination, and policy enforcement, which is not assumed to be impossible to measure. Each of these properties is discussed in considerable detail below.

2. Threat Landscape in RAG-Based Systems

It is crucial to appreciate the threat scenario unique to the RAG pipelines to be able to design appropriate controls. RAG contrasts with the traditional API-based data access model in which authorization gates are clearly defined and binary, in addition to probabilistic retrieval, the outcome of which relies on the semantic closeness in a high-dimensional embedding space. This probabilistic aspect gives rise to new types of risks beyond the traditional access control failures, which are further input manipulation, retrieval compromise, and generation-stage leakage, all of which interact with each other throughout the pipeline.

2.1 Input Manipulation Threats

The most ubiquitous and structurally relevant threat to systems based on LLMs is the case of prompt injection. Direct injection is an attack in which the attacker designs a query that bypasses system-level instructions. System-level instructions are instructions that are embedded in what appears to be a harmless input by a user to cause the model to act in a way that it is not supposed to. Indirect injection is more covert; there are malicious instructions embedded amongst the documents stored in the knowledge base so that when the retriever brings up those documents and drops them into the context window of the model, the instructions are executed at inference time with no indication of it in the original query. OWASP Top 10 Large Language Model Applications lists prompt injection as the most prevalent vulnerability in all LLM deployments, observing that both direct and indirect versions thereof may permit privilege escalation, unauthorized access to data, and agentic execution of actions [3].

An experimental study of immediate injection with the PROMPTINJECT framework, a modular adversarial prompt composition framework trained on 35 base prompts, indicated that low-sophistication adversarial signals also have the ability to effectively misalign production LLM behavior [4]. Namely, goal hijacking attacks scored an estimated success rate of 58.6% on the text-davinci-002 model, i.e., over half of the experimented prompts were redirected to attacker-specified outputs. In comparison, weaker models like text-ada-001 had a success rate of only 13.8% in the same attack models; the researchers believe this is due to the inverse scaling phenomenon, where the more powerful the instruction-following models, the more vulnerable they are to following injected instructions [4]. The results have a direct influence on RAG threat modeling: the better the generation model, the higher the vulnerability to indirect injection with the help of retrieved document content.

2.2 Retrieval Compromise

In case of Entitlement bypass using retrieval ACL enforcement is not provided at the index of the vector, but it is enforced at the application layer. In the absence of metadata filtering based on subject attributes at query time, identical similarity scores are used to select what sensitive material to include in the context window, and a structural pathway to unauthorized data disclosure that cannot be entirely redressed by any output-layer control exists. Index poisoning is a supply-chain-level extension of this threat: when the ingestion pipeline is not integrity-verified, maliciously authored content may be added to the knowledge base and be returned by any downstream query, the embedding of which is semantically proximate to the poisoned chunk. The OWASP Top 10 list of vulnerabilities applies to LLC applications,

giving 10 categories, where training data poisoning and vulnerabilities in supply chains are some of the most significant in the context of RAG architectures in particular [3].

2.3 Generation Leakage

Generation-stage leakage is a residual risk even in cases where the retrieval controls are strong. One of the ways retrieved pieces may be synthesized by a model is to form sensitive information that is not found in one of the pieces, what the PROMPTINJECT study calls an emergent reconstruction vector, where parametric knowledge of the model is combined with the retrieved context to generate inferential revelations [4]. The evaluation of output safety needs to differentiate between verbatim disclosure and inferential disclosure because both have regulatory implications in terms of regulated mandates of data protection in the regulated environment.

Threat Type	What It Is	Risk
Input Manipulation	Direct/indirect prompt injection via user query or retrieved docs	Instruction hijack, unauthorized actions
Retrieval Compromise	Weak ACLs, missing metadata filtering, index poisoning	Unauthorized sensitive data retrieval
Generation Leakage	Verbatim or inferential disclosure from model outputs	Data leakage, compliance violations

Table 1: Threat Landscape in RAG-Based Systems [3, 4]

3. Secure RAG Architecture and Control Mechanisms

A production-scale Secure RAG architecture should impose constraints at each point through the data lifecycle: ingestion, classification, indexing, retrieval, pre-generation processing, generation, and output processing. Point solutions that are implemented at the generation layer are not enough since decisions regarding upstream retrieval have already been made that dictate what sensitive content is going into the context window of the model. The structure of a two-plane design (distinct control plane and data plane) offers the basis of predictable enforcement, which is auditable. In a zero trust architecture, it is stipulated by NIST Special Publication 800-207 that the policy enforcing points, policy administrator, and policy engine must be located logically apart and that application data flows across a separate data plane [5]. Such a separation principle directly corresponds to Secure RAG: all identity evaluation, entitlement decisions, and policy enforcement fall into the control plane, and document retrieval, context assembly, and generation fall into the data plane governed by control-plane policies.

3.1 Control Plane Components

The control plane concentrates on the identity management, attribute-based access control, policy evaluation and evidence handling. NIST SP 800-207 considers zero trust with 7 foundational tenets; the most operationally critically important in the deployment of RAG systems is the principle that individual enterprise resources can be accessed only on a per-session basis, with trust in the requester considered before a resource can be accessed and not based on network location or on a pre-existing authentication state [5]. In the case of RAG, it implies that the retriever should consider a new entitlement decision for every request, no authorization of a previous session should be forwarded. In their contextual form, the zero trust trust algorithm is also able to identify abnormal retrieval patterns: an account, which typically requests 20 to 30 document chunks per session, sending an alert when the requests suddenly increase to 100 or higher, which can be an indication of credential compromising or policy probing [5].

The identity and entitlement module submits subject attributes, role, business unit, clearance level, and geographic region, to the policy decision point (PDP), which compares every retrieval request against versioned policy rules. NIST Special Publication 800-162 describes the concept of the Attribute Based Access Control (ABAC) as a form of access control in which permissions are granted or denied based on the assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies expressed in terms of attributes and conditions [6]. Object attributes in a RAG environment are classification labels, residency constraints and retention flags assigned to every document chunk during ingestion. Role and clearance are also subject attributes. The overlap of these sets of attributes, computed at query time through the PDP determines the eligibility of a chunk to be retrieved.

3.2 Data Plane Components

The ingestion, embedding, indexing, retrieval, and generation are carried out in the data plane, and the control plane continuously governs them. NIST SP 800-162 means that the access control mechanism consists of 4 functional components, namely the Policy Decision Point (PDP), Policy Enforcement Point (PEP), Policy Information Point (PIP), and Policy Administration Point (PAP), the distributed implementation of which across an enterprise enables centralized decision services to control a multiplicity of applications without necessarily each application having to support its own authorization infrastructure [6]. When used in Secure RAG, the PEP is placed at the query boundary of the vector index query, and filters metadata of ACLs are applied during retrieval prior to any content being passed to the context assembly layer. After retrieval, a redaction gate obtains information using pattern matching of structured sensitive data on a deterministic basis and unstructured PII detection with the help of ML and then transfers the context to the LLM. An end-product safety gate implements citation validity and limited-topic refusal logic prior to delivery of responses.

Area	Main Controls	Purpose
Control Plane	Identity, ABAC policy evaluation (PDP/PAP/PIP), session-based authorization	Trust and entitlement decisions per request
Data Plane	Ingestion → indexing → retrieval → context assembly → generation	Executes retrieval/generation under enforced policies
Safety Gates	PEP metadata filtering, redaction + PII detection, output safety checks	Prevents sensitive data exposure and unsafe outputs

Table 2: Secure RAG Architecture [5, 6]

4. Policy Enforcement and Provenance Management

Implementation of policy on a Secure RAG deployment should be full and auditable. Enforcement that is solely enforced in application code, instead of externalized and versioned policy rules, is weak: it cannot be audited, cannot be updated, and cannot be tested with historical queries to prove it is correct. To overcome this structural fragility, policy-as-code systems represent data handling policies in declarative form, under version control, and testable and comprehensible by the policy decision point at runtime. The outcome is the enforcement model that is not only responsive but also proactively auditable throughout the pipeline stages.

4.1 Policy-as-Code Enforcement

Open Policy Agent (OPA) is a decoupled policy engine, built as a general-purpose and open-source policy engine that is a graduated project of the Cloud Native Computing Foundation (CNCF) and which decouples policy decision-making and policy enforcement across the software stack [7]. Instead of implementing authorization logic on a per-service basis, OPA concentrates policy evaluation by taking structured input in the form of a JSON message to any service making a query and returning a policy decision, which could be a simple allow or deny response or arbitrary structured output, based on rules in a purpose-designed declarative language, Rego [7]. Secure RAG is architecturally sensitive to this decoupling. instead of having enforcement logic distributed through the ingestion service, the vector index query layer, the redaction gate, and the output safety gate, one OPA instance, or a distributed collection of OPA instances operating in server mode, is the sole authority determining all pipeline stages concurrently.

OPA policy rules are used to enforce each different step in a Secure RAG pipeline, which includes ingestion-time classification and tagging, index-time ACL assignment, retrieval-time entitlement filtering, and generation-time output restrictions. Any point of enforcement sends a query to OPA with the bundle of attributes of the subject and the metadata of the classification of the object, and OPA compares the rules in the Rego ruleset with the current policy data to generate a structured response. Upon any change in a regulatory requirement (e.g., a new data residency requirement applicable to a given level of document classification), the versioned policy repository stores the updated Rego rule and is instantly accessible to all enforcement points without the need to orchestrate application redeployments [7]. This model of propagation removes the policy drift risk of inline enforcement whereby enforced behavior may fail silently to match documented policy across independently deployed service components.

4.2 Provenance and Evidence Bundles

A provenance and evidence bundle is a bundle of elements that contains provenance, evidence, and history.

According to the IEEE P7001 standard development paper, a survey of AI ethics guidelines identified by Jobin et al. found that transparency was present in 73 of 84 sets of guidelines they reviewed and thus has an 87% prevalence rate, making transparency the most commonly mentioned ethical value throughout the global AI governance landscape [8], and 11 guidelines were not included in the ethical principle. This observation gives the normative basis that provenance should be treated as an optional addition to Secure RAG but it is a structural requirement to deploy it. The IEEE P7001 standard of transparency of autonomous systems assigns the meaning of transparency to the transfer of information to a stakeholder by an autonomous system, or by the designers of that autonomous system, in an honest, information-relevant manner, and at a degree of abstraction that is significant to the stakeholder [8]. This definition is directly applicable to RAG, in that the bundle is required to be reproducible, truthful in reflecting the reality of the real retrieval situation, and understandable by the regulatory or audit stakeholder on whom the bundle is reported.

P7001 identifies 5 stakeholder categories: end users, the rest of the population and bystanders, safety certification agencies, incident and accident investigators, and lawyers and expert witnesses, where each stakeholder category needs a varying form and degree of transparency by the autonomous system [8]. The group of incident and accident investigators with P7001 specifying a cumulative transparency scale of 0 to 5 starting with event data recording and moving on to timestamped recording of sensor inputs, high-level decisions, and decision reasons is the operationally most critical in regulated RAG deployments [8]. Conversion into the higher levels of the Secure RAG terms stipulates that all evidence bundles include identifiers and version hashes of source documents, coordinates of chunk boundaries, similarity scores at retrieval, ACL filter settings used, policy selections regarding each retrieved chunk, model identifier and prompt template version, millisecond-precise timestamps, and a cryptographic hash of the generated output. These bundles support three audit-critical functions, namely, precise reproduction of the retrieval context of any historical query, showing that a specific document was or was not in the context window of the model when the response that was disputed was given, and rollback control, which identifies all the responses that were affected by a document that has since been determined to be misclassified or inaccurate.

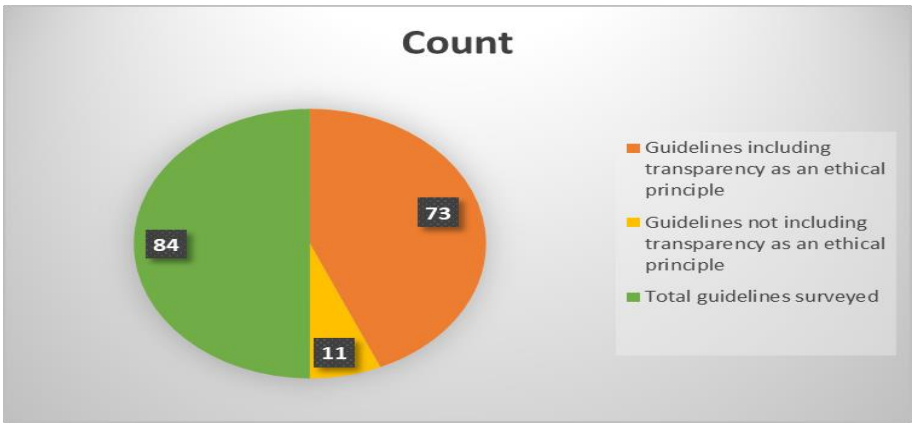

Fig 1: Transparency Prevalence in AI Ethics Guidelines [7, 8]

5. Evaluation Framework and Future Directions

Lack of a strict evaluation methodology in the deployment of a Secure RAG will not allow it to prove to regulators, internal risk functions, or audit bodies that it is in a secure posture. The assessment should be ongoing as opposed to periodic, and it should include the result of security and the viability of the operations. The article AAGATE illustrates how the 4 core capabilities of the NIST AI RMF, namely Govern, Map, Measure, and Manage, can be refined into workable architectural patterns and engineering controls of AI systems in production settings [9]. This four-function structure is applied to Secure RAG, giving the policy base and accountability framework outlining the natural scaffold of a security harness mapping that identifies the full risk surface of the retrieval pipeline, a measurement that quantifies security performance in real-time, and management that implements containment and remediation in the event of a defined threshold being violated.

5.1 Security-Focused Evaluation Metrics

A Secure RAG system should have its evaluation harness measure four key outcomes of security that are correlated to the NIST AI RMF Measure function. Leakage rate The proportion of test queries that reply with restricted content in any context or the ultimate generated output when the requester is not entitled to it. Entitlement violation rate is a fraction of unauthenticated access attempts, which succeed against the vector index, separating ACL filtering performance with no redaction controls in the downstream. Provenance fidelity quantifies the proportion of generated responses that have corresponding citations that successfully map to precise pieces of text retrieval, which assures that citations are based on real retrieval and not hallucinated attribution. Assessed by Refusal incorrectness determines how the output safety gate handles restricted-topic queries correctly and false-permitted queries correctly, which produces a minimum number of false refusals and false compliances.

The AAGATE platform implements the Measure function as a two-part pipeline: the UEBA Behavior Profiler creates behavioral fingerprints (anomaly scaled) of every agent, and the Compliance Agent forwards events at the Tool-Gateway and runs policy checks against PII leakage and policy drift [9]. Incidents with a score are then processed by the Governing-Orchestrator Agent in a form of structured decision tree, turning response into track, alert or quarantine, which is a tiered response model directly aligned to the structure of Secure RAG, with retrieval anomalies, ACL bypass attempts, and responses requiring policy violation in the generation stage, each having a level-specific response escalation path.

5.2 Operational Metrics and Latency Considerations

Security results need to be quantified in addition to the security controls imposing a latency cost. The ACL metadata filtering incurs overhead at query time, the redaction gate incurs processing delay at the time of generation, and the output safety gate incurs processing delay at the time of post-generation evaluation. An actual assessment harness should measure p95 retrieval latency, end-to-end response latency, and per-query token cost. To do so, the AAGATE architecture is tackling Redis caching in the UEBA pipeline, where low latency has been named a structural requirement to enable a successful real-time security reaction [9].

The larger investment trend of attribute-based enforcement infrastructure speaks of the magnitude of the issue. The ABAC market in the worldwide market was estimated at 1.8 billion dollars in 2023 and is forecasted to experience a growth of 24.9 percent to 5.4 billion dollars by 2028 [10]. This increase can be attributed to the understanding that the real-time attribute evaluation that ABAC provides, including the attributes of the subject, the classification of a particular object, the type of the action, and environmental conditions, can provide the granular and context-specific enforcement needed by the secure RAG pipelines along the vector index query boundary. The functional architecture of the 4-component access control mechanism of ABAC, which is made of Policy Decision Point, Policy Enforcement Point, Policy Information Point, and Policy Administration Point, gives the ability to enforce using consistent mechanisms across distributed stages of the RAG pipeline [10].

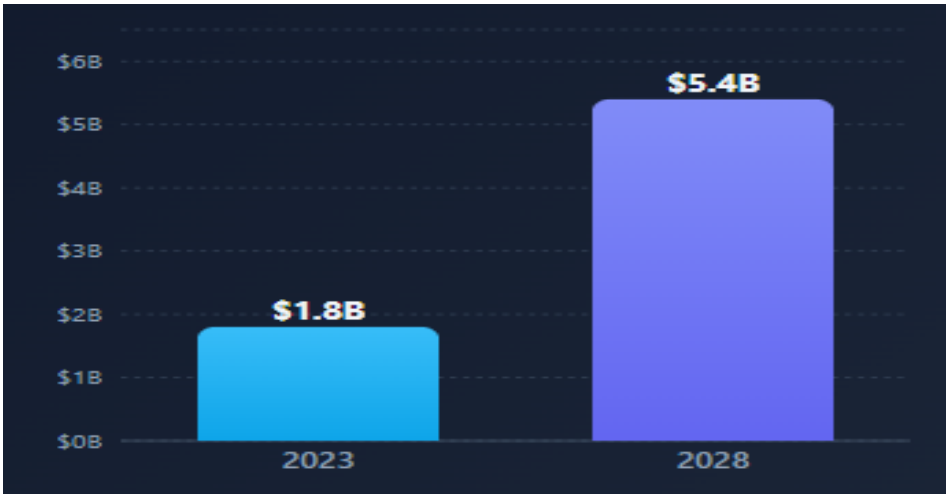

Fig 2: Attribute-Based Access Control (ABAC) [9, 10]

5.3 Future Directions

There are a number of open research issues. Embedding space techniques that use differential privacy can minimize the threat of recreating sensitive information under semantic proximity queries. Isolating by classification level Document indexes in federated retrieval architectures can be physically segregated and guaranteed to a higher degree of isolation than by metadata-based filtering alone. The AAGATE framework recognizes 7 MAESTRO architectural layers as a systematic prism to examine threat modeling multi-component AI systems, a framework that is naturally extended to RAG pipelines where the threats cut across the vulnerability of foundation models, data operations, agent frameworks, and deployment infrastructure [9]. With agentic RAG architectures, which bring in the ability to do multiple steps of retrieval and tool use, the attack surface increases further and the need to have integrated behavioral analytics, policy-as-code enforcement, and audit trails that are verifiable and run as a single, continuously active plane of governance.

Conclusion

Ensuring RAG deployments in controlled settings will need more than generation-layer guardrails, it will need enforcement at each point of the data lifecycle, including ingestion-time classification and retrieval-time entitlement filtering as well as policy validation at the output stage. Attribution-based access control, the principle of zero trust architecture, and policy-as-code implementation give the technical basis of a RAG pipeline whereby no material can arrive at the context window of the model without an explicit and auditable decision on entitlement. Documents that can be verified assemble provenance as a documentation of final thought into a structural auditable capability, with reproducibility, dispute capabilities, and controlled rollback at regulatory grade. Ongoing testing This is done by continuous evaluation of leakage rate, entitlement violation rate, provenance fidelity, and refusal correctness against a harness tuned to the NIST AI RMF Govern, Map, Measure, and Manage functions to ensure that security posture is not merely assumed. Since agentic RAG architectures increase the attack surface due to multi-step retrieval and use of autonomous tools, the governance model needs to change in tandem with behavioral analytics, formal policy verification, and federated index architecture. The principles set here give a sustainable basis to that evolution: quantifiable enforcement, provisibly proven provenance, and structurally constrained leakage risk throughout the entire RAG implementation life cycle.

References

[1] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

[2] Patrick Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv:2005.11401v4, 2021. [Online]. Available: https://arxiv.org/pdf/2005.11401

[3] Mohammad Fasha et al., "Mitigating the OWASP Top 10 For Large Language Models Applications using Intelligent Agents". [Online]. Available: https://www.arxiv.org/pdf/2601.18105

[4] Fábio Perez and Ian Ribeiro, "Ignore Previous Prompt: Attack Techniques For Language Models," arXiv:2211.09527v1, 2022. [Online]. Available: https://arxiv.org/pdf/2211.09527

[5] Scott Rose et al., "Zero Trust Architecture," NIST, 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf

[6] Vincent C. Hu et al., "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST, 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf

[7] Open Policy Agent, "Open Policy Agent (OPA)." [Online]. Available: https://www.openpolicyagent.org/docs

[8] Alan Frank Thomas Winfield et al., "IEEE P7001: A Proposed Standard on Transparency," Frontiers, 2021. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.665729/full?trk=public_post_comment-text

[9] Ken Huang et al., "AAGATE: A NIST AI RMF-Aligned Governance Platform for Agentic AI," arXiv:2510.25863, 2025. [Online]. Available: https://arxiv.org/abs/2510.25863v2

[10] Dinesh Rajasekharan et al., "Simplifying Attribute-Based Access Control (ABAC) for Modern Enterprises," IJSRCSEIT, 2025. [Online]. Available: https://ijsrcseit.com/index.php/home/article/view/CSEIT251112332