LLM Embedding-based Attribution (LEA): Quantifying Source Contributions to Generative Model's Response for Vulnerability Analysis

Reza Fayyazi
Rochester Institute of Technology
Rochester, NY, USA
rf1679@rit.edu

Michael Zuzak
Rochester Institute of Technology
Rochester, NY, USA
mjzeec@rit.edu

Shanchieh Jay Yang
Gonzaga University
Spokane, WA, USA
yangj@gonzaga.edu

***Abstract*—Security vulnerabilities are rapidly increasing in frequency and complexity, creating a shifting threat landscape that challenges cybersecurity defenses. Large Language Models (LLMs) have been widely adopted for cybersecurity threat analysis. When querying LLMs, dealing with new, unseen vulnerabilities is particularly challenging as it lies outside LLMs' pre-trained distribution. Retrieval-Augmented Generation (RAG) pipelines mitigate the problem by injecting up-to-date authoritative sources into the model context, thus reducing hallucinations and increasing the accuracy in responses. Meanwhile, the deployment of LLMs in security-sensitive environments introduces challenges around trust and safety. This raises a critical open question: How to quantify or attribute the generated response to the retrieved context versus the model's pre-trained knowledge? This work proposes *LLM Embedding-based Attribution (LEA)* – a novel, explainable metric to paint a clear picture on the 'percentage of influence' the pre-trained knowledge vs. retrieved content has for each generated response. We apply LEA to assess responses to 100 critical CVEs from the past decade, verifying its effectiveness to quantify the *insightfulness* for vulnerability analysis. Our development of LEA reveals a progression of independency in hidden states of LLMs: heavy reliance on context in early layers, which enables the derivation of LEA; increased independency in later layers, which sheds light on why scale is essential for LLM's effectiveness. This work provides security analysts a means to audit LLM-assisted workflows, laying the groundwork for transparent, high-assurance deployments of RAG-enhanced LLMs in cybersecurity operations.** [1]

***Index Terms*—LLM, Vulnerabilities, RAG, Hidden States, Linear Independency, CVE, Rank, Embeddings, LEA.**

1. Introduction

Security professionals and vendors rely on vulnerability intelligence to defend against evolving threats. However, the threat landscape is becoming increasingly dynamic, with a sharp rise in both the volume and complexity of vulnerabilities [1]. The CVE framework [2] has cataloged over 300,000 vulnerabilities since 1999, creating challenges in identifying patterns, assessing risks, and deploying timely mitigations. Public databases like NIST NVD [3] are essential for sharing this information, but rely heavily on manual analysis - an approach that is slow and prone to delays in recognizing and responding to emerging threats [1].

Large Language Models (LLMs) have been widely adopted for cybersecurity threat analysis and deployment of mitigations in enterprise systems [4]–[14]. They have been particularly effective in cyber threat intelligence analysis [7], [8], LLM-assisted attacks [6], [9], [10], log-based anomaly detection [11], [14], and vulnerability detection [4], [5], [12], [13]. LLMs are inherently constrained by their training data, which is time-gated – meaning their knowledge reflects only what was available up to a fixed cutoff date. Meanwhile, thousands of new CVEs are disclosed each year. This creates a persistent out-of-distribution problem: LLMs lack awareness of emerging vulnerabilities that appear after their training cutoff. In cybersecurity, where timely and accurate threat intelligence is critical, this gap can have serious consequences. This issue is especially pressing as more organizations integrate LLMs into their SECDEVOPS pipelines [15], [16], and addressing this knowledge gap is essential for a safe and effective use of LLMs in real-time cybersecurity operations.

Retrieval-Augmented Generation (RAG) pipelines address the out-of-distribution shortfall by injecting up-to-date, authoritative data directly into the model's context. RAG grounds outputs in verifiable evidence, which significantly reduces the likelihood of hallucinations [17], [18]. Given the rapid evolution of threats, analysts will continue to rely on RAG pipelines for real-time access to current information about vulnerabilities. New models have added automatic RAG capabilities to help address this. However, over reliance on retrieved knowledge can still cause hallucinations, either due to ambiguous queries, or a low-quality knowledge base [19]. In addition, the deployment of LLMs in

1. Code and data: https://github.com/RezzFayyazi/LEA

security-sensitive environments introduces new challenges around trust, interpretability, and safety. For example, inaccurate or hallucinated code suggestions can lead to insecure configurations or flawed mitigations. Thus, beyond retrieval, it is essential to measure and communicate the confidence and source attribution of LLM-generated responses.

Since vulnerabilities are often novel, nuanced, and evolve rapidly, distinguishing between retrieved factual content and internal model reasoning is critical in cybersecurity, where inaccuracies can result in security breaches. This leads us to a critical question: Can we quantify the degree to which an LLM's response relies on retrieved knowledge versus its internal parameters – particularly in the context of novel vulnerabilities? In other words, when an LLM generates a response, how can we determine how much of that output is grounded in retrieved and factual content, and how much stems from the model's pre-trained (internal) knowledge? Quantifying this dependency is essential for building transparent, auditable AI systems that security analysts can trust in high-stake environments.

One way to quantify this can be with token-level probability differences. However, as vulnerability jargon often has multiple forms for the same concept, e.g., 'Cross-Site Scripting' vs. its abbreviation 'XSS' or 'SQL Injection' vs. 'SQLi'. An LLM can assign different token-level probabilities even though these pairs convey identical meaning. For this reason, the vulnerability analysis problem is unique from broader retrieval tasks. Researchers have observed that transformer weight matrices often have redundant or low-information directions, which can be exploited to reduce model size [20]. Low-Rank Adaptation (LoRA) [20] is a parameter-efficient fine-tuning method that injects a pair of learnable low-rank update matrices into each transformer layer to fine-tune large models like *Gemma-3* [21], *Deepseek-R1* [22]. This shows that LLMs exhibit substantial linear redundancy – many weight vectors are linearly dependent. From a theoretical perspective, the embedding vectors inhabit nearly the same subspace, so one can be expressed as a linear combination of the other. The 'rank' of a matrix reflects the number of linearly independent vectors it contains, offering a measure of the 'unique' or 'non-redundant' information present. If a vector can be expressed as a linear combination of others, it contributes no new information, indicating redundancy. Conversely, a set of linearly independent vectors implies that each vector adds distinct value to the representation space.

Built upon the above insights, we introduce a novel, explainable metric – **LEA** – based on the linear dependency of hidden state representations within LLMs. By measuring the rank of a matrix constructed from both retrieved-augmented representations and the latent hidden states of the LLM, we can quantitatively assess the relative contribution (percentage of influence) of each source to the final output. In fact, we show that LLMs exhibit strong context dependence in the early layers of the hidden state, which enables the construction of LEA to examine the linear dependencies of the generated tokens against the retrieved context vs. LLM's internal knowledge. This approach provides a transparent view into the influence balance between externally retrieved evidence (e.g., RAG) and the model's internal, pre-trained knowledge, and thus a principled way to disentangle genuine knowledge injection from mere rephrasing in RAG-enhanced cybersecurity workflows, improving trust and interpretability in model outputs.

We curated 100 critical and high severity CVEs between 2016 to 2025, and conducted a thorough analysis across the layers of four instruction-tuned LLMs of varying sizes: *Gemma-3-27B-IT* [21], *Mistral-Small-24B-Instruct-2501* [23], *DeepSeek-R1-Distill-Llama-8B* [24], and *LLaMA-3.2-3B-Instruct* [25]. Our results reveal a common trend across all evaluated models: as long as the generated response captures the factual, verified content from RAG, LEA shows a relatively balanced distribution between the retrieved context and the LLM's internal knowledge. Conversely, when the model disregards verified RAG embeddings and generates a response with little to no alignment with the retrieved context, LEA exhibits significantly low percentage value associated with the RAG. Furthermore, by analyzing the LEA distributions for the CVE analysis from 2016 to 2025, we discover the limitations of generic use of LLMs in memorizing the vast majority of documented CVEs. This indicates the inherent constraints of relying solely on foundational (internal) knowledge for comprehensive and up-to-date threat intelligence. We summarize our contributions as follows:

- **Explainable Metric:** A novel metric, **LEA**, is developed to reveal how LLM generated responses depend on the foundational knowledge versus retrieved context by exploiting the inherent transformers architecture.
- **Verifiable Uses for Vulnerability Analysis:** We apply LEA to verify that LLM-generated responses for vulnerability analysis are *insightful*, by showcasing the *expected distribution* of the responses when applied with verifiable sources, using 100 critical CVEs over a 10-year period.
- **Dependency Progression in Hidden States:** We demonstrate that LLMs show context dependence in early layers (especially layer-0), but increasingly treat tokens independently in the middle to final layers, indicating why large model sizes are needed to capture complex interdependencies.

2. Preliminaries

2.1. Large Language Models

The remarkable progress in LLMs is due to the adoption of the Transformers architecture [26]. State-

of-the-art models such as OpenAI's GPT series [27] and Deepseek-R1 [22] exemplify this advancement. Their ability to tackle a wide range of complex natural language tasks has been well documented in recent studies [28], [29]. However, despite their effectiveness, LLMs remain susceptible to hallucinations [15], [16]. This issue is often rooted in the models' reliance on their static pre-trained knowledge, which introduces a critical limitation, and that is the knowledge cutoff of the training data. As a result, LLMs may struggle to reason about or accurately respond to emerging events or newly discovered vulnerabilities.

To mitigate these shortcomings and enhance factual accuracy, RAG techniques were introduced [17]. RAG combines traditional information retrieval techniques with LLM prompting by incorporating relevant, up-to-date external content at inference time, thereby grounding model outputs with relevant context. Building on this foundation, several advanced RAG techniques have been proposed to further improve precision, adaptability, and robustness in response generation [30]–[32].

However, excessive reliance on retrieved knowledge can still lead to hallucinations – particularly when the input queries are ambiguous or when the retrieved information comes from a low-quality knowledge base [19]. This raises the need to quantify how much the models depend on the retrieved tokens when generating a response.

2.2. LLMs in Vulnerability Analysis

Several recent studies have explored the integration of LLMs into vulnerability analysis workflows [4], [5], [13], [33]. Cheshkov et al. [5] applied GPT-based models to detect Common Weakness Enumeration (CWE) vulnerabilities in Java code, uncovering that LLMs often struggle with reliably identifying vulnerable patterns—especially in complex or obfuscated codebases. In contrast, Khare et al. [4] demonstrated that LLMs can outperform traditional deep learning models in vulnerability detection when guided by well-crafted prompt strategies, which indicates the critical importance of prompt engineering in effectively leveraging LLMs for cybersecurity tasks.

Extending these efforts, Du et al. [33] introduced Vul-RAG, a retrieval-augmented framework that constructs a multi-dimensional knowledge base from historical CVE reports and integrates it with LLMs to improve contextual understanding during vulnerability analysis. ChatNVD [13] leverages the NVD as an external knowledge source and utilizes a GPT-4 variant to generate concise, human-readable summaries of CVE entries. This system aims to help practitioners rapidly understand a vulnerability's nature, exploitation potential, and impact without having to parse through dense technical descriptions.

Despite these promising advances, none of these works quantified the degree to which an LLM's output relies on retrieved knowledge versus its internal representations. This lack of quantification constitutes a significant gap in explainability and trust, particularly in cybersecurity domain, where understanding the provenance of generated content is essential for reliable and auditable decision-making.

3. Motivation to use Linear Independence

To better understand the influence of the RAG context on the generation process, we begin by analyzing the differences in token-level probabilities before and after incorporating RAG. This analysis reveals how RAG affects the model's confidence in generating specific tokens and provides insight into the contextual dependencies introduced by the retrieved information. Figure 1 illustrates how the token-level probabilities in the generated response shift when RAG-provided context is incorporated. Notably, the probabilities of key tokens related to *CVE-2025-30066* exhibit substantial changes, which indicates the influence of the retrieved context on the model's output.

However, while informative, token-level probability changes alone do not offer a principled or quantifiable measure of how much the model relies on retrieved information. One limitation arises from the linguistic variability inherent in cybersecurity jargon. For instance, terms such as "malicious commit" and "backdoor commit" may be semantically equivalent, yet the model may assign distinct token probabilities to each. This semantic ambiguity complicates efforts to attribute generation influence based purely on token probability shifts.

To overcome this challenge, we introduce a more robust and interpretable approach grounded in the internal structure of the model's hidden representations. Prior work has shown that transformer models often exhibit redundancy in their weight matrices and hidden states, with many directions in these high-dimensional spaces contributing little unique information [20], [34], [35]. One notable method that capitalizes on this property is LoRA [20]. LoRA has gained widespread adoption for fine-tuning LLMs as it achieves performance comparable to full fine-tuning while requiring updates to only a few million parameters. Model pruning is another prominent area where the concept of linear dependence is exploited. Pruning techniques aim to remove redundant parameters or structures—such as attention heads or neurons—based on the insight that many components in LLMs learn features that are linear combinations of others, and thus do not contribute uniquely to model output. Recent research in attention head pruning has shown that a substantial number of attention heads can be removed with minimal impact on performance [34], [35]. This insight motivates the use of matrix rank—specifically, the rank of hidden-state matrices—as a metric to assess linear independence and information diversity across layers. The rank of a hidden-

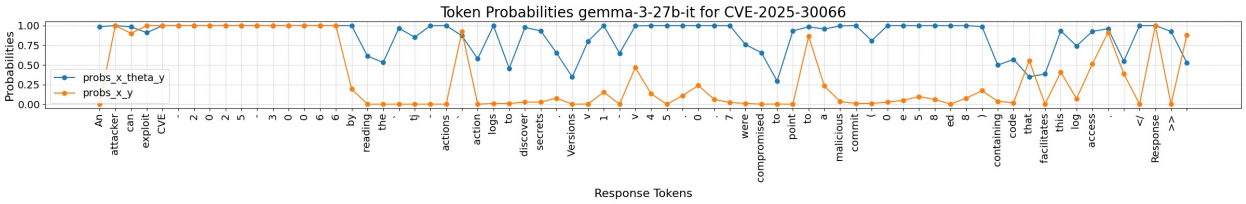

Figure 1: The probability differences for the RAG-generated response of CVE-2025-30066 before and after including RAG tokens (refer to Table 1 for the terms).

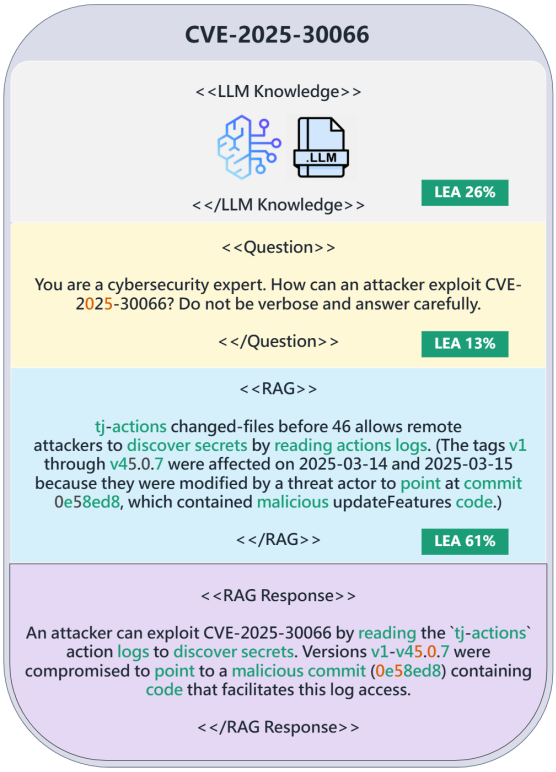

Figure 2: LEA's dependency distribution of the Response with respect to the model's knowledge, RAG context, and the Question.

state matrix quantifies the number of linearly independent vectors, effectively capturing the dimensionality of the informative subspace in a given layer. A lower rank implies redundancy or strong dependency among token representations, while a higher rank suggests that the tokens are contributing distinct information to the representation space. By computing the rank at each transformer layer, we gain visibility into how token interactions evolve with depth, and how the presence of RAG-derived context modulates these interactions.

This leads us to consider how the concept can be applied in the analysis of vulnerabilities to detect which token generations in the responses were from the retrieved context (i.e., RAG). In this setting, rank-based analysis can help determine whether the model's response about a CVE (e.g., its exploitability, impact, or mitigation strategies) is primarily grounded in retrieved evidence or generated using internal knowledge. If token representations introduced by RAG significantly affects the rank of hidden states, this suggests that the retrieved content is actively enriching the model's representational space and guiding generation. Conversely, if the model's internal representations already exhibit high rank without RAG context, it may indicate that the model possesses intrinsic understanding of the vulnerability and is less reliant on external information.

Figure 2 illustrates the dependency distribution derived from our proposed metric (LEA) for *CVE-2025-24472*. In this example, 26% of the generated content is attributed to the model's pre-trained knowledge, while 13% and 61% are influenced by the user query and retrieved documents, respectively. This percentage of influence is critical for downstream cybersecurity tasks. If the dependency distribution with retrieved tokens closely mirrors that seen in the absence of retrieval, it could indicate that the model possesses internal insights about the CVE and retains semantically relevant associations from its foundational knowledge. In contrast, when the model overlooks verified RAG embeddings and produces responses with minimal alignment to the retrieved context, it could indicate hallucination or a generic generation. In the context of vulnerability analysis, a poorly aligned response can result in misleading interpretations or a lack of actionable insights. Therefore, detecting and filtering poorly grounded responses is essential before dissemination to ensure the reliability of AI-assisted cybersecurity workflows.

4. LEA: Theory and Derivation

In this section, we show how the proposed LEA metric works based on the linear dependency of hidden state representations within LLMs. By measuring the rank of a matrix constructed from both retrieved-augmented representations and the latent hidden states of the LLM, we can quantitatively assess the relative contribution of each source to the final output. To understand this, we first need to discuss how text progresses through the transformers architecture.

Let a prompt be tokenized in the ordered sequence $T = (t_1, t_2, \ldots, t_L)$ of length L.

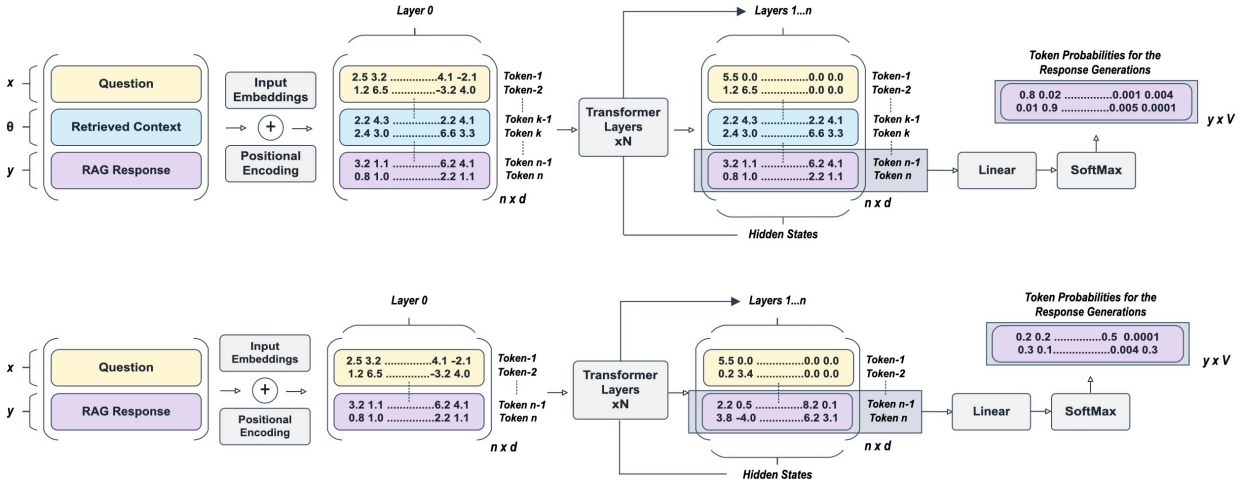

Figure 3: The step-by-step of getting the hidden state progression and probability differences.

1) *Token embedding.* Each discrete token t_i is mapped to a continuous vector via an embedding matrix $E \in \mathbb{R}^{|\mathcal{V}|\times d}$:

$$\mathbf{e}_i \;=\; E[t_i] \in \mathbb{R}^d,$$

where d is the model (hidden) dimension and $\mathcal{V}$ is the vocabulary.

2) *Positional encoding.* Because the self-attention mechanism is permutation invariant, positional information must be injected. For position i we add a vector $\mathbf{p}_i \in \mathbb{R}^d$ (either a fixed sinusoidal code or a learned embedding):

$$\mathbf{x}_i^{(0)} \;=\; \mathbf{e}_i \;+\; \mathbf{p}_i, \qquad i = 1, \ldots, L.$$

The matrix $X^{(0)} = [\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_L^{(0)}] \in \mathbb{R}^{L\times d}$ constitutes the *layer-0 hidden state*. It encodes *both* lexical identity and position.

3) *Stack of Transformer blocks.* For $\ell = 1, \ldots, N$ (where N is the number of layers) the hidden state is updated by a residual sublayer comprising multi-head self-attention (MHA), feed-forward network (FFN), and layer normalisation (LayerNorm):

$$\tilde{X}^{(\ell)} = X^{(\ell-1)} \;+\; \mathrm{MHA}\big(\mathrm{LayerNorm}(X^{(\ell-1)})\big),$$

$$X^{(\ell)} = \tilde{X}^{(\ell)} \;+\; \mathrm{FFN}\big(\mathrm{LayerNorm}(\tilde{X}^{(\ell)})\big),$$

yielding $X^{(\ell)} \in \mathbb{R}^{L\times d}$, the *layer-$\ell$ hidden state*. Each row $\mathbf{x}_i^{(\ell)}$ is a context-dependent representation of the i-th token, refined through ℓ rounds of self-attention and nonlinear mixing.

4) *Output (logits).* After the final layer N, the hidden state $X^{(N)}$ is projected back to vocabulary space through a linear map (often the transpose of E) followed by a soft-max:

$$Z \;=\; X^{(N)}\, W_{\text{out}} + \mathbf{b}_{\text{out}} \;\in\; \mathbb{R}^{n\times|\mathcal{V}|},$$

$$P \;=\; \mathrm{softmax}(Z)$$

yielding for every position i a probability distribution $P_{i,:}$ over the next token. During autoregressive generation the model typically uses only the *last* row of $X^{(L)}$ (the most recent token) to predict the next token in the sequence.

Therefore, every Transformer layer re-encodes the entire prompt, allowing each token's representation to attend to, and hence depend on, *all other tokens*. Consequently the hidden states evolve from locally grounded embeddings ($\mathbf{e}_i$) to highly contextual vectors $\mathbf{x}_i^{(N)}$ (hidden states) that capture syntax, semantics, and long-range dependencies of the prompt.

Now, to understand how we can get a representation of these highly contextual vectors, we need to discuss about linear independency. A set of vectors is linearly independent if no vector in the set can be expressed as a linear combination of the other vectors in the set. In simpler terms, this means that the vectors are not "redundant" or "dependent" on each other, and hence, adding 'new information'. Let $\{v_1, v_2, \ldots, v_n\}$ be a set of vectors in a vector space V. The vector v_i is said to be *linearly independent of the remaining vectors* if

$$\sum_{\substack{j=1\\ j\neq i}}^{n} \alpha_j\, v_j \;=\; v_i \;\;\Longrightarrow\;\; \alpha_j = 0 \quad \text{for all } j \neq i.$$

Equivalently, there are no scalars α_j (not all zero) with $j \neq i$ such that a linear combination of the other vectors equals v_i.

The 'rank' of a matrix is how many vectors are needed to describe the original matrix. Let $A \in \mathbb{F}^{m\times n}$ be a matrix over a field $\mathbb{F}$: For $A \in \mathbb{F}^{m\times n}$ the *rank* of A, written $\mathrm{rank}(A)$, is the dimension of its row-space

$$\mathrm{row}(A) = \mathrm{span}\{A_{1,:}, \ldots, A_{m,:}\} \subseteq \mathbb{F}^n.$$

By equality of row- and column-rank this equals the dimension of the column-space

$$\mathrm{col}(A) = \mathrm{span}\{A_{:,1}, \ldots, A_{:,n}\} \subseteq \mathbb{F}^m.$$

In other words, $\mathrm{rank}(A)$ counts the number of *fundamental directions* needed to reconstruct every row (or, equivalently, every column) of A. Any additional vector lying in the span of those directions adds no new information.

Duplicate information. If $\mathrm{rank}(A) = r < m$, then exactly $m - r$ rows are redundant. Each redundant row can be written as a linear combination of r independent rows:

$$A_{j,:} = \sum_{i=1}^{r} \alpha_i A_{k_i,:},$$

where $k_1, \ldots, k_r$ index linearly independent rows and $\alpha_1, \ldots, \alpha_r \in \mathbb{F}$.

Therefore, this concept can be applied in the context of LLMs, as each token has a vector representation. More specifically, a prompt is converted into an embedding matrix representing the tokens in the prompt, with the dimension of $\mathbb{R}^{L \times d}$, where *L* is the number of tokens and *d* is the dimension of the model. As **E** propagates through the model's Transformer blocks, self-attention and feed-forward layers iteratively refine these vectors, producing a sequence of hidden states—one per token—that captures progressively richer semantic and syntactic context. Because decoder-only LLMs are trained with a causal mask, the hidden state at position t is conditioned exclusively on tokens $1, \ldots, t-1$, ensuring the model's predictions remain autoregressive.

This structural property opens a pathway for fine-grained attribution analysis: by examining how hidden states evolve in the presence of retrieved context regarding new vulnerabilities, we can quantify the extent to which specific token generations in the model's output are influenced by the retrieved evidence. To do so, we first concatenate the original question (x) with the RAG-generated response (y) to form the combined sequence xy, and we compare with adding the retrieved context (θ) to get the sequence $x\theta y$. Figure 3 show how these representations progress in the transformers architecture. We then evaluate the contextual dependency of each token within this sequence. Specifically, we compare the token-level dependency annotations before and after incorporating θ. Table 1 provides a brief description of the terms used in the paper.

In Table 2, we provide the evolution of the dependency patterns across transformer layers for a vulnerability (*CVE-2025-30066*). As can be seen, when the model progresses through the intermediate layers, a shift occurs: tokens are increasingly treated as independent variables, and the trend continues until the final layers. This likely arises from the attention mechanism's independent token weighting, which encourages treating tokens in isolation and may explain why LLMs must be so large to capture complex interdependencies. Interestingly, we observe that Layer-0, which includes both token embeddings and positional encodings, provides a signal of how dependent the model is on the input prompt. At this layer, the model's representations reflect meaningful contextual relationships between tokens. This indicates that Layer-0 can be especially useful for discerning how the model interprets and distinguishes between question tokens and retrieved content. In Appendix A, we show how these dependencies follow the same trend with another CVE and more analysis. Therefore, **we define the *LEA* metric as the transition of the dependency from xy to $x\theta y$ at Layer-0**. The progressive transformation from independent to dependent in the responses at layer-0 serves as a measure in understanding the foundational behavior of LLMs in vulnerability analysis. The dependency transitions are summarized in Table 3, and in Sec. 5.2 we explain on how the interpretation of these transitions are derived.

TABLE 1: Brief description of the used terms throughout the paper

Term	Description
x	main question of the prompt
θ	retrieved context (RAG)
y	RAG-generated response
y'	base response (no RAG involved)
xy	concatenation of the question and RAG-generated response
$x\theta y$	concatenation of the question, retrieved context, and RAG-generated response
xy'	concatenation of the question and base-generated response
$x\theta y'$	concatenation of the question, retrieved context, and base-generated response

5. Experimental Design

5.1. Dataset

We curated a dataset of CVEs with high or critical severity reported in the past 10 years. We focused on CVEs with a Common Vulnerability Scoring System (CVSS) score of 7.0 or higher. In total, the curated dataset comprised of **100 CVEs** from 2016 to 2025. For each year, we curated 10 CVEs to see how the models perform when dealing with the most recent vs. the older ones. For each CVE, we assumed an ideal RAG pipeline that returned only the most relevant, non-redundant, and verified information from authoritative web sources. What this indicates is that we only provided the most relevant information for the CVE in question from the NVD website [3]. Moreover, since the selected LLMs have a training cut-off up until mid-2024, we expect them to rely more heavily on RAG for the 2025 CVEs. However, older vulnerabilities are not guaranteed to appear in the pre-training corpora of models, therefore, some years may exhibit a similar distribution of RAG-generated tokens (especially since we are only putting the CVE-ID in the question).

TABLE 2: Layer–by–layer rank evolution for each model for CVE-2025-30066

	Llama-3.2-3B-Instruct	DeepSeek-R1-Distill-Llama-8B	Mistral-Small-24B-Instruct-2501	Gemma-3-27b-it
	28 Layers	**32 Layers**	**40 Layers**	**62 Layers**
	Layer 0	***Layer 0***	***Layer 0***	***Layer 0***
rank $x\theta y$	48%	13%	48%	47%
rank x_y	50%	12%	57%	64%
	Layers 1-2	***Layers 1-24***	***Layers 1-5***	***Layers 1-23***
rank $x\theta y$	98%	79%	95%	88%
rank x_y	98%	78%	98%	94%
	Layers 3-27	***Layers 25-31***	***Layers 6-39***	***Layers 24-61***
rank $x\theta y$	100%	100%	100%	100%
rank xy	100%	100%	100%	100%

TABLE 3: Independency conditions, their interpretations, and corresponding LEA values

xy	$x\theta y$	**Interpretation**	**Abbr.**
True	**True**	Foundational Knowledge	*LEA*(fnd)
True	**False**	Retrieved Context	*LEA*(rag)
False	**False**	Question	*LEA*(q)
False	**True**	Invalid	—

It is worth noting that we conducted our experiments on a system equipped with two Intel Xeon E5-2650 CPUs, 256 GB of RAM, two NVIDIA Tesla P40 GPUs, and one NVIDIA Tesla V100 GPU.

5.2. Experimental Settings

For our experiments, first, we will design a prompt to ask about exploitation of different CVEs, along with the retrieved context, to obtain the generations. We will test on 4 different LLMs of varying sizes: *Gemma-3-27B-IT*, *Mistral-Small-24B-Instruct-2501*, *DeepSeek-R1-Distill-Llama-8B*, and *LLaMA-3.2-3B-Instruct*. Note that for the generations, we get the highest probable token (i.e., the token with the highest logit value) to get the most deterministic responses from the LLM. The designed prompt is as follows:

> *You are a cybersecurity expert. How can an attacker exploit CVE-xxx-xxx? Do not be verbose and answer carefully.*
>
> `<<RAG>>`
> *Retrieved Context*
> `<</RAG>>`
>
> `<<Response>>`

We put each CVE-ID in the question, and we will put the retrieved context in RAG. After getting the generations, we now have the question (x), the retrieved context (θ), and the generated response (y).

Our investigation into linear dependence at layer-0 stage aims to determine whether the model inherently relies on this raw contextual information for subsequent token generation. This layer is critical because it directly processes the input embeddings augmented by positional encodings, which reflects the model's initial view of token identity and order.

Figure 4 and Table 3 show how we quantify the dependency of the generated tokens in our proposed metric using *LEA*(fnd), *LEA*(rag), and *LEA*(q). A token initially marked as independent (i.e., True) that becomes dependent (i.e., False) after adding θ indicates that the retrieved context introduced a new dependency (*LEA*(rag)). In this case, θ contributes meaningful contextual information influencing that token. Conversely, if a token remains independent both before and after adding θ, it suggests that the token's interpretation is self-contained within the original question and unaffected by the retrieved content *LEA*(fnd). If a token remains dependent (i.e., False to False), this implies that the dependency is solely attributed to the information present in the original input x, and the retrieved content did not alter this relationship *LEA*(q). Lastly, if adding θ changes a token from False (dependent) to True (independent), this suggests an inconsistency. A token initially identified as dependent should not become independent with the addition of more context. Such a transition may indicate invalid behavior, as it undermines the stability of the token's contextual dependency assessment.

6. Usage of LEA for CVE Analysis

As outlined in Sec. 5, LEA quantifies an LLM's reliance on retrieved context when generating responses. To validate its utility and robustness, we present four complementary sets of results. First, we will get the LEA distribution over the entire response. Next, we compare the distribution with a generic RAG. Third, we will apply some filtering to focus on content-bearing tokens. Finally, we compare these LEA distributions of RAG-response with a Base-response (i.e., no retrieval).

6.1. Full-Response Dependency Distribution

In Figure 5, we summarize our findings for CVEs spanning several years and evaluated across multiple

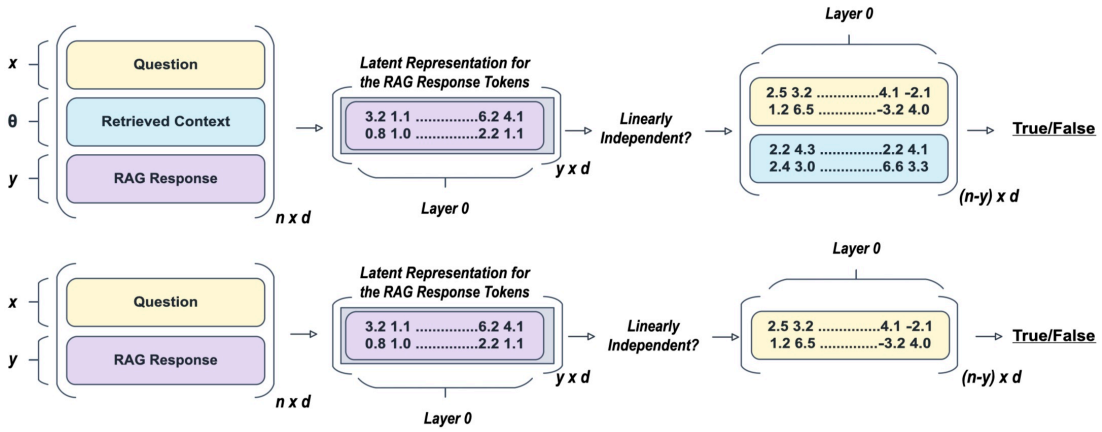

Figure 4: The process to derive LEA with the dependent vectors with (top) and without (bottom) retrieved context.

state-of-the-art LLMs. For every token in each RAG-generated answer, we calculate the linear dependence of its layer-0 representation on the embedding sets of (i) the user query and (ii) the retrieved passages, following the workflow depicted in Figure 4. This token-level analysis quantifies how strongly the model's output leans on newly retrieved evidence versus its internal knowledge. The results reveal a broadly distributed dependency of tokens across all the years and the models. From the results, we can see that, first, the smaller model (*Llama-3.2-3B-Instruct*) exhibits a greater reliance on its *LEA*(fnd) compared to the larger models. This suggests that, despite its limited capacity, the smaller model tends to utilize pre-trained knowledge more than retrieved context. Second, across all models, we observe a greater dependence on *LEA*(rag) context when responding to CVEs from 2023 and 2025. This trend likely reflects the fact that these CVE identifiers are relatively recent and may not have been present during the pre-training phase; therefore, the models lean more heavily on retrieved information to compensate for the lack of prior exposure.

Moreover, it is important to note that in the experiments, the question is only asking about the CVE ID (e.g., *CVE-2025-30066*) without additional context. Given that the global CVE database contains over 300,000 unique IDs, it is highly improbable that any LLM could memorize and internalize detailed knowledge for each ID. Thus, the relatively uniform distribution of dependency across CVE years is expected, which justifies the reliance of LLMs on external retrieval knowledge bases in cyber operations to provide accurate and contextually grounded responses for most CVEs. Next, we test with a generic RAG to measure what might change with LEA.

6.2. Generic RAG Context Evaluation

To assess the sensitivity and effectiveness of our metric, we test the model's response behavior when provided with a generic or weakly relevant RAG context. Here is the text for the generic RAG:

> *CVE, short for Common Vulnerabilities and Exposures, is a list of publicly disclosed computer security flaws. When someone refers to a CVE, they mean a security flaw that's been assigned a CVE ID number.*

The results are presented in Figure 6. Compared to Figure 5, we observe a significant decrease in *LEA*(rag). This decline supports the validity of our approach by demonstrating that the proposed LEA metric is sensitive to the informativeness of the retrieved context. In other words, when meaningful retrieval is absent, the model exhibits a measurable drop in *LEA*(rag) – reinforcing our metric's utility in distinguishing context-driven generation from reliance on internal knowledge. The results also show that *LEA*(q) remains consistent (refer to Figure 5). This is expected – since the question is unchanged, the influence of its tokens remains stable. However, when presented with a more generic RAG input, we observe that the models tend to rely more heavily on their foundational knowledge, i.e., higher *LEA*(fnd). Moreover, note that not all tokens are equally informative – common stop words like 'on' or 'the' carry little semantic value and may skew aggregate metrics. Therefore, in the next section, we refine our analysis by focusing on content-bearing tokens and examining how the distributions shift.

6.3. Stop-word Filtering and Thresholding

To refine our analysis and focus on semantically meaningful content, we apply stopword filtering (e.g., removing tokens such as "on", "the") and introduce a probability-based thresholding mechanism. This dual filtering ensures that our metric emphasizes content-bearing tokens that meaningfully contribute to the model's output, while minimizing noise introduced by

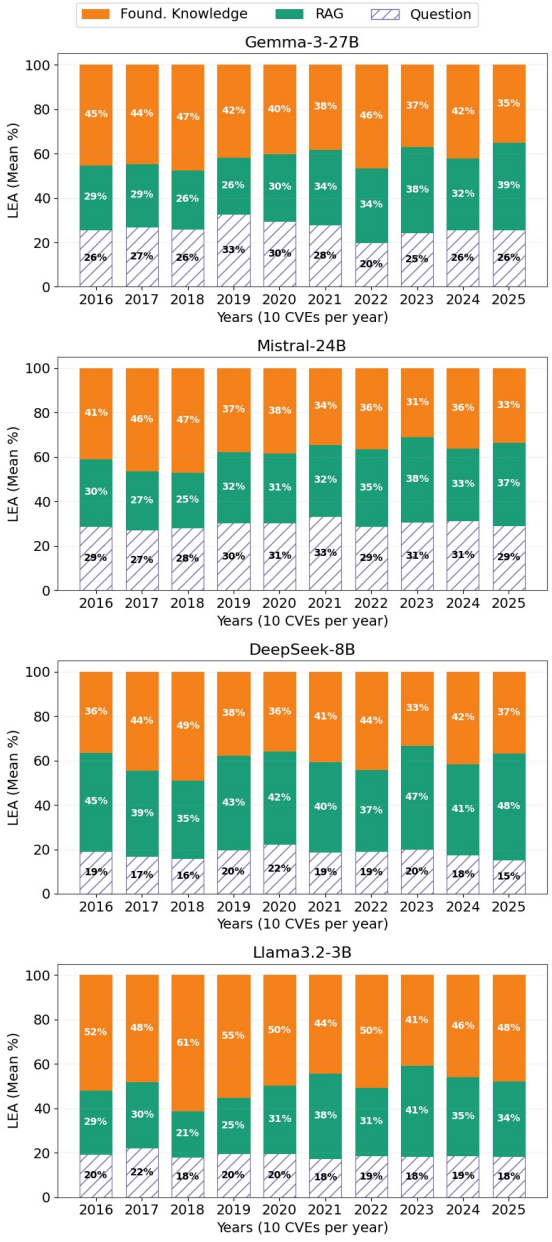

Figure 5: The LEA distribution for verified RAG-based CVE queries across four LLMs

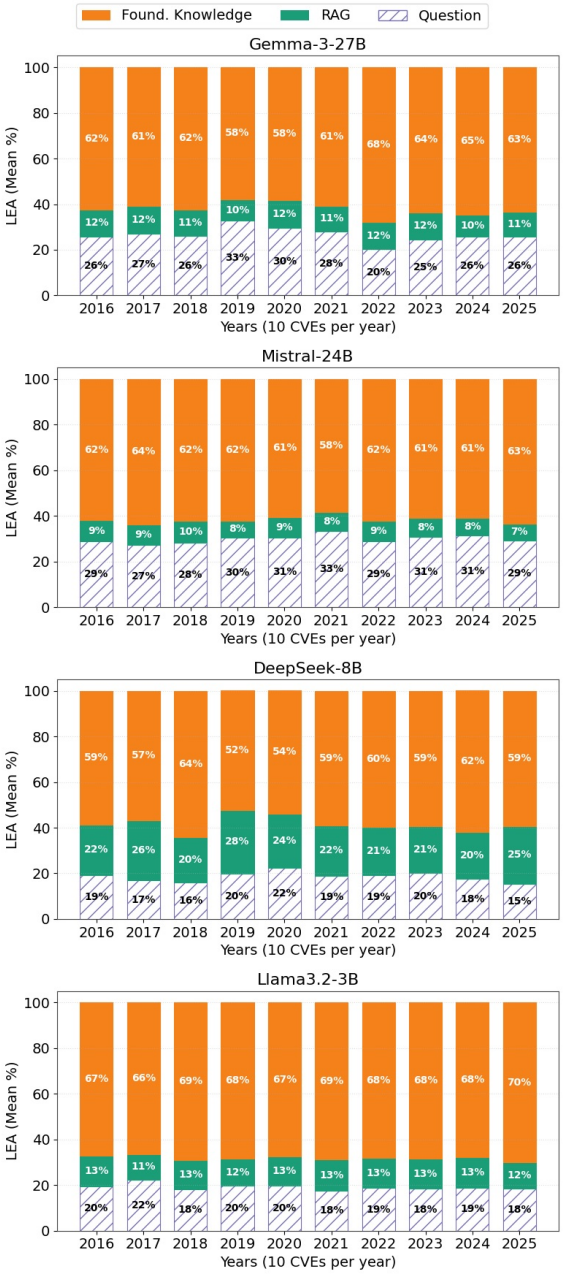

Figure 6: The LEA distribution for generic RAG-based CVE queries across four LLMs

high-frequency but low-informational words. For example, the following Table 4, shows the token-level probability deltas ($\Delta p = x\theta y - xy$), where p_1 and p_2 represent token probabilities before and after removing the RAG context, respectively. Filtering out these non-informative tokens results in a noticeable decrease in the proportion of *LEA*(q), as shown in Figure 7. This outcome is expected, as the question tokens primarily consist of the CVE identifier (e.g., *CVE-2025-30066*), which remain invariant with or without RAG context.

TABLE 4: Token-level probability deltas ($\Delta p = x\theta y - xy$) with and without RAG context

Token	**Text**	$x\theta y$	Δp $(x\theta y - xy)$
257	`attacker`	1.000	0.0000
258	`exploits`	0.770	0.7695
259	`CVE`	1.000	0.0000
260	–	1.000	0.0000
261–264	`2025`	1.000 (each)	0.0000 (each)

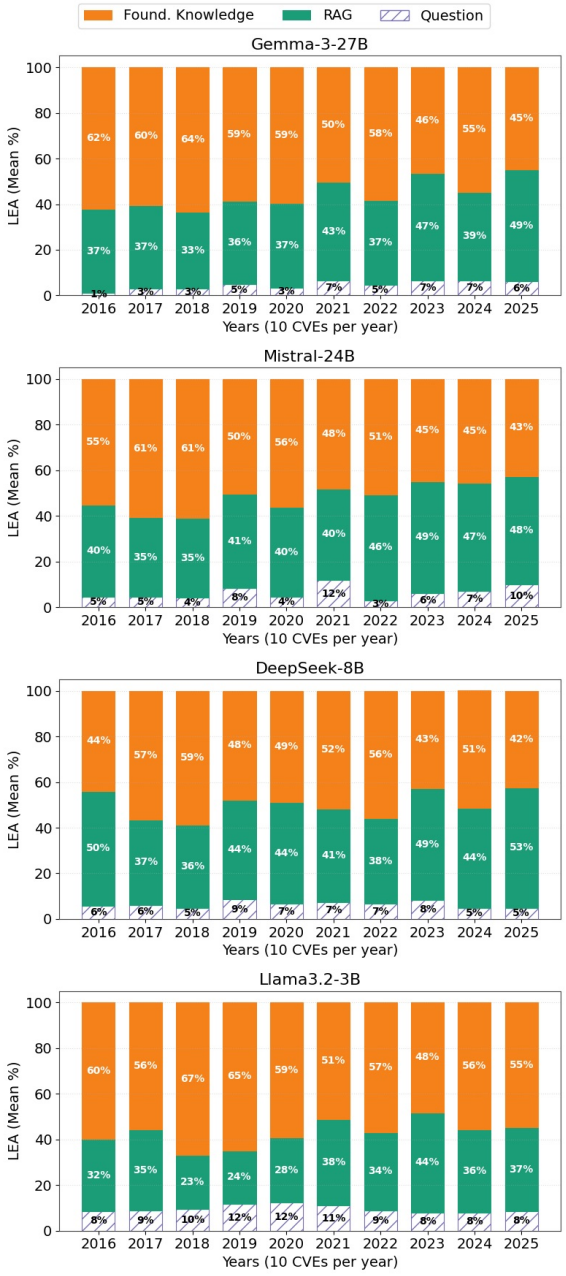

Figure 7: The LEA distribution of verified RAG-based CVE queries across four LLMs with Stop-word Filtering & Thresholding

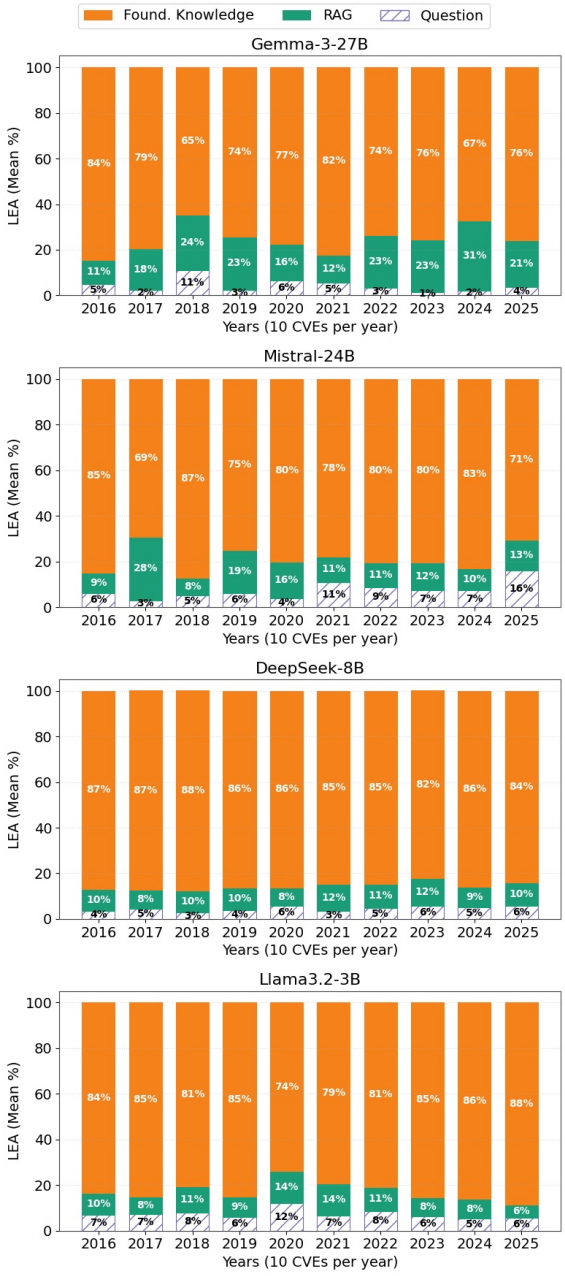

Figure 8: The LEA distribution of Base-generated (y') Responses with Stop-word Filtering and Thresholding

While these identifiers are syntactically important, they carry minimal semantic weight in guiding generation. From a practical standpoint, security analysts are less concerned with surface-level identifiers and more interested in whether and how the retrieved context influences the model's substantive understanding of the vulnerability.

When comparing these results with those in Figure 5, we observe a relative increase in *LEA*(rag). This shift reinforces the value of our filtered metric: it demonstrates the extent to which informative content—rather than syntactic tokens—relies on external retrieval. Moreover, by analyzing tokens with significant Δp values, such as 'exploits' in the example above, we can precisely identify which parts of the model's output are context-dependent versus internally derived (refer to Figure 2). Overall, this demonstrates the effectiveness

TABLE 5: Specific examples of RAG-generated (y) vs. Base-generated (y') responses for old and new CVEs

CVE-2016-8655	***Gemma-3-27B* Output**
Base–Response	CVE-2016-8655 is a vulnerability in the Microsoft Graphics Component. An attacker can exploit it by crafting a malicious image file (e.g., TIFF, JPEG, PNG) that, when opened by a vulnerable application (like Windows Photo Viewer, or an application using the graphics component), allows for remote code execution. The attack requires user interaction 2013 the user must open the crafted image.
RAG–Response	An attacker with 'CAP_NET_RAW' can exploit CVE-2016-8655 by rapidly changing a packet socket's version while the kernel is processing it. This race condition leads to a use-after-free vulnerability. Successful exploitation can result in privilege escalation or a denial of service.
CVE-2025-24472	***Gemma-3-27B Output***
Base–Response	CVE-2025-24472 is a critical remote code execution vulnerability in the widely used LibImageLibrary. It stems from a heap buffer overflow during the processing of malformed TIFF images. **Exploitation:** 1. **Craft Malformed TIFF:** Create a TIFF image with specifically crafted, oversized tag data. 2. **Trigger Processing:** Deliver the image to an application using LibImage (e.g., image viewer, editor, server). 3. **Overflow & Control:** The overflow overwrites adjacent heap metadata, allowing control of program execution.4. **RCE:** Redirect execution to shellcode injected within the image or existing code. Successful exploitation leads to arbitrary code execution with the privileges of the affected application.
RAG–Response	An attacker can exploit CVE-2025-24472 by crafting malicious CSF proxy requests. These requests bypass normal authentication checks, granting the attacker super-admin privileges on vulnerable FortiOS and FortiProxy systems. This allows full control of the device.

of our stopword-filtered and thresholded analysis in surfacing meaningful differences in generation behavior, making it more aligned with real-world interpretability needs in cybersecurity applications.

6.4. LEA of RAG-Gen. vs. Base-Gen. Response

So far, our analysis has focused on the LEA distribution in RAG-generated responses to quantify how much the model relies on the retrieved context. However, an important complementary question arises: How does the model perform when asked directly about a CVE without access to retrieved context? To explore this, we generate a baseline response y' from the model using only the question (i.e., no RAG), and then assess the extent to which this response aligns with the verified retrieved content.

Figure 8 illustrates this alignment by showing the dependency of tokens in the y' response with respect to the verified source context (θ). As can be seen, the *LEA*(rag) significantly decreases when using the y' response (compared to y). This analysis serves as a proxy for what an insightful, contextually relevant response should look like, and highlights how the model internally reasons about a vulnerability when retrieval is not available. If the dependency distribution for y' resembles that of Figure 7 – where the model meaningfully engages with retrieved tokens – it suggests that the model has internal insights about that CVE and retains some semantically relevant associations from its internal knowledge. Conversely, if the response is entirely ignoring the verified RAG embeddings, it suggests that the model either hallucinated or produced a response with minimal factual relevance, which indicates the necessity of retrieval for reliable and informative outputs in cybersecurity applications.

Table 5 provides illustrative examples of this behavior. Across both recent and older CVEs, the model frequently hallucinates or fails to generate responses grounded in factual detail when retrieval is disabled. This is expected: given the existence of over 300,000 CVEs, it is unreasonable to assume that a language model can memorize or accurately recall all of them—particularly when prompted with only a CVE identifier. Unlike a search engine, the model lacks indexed recall capabilities, further emphasizing the limitations of relying solely on internal knowledge. These findings underscore two important takeaways: (1) retrieval is indispensable for producing informative, trustworthy outputs in vulnerability analysis, and (2) the proposed LEA metric offers a robust measure for quantifying model-context reliance, which allows for informed decision-making when interpreting responses about unseen or complex vulnerabilities.

7. Conclusion

We developed a novel, explainable metric called **LEA** to reveal how LLM generated responses depend on the foundational knowledge versus retrieved context by exploiting the inherent transformer architecture. We demonstrated its verifiable use for vulnerability analysis to show LLM generated responses are insightful by quantifying the expected distribution when properly retrieved from verifiable sources. We tested on 100 critical and high-severity CVEs across 10-year span. We also demonstrated that LLMs in layer-0 show dependency on the retrieved context, however, they tend to treat every token as an independent variable as they progress through middle layers until the final layers. This inherently showcases the model's tendency to treat tokens independently, and hence, possibly explaining why LLMs need to be very large to effectively capture complex interdependencies. Finally, as LLMs become more deeply integrated into cybersecurity workflows, LEA offers a transparent, trust-building mechanism for interpreting model outputs and supporting informed decision-making.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 2344237

and No. 2228001.

Ethics Statement

This work introduces an explainable metric (LEA) to quantify the extent to which LLM-generated responses rely on foundational knowledge versus retrieved context in the domain of cybersecurity vulnerabilities. The research poses no ethical concerns, as it does not reveal or facilitate the exploitation of any new or existing vulnerabilities. Instead, it focuses on analyzing internal model representations to assess the insightfulness/accuracy of responses. As LLMs are further integrated into cybersecurity systems, LEA serves as a mechanism to enhance trust and interpretability.

References

[1] A. Okutan, P. Mell, M. Mirakhorli, I. Khokhlov, J. C. Santos, D. Gonzalez, and S. Simmons, "Empirical Validation of Automated Vulnerability Curation and Characterization," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3241–3260, 2023.

[2] MITRE, "CVE - Common Vulnerabilities and Exposures," *https://cve.mitre.org/*, 2024, accessed: 2024-11.

[3] NIST, "National Vulnerability Database (NVD)," *https://nvd.nist.gov/*, 2024, accessed: 2024-11.

[4] A. Khare, S. Dutta, Z. Li, A. Solko-Breslin, R. Alur, and M. Naik, "Understanding the Effectiveness of Large Language Models in Detecting Security Vulnerabilities," *arXiv preprint arXiv:2311.16169*, 2023.

[5] A. Cheshkov, P. Zadorozhny, and R. Levichev, "Evaluation of ChatGPT Model for Vulnerability Detection," *arXiv preprint arXiv:2304.07232*, 2023.

[6] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing," in *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 2024.

[7] S. Mitra, S. Neupane, T. Chakraborty, S. Mittal, A. Piplai, M. Gaur, and S. Rahimi, "LOCALINTEL: Generating Organizational Threat Intelligence from Global and Local Cyber Knowledge," *arXiv preprint arXiv:2401.10036*, 2024.

[8] F. Perrina, F. Marchiori, M. Conti, and N. V. Verde, "AGIR: Automating Cyber Threat Intelligence Reporting with Natural Language Generation," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 3053–3062.

[9] P. Sharma and B. Dash, "Impact of Big Data Analytics and ChatGPT on Cybersecurity," in *2023 4th International Conference on Computing and Communication Systems (I3CS)*. IEEE, 2023, pp. 1–6.

[10] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, "From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy," *IEEE Access*, 2023.

[11] E. Karlsen, X. Luo, N. Zincir-Heywood, and M. Heywood, "Benchmarking Large Language Models for Log Analysis, Security, and Interpretation," *Journal of Network and Systems Management*, vol. 32, no. 3, p. 59, 2024.

[12] S. Ullah, M. Han, S. Pujar, H. Pearce, A. Coskun, and G. Stringhini, "LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks," in *IEEE Symposium on Security and Privacy*, 2024.

[13] S. Chopra, H. Ahmad, D. Goel, and C. Szabo, "ChatNVD: Advancing Cybersecurity Vulnerability Assessment With Large Language Models," *arXiv preprint arXiv:2412.04756*, 2024.

[14] J. Qi, S. Huang, Z. Luan, S. Yang, C. Fung, H. Yang, D. Qian, J. Shang, Z. Xiao, and Z. Wu, "LogGPT: Exploring ChatGPT for Log-Based Anomaly Detection," in *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2023, pp. 273–280.

[15] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," *arXiv preprint arXiv:2311.05232*, 2023.

[16] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. Tonmoy, A. Chadha, A. P. Sheth, and A. Das, "The Troubling Emergence of Hallucination in Large Language Models–An Extensive Definition, Quantification, and Prescriptive Remediations," *arXiv preprint arXiv:2310.04988*, 2023.

[17] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, "Improving Language Models by Retrieving from Trillions of Tokens," *International Conference on Machine Learning*, pp. 2206–2240, 2022.

[18] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.

[19] W. Zhang and J. Zhang, "Hallucination Mitigation for Retrieval-Augmented Large Language Models: A Review," *Mathematics*, vol. 13, no. 5, p. 856, 2025.

[20] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," 2021.

[21] Google, "Introducing Gemma 3: The Most Capable Model You Can Run on a Single GPU or TPU," *https://blog.google/technology/developers/gemma-3/*, 2025, accessed: 2025-04.

[22] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," *arXiv preprint arXiv:2501.12948*, 2025.

[23] Mistral AI, "Mistral Small 3," *https://mistral.ai/news/mistral-small-3*, 2025, accessed: 2025-03.

[24] HuggingFace, "DeepSeek-R1-Distill-Llama-8B," *https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B*, 2025, accessed: 2025-03.

[25] Meta, "Llama 3.2: Revolutionizing edge AI and vision with open, customizable models," *https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/*, 2025, accessed: 2025-01.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] OpenAI, "Models-OpenAI," *https://platform.openai.com/docs/models/gpt-3-5*, 2024, accessed: 2023-11.

[28] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.

[29] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A Survey of Large Language Models," *arXiv preprint arXiv:2303.18223*, 3 2023. [Online]. Available: https://arxiv.org/abs/2303.18223v10

[30] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity," *arXiv preprint arXiv:2403.14403*, 2024.

[31] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," *The Twelfth International Conference on Learning Representations*, 2023.

[32] J. Kim, J. Nam, S. Mo, J. Park, S.-W. Lee, M. Seo, J.-W. Ha, and J. Shin, "SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs," *arXiv preprint arXiv:2404.13081*, 2024.

[33] X. Du, G. Zheng, K. Wang, J. Feng, W. Deng, M. Liu, B. Chen, X. Peng, T. Ma, and Y. Lou, "Vul-RAG: Enhancing LLM-based Vulnerability Detection via Knowledge-level RAG," *arXiv preprint arXiv:2406.11147*, 2024.

[34] X. Ma, G. Fang, and X. Wang, "LLM-Pruner: On the Structural Pruning of Large Language Models," *Advances in neural information processing systems*, vol. 36, pp. 21 702–21 720, 2023.

[35] X. Ding, Y. Zhu, Y. Zhang, and C. Xie, "A Sliding Layer Merging Method for Efficient Depth-Wise Pruning in LLMs," *arXiv preprint arXiv:2502.19159*, 2025.

Appendix

1. The Independency in LLMs' Hidden States

In Table 2, we showed that the LLM for CVE-2025-30066 (with and without RAG) increasingly treats each token as an independent variable as depth grows, with a notable degree of independence already present in the earliest layers. To support the generality of this phenomenon across different vulnerabilities, we included Table 6 for CVE-2025-24472, which demonstrates a similar pattern and thus reinforces our original observation. Additional tests on various other CVEs further confirmed the consistency of this trend. As the input propagates through successive transformer blocks, the model increasingly treats tokens as independent vectors, suggesting that deeper layers re-encode representations in ways that abstract away initial inter-token dependencies. We hypothesize that this behavior stems from the attention mechanism itself, which is designed to assign independent attention weights to each token. This mechanism inherently encourages the model to treat tokens independently, which could partly explain why LLMs need to be so large to effectively capture complex interdependencies and deliver strong performance.

TABLE 6: Layer–by–layer rank evolution for each model for CVE-2025-22472

(a) Llama-3.2-3B-Instruct (28 Layers)

	28 Layers
Layer 0	
rank $x\theta y$	54%
rank x_y	67%
Layers 1–3	
rank $x\theta y$	97%
rank x_y	99%
Layers 4–27	
rank $x\theta y$	100%
rank xy	100%

(b) DeepSeek-R1-Distill-Llama-8B (32 Layers)

	32 Layers
Layer 0	
rank $x\theta y$	14%
rank x_y	15%
Layers 1–14	
rank $x\theta y$	78%
rank x_y	80%
Layers 15–31	
rank $x\theta y$	100%
rank xy	100%

(c) Mistral-Small-24B-Instruct-2501 (40 Layers)

	40 Layers
Layer 0	
rank $x\theta y$	48%
rank x_y	68%
Layers 1–5	
rank $x\theta y$	96%
rank x_y	98%
Layers 6–39	
rank $x\theta y$	100%
rank xy	100%

(d) Gemma-3-27b-it (62 Layers)

	62 Layers
Layer 0	
rank $x\theta y$	45%
rank x_y	65%
Layers 1–28	
rank $x\theta y$	86%
rank x_y	94%
Layers 29–61	
rank $x\theta y$	100%
rank xy	100%