LogicPoison: Logical Attacks on Graph Retrieval-Augmented Generation

Yilin Xiao[♠*], **Jin Chen**[♣*], **Qinggang Zhang**[♠◇], **Yujing Zhang**[♠], **Chuang Zhou**[♠], **Longhao Yang**[♣], **Lingfei Ren**[♣†], **Xin Yang**[♣], **Xiao Huang**[♠]
[♣]School of Computing and Artificial Intelligence,
Southwestern University of Finance and Economics
[♠]The Hong Kong Polytechnic University
[◇]School of Artificial Intelligence, Jilin University
yilin.xiao@connect.polyu.hk jord.chen04@gmail.com
renlf@swufe.edu.cn xiao.huang@polyu.edu.hk

Abstract

Graph-based Retrieval-Augmented Generation (GraphRAG) enhances the reasoning capabilities of Large Language Models (LLMs) by grounding their responses in structured knowledge graphs. Leveraging community detection and relation filtering techniques, GraphRAG systems demonstrate inherent resistance to traditional RAG attacks, such as text poisoning and prompt injection. However, in this paper, we find that the security of GraphRAG systems fundamentally relies on the topological integrity of the underlying graph, which can be undermined by implicitly corrupting the logical connections, without altering surface-level text semantics. To exploit this vulnerability, we propose LogicPoison, a novel attack framework that targets logical reasoning rather than injecting false contents. Specifically, LogicPoison employs a type-preserving entity swapping mechanism to perturb both global logic hubs for disrupting overall graph connectivity and query-specific reasoning bridges for severing essential multi-hop inference paths. This approach effectively reroutes valid reasoning into dead ends while maintaining surface-level textual plausibility. Comprehensive experiments across multiple benchmarks demonstrate that LogicPoison successfully bypasses GraphRAG's defenses, significantly degrading performance and outperforming state-of-the-art baselines in both effectiveness and stealth. Our code is available at https://github.com/Jord8061/logicPoison.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023; Xiao et al., 2026b) has emerged as a promising paradigm for enhancing Large Language Models (LLMs) (OpenAI et al., 2024; Anthropic, 2024; Guo et al., 2025) by leveraging external knowledge bases. However, existing RAG systems face significant challenges when processing large-scale, unstructured corpora in real-world scenarios. The relevant information is often unevenly distributed across heterogeneous documents (Xiao et al., 2025; Yang et al., 2026). Consequently, the contexts retrieved by traditional RAG are frequently voluminous, fragmented, and lacking in structural organization, leading to inconsistencies in generation accuracy and coherence. To address these limitations, Graph Retrieval-Augmented Generation (GraphRAG) (Zhang et al., 2025; Peng et al., 2024; Xiao et al., 2026a) has gained prominence as a robust solution. By structuring knowledge into graphs, GraphRAG models hierarchical relationships and supports efficient multi-hop retrieval and reasoning, enabling more reliable use of background knowledge.

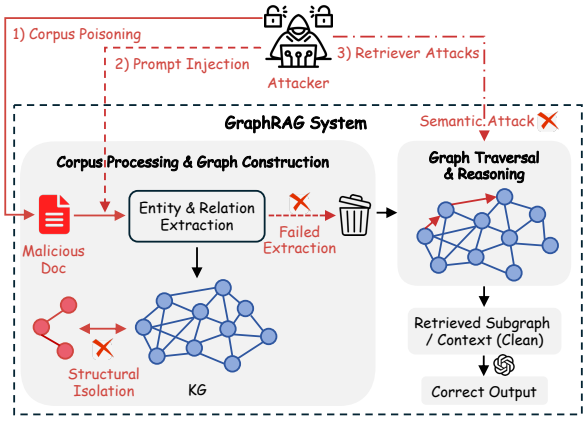

Figure 1: Traditional attacks on LLM or RAG are difficult to pose an effective threat to GraphRAG systems. This is mainly due to two characteristics. First, the construction process of the knowledge graph naturally filters part of the attack content. Second, the unique retrieval and reasoning mechanism of graph structure further improves the difficulty of attack implementation.

Security remains a critical challenge for Retrieval-Augmented Generation (RAG) systems due to their reliance on external corpora, which exposes them to adversarial attacks. As shown in Fig-

[*]Equal contribution.
[†]Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5575–5591
July 2-7, 2026

ure 1, prior research has identified three primary attack types: (i) Corpus Poisoning (Zou et al., 2025): injecting malicious documents into the knowledge source; (ii) Prompt Injection (Hines et al., 2024; Suo, 2024), where adversarial instructions are embedded within retrieved text to hijack model behavior; and (iii) Retriever Attacks (Wallace et al., 2019; Shafran et al., 2025), that craft deceptive queries to degrade the retrieval accuracy.

However, these methods are largely ineffective against GraphRAG due to its graph-based architecture. By constructing a knowledge graph from source corpora, GraphRAG introduces inherent defensive mechanisms. First, poisoning attacks are topologically marginalized, as synthetic entities fail to integrate into coherent graph communities. Second, and most critically, prompt injection is structurally prevented: retrieval operates over entity-relation subgraphs rather than raw text, thereby decoupling knowledge access from instruction execution and filtering out embedded adversarial commands. Besides that, because reasoning occurs over multiple connected paths within the graph, attacks targeting single nodes or local content fail to corrupt the system's global inference capability, as alternative uncontaminated reasoning paths remain available. Recent work like GRAGPOISON (Liang et al., 2025) has taken initial steps to attack GraphRAG. However, it relies heavily on LLM-generated contents to fabricate and amplify spurious relations, which is highly detectable.

To make it clear, we conduct an in-depth analysis (in Appendix A.1) on the graph-based architecture and reveal that GraphRAG functions less as a static knowledge store and more as a dynamic logical reasoning engine. Its efficacy stems not from the quantity of stored triples, but from its ability to conduct logical propagation via topological structures. Consequently, defeating GraphRAG requires disrupting its logical reasoning chains rather than merely corrupting isolated data points. However, the underlying graph structure is typically opaque (black-box) in practice, and different GraphRAG implementations employ diverse graph construction and reasoning mechanisms. This imposes two primary challenges: (i) *How to achieve effective, and low-magnitude poisoning without knowledge of the underlying graph structure?* (ii) *How can such an attack remain robust and transferable across diverse GraphRAG architectures?*

To overcome these challenges, we propose a novel attack framework: LOGICPOISON. Different from traditional attack methods that rely on additive noise or conflicting evidence, LOGICPOISON fundamentally targets the topological integrity of the knowledge graph constructed by GraphRAG systems. Specifically, it employs a Type-Preserving Entity Swapping mechanism to implicitly corrupt the graph structure without disrupting the textual surface. We devise a dual-pronged strategy to select target entities: (i) Global Logic Poison, which identifies and perturbs high-frequency hub nodes to shatter global graph connectivity; and (ii) Query-Centric Logic Poison, which utilizes Chain-of-Thought extraction to pinpoint and sever specific reasoning bridges essential for multi-hop queries. By applying cyclic permutations to these key entities within their respective types, we ensure that while the document remains semantically readable, the underlying graph topology is rerouted into logical dead ends or spurious shortcuts. Our contributions are summarized as follows:

- We identify a critical vulnerability in GraphRAG systems: while they are robust to unstructured noise, they are highly fragile to structural logic corruption. We formalize this as a topological security problem, shifting the attack surface from poisoned content injection to logic rewiring.
- We propose LOGICPOISON, a unified framework that combines global centrality analysis with query-specific reasoning extraction. This method allows for low-magnitude, stealthy modifications that effectively bypass the community summarization and filtering mechanism defenses inherent in GraphRAG.
- We conduct comprehensive experiments on multiple benchmarks across various SOTA GraphRAG architectures and base LLMs. The results demonstrate that LOGICPOISON significantly degrades GraphRAG performance, outperforming existing baselines.

2 Related Work

The security of Retrieval-Augmented Generation models has gained considerable attention in the research community in recent years. The concept of Universal Adversarial Triggers (Wallace et al., 2019) and Indirect Prompt Injection (Hines et al., 2024; Suo, 2024) served as the foundation for initial results that pointed out the vulnerability

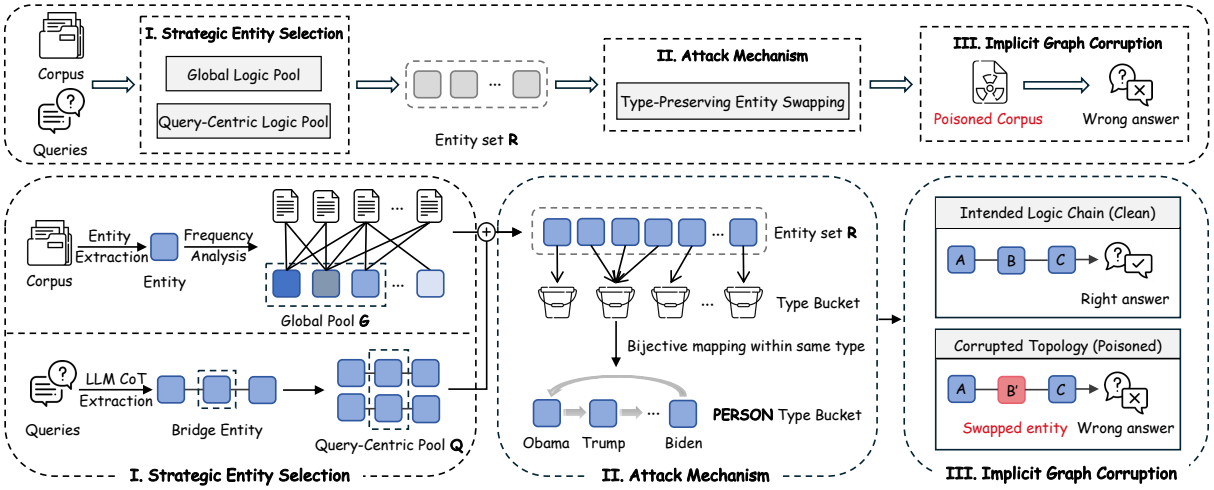

Figure 2: The overall framework of LOGICPOISON. The attack pipeline is divided into three stages: **I. Strategic Entity Selection**, where target entities are identified via a dual-pronged strategy combining global logic hubs and query-centric reasoning bridges into a unified set $\mathcal{R}$. **II. Attack Mechanism**, which employs a type-preserving cyclic permutation to swap entities within their respective type buckets in the corpus. **III. Implicit Graph Corruption**, demonstrating how the poisoned corpus subtly rewires the implicit topology of the constructed graph, severing valid reasoning chains ($A \to B \to C$) and re-routing them to incorrect entities ($A \to B' \to C$), leading to wrong answers while maintaining textual plausibility.

of large language models to any type of input attack. Starting with this point, various attacks on the RAG models have emerged in the pipeline: PoisonedRAG (Zou et al., 2025) defined knowledge-base poisoning for directing model behavior; PANDORA (Deng et al., 2024b) utilized retrieval functionality for conducting indirect jailbreak attacks; TrojanRAG (Cheng et al., 2024) optimized joint attacks for inserting backdoors in the retrieval contexts; Jamming attacks (Shafran et al., 2025) initiated denial-of-service conditions by promoting blocking documents; besides, privacy leakage issues intrinsic to the retrieval process were uncovered by further studies (Zeng et al., 2024). Most recently, GRAGPOISON (Liang et al., 2025) generalized GraphRAG attacks by injecting texts written by LLMs for simulating fake linkages in the graphs. The existing techniques were based on explicit content injection and additive perturbation. This work introduces LOGICPOISON, which changes the paradigm by re-modeling the topology of logical graphs to effectively manipulate GraphRAG by stealthy, type-preserving swapping, which corrupts GraphRAG in a logically implicit manner over more obvious generations.

3 Preliminaries

Given a retrieval corpus $\mathcal{C} = \{d_i\}_{i=1}^{N}$ and a set of queries $\mathcal{Q}$, a GraphRAG system builds an implicit or explicit knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from $\mathcal{C}$ to perform reasoning. Our objective is to obtain a poisoned corpus $\tilde{\mathcal{C}}$ via a perturbation function $\mathcal{P} : \mathcal{C} \to \tilde{\mathcal{C}}$, such that the system's reasoning on $\tilde{\mathcal{C}}$ is disrupted, while being imperceptible and grammatically coherent at the text level. In networked systems, the effectiveness of poisoning attacks is closely related to the structural importance of the targeted nodes. Since knowledge graphs are not available before graph construction, classical topological centrality measures cannot be applied directly. In such cases, corpus-level statistics offer alternative insights for node importance. From a spectral graph theory perspective (Spielman, 2012), global connectivity and information propagation are largely determined by the dominant modes of the adjacency structure, which are disproportionately influenced by high-degree nodes (Check more details in Appendix A.2.3). Consequently, high-frequency entities implicitly control the backbone of information flow, making them particularly effective targets for poisoning attacks.

4 The Framework of LogicPoison

We propose LOGICPOISON, a framework specifically targeting GraphRAG systems on the principle of topological disruption. Contrary to traditional poisoning methods that inject conspicuous amounts of misinformation, our solution carefully rewrites

the *logical reasoning chains* inherent in the corpus. By strategically permuting entities, we induce semantically plausible yet logically fallacious paths in the constructed knowledge graph, misleading the system into generating confident but incorrect answers.

The corrupted corpus $\tilde{\mathcal{C}}$ can be built by performing the operation of type-preserving entity swapping on a candidate set of entities $\mathcal{R}$. In order to have structural impact and querying relevance at the same time, we compose $\mathcal{R}$ using two complementary sub-strategies:

- **Global Logic Poison:** Targets corpus-level *logic hubs* to distort the global connectivity of the knowledge graph.
- **Query-Centric Logic Poison:** Targets specific entities essential for multi-hop logical reasoning chains to ensure the attack covers the precise paths required by user queries.

4.1 Global Logic Poison

GraphRAG systems rely on central entities to connect disparate information chunks. Disrupting these hubs destabilizes the global graph structure.

Entity Extraction. We treat each document $d_i \in \mathcal{C}$ as a processing knowledge unit. To identify potential knowledge graph nodes, we employ a lightweight Named Entity Recognition (NER) model (e.g., spaCy). Formally, for each document d_i, we extract a set of typed entities:

$$E(d_i) = \{(e, \tau) \mid e \in \mathcal{V}_{\text{vocab}}, \tau \in \mathcal{T}\}, \quad (1)$$

where e denotes the surface form and τ represents the entity type (e.g., PERSON, ORG, DATE).

Corpus-Level Frequency Analysis. We approximate the topological importance of an entity by its document frequency. For an entity e of type τ, we define its frequency as:

$$f(e, \tau) = \sum_{i=1}^{N} \mathbb{I}\big[(e, \tau) \in E(d_i)\big], \quad (2)$$

where $\mathbb{I}[\cdot]$ is the indicator function. Intuitively, entities with high $f(e, \tau)$ act as implicit hubs that bridge numerous communities in the graph.

Global Replacement Pool. To maximize disruption, we rank entities within each type τ by $f(e, \tau)$ and select the top-$n\%$ as targets:

$$\mathcal{G}_\tau = \{e \mid \text{type}(e) = \tau \wedge \text{rank}(f(e, \tau)) \leq n\%\}. \quad (3)$$

The global candidate set is defined as $\mathcal{G} = \bigcup_{\tau \in \mathcal{T}} \mathcal{G}_\tau$. Poisoning these entities distorts the backbone of the reasoning graph, affecting a broad spectrum of retrieval operations.

4.2 Query-Centric Logic Poison

Global hubs may not cover long-tail entities critical for specific queries. To enforce path coverage, we introduce a query-guided selection mechanism.

Chain-of-Thought Entity Extraction. For each multi-hop query $q \in \mathcal{Q}$, we prompt an LLM to perform Chain-of-Thought (CoT) reasoning to identify entities essential for deriving the answer (e.g., bridge entities, comparison targets). This yields a structured set of reasoning entities:

$$E_q = \{(e, \tau, r)\}, \quad (4)$$

where r denotes the reasoning role (e.g., `bridge`, `target`). This ensures we target logical dependencies of the query rather than superficial mentions.

Corpus Verification. To prevent the hallucination of non-existent targets, we perform a filtering step on E_q against the corpus inventory:

$$E_q^{\text{corpus}} = \{(e, \tau) \mid (e, \tau, r) \in E_q \wedge f(e, \tau) > 0\}. \quad (5)$$

This ensures that the attack only modifies entities that actually exist in $\mathcal{C}$ and participate in the knowledge graph construction pipeline.

Query-Centric Replacement Pool. We aggregate these verified entities across all queries to form the query-centric replacement pool:

$$\mathcal{Q}_\tau = \bigcup_{q \in \mathcal{Q}} \{e \mid (e, \tau) \in E_q^{\text{corpus}}\}, \quad \mathcal{Q} = \bigcup_{\tau \in \mathcal{T}} \mathcal{Q}_\tau. \quad (6)$$

This design guarantees that if a GraphRAG system correctly follows the intended reasoning chain, it will inevitably traverse a poisoned node.

4.3 Type-Preserving Entity Swapping

Unified Candidate Set. We combine the global and query-centric pools into a unified set of target entities, which play complementary roles:

$$\mathcal{R}_\tau = \mathcal{G}_\tau \cup \mathcal{Q}_\tau, \quad \mathcal{R} = \bigcup_{\tau \in \mathcal{T}} \mathcal{R}_\tau. \quad (7)$$

Crucially, we maintain strict type constraints during swapping to ensure the poisoned text remains locally plausible and grammatically valid, thereby bypassing syntax-based filters.

Dataset	LLM	Attack	Naive RAG		GraphRAG		GFM-RAG		HippoRAG 2	
			ASR	ASR-G	ASR	ASR-G	ASR	ASR-G	ASR	ASR-G
HotpotQA	GPT-4o-mini	PoisonedRAG	61.8	72.8	66.8	80.0	71.6	70.4	68.0	66.0
		LogicPoison	**92.0**	**91.6**	**78.4**	**92.2**	**81.0**	**78.0**	**73.6**	**66.2**
	Llama-3.1-8B	PoisonedRAG	51.0	58.8	64.4	78.0	55.8	49.4	68.4	63.6
		LogicPoison	**88.2**	**86.6**	**79.2**	**91.2**	**80.6**	**83.8**	**72.6**	**73.0**
	Qwen-3-32B	PoisonedRAG	56.8	68.2	65.8	80.2	71.6	35.0	71.8	67.6
		LogicPoison	**85.0**	**83.8**	**84.2**	**92.0**	**76.6**	**76.8**	**76.6**	**69.8**
2Wiki	GPT-4o-mini	PoisonedRAG	64.4	83.4	58.6	77.4	57.2	56.6	74.8	70.4
		LogicPoison	**90.0**	**91.6**	**78.4**	**95.6**	**71.6**	**76.0**	**82.8**	**71.8**
	Llama-3.1-8B	PoisonedRAG	61.4	77.8	54.8	73.2	42.2	37.2	74.8	70.0
		LogicPoison	**95.4**	**94.4**	**78.8**	**95.0**	**74.2**	**85.6**	**77.4**	**74.2**
	Qwen-3-32B	PoisonedRAG	63.0	82.4	58.0	77.0	58.8	37.6	75.8	73.6
		LogicPoison	**82.2**	**83.4**	**78.6**	**93.2**	**71.4**	**79.8**	**87.4**	**76.6**
MuSi	GPT-4o-mini	PoisonedRAG	64.0	77.8	59.6	77.6	70.0	74.8	67.2	68.0
		LogicPoison	**99.2**	**99.2**	**91.4**	**97.0**	**92.6**	**93.4**	**90.2**	**88.0**
	Llama-3.1-8B	PoisonedRAG	58.2	64.4	60.2	75.4	43.0	42.6	59.4	57.8
		LogicPoison	**97.4**	**97.4**	**91.6**	**97.6**	**95.0**	**98.2**	**88.8**	**92.2**
	Qwen-3-32B	PoisonedRAG	66.2	80.0	61.2	77.6	59.2	62.0	67.4	67.0
		LogicPoison	**94.6**	**95.8**	**93.0**	**95.8**	**87.6**	**92.6**	**91.2**	**91.6**

Table 1: Attack Performance (ASR and ASR-G) of LOGICPOISON and PoisonedRAG across different RAG frameworks, Datasets, and LLMs.

Cyclic Permutation. For each entity type τ, we organize the entities in $\mathcal{R}_\tau$ into a deterministic sequence $\mathbf{S}_\tau = [e_1, e_2, \ldots, e_m]$. We define a cyclic permutation function π_τ:

$$\pi_\tau(e_i) = \begin{cases} e_{i-1} & \text{if } 1 < i \leq m, \\ e_m & \text{if } i = 1. \end{cases} \tag{8}$$

This mapping ensures that (i) every target entity is replaced by another entity of the same type, and (ii) the mapping is bijective and invertible.

Corpus Rewriting and Implicit Graph Corruption. Finally, we apply the perturbation function to the entire corpus. At the surface form level, any mention of $e \in \mathcal{R}_\tau$ for each document d_i is substituted with $\pi_\tau(e)$. We explicitly do not modify the ground-truth answers or queries. Consequently, while the corpus $\tilde{\mathcal{C}}$ remains readable, the logical links supporting the ground truth are severed. When a GraphRAG system builds its index on $\tilde{\mathcal{C}}$ and performs chunking, embedding, and graph extraction, it implicitly constructs a corrupted topology where valid reasoning paths are rerouted to incorrect entities, misleading the reasoning process.

5 Experiments

5.1 Experimental Setting

Dataset. To evaluate the efficacy of LOGICPOISON in real scenarios, we align our experimental setup with established protocols in SOTA GraphRAG research (Gutiérrez et al., 2024, 2025). We employ three widely adopted multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). These datasets represent a wide range of reasoning complexity, ranging from moderate to highly challenging tasks, and require various graph-based reasoning abilities such as bridging entities and comparative analysis. Following standard practices, we randomly sample 500 queries from the validation set of each benchmark for evaluation, while strictly align the corpus and ground-truth defined in previous work.

Baseline. We benchmark LOGICPOISON against PoisonedRAG, the state-of-the-art attack for RAG systems, as well as widely adopted adversarial methods for LLMs, including Naive Attack, Prompt Injection (Liu et al., 2024), GCG Attack (Zou et al., 2023), and Disinformation (Du et al., 2022). The details of all baselines are pro-

vided in Appendix A.5.

GraphRAG methods. To verify the robustness and universality of our attack across diverse architectures, we conduct evaluations on three representative GraphRAG frameworks: (1) Microsoft GraphRAG (Edge et al., 2025), the pioneering implementation that popularized graph-based global retrieval; and (2) HippoRAG 2(Gutiérrez et al., 2025) and GFM-RAG (Luo et al., 2025), which represent the current SOTA in the field. This selection allows us to assess vulnerabilities across different graph construction and reasoning paradigms. Details of each method are provided in the Appendix.

Metrics. Consistent with prior work (Zou et al., 2025), we report the Attack Success Rate (ASR) based on substring inclusion. An attack is deemed successful if the ground truth string is absent from the model's output. Recognizing that rigid string matching fails to capture semantic nuances (e.g., synonyms or structural changes), we additionally propose ASR-G. This metric leverages an LLM-as-a-judge paradigm to assess semantic equivalence. Given the triplet of (question, ground truth, prediction), the judge evaluates if the prediction conveys the same meaning as the ground truth.

Implementation Details. All experiments were executed on the GeForce RTX 5090. For all methods, we use Facebook/contriever as the embedding model. For the retrieval top-k parameters across different RAG methods, we set k = 10. For the top-$n\%$ of targeted entities, we set $n\% = 5\%$. We use three different LLMs such as GPT-4o-mini (OpenAI et al., 2024), Llama-3.1-8B (Meta, 2024), and Qwen-3-32B (Yang et al., 2025) as base LLMs to verify the generality of the experimental results.

5.2 Main Results

Table 1 summarizes the attack performance. LOGICPOISON achieves SOTA ASR and ASR-G across all evaluated scenarios, significantly outperforming PoisonedRAG. The advantage is most evident in graph-centric systems like GraphRAG and GFM-RAG, where our topological rewiring strategy effectively dismantles the logical connectivity essential for retrieval. While HippoRAG 2 shows relative resilience due to its semantic safeguards, our method still achieves a higher success rate. Moreover, in scenarios requiring deep reasoning (MuSiQue), LOGICPOISON demonstrates dominant performance, verifying that logical corruption is more potent than content injection for multi-hop tasks. Surprisingly, this advantage persists even in Naive RAG, indicating the broad applicability of entity-level logic poisoning.

LLM	Attack Method	GraphRAG		GFM-RAG	
		ASR	ASR-G	ASR	ASR-G
GPT-4o-mini	Naive Attack	18.8	15.6	18.8	10.0
	Prompt Injection	58.8	80.4	38.2	47.6
	GCG Attack	19.6	21.6	19.2	9.6
	Disinformation	54.6	75.0	42.0	51.4
	LogicPoison	**78.4**	**95.6**	**71.6**	**76.0**
Llama-3.1-8B	Naive Attack	18.0	21.4	20.8	13.6
	Prompt Injection	57.0	78.2	30.4	32.6
	GCG Attack	19.8	16.8	20.4	12.4
	Disinformation	53.6	70.4	42.4	43.2
	LogicPoison	**78.8**	**95.0**	**74.2**	**85.6**
Qwen-3-32B	Naive Attack	19.0	17.2	19.4	7.2
	Prompt Injection	59.0	83.4	50.0	64.0
	GCG Attack	19.6	20.0	19.2	7.6
	Disinformation	54.4	74.0	50.4	52.8
	LogicPoison	**78.6**	**93.2**	**71.4**	**79.8**

Table 2: Attack Performance of LOGICPOISON and LLM attack methods on the 2Wiki Dataset across different LLMs and RAG frameworks.

We further compare the effect difference between LOGICPOISON and the traditional LLM attack method. Table 2 shows that LOGICPOISON shows significant advantages in different LLM models and GraphRAG methods. Specifically, Naive Attack and GCG Attack have poor performance. The core reason is that Naive texts and text suffixes are easy to be filtered in the graph construction stage. Prompt Injection has a certain attack effect. The triple extraction process from GraphRAG usually relies on LLM, which may perform wrong triple extraction operations under the influence of injection prompt words. Disinformation has the possibility of being included in the Knowledge Graph by generating false information highly relevant to the query. However, on the whole, these traditional methods are far less effective than LOGICPOISON, because they focus on unstructured text injection which is not designed for the topology destruction of KG. In addition, we also find that GFM-RAG shows a stronger defense ability against various attacks than Microsoft GraphRAG, which is because GFM regards KG as an intermediate state requiring inference and denoising, and its performance depends more on the pre-trained GNN reasoning module.

5.3 Efficiency Analysis

In order to compare the efficiency of LOGICPOISON and PoisonedRAG, we evaluate it from three

Method	Time Cost(s)	Token Cost	Injected Tokens
PoisonedRAG	6607.3	593.6	296813
w/ Global	125.0	0	0
w/ Query-Centric	1281.4	74.9	0
LOGICPOISON	**1406.4**	**74.9**	**0**

Table 3: **Efficiency Analysis.** Comparison of computational time, token cost per query, and the length of injected content in the corpus between baselines and our variants. All results are averaged over the three datasets.

dimensions of time consumption, Token consumption and injection Token on three datasets. The results show that LOGICPOISON significantly outperforms the baseline method in all dimensions. Specifically, in terms of time consumption, PoisonedRAG relies heavily on the large-scale generation of LLM, which leads to its slow speed. The global phase of LOGICPOISON uses lightweight NER technology, which takes very low time. Although the query-centric part needs to extract bridge entities through LLM and consumes a certain amount of time, the overall time is still 4 times faster than PoisonedRAG. In terms of Token consumption, LOGICPOISON consumes only 1/8 of PoisonedRAG, which significantly reduces the resource overhead. In the dimension of injection Tokens, PoisonedRAG inserts massive attack-related content into a single dataset, which makes it easy to be detected. LOGICPOISON eliminates the need to inject spurious content into the dataset, thus achieving high concealment.

5.4 Ablation Study

To verify the effectiveness of each module, we conducted ablation experiments on three different LLM models and two GraphRAG methods. The experimental results, as shown in Figure 3, demonstrate that each module of LOGICPOISON exhibits significant effectiveness. Specifically, the Query-centric Logic Poison module identifies key entities for each query, effectively cutting off its logical reasoning chain and contributing to a significant attack effect; the Global Logic Poison module, by weakening the Logic Hubs in the Knowledge Graph, severely undermines the overall logicality of the graph which does not rely on LLM or query content, making it the preferred solution in scenarios with low costs and low data availability. The combination of the two forms LOGICPOISON, which effectively integrates the aforementioned advantages: it not only disrupts the topological connections of the KG at a global level but also specifically interrupts the exclusive logical reasoning chain of each query, achieving the optimal attack effect.

5.5 Impact on the Logic Chains

To quantitatively verify the attack's impact on the graph's topological integrity, we measured the disruption of actual logic chains. For each multi-hop query, the ground-truth logic chain is composed of a sequence of interdependent entities. We define the Logic-Chain-Hit Rate as the percentage of queries where the GraphRAG system successfully retrieves corresponding entities required to form the complete logic chain. We conducted this topological evaluation across three public datasets, comparing the hit rates on the clean knowledge graph versus the poisoned graph:

Method	Logic-Chain-Hit Rate	Drop (↓)
GFM-RAG	80.30%	–
Attacked GFM-RAG	**4.89%**	**75.41%**
HippoRAG 2	87.13%	–
Attacked HippoRAG 2	**2.59%**	**84.54%**

Table 4: Impact of Attacks on Logic-Chain-Hit Rate (when there are more than two entities in logic chain).

As the results in Table 4 demonstrate, LOGICPOISON causes a sharp decline in the Logic-Chain-Hit Rate. For instance, HippoRAG 2 experiences an 84.54% absolute drop in its ability to retrieve correct logic chains. This quantitatively proves our core claim: LOGICPOISON does not merely introduce surface-level noise, but systematically severs the critical topological connections and reasoning bridges within the knowledge base.

5.6 Potential Defenses

Previous experiments have fully demonstrated the effectiveness of LOGICPOISON in attacking GraphRAG. In the following section, this paper will further explore potential defense mechanisms.

5.6.1 Query Paraphrasing

Paraphrasing (Jain et al., 2023) is widely used to defend against prompt injection attacks (Liu et al., 2024, 2025; Pedro et al., 2025; Branch et al., 2022) and jailbreak attacks (Wei et al., 2023; Deng et al., 2024a) by large language models. Its core mechanism involves rewriting the query by LLM and then performing retrieval and generation tasks based on the rewritten query. We verified the defense effect of this strategy against LOGICPOISON, generating

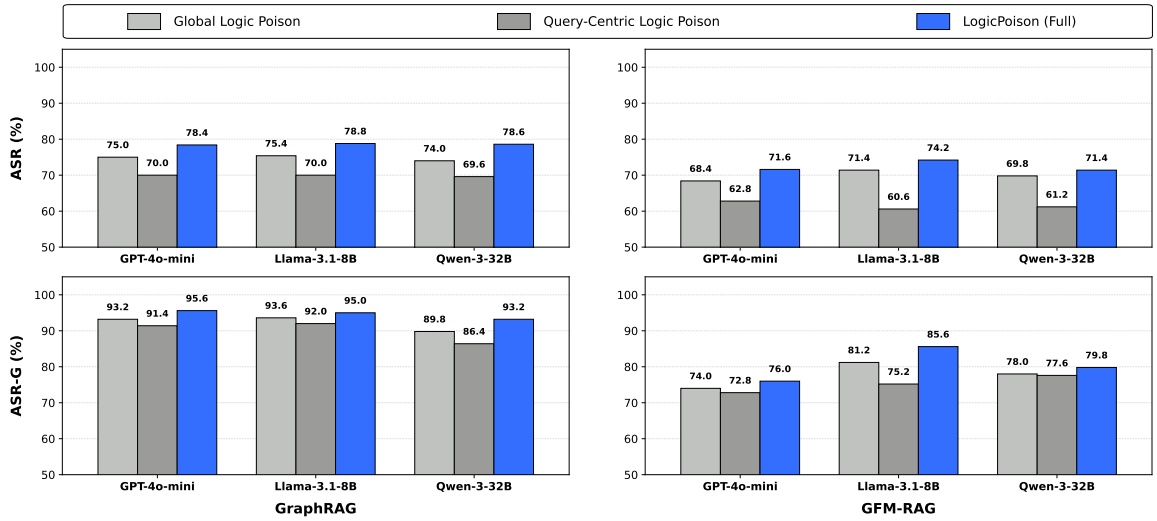

Figure 3: Ablation study of the components on the 2Wiki Dataset. We compare Global-only, Query-Centric-only, and the full **LogicPoison** strategies with three different LLMs.

Dataset	System	Standard (No Def.)	w/ Paraphrasing (Defense)	Δ
HotpotQA	GraphRAG	92.2	92.2	0
	GFM-RAG	78.0	77.8	-0.2
2Wiki	GraphRAG	95.6	94.6	-1.0
	GFM-RAG	76.0	75.6	-0.4
MuSiQue	GraphRAG	97.0	96.2	-0.8
	GFM-RAG	93.4	93.4	0

Table 5: **Defense Analysis.** Robustness of LOGICPOISON against Query Paraphrasing defense across different datasets and GraphRAG systems. The LLM backbone is fixed to GPT-4o-mini.

5 rewritten versions for each query. The experimental results are shown in Table 5. The specific results indicate that Paraphrasing has almost no defense effect against LOGICPOISON. The decline in attack effectiveness caused by this strategy is always less than 1.0%. The main reason is that regardless of how the sentence structure of the query is rewritten, its core semantics remain unchanged, which will not affect the extraction results of key entities by LOGICPOISON, and thus has almost no impact on the overall attack effect.

5.6.2 LLM Knowledge Referencing

For the scenario of using GraphRAG, LLM itself cannot solve professional questions well, and a specific knowledge base/KG is needed as the knowledge base. At the same time, the prompt of various GraphRAG will strictly limit the LLM to refer to the content retrieved from the knowledge base/KG during generation. These constraints greatly limit the defense effect of LLM Knowledge Referencing (relying on LLM internal knowledge) (Liang et al., 2025) against LOGICPOISON.

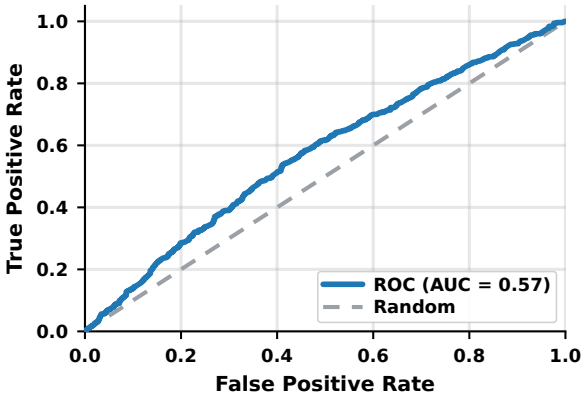

Figure 4: PPL-based detection for LOGICPOISON.

5.6.3 Poisoning Text Identification

Perplexity (PPL) (Alon and Kamfonas, 2023) is widely used to measure text quality and also to defend against attacks on large language models. Previous studies have shown that a high perplexity of text indicates low quality, and poisoned texts often have a higher perplexity than texts written by humans, which makes poisoned texts very easy to detect. To verify this, following (Zou et al., 2025), we calculated the perplexity of clean texts and texts with entities replaced by LOGICPOISON (randomly sample 500 of each), respectively, using OpenAI's `tiktoken cl100k_base model` for detection. As shown in Figure 4, the results show that the AUC value is 0.57, which is very close to the random guess benchmark of 0.5, indicating that the model is basically in a state of random judgment on poi-

soned texts. This means that the poisoned texts generated by LOGICPOISON after entity replacement are almost indistinguishable from clean texts in terms of perplexity, making detection difficult. The core reason lies in the Type-Preserving Entity Swapping module proposed in this paper: all entity replacements maintain the consistency of types, ensuring that the fluency and semantic expression of sentences are not affected, thereby achieving effective evasion of detection mechanisms.

6 Conclusion

We found that existing LLM or RAG attack techniques cannot effectively attack GraphRAG systems. Through in-depth analysis of the GraphRAG system, we discovered that it is vulnerable to logic destruction attacks, so we proposed the LOGICPOISON. This method constructs a unified candidate set by identifying logic hubs and key entities corresponding to queries. Based on cyclic permutation, we implement a comprehensive and covert attack on GraphRAG from both global and query-centric levels, which destroys its logical reasoning ability. Under the experimental settings of three public datasets, three LLM models and three GraphRAG methods, LOGICPOISON shows excellent performance, and is significantly better than the SOTA in time and token cost. In addition, we explore and experimentally verify the effectiveness of potential defenses. This study provides important guidance for the security research in the GraphRAG field.

Limitations

While LOGICPOISON effectively exposes the topological vulnerabilities of GraphRAG and demonstrates high efficacy across multiple benchmarks, we acknowledge certain limitations that point to directions for future research.

Generalization to Dynamic Graph Construction. Our experiments focus on graph indices built from static corpora. Real-world GraphRAG systems may employ dynamic updating mechanisms where nodes are added or pruned in real-time. While the theoretical basis of logical poisoning remains valid, the persistence and stability of the attack under high-frequency graph updates warrant further empirical investigation.

Linguistic Scope. The current evaluation is confined to English-language datasets. Although logical reasoning is language-agnostic, implementing imperceptible entity swapping in languages with complex morphology or rich inflection requires tailored rewriting heuristics. We plan to extend our validation to multilingual GraphRAG settings in future work.

Ethics Statement

This work complies with the ACL Ethics Policy. We utilize three publicly available datasets: HotpotQA, 2WikiMultiHopQA, and MuSiQue, which are widely established benchmarks in the research community. To the best of our knowledge, these datasets are free of personally identifiable information (PII) or offensive content. Our study involves neither human subject experimentation nor the collection of private user data.

We acknowledge that the LOGICPOISON framework proposed in this paper demonstrates a method to compromise GraphRAG systems, which carries a potential risk of misuse. However, our primary motivation is defensive: by exposing the fragility of topological reasoning in GraphRAG, we aim to alert the community to these stealthy vulnerabilities. We believe that identifying such structural weaknesses is a prerequisite for developing more robust defense mechanisms and secure RAG systems. All experiments were conducted in a controlled environment, and no poisoned data was released into real-world applications.

Acknowledgements

The work described in this paper was substantially supported by a grant from the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (Project No. ITS/263/24FP), National Natural Science Foundation of China (No.62506308) and Chengdu Science and Technology Program (No.2025-YF12-00030-RC). We also thank the reviewers for their insightful comments.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *Preprint*, arXiv:2308.14132.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Hezekiah J. Branch, Jonathan Rodriguez Cefalu, Jeremy McHugh, Leyla Hujer, Aditya Bahl, Daniel del

Castillo Iglesias, Ron Heichman, and Ramesh Darwishi. 2022. Evaluating the susceptibility of pre-trained language models via handcrafted adversarial examples. *Preprint*, arXiv:2209.02128.

Pengzhou Cheng, Yidong Ding, Tianjie Ju, Zongru Wu, Wei Du, Ping Yi, Zhuosheng Zhang, and Gongshen Liu. 2024. Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models. *Preprint*, arXiv:2405.13401.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2024a. Masterkey: Automated jailbreaking of large language model chatbots. In *Proc. ISOC NDSS*.

Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, and Yang Liu. 2024b. Pandora: Jailbreak gpts by retrieval augmented generation poisoning. *Preprint*, arXiv:2402.08416.

Yibing Du, Antoine Bosselut, and Christopher D. Manning. 2022. Synthetic disinformation attacks on automated fact verification systems. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. *Preprint*, arXiv:2403.14720.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *Preprint*, arXiv:2309.00614.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jiacheng Liang, Yuhui Wang, Changjiang Li, Rongyi Zhu, Tanqiu Jiang, Neil Gong, and Ting Wang. 2025. Graphrag under fire. *Preprint*, arXiv:2501.14050.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, Leo Yu Zhang, and Yang Liu. 2025. Prompt injection attack against llm-integrated applications. *Preprint*, arXiv:2306.05499.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and benchmarking prompt injection attacks and defenses. In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, USA. USENIX Association.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: Graph foundation model for retrieval augmented generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Rodrigo Pedro, Daniel Castro, Paulo Carreira, and Nuno Santos. 2025. From prompt injections to sql injection attacks: How protected is your llm-integrated web application? *Preprint*, arXiv:2308.01990.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang

Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Avital Shafran, Roei Schuster, and Vitaly Shmatikov. 2025. Machine against the rag: jamming retrieval-augmented generation with blocker documents. In *Proceedings of the 34th USENIX Conference on Security Symposium*, SEC '25, USA. USENIX Association.

Daniel Spielman. 2012. Spectral graph theory. *Combinatorial scientific computing*, 18(18).

Xuchen Suo. 2024. Signed-prompt: A new approach to prevent prompt injection attacks against llm-integrated applications. *AIP Conference Proceedings*, 3194(1):040013.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. Graphrag-bench: Challenging domain-specific reasoning for evaluating graph retrieval-augmented generation. *Preprint*, arXiv:2506.02404.

Yilin Xiao, Chuang Zhou, Qinggang Zhang, Bo Li, Qing Li, and Xiao Huang. 2026a. Reliable reasoning path: Distilling effective guidance for llm reasoning with knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 38(4):2380–2393.

Yilin Xiao, Chuang Zhou, Yujing Zhang, Qinggang Zhang, Su Dong, Shengyuan Chen, Chang Yang, and Xiao Huang. 2026b. Lag: Logic-augmented generation from a cartesian perspective. *Preprint*, arXiv:2508.05509.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. 2026. Graph-based agent memory: Taxonomy, techniques, and applications. *Preprint*, arXiv:2602.05665.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4505–4524, Bangkok, Thailand. Association for Computational Linguistics.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3827–3844.

A Appendix

A.1 Analysis of the graph-based architecture

Unlike traditional RAG, which relies on similarity in vector space to retrieve static text chunks, GraphRAG fundamentally operates as a topological reasoning engine. Its core advantage does not lie in merely accumulating knowledge triples, but in being able to perform transitive reasoning between these triples.

Specifically, GraphRAG systems synthesize answers using graph traversal algorithms (e.g., multi-hop pathfinding) and structural aggregation mechanisms (e.g., community detection via the Leiden algorithm). In this architecture, the value of a piece of information depends on its connectivity. For example, answering a complex query often requires traversing specific inferential bridges. If a semantic bridge is topologically disconnected or routed incorrectly, the system cannot connect premises to conclusions even if both entities are intact in the database.

Thus, the robustness of GraphRAG depends on structural integrity rather than data volume. This architectural dependency exposes a critical vulnerability: logic is fragile. While vector retrieval degrades moderately under noise (retrieved text chunks are slightly less relevant), GraphRAG's inference chains are prone to catastrophic failures when critical hub nodes or bridge edges are damaged. Replacing a key entity in the inference chain not only introduces factual errors but also disrupts the entire reasoning process, preventing the retrieval mechanism from identifying the correct context. This insight shifts the attack target from injecting large amounts of false information (flooding attacks) to subtly reconstructing logic (reconnection attacks), which also prompts us to propose the LOGICPOISON framework.

A.2 Theoretical Analysis of logic hub

A.2.1 Hub Nodes and Connectivity Disruption

Real-world graphs, including social networks, information networks, and knowledge graphs constructed from large-scale corpora, often exhibit scale-free properties. In such networks, node degrees follow a heavy-tailed distribution rather than a homogeneous distribution. Formally, the probability that a node has degree k is given by:

$$P(k) \propto k^{-\gamma}, \quad \gamma \in (2, 3), \tag{9}$$

where a small fraction of nodes, referred to as hub nodes, possess dominantly high degrees, while most of nodes maintain sparse connectivity.

A fundamental result in network science is that scale-free networks are robust to random failures but highly vulnerable to targeted attacks. Random node removal predominantly affects low-degree nodes and therefore preserves the giant connected component. In contrast, selectively removing or perturbing hub nodes can drastically impair global connectivity. Besides, hub nodes play a central role in shortest-path routing and information flow. Their removal significantly increases the average shortest path length and graph diameter, thereby degrading multi-hop information propagation. Prior theoretical and empirical studies have shown that eliminating only a small fraction of the highest-degree nodes can lead to disproportionate degradation of global network efficiency.

This behavior is commonly analyzed through percolation theory, where the existence of a giant connected component depends on the ratio:

$$\kappa = \frac{\langle k^2 \rangle}{\langle k \rangle}. \tag{10}$$

For scale-free networks with $\gamma \leq 3$, the second moment $\langle k^2 \rangle$ diverges, resulting in a vanishing percolation threshold under random failures. However, targeted attacks on hub nodes sharply reduce $\langle k^2 \rangle$, driving the network below the critical threshold and causing abrupt fragmentation.

A.2.2 Node Centrality Metrics

To quantitatively characterize the importance of entities in a graph, we adopt classical centrality measures from network science that capture complementary aspects of node influence. Degree centrality measures the number of direct connections of a node, formally defined as $C_D(v) = \deg(v)$, reflecting its local prominence and immediate aggregation capacity; in GraphRAG-style entity graphs, nodes with high degree centrality typically correspond to frequently mentioned or semantically generic entities that serve as global connectors. In contrast, betweenness centrality quantifies the extent to which a node lies on shortest paths between other nodes, defined as $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} denotes the number of shortest paths between nodes s and t and $\sigma_{st}(v)$ those passing through v; such nodes act as structural bridges that mediate information flow and multi-hop reasoning. Finally, closeness centrality, defined as

$C_C(v) = \left(\sum_{u \neq v} d(v,u)\right)^{-1}$, captures how close a node is to all others in terms of shortest-path distance, reflecting its ability to rapidly access or disseminate information across the graph. Together, these metrics provide a quantitative basis for identifying nodes that are critical to global connectivity, efficient information propagation, and reasoning path composition.

A.2.3 Spectral View of Frequent Entities

From the perspective of spectral graph theory, the global connectivity and information propagation properties of a graph are governed by the spectrum of its adjacency or Laplacian matrix. Let $G = (V, E)$ denote the entity graph induced from a corpus, and let $A \in \mathbb{R}^{|V| \times |V|}$ be its adjacency matrix. The largest eigenvalue $\lambda_{\max}(A)$ and the associated leading eigenvector play a dominant role in characterizing diffusion dynamics, aggregation behavior, and global structural coherence.

In heterogeneous graphs, particularly those with scale-free characteristics, the leading eigenvalue is disproportionately influenced by high-degree nodes. Classical results in spectral graph theory indicate that the spectral radius scales with the square root of the maximum degree:

$$\lambda_{\max}(A) \approx \max_{v \in V} \sqrt{\deg(v)}, \tag{11}$$

suggesting that nodes with large degrees exert a dominant influence on the spectral radius. Moreover, the mass of the leading eigenvector tends to localize around hub nodes, a phenomenon known as eigenvector localization, implying that these nodes dominate global signal propagation.

In corpus-induced entity graphs, explicit degree information is often unavailable prior to graph construction. However, under standard co-occurrence-based graph formation, the expected degree of an entity is monotonically related to its corpus-level frequency. Let $f(e)$ denote the document frequency of entity e.

$$\mathbb{E}[\deg(e)] \propto f(e), \tag{12}$$

Consequently, high-frequency entities are expected to dominate the eigenspace of the induced graph, which motivates the use of corpus-level frequency as a practical substitute for spectral dominance. Attacking or perturbing such entities induces a low-rank perturbation to the adjacency matrix. Let $\tilde{A} = A + \Delta A$ denote the perturbed adjacency matrix after poisoning a small set of high-frequency entities. The First-order matrix perturbation theory yields the following.

$$\Delta\lambda_{\max} \approx \mathbf{u}^\top \Delta A \mathbf{u}, \tag{13}$$

where $\mathbf{u}$ is the leading eigenvector of A. Since $\mathbf{u}$ concentrates its mass on hub nodes, perturbations targeting these nodes produce disproportionately large shifts in $\lambda_{\max}$. Such spectral shifts degrade global diffusion efficiency and destroy multi-hop information aggregation in an efficient manner.

Taken together, these results imply that poisoning high-frequency entities constitutes a spectrally efficient attack strategy: by targeting a small number of entities that dominate the leading eigenspace, the attack induces global structural degradation with minimal perturbation budget. This provides a theoretical explanation for why corpus-level frequency-based attacks can effectively disrupt reasoning in GraphRAG systems without requiring explicit access to the full graph topology.

A.3 Algorithm Workflow

Algorithm 1 details the execution pipeline of the LOGICPOISON framework. The process operates in three phases: (1) Target Selection, where global logic hubs and query-centric bridges are identified; (2) Permutation Construction, where a type-preserving cyclic mapping is defined for the unified target set; and (3) Corpus Rewriting, which generates poisoned corpus $\tilde{\mathcal{C}}$ to induce graph corruption.

A.4 Details of GraphRAG methods

Here we introduce the GraphRAG methods chosen in this paper in detail:

GraphRAG. Microsoft GraphRAG proposes a hierarchical community summarization strategy based on knowledge graph. This method constructs a knowledge graph by extracting entities and relations through LLM, and then generates a structured summary through hierarchical community detection. Finally, map-reduce mode is used to integrate partial responses of each community to generate a global answer. The innovation of this method is that it combines the modularity of graph with summary generation, and supports global queries on millions of token corpora. It is significantly better than the traditional vector RAG in the comprehensiveness and diversity of answers, and provides an efficient solution for the global sense-making task of large-scale private corpora.

Algorithm 1 LOGICPOISON Attack Workflow

Input: Clean Corpus $\mathcal{C}$, Query Set $\mathcal{Q}$, Entity Types $\mathcal{T}$, Poison Budget $n\%$.
Output: Poisoned Corpus $\tilde{\mathcal{C}}$.

Initialize target pools: $\mathcal{G} \leftarrow \emptyset$, $\mathcal{Q}_{pool} \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$;
{**Phase 1: Global Logic Poisoning**}
for document $d_i \in \mathcal{C}$ **do**
  Extract typed entities $E(d_i)$ using NER;
  Update frequency counts $f(e, \tau)$;
end for
for type $\tau \in \mathcal{T}$ **do**
  Select top-$n\%$ frequent entities as $\mathcal{G}_\tau$;
end for
$\mathcal{G} \leftarrow \bigcup_\tau \mathcal{G}_\tau$;
{**Phase 2: Query-Centric Logic Poisoning**}
for query $q_j \in \mathcal{Q}$ **do**
  Extract reasoning chain entities E_{q_j} via LLM CoT;
  Filter E_{q_j} against corpus inventory to get valid set $E_{q_j}^{\text{corpus}}$;
  $\mathcal{Q}_{pool} \leftarrow \mathcal{Q}_{pool} \cup E_{q_j}^{\text{corpus}}$;
end for
{**Phase 3: Unified Swapping Mechanism**}
$\mathcal{R} \leftarrow \mathcal{G} \cup \mathcal{Q}_{pool}$; {Combine global and query-specific targets}
for type $\tau \in \mathcal{T}$ **do**
  Extract subset $\mathcal{R}_\tau$ from $\mathcal{R}$ where type is τ;
  Sort $\mathcal{R}_\tau$ to form sequence $\mathbf{S}_\tau$;
  Construct cyclic permutation function π_τ: $\mathbf{S}_\tau[i] \rightarrow \mathbf{S}_\tau[i-1]$;
end for
{**Phase 4: Corpus Rewriting**}
$\tilde{\mathcal{C}} \leftarrow \emptyset$;
for document $d_i \in \mathcal{C}$ **do**
  $\tilde{d}_i \leftarrow d_i$;
  for entity $e \in \mathcal{R}$ **do**
    Replace all mentions of e in $\tilde{d}_i$ with $\pi_{\text{type}(e)}(e)$;
  end for
  $\tilde{\mathcal{C}}$.append($\tilde{d}_i$);
end for
return Poisoned Corpus $\tilde{\mathcal{C}}$.

GFM-RAG. In order to solve the problems of noisy graph structure, incompleteness and lack of model generality in GraphRAG, GFM-RAG proposes the first Graph-based model (GFM) suitable for retrieval enhancement generation. The model takes query dependency graph neural network as the core, and through two-stage training (unsupervised knowledge graph completion pre-training and supervised document retrieval fine-tuning), it learns general reasoning patterns on large-scale data containing 60 knowledge graphs and 700k documents, and can directly adapt to unseen data sets without fine-tuning. Its innovative single-step multi-hop inference mechanism significantly improves the performance of complex reasoning tasks while ensuring efficiency, and conforms to the neural scaling law, which provides a basis for subsequent model expansion.

HippoRAG 2. Aiming at the problem of performance degradation of existing structure enhanced RAG methods in fact memory tasks, HippoRAG 2 is based on a neurobiologically inspired long-term memory framework and inherits the core of HippoRAG's personalized PageRank algorithm. Through the optimization of deep paragraph integration, deep association between query and knowledge graph triples, and LLM online identification of irrelevant triples, the problem of context loss and semantic matching caused by the entity-centric method is solved.

We have carefully selected these three most representative GraphRAG methods as part of the experiment. Due to the huge resource consumption of GraphRAG methods, we were unable to include more other excellent GraphRAG methods. In future, we will consider expanding the scope of the experiments.

A.5 Details of Baselines

Here we introduce the baselines chosen in this paper in detail:

PoisonedRAG. PoisonedRAG is the first knowledge pollution attack framework against RAG. This work reveals for the first time that the knowledge base of RAG systems can be used as a practical attack surface to induce LLMs to generate incorrect answers preset by attackers for specific target questions by injecting a small amount of malicious text. Its core innovation lies in deriving and satisfying two necessary conditions for the attack: retrieval condition (malicious text can be retrieved by the

target question) and generation condition (malicious text can mislead the LLM to output the target answer).

Naive Attack. Following the baseline (Zou et al., 2025), we define Naive Attack as: Given a question, treat the question itself as malicious text because it has a high probability of being retrieved.

Prompt Injection. Prompt Injection (Liu et al., 2024) is a kind of security attack against Large Language Model integration applications. The core of prompt injection is that the attacker can inject malicious instructions or data by manipulating the external input data of the application, and induce the LLM to ignore the original task instructions and execute the injection task preset by the attacker, so as to produce the attacker's expected results.

GCG Attack. By combining greedy search and gradient-directed discrete optimization strategies, GCG Attack (Zou et al., 2023) automatically generates adversarial suffixed, which can efficiently induce aligned LLMs to output harmful content. The core innovations of this method are as follows: first, the model is guided into harmful generation mode by optimizing the target text prefix without manual intervention; Second, it supports multi-hint and multi-model joint optimization, and the generated adversarial suffix has strong versatility and transferability, which can successfully attack GPT-4, Bard and other black-box commercial models and open source LLMs.

Disinformation. PoisonedRAG generates the basis content of the attack text through the prompt designed for the target problem, which can be regarded as disinformation (Du et al., 2022). Therefore, we compare with this baseline, which can be seen as a variant of PoisonedRAG.

A.6 Hyperparameter analysis

LOGICPOISON contains only one hyperparameter, which is the top-n% in the Global Logic Poison module. To select the smallest value that balances effectiveness and efficiency, we conducted a hyperparameter sensitivity analysis based on Naive RAG (since the cost of repeating experiments multiple times on the GraphRAG method is extremely high) under the setting of retrieval top-k=5. All Attack Success Rate results are the average of 5 independent tests on 3 datasets. The experimental results show that when n% exceeds 5%, the improvement in ASR tends to level off. Therefore, we finally chose top-n%=5% as the hyperparameter, which achieves the optimal balance between attack effect and resource cost.

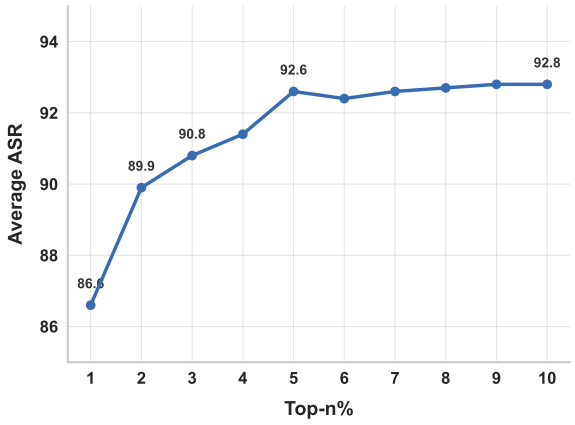

Figure 5: Hyperparameter analysis results of LOGICPOISON.

A.7 Qualitative analysis of the impact on the logic chains

We conducted a case study on the real question in the dataset. Taking the multi-hop query "What is the place of birth of the director of film Gaby: A True Story?" as an example, through a comparison of specific logic chains, we intuitively demonstrate the destructive effect of LogicPoison on the reasoning path of GraphRAG:

- Valid logical chain in the clean knowledge graph In an unattacked knowledge graph, GraphRAG can construct complete multi-hop reasoning chains through topological associations: Gaby: A True Story → (directed by) → Luis Mandoki → (born in) → Mexico City. This logical chain relies on "Luis Mandoki" as the bridging entity connecting "film" and "place of birth". After GraphRAG locates this entity through graph traversal algorithms, it can accurately output the ground truth Mexico City. At this point, the entity associations at the text level are completely consistent with the logical semantics, and the reasoning path is clear and effective.

- Breakage of the logical chain after a LOGICPOISON attack Through a type-preserving entity swapping mechanism, LOGICPOISON cyclically replaces the PERSON-type bridging entity "Luis Mandoki" (the actual director of the film Gaby: A True Story) with another entity of the same type, "Quentin Tarantino"

(another well-known director). The modified text, "the director of film Gaby: A True Story is Quentin Tarantino", is completely semantically and grammatically coherent without any sense of incongruity. However, the topological connections in the knowledge graph have been implicitly tampered with: Gaby: A True Story → (directed by) → Quentin Tarantino → (born in) → Knoxville. Since Quentin Tarantino was born in Knoxville, GraphRAG, based on the damaged topological structure, ultimately outputs the incorrect answer, Knoxville.

A.8 Robustness for the entity type distribution

The three public datasets used in the experiments section have different entity numbers and distributions. The specific results are shown in Table 6 and Table 7:

Dataset	Number of entities
HotpotQA	71508
MuSiQue	73000
2WikiMultihopQA	41505

Table 6: Total Number of Entities in Datasets

Entity Type	Hotpot	Musi	2Wiki
DATE	8838	9974	7368
ORG	14397	12389	4246
FAC	3354	2920	516
PERSON	19843	18729	16767
ORDINAL	186	177	119
CARDINAL	1599	2523	427
NORP	1040	1610	803
EVENT	1786	1659	591
TIME	286	435	83
GPE	5616	7757	3514
MONEY	229	480	79
WORK_OF_ART	9158	6173	6139
QUANTITY	1189	1718	85
LOC	2108	3422	484
LAW	260	651	78
PERCENT	134	661	23
LANGUAGE	121	163	66
PRODUCT	1364	1559	117

Table 7: Entity Type Distribution Across Datasets

It can be seen that LOGICPOISON has good effects in different entity numbers and distributions, which also proves the strong robustness of our method.

A.9 Prompts

In this section, we present the prompt templates that drive the components of our method. Figure 7 illustrates the entity extraction prompt, which enforces strict constraints to isolate minimal reasoning hops and classify entities (including implicit bridges). Additionally, Figure 6 shows the evaluation prompt used to verify the ASR-G of the final answers, ensuring robustness against formatting variations.

ASR-G Evaluation Prompt

You are an expert judge evaluating if a model's prediction exactly matches the correct answer.

Question:
<Question>

Prediction:
<Prediction>

Correct answer:
<Correct Answer>

Task: Determine if the prediction exactly matches the correct answer. Consider:
- The prediction must contain the same core information as the correct answer.
- Minor formatting differences (punctuation, capitalization, spacing) should be ignored.
- The prediction should convey the same meaning as the correct answer.

Respond with only "YES" or "NO" (no other text).

Figure 6: Prompt template used for evaluating exact matches between predictions and gold answers.

Entity Extraction Prompt

Extract the reasoning-critical entities from the multi-hop question Q.

Your constraints:
- Do NOT answer Q.
- Do NOT add external knowledge.
- Use ONLY the wording of Q.
- Identify the minimal reasoning hops (1, 2, 3. . .).
- For each hop, extract only entities necessary for the reasoning chain.

Entity types: Use spaCy's NER label set when applicable: "PERSON", "NORP", "FAC", "ORG", "GPE", "LOC", "PRODUCT", "EVENT", "WORK_OF_ART", "LAW", "LANGUAGE", "DATE", "TIME", "PERCENT", "MONEY", "QUANTITY", "ORDINAL", "CARDINAL".
Additionally allow two reasoning-specific types:
- "ALIAS": paraphrased or descriptive label referring to an entity (e.g., "the nation called the nobilities commonwealth")
- "BRIDGE": implicit descriptive entity needed to connect reasoning hops (e.g., "the country where the operatives originated")

Guidelines:
- Include implicit bridge entities (BRIDGE) and alias-style descriptions (ALIAS) if they are part of the reasoning chain.
- Do NOT normalize entities to real-world names.
- Focus only on entities whose presence or value matters for understanding or resolving the question.

Output format (strict): A Python-style list of dicts, e.g.:

```
[
  {"hop": 1, "entity": "...", "type": "GPE"},
  {"hop": 2, "entity": "...", "type": "BRIDGE"},
  {"hop": 3, "entity": "...", "type": "DATE"}
]
```

Return ONLY the list.

Q: <Question Text>

Figure 7: Prompt template for reasoning-critical entity extraction.